ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

22Υ103
ΕΙΣΑΓΩΓΗ ΣΤΟΥΣ ΥΠΟΛΟΓΙΣΤΕΣ

Εργαστηριακές Ασκήσεις

Κ. Σγάρμπας, Ν. Αβούρης, Π. Σταθοπούλου

Πάτρα - Σεπτέμβριος 2010
(Έκδοση 3.01)

Περιεχόμενα

1. Εισαγωγή

Γενικά

Στόχος των εργαστηριακών ασκήσεων του μαθήματος "22Υ103 Εισαγωγή στους Υπολογιστές" που διδάσκεται στο $1^{ο}$ εξάμηνο, είναι η γνωριμία και πρακτική άσκηση των φοιτητών με τους υπολογιστές και τα βασικά εργαλεία υπολογιστικής υποδομής που θα χρειαστούν κατά την διάρκεια των σπουδών τους, καθώς και η εξισορρόπηση των γνώσεων φοιτητών που προέρχονται από διαφορετικές κατευθύνσεις της δευτεροβάθμιας εκπαίδευσης. Ιδιαίτερη έμφαση δίδεται στην εισαγωγή στην αλγοριθμική σκέψη και τον προγραμματισμό με τη χρήση της γλώσσας υψηλού επιπέδου Python.

Στα πλαίσια του εργαστηρίου, ο φοιτητής θα γνωρίσει και θα ασκηθεί πρακτικά στις εξής περιοχές:

- Εξοικείωση με τους υπολογιστές και τις υπηρεσίες του Υπολογιστικού Κέντρου του Τμήματος και τους κανόνες λειτουργίας του
- Ασκήσεις δημιουργίας και διερεύνησης αλγορίθμων επίλυσης δοθέντων προβλημάτων
- Εισαγωγή στον προγραμματισμό με χρήση της γλώσσας προγραμματισμού Python.
- Εισαγωγή στη χρήση, τις βασικές έννοιες και τον προγραμματισμό του διαδικτύου, καθώς και λειτουργίες σχετικά με αυτό

Στο δεύτερο μισό του εξαμήνου θα ζητηθεί να γίνει μια ομαδική εργασία (project) με αντικείμενο τον προγραμματισμό. Η εργασία αυτή απαιτεί από τους νέους φοιτητές να γνωριστούν μεταξύ τους καθώς και να εργαστούν ομαδικά, ώστε να αντιμετωπίσουν σε περισσότερο βάθος κάποιο θέμα που σχετίζεται με επίλυση ενός πραγματικού προβλήματος στην περιοχή της επιστήμης του μηχανικού, χρησιμοποιώντας τις γνώσεις που έχουν αποκτήσει στα πλαίσια του μαθήματος.

Ο σπουδαστής στην φάση των εργαστηριακών ασκήσεων θα χρησιμοποιήσει το Υπολογιστικό Κέντρο του Τμήματος. Επίσης θα μπορεί να το χρησιμοποιήσει αν το επιθυμεί στις ώρες που είναι διαθέσιμο για την ομαδική εργασία ή για εξάσκηση. Για περισσότερες πληροφορίες για το Υπολογιστικό Κέντρο και τις διαδικασίες που θα πρέπει να γνωρίζετε προκειμένου να διεκπεραιώσετε τις εργαστηριακές ασκήσεις μεταβείτε στο κεφάλαιο 4 – Χρήση του Υπολογιστικού Κέντρου.

Χρονοδιάγραμμα

Το εργαστήριο περιλαμβάνει δύο φάσεις οι οποίες εξελίσσονται παράλληλα. Η συνολική του διάρκεια είναι 13 ακαδημαϊκές εβδομάδες.

Στη φάση Α γίνεται η εκπόνηση εργαστηριακών ασκήσεων στο χώρο του Υπολογιστικού Κέντρου. Ο αριθμός των εργαστηριακών ασκήσεων είναι δέκα(10)

και θα διεξάγεται μία (1) ανά εβδομάδα. Συγκεκριμένα τα θέματα των εργαστηριακών ασκήσεων είναι κατά σειρά τα εξής:

Εβδομάδα	Εργαστηριακή Άσκηση
1η	Γνωριμία με το εργαστήριο και τις υποδομές του Κέντρου
2η	Το Διαδίκτυο
3η	Σχεδίαση αλγορίθμων 1
4η	Σχεδίαση αλγορίθμων 2
5η	Άσκηση Προγραμματισμού Python 1
6η	Άσκηση Προγραμματισμού Python 2
7η	Άσκηση Προγραμματισμού Python 3
8η	Άσκηση Προγραμματισμού Python 4
9η	Άσκηση Προγραμματισμού Python 5
13η	Τελική Εξέταση Εργαστηρίου

Στη φάση Β που διαρκεί από την εβδομάδα 7 έως 12 γίνεται η εκπόνηση των ομαδικών εργασιών. Η φάση αυτή θα αρχίσει την 7η εβδομάδα του εξαμήνου και διαρκεί εξι (6) εβδομάδες. Τα θέματα των ομαδικών εργασιών, η δημιουργία των ομάδων και η ημερομηνία κατάθεσης και εξέτασής τους θα ανακοινωθούν από τους διδάσκοντες.

Κανόνες διεξαγωγής

Η παρακολούθηση του εργαστηρίου, η εκπόνηση τόσο όλων των εργαστηριακών ασκήσεων και των ομαδικών εργασιών και η έγκαιρη κατάθεση των εργασιών που ανατίθενται είναι, σύμφωνα με κανονισμό του Τμήματος, υποχρεωτικές και η επιτυχής τους ολοκλήρωση είναι προϋπόθεση για συμμετοχή στην τελική εξέταση του μαθήματος. Δεν επιτρέπονται απουσίες στο εργαστήριο. Αν για λόγους ανωτέρας βίας δεν είναι δυνατή η συμμετοχή σε κάποια εργαστηριακή άσκηση στο τέλος του εξαμήνου θα δοθεί η δυνατότητα αναπλήρωσης της κατά την εβδομάδα των επαναληπτικών ασκήσεων. Το εργαστηριακό σκέλος του μαθήματος συμμετέχει κατά 30% στον τελικό βαθμό του μαθήματος «22Υ103 Εισαγωγή στους Υπολογιστές Ι».

Στο διάστημα εκπόνησης της ομαδικής εργασίας, οι φοιτητές κάθε ομάδας, που αποτελείται από 4-5 άτομα, δρουν με δική τους πρωτοβουλία και με συναντήσεις που οι ίδιοι συντονίζουν, θα πρέπει να προγραμματίσουν και να οργανώσουν τις επιμέρους εργασίες καθώς και το υλικό που θα αναλάβει το κάθε μέλος της ομάδας, ώστε στο τέλος να παραδοθεί έγκαιρα η ομαδική εργασία με ενιαία δομή, εμφάνιση και περιεχόμενο. Είναι ευθύνη των μελών κάθε ομάδας να μοιράσουν ισόποσα τον εργασιακό φόρτο μεταξύ τους ώστε να μην υπάρχει αδικία στην τελική αξιολόγηση της ομαδικής τους εργασίας, μιας και ο βαθμός της ομαδικής εργασίας είναι συνήθως κοινός για όλα τα μέλη της ομάδας. Προβλέπεται εξέταση της εργασίας αυτής στο τέλος του εξαμήνου.

Στο διάστημα της φάσης εκπόνησης των εργαστηριακών ασκήσεων στο Υπολογιστικό Κέντρο, οι φοιτητές χωρίζονται σε διαφορετικές ομάδες 50 περίπου φοιτητών η κάθε μια, στις οποίες διατίθεται μία συγκεκριμένη μέρα και συγκεκριμένες ώρες κατά τις οποίες εκτελούν την εργαστηριακή άσκηση στο

Υπολογιστικό Κέντρο. Οι ομάδες και το ωρολόγιο πρόγραμμα των εργαστηριακών ασκήσεων θα ανακοινωθούν από τους διδάσκοντες. Η διάρκεια που διατίθεται για την εκπόνηση της κάθε εργαστηριακής άσκησης είναι δύο (2) ώρες. Οι φοιτητές θα πρέπει **να έχουν προετοιμαστεί από πριν**, να έχουν μελετήσει το αντίστοιχο κεφάλαιο θεωρίας από το διδακτικό βοήθημα και άλλες πηγές, αφού ο χρόνος άσκησης στο χώρο του Υπολογιστικού Κέντρου πιθανόν να μην είναι αρκετός για να λυθούν όλες τους οι απορίες.

Φυλλάδιο

Στο φυλλάδιο αυτό περιλαμβάνονται οδηγίες για την εκπόνηση των εργαστηριακών ασκήσεων και των ομαδικών εργασιών. Πρόσθετες πληροφορίες για το μάθημα μπορείτε να βρείτε στην ιστοσελίδα (http://hci.ece.upatras.gr) και για το εργαστήριο στον ηλεκτρονικό χώρο υποστήριξης του εργαστηρίου (http://hci.ece.upatras.gr/labs), στον πίνακα ανακοινώσεων έξω από το Υπολογιστικό Κέντρο ΚΥΠΕΣ του Τμήματος. Ο κανονισμός λειτουργίας του Υπολογιστικού Κέντρου ΚΥΠΕΣ μπορείτε να βρείτε στην ιστοσελίδα του http://kypes.ece.upatras.gr/.

Ευχαριστίες

Ειδική αναφορά πρέπει να γίνει στους μεταπτυχιακούς σπουδαστές του Τμήματος που δούλεψαν μαζί με τους διδάσκοντες για την παραγωγή του υλικού αυτού. Ο **Νίκος Τσέλιος** στα θέματα του διαδικτύου, η **Ελένη Βογιατζάκη** στα διαγράμματα ροής αλγορίθμων. Επίσης οι διδάσκοντες εκφράζουν τις ευχαριστίες τους στον προπτυχιακό φοιτητή **Δημήτρη Σκρεπετό** για τις ιδέες και τις υποδείξεις του.

2. Εργαστηριακές ασκήσεις

Συχνές ερωτήσεις

Τι αντικείμενο έχουν οι εργαστηριακές ασκήσεις;

Οι σπουδαστές είναι υποχρεωμένοι να εκπονήσουν όλες τις εργαστηριακές ασκήσεις, σύμφωνα με το πρόγραμμα του εργαστηρίου που διαρκεί 10 εβδομάδες. Το αντικείμενο των εργαστηριακών ασκήσεων είναι κατά σειρά:

1η Γνωριμία με τις ηλεκτρονικές υπηρεσίες του εργαστηρίου
2η Διαδίκτυο: FTP, Email, Φυλλομετρητές, HTML
3η Σχεδίαση διαγραμμάτων ροής αλγορίθμων: Βασικές Έννοιες
4η Σχεδίαση διαγραμμάτων ροής αλγορίθμων: Προχωρημένα Θέματα
5η Άσκηση Προγραμματισμού 1
6η Άσκηση Προγραμματισμού 2
7η Άσκηση Προγραμματισμού 3
8η Άσκηση Προγραμματισμού 4
9η Άσκηση Προγραμματισμού 5
10η Εξέταση Εργαστηρίου

Πόσες απουσίες μπορώ να κάνω στο εργαστήριο;

Στο εργαστήριο δεν δικαιολογούνται απουσίες. Θα γίνει προσπάθεια όμως να δοθεί η δυνατότητα αναπλήρωσης δικαιολογημένα χαμένων εργαστηριακών ασκήσεων <u>αν υπάρχουν</u> ελεύθερες εβδομάδες στο τέλος του εξαμήνου.

Ποιο είναι το παραδοτέο σε κάθε εργαστηριακή άσκηση;

Σε κάθε εργαστηριακή άσκηση θα πρέπει να παραδοθεί μία **Έκθεση** (συνήθως με ηλεκτρονικό τρόπο) το αργότερο μέχρι το τέλος της ώρας άσκησης, εκτός αν σας δοθούν διαφορετικές οδηγίες κατά τη διάρκεια του εργαστηρίου. Οι εκθέσεις που παραδίδετε βαθμολογούνται από τους διδάσκοντες του εργαστηρίου με άριστα το 10, ενώ οι διδάσκοντες θα αναφέρουν τυπικά λάθη και παρανοήσεις στο επόμενο εργαστήριο.

Πώς υπολογίζεται ο βαθμός του εργαστηρίου;

Η κάθε έκθεση θα βαθμολογηθεί ξεχωριστά και στο τέλος βγαίνει ο μέσος όρος των εκθέσεων των εργαστηρίων. Ο τελικός Βαθμός εργαστηρίου προκύπτει ως μέσος όρος αφενός του βαθμού των εργαστηριακών εκθέσεων και του βαθμού της ομαδικής εργασίας και αφετέρου του βαθμού της τελικής εξέτασης εργαστηρίου που θα γίνει την τελευταία βδομάδα.

Πώς υπολογίζεται ο βαθμός του μαθήματος;

Ο συνολικός βαθμός του μαθήματος *22Y103 Εισαγωγή στους Υπολογιστές Ι* προκύπτει από τον **Βαθμό εργαστηρίου (40%)**, **και το βαθμό της Τελικής γραπτής εξέτασης**

του μαθήματος (60%).

Εξεταζόμαστε κατά τη διάρκεια του εργαστηρίου;

Οι διδάσκοντες συχνά κατά τη διάρκεια των εργαστηρίων μπορεί να κάνουν ερωτήσεις στους συμμετέχοντες. Όμως θα πρέπει να γίνει κατανοητό ότι οι εργαστηριακές ασκήσεις δεν είναι διαγωνισμός γνώσης και δεν έχουν το χαρακτήρα εξέτασης. Το κύριο ζητούμενο είναι ο φοιτητής να πειραματιστεί και να *μάθει* και για το λόγο αυτό οι βοηθοί και οι διδάσκοντες θα βοηθήσουν τους φοιτητές να λύσουν απορίες και να κάνουν τις ασκήσεις, ενώ συχνά οι φοιτητές ενθαρρύνονται να συνεργάζονται στην επίλυση των ασκήσεων. Εξάλλου στο τέλος του ακαδημαϊκού εξαμήνου θα γίνει γραπτή εξέταση του εργαστηριακού έργου με βαρύτητα 50% του τελικού βαθμού εργαστηρίου, οπότε εκεί θα ελεγχθεί και θα πιστοποιηθεί η γνώση και οι δεξιότητες που κατακτήθηκαν κατά τη διάρκεια του έτους. Για το λόγο αυτό είναι καλό να επικεντρωθείτε κατά τη διάρκεια των ασκήσεων στην ουσιαστική κατανόηση και εκπόνηση των ασκήσεων και όχι απλά στην διεκπεραίωση τους. Είναι καλή πρακτική να κρατάτε σημειώσεις και αντίγραφα των εργασιών που παραδίδετε.

Πώς προετοιμαζόμαστε για κάθε άσκηση;

Για κάθε μια από τις ασκήσεις αυτές θα **πρέπει να έχει γίνει προετοιμασία**, γιατί ο χρόνος άσκησης δεν είναι αρκετός για την μελέτη της θεωρίας και των τεχνικών που θα χρειαστεί να χρησιμοποιηθούν. Ο υπεύθυνος του εργαστηρίου στην αρχή κάθε εργαστηριακής άσκησης θα κάνει εισαγωγή στη θεωρία και τους στόχους της άσκησης αλλά αυτή δεν μπορεί να είναι σε μεγάλο βάθος. Εξάλλου οι διδάσκοντες έχουν συζητήσει θέματα που σχετίζονται με το εργαστήριο στο αμφιθέατρο κατά τη διδασκαλία του μαθήματος.

Μπορώ να συμμετάσχω στην τελική εξέταση της θεωρίας αν δεν πάρω καλό βαθμό στο εργαστήριο;

Ναι μπορείς. Προυπόθεση για συμμετοχή στην τελική εξέταση δεν είναι να πάρετε προβιβάσιμο βαθμό στο εργαστήριο, αλλά να έχετε συμμετάσχει στις ασκήσεις και την ομαδική εργασία.

Αν έχετε κάνει απουσίες ή δεν παραδώσετε εργασία θα πρέπει να επαναλάβετε το εργαστήριο του χρόνου και δεν μπορείτε να δώσετε την τελική εξέταση του μαθήματος. Αν έχετε ολοκληρώσει τις ασκήσεις και την εργασία, ανεξάρτητα από τον τελικό βαθμό σας στο εργαστήριο, μπορείτε να δώσετε τελική εξέταση του μαθήματος.

Αν χάσω την τελευταία άσκηση (εξέταση του εργαστηρίου) μπορώ να την επαναλάβω το Σεπτέμβρη ή κάποια άλλη μέρα;

Όχι, δεν προβλέπεται επαναληπτική εξέταση.

Αν γίνονται συνελεύσεις των φοιτητικών συλλόγων γίνονται εργαστηριακές ασκήσεις;

Ναι μετά από απόφαση της Γενικής Συνέλευσης του Τμήματος, οι ασκήσεις γίνονται κανονικά όταν υπάρχουν τέτοιες εκδηλώσεις, απλά η απουσία θεωρείται δικαιολογημένη για όσους θέλουν να συμμετάσχουν στη Συνέλευση, αλλά θα πρέπει να αναπληρώσουν την άσκηση όταν δοθεί ευκαιρία στο τέλος του εξαμήνου.

Εργαστήριο 1: Εισαγωγή στις ηλεκτρονικές υπηρεσίες του Πανεπιστημίου, Τμήματος, Εργαστηρίου

1.1 Γενικά

Στην διάρκεια του πρώτου εργαστηρίου θα γίνει η επαφή των πρωτοετών φοιτητών με τον χώρο του **Κέντρου Υπολογιστικών και Επικοινωνιακών Συστημάτων (ΚΥΠΕΣ)**[1] του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών και τις **Ηλεκτρονικές Υπηρεσίες** που υποστηρίζουν το Εργαστήριο του μαθήματος **«22Υ103 Εισαγωγή στους Υπολογιστές Ι».** Επιπρόσθετα θα πραγματοποιηθεί η ανάθεση των θεμάτων των ομαδικών εργασιών. Στο συγκεκριμένο εργαστήριο δεν θα παραδοθεί Έκθεση εργαστηρίου.

Οι κυριότερες ιστοδιευθύνσεις που κρίνεται απαραίτητο οι πρωτοετείς φοιτητές να ανατρέξουν για να πληροφορηθούν για τις **Ηλεκτρονικές Υπηρεσίες** σε επίπεδο Πανεπιστημίου και Τμήματος παρουσιάζονται στο παρακάτω πίνακα.

Ιστότοπος Τμήματος ΗΜΤΥ ***http://www.ece.upatras.gr***
Δικτυακή Πύλη Τμήματος ΗΜΤΥ ***http://myece.ece.upatras.gr***
Ιστότοπος Τμήματος Δικτύων Πανεπιστημίου (UPNet) ***http://www.upnet.gr***
Πλατφόρμα e-class ***http://eclass.upatras.gr/***
Ιστότοπος Πανεπιστημίου Πατρών ***http://www.upatras.gr***

Για την διαχείριση του μαθήματος **«22Υ103 Εισαγωγή στους Υπολογιστές Ι»** χρησιμοποιούνται δύο διαδικτυακοί Ιστότοποι.
Ο πρώτος είναι η πλατφόρμα **eClass** η οποία αποτελεί ένα ολοκληρωμένο Σύστημα Διαχείρισης Ηλεκτρονικών Μαθημάτων, ασύγχρονης τηλεκπαίδευσης του Πανεπιστημίου Πατρών. Η πλατφόρμα **eClass** χρησιμοποιείται για τα περισσότερα μαθήματα του Τμήματος και θα χρησιμέψει κυρίως για τη διδασκαλία του θεωρητικού σκέλους του μαθήματος.
Ο δεύτερος ιστότοπος είναι ο χώρος υποστήριξης κυρίως; ηλεκτρονικών εργαστηριακών μαθημάτων που έχει αναπτύξει η ομάδα **Αλληλεπίδρασης Ανθρώπου-Υπολογιστή** για τα μαθήματα που έχει αναλάβει να υποστηρίζει.

Είναι όμως απαραίτητο καταρχήν να πραγματοποιηθεί μια σύντομη αλλά πλήρης παρουσίαση όλων των ηλεκτρονικών υπηρεσιών του πανεπιστήμιου Πατρών η οποία θα επιτρέψει στον πρωτοετή φοιτητή να αποκτήσει μια ολοκληρωμένη εικόνα των

[1] Ειδικά διαμορφωμένος χώρος και με κατάλληλη υποδομή (computer-room) για την εκπαίδευση των φοιτητών με χρήση υπολογιστών.

αξιόλογων υπηρεσιών που οι ψηφιακές τεχνολογίες του προσφέρουν σε όλα τα επίπεδα της φοιτητικής του ζωής και θα τον διευκολύνουν άμεσα αλλά και στο μέλλον.

1.2 Γνωριμία με την πλατφόρμα eClass (http://eclass.upatras.gr)

Στα πλαίσια του θεωρητικού σκέλους του μαθήματος θεωρείται υποχρεωτική η εγγραφή των φοιτητών στο σύστημα ασύγχρονης τηλεκπαίδευσης του Πανεπιστημίου Πατρών **eClass**. Σκοπός της χρήσης του είναι να διευκολύνει την συνεργασία μεταξύ διδάσκοντα και φοιτητή στα πλαίσια της εκπαιδευτικής διαδικασίας για την όλη διεξαγωγή του μαθήματος **«22Υ103 Εισαγωγή στους Υπολογιστές Ι»,** με κάποιες υπηρεσίες που αυτό προσφέρει.
Συγκεκριμένα μέσα από το **eClass** οι φοιτητές θα μπορούν να ενημερώνονται για την ατζέντα του μαθήματος και τις διάφορες ανακοινώσεις, να «κατεβάζουν» κείμενα που τυχόν να χρειαστούν κατά την πορεία και τέλος να παραδίδουν τις φροντιστηριακές ασκήσεις που οι διδάσκοντες αναθέτουν. Βασικό πλεονέκτημα του **eClass** είναι η δυνατότητα απομακρυσμένης πρόσβασης, με αποτέλεσμα να μην απαιτείται η φυσική επικοινωνία των φοιτητών με τους υπεύθυνους του μαθήματος και των εργαστηρίων για να ενημερωθούν για τις τρέχουσες εξελίξεις ή για να παραδώσουν τις φροντιστηριακές ή ομαδικές ασκήσεις. Αντίθετα κάθε φοιτητής με πρόσβαση στο **Internet** μπορεί να συνδεθεί με το **eClass** και συγκεκριμένα στην διεύθυνση http://eclass.upatras.gr, και να έχει άμεση και έγκαιρη ενημέρωση για τα μαθήματα που τον ενδιαφέρουν .

Προσοχή!!!! το σύστημα αυτό δεν έχει σκοπό και δεν μπορεί να αντικαταστήσει την παρακολούθηση του μαθήματος.

Για να συνδεθεί κανείς με την πλατφόρμα **eClass** και να γραφτεί ως φοιτητής του μαθήματος «**22Υ103 Εισαγωγή στους Υπολογιστές Ι**», απαιτείται μία συγκεκριμένη διαδικασία, την οποία πρέπει να ακολουθήσει και η οποία περιγράφεται στις παρακάτω παραγράφους.
Για πρόσβαση στο σύστημα, καταρχήν χρειάζεται ένας λογαριασμός, τον οποίον αναλαμβάνει να δημιουργήσει η Υπηρεσία Διαχείρισης Δικτύου του Πανεπιστημίου Πατρών. Τα στοιχεία του λογαριασμού, όνομα χρήστη (**username**) και συνθηματικό (**password**), του κάθε σπουδαστή, παραδίδονται σε αυτόν από την Γραμματεία του τμήματος Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών κατά την πρώτη του εγγραφή στο τμήμα.
Έχοντας τα στοιχεία αυτά ο φοιτητής θα πρέπει να ανατρέξει στη ιστοδιεύθυνση **http://eclass.upatras.gr** με την βοήθεια ενός φυλλομετρητή (**web browser, Internet Explorer, Mozilla Firefox**) και να εγγραφεί χρησιμοποιώντας την οθόνη που φαίνεται στο σχήμα 1.
Στην περίπτωση πρώτης πρόσβασης στην πλατφόρμα **eClass** ο φοιτητής θα πρέπει να επιλέξει την ένδειξη **Εγγραφή Φοιτητή** που βρίσκεται στο μενού των επιλογών της αριστερής πλευράς της οθόνης του σχήματος 1. Η επιλογή αυτή αυτόματα θα οδηγήσει τον φοιτητή στην οθόνη του σχήματος 2 και στην συνέχεια επιλέγοντας

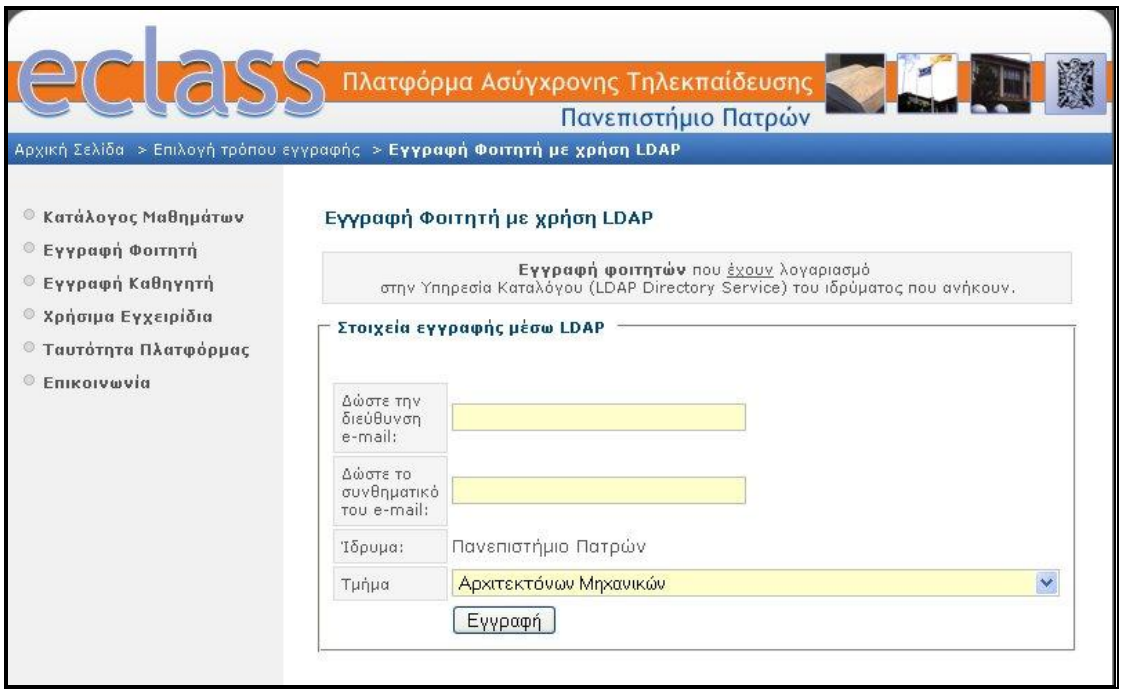
Σχήμα 1. Οθόνη Εγγραφής / Σύνδεσης Φοιτητή.

την μοναδική ενεργή επιλογή **Εγγραφή Φοιτητή με χρήση LDAP** και η οποία θα τον οδηγήσει στην οθόνη του σχήματος 3 στην φόρμα της οποίας θα πρέπει να καταχωρίσει τα στοιχεία τα οποία του υπαγορεύει.

Σχήμα 2. Οθόνη επιλογής τρόπου εγγραφής.

Τα πεδία τα οποία θα πρέπει να συμπληρώσει ο φοιτητής είναι τρία εκ των οποίων τα δύο πρώτα είναι μοναδικά για τον κάθε φοιτητή και θα τα αναζητήσει στο έντυπο που του επιδίδεται κατά την εγγραφή του[2]. Το τρίτο πεδίο αποτελεί ένα πτυσσόμενο παράθυρο το οποίο επιτρέπει στον φοιτητή να διευκρινίσει το τμήμα στου οποίου τα μαθήματα επιθυμεί την εγγραφή.

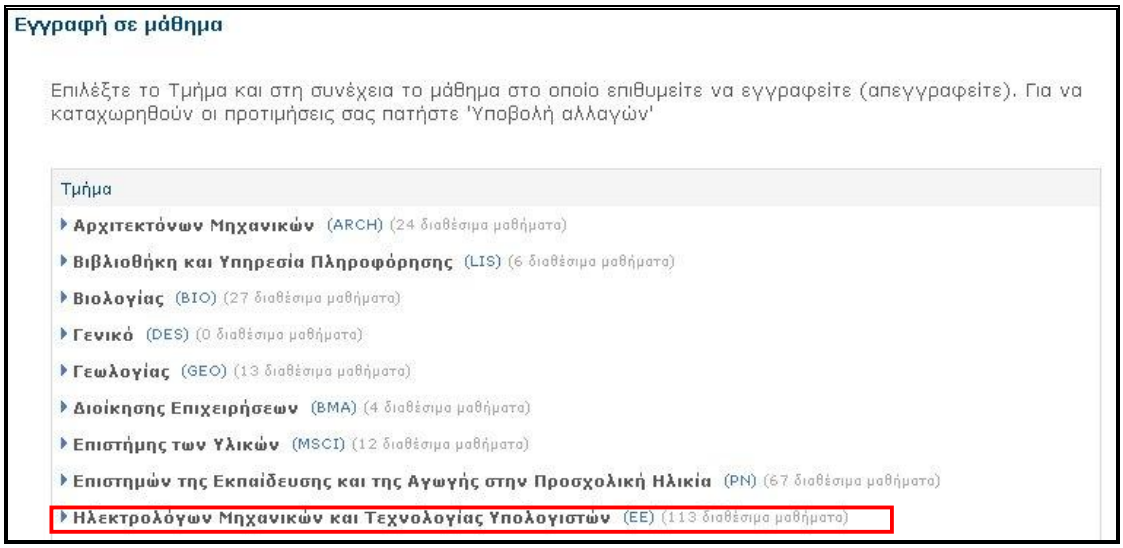
[2] Τίτλος εντύπου: «Ο Κόσμος του Διαδικτύου στο Πανεπιστήμιο Πατρών»



Σχήμα 3. Οθόνη εγγραφής Φοιτητή με χρήση **LDP.**

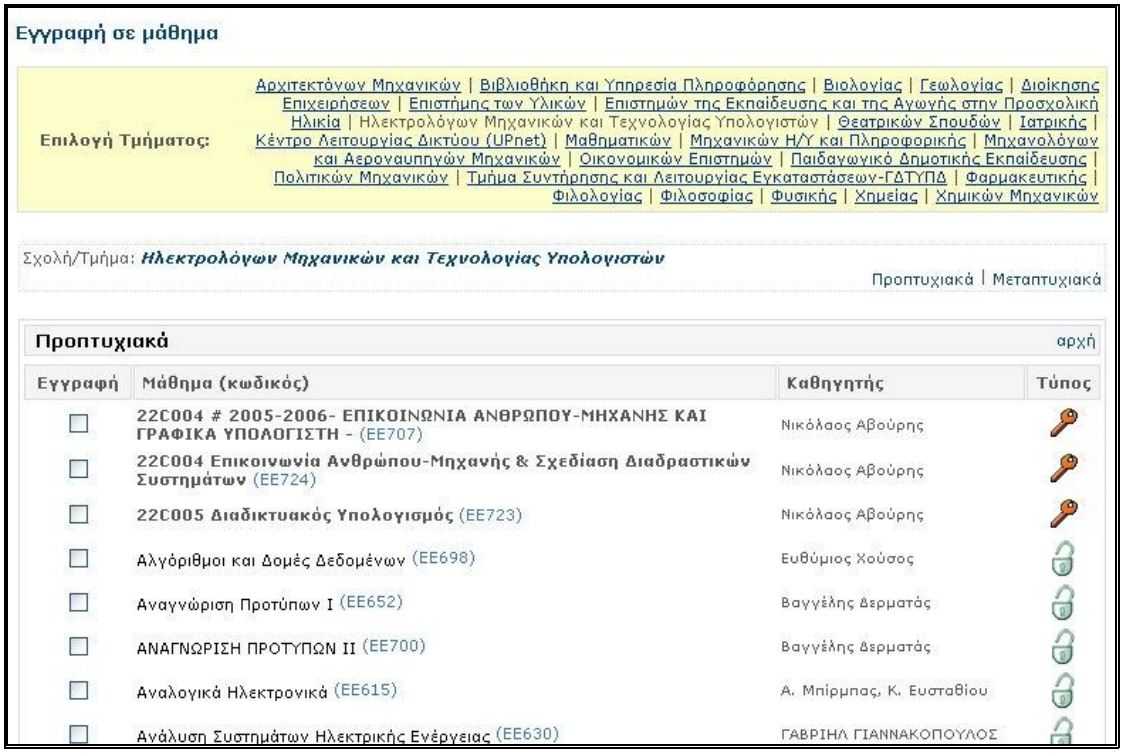
- **Διεύθυνση e-mail:** **ee6705@upnet.gr**
- **Συνθηματικό του e-mail:** **xxxxxxx**
- **Τμήμα** **Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών**

Από την λίστα τμημάτων/σχολών του πανεπιστημίου Πατρών που θα εμφανισθεί, θα επιλεγεί το τμήμα των «**Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών**». Η συμπλήρωση της συγκεκριμένης φόρμας θα πρέπει να γίνει με ιδιαίτερη προσοχή και αφού ο φοιτητής έχει φροντίσει να συλλέξει σωστά τα στοιχεία που του ζητάει. Βεβιασμένες κινήσεις συμπλήρωσης της συγκεκριμένης φόρμας θα οδηγήσουν τον φοιτητή σε περιπέτειες και πιθανόν καθυστερήσεις εγγραφής του στην πλατφόρμα **eClass.**



Σχήμα 4. Τμήμα Οθόνης λίστας τμημάτων πανεπιστημίου Πατρών**.**

Μετά την συμπλήρωση της φόρμας ενεργοποιούμε την εγγραφή μας πιέζοντας το πλήκτρο **Εγγραφή** και αυτόματα μετακινούμαστε στην οθόνη του σχήματος 4, όπου εμφανίζονται όλα τα τμήματα του πανεπιστημίου Πατρών λίστα των ηλεκτρονικών μαθημάτων του τμήματος. Επιλέγοντας το τμήμα **Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών** μετακινούμαστε στην οθόνη του σχήματος 5 όπου εμφανίζεται η λίστα όλων των ηλεκτρονικών μαθημάτων του τμήματος.



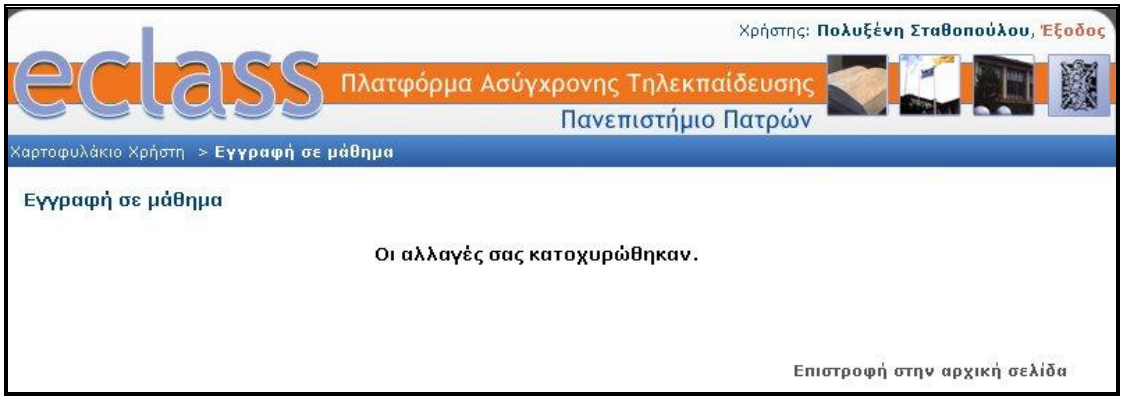
Σχήμα 5. Τμήμα Οθόνης λίστας μαθημάτων του τμήματος ΤΗΜΤΥ.

Τα ηλεκτρονικά μαθήματα διακρίνονται σε τρεις κατηγορίες:

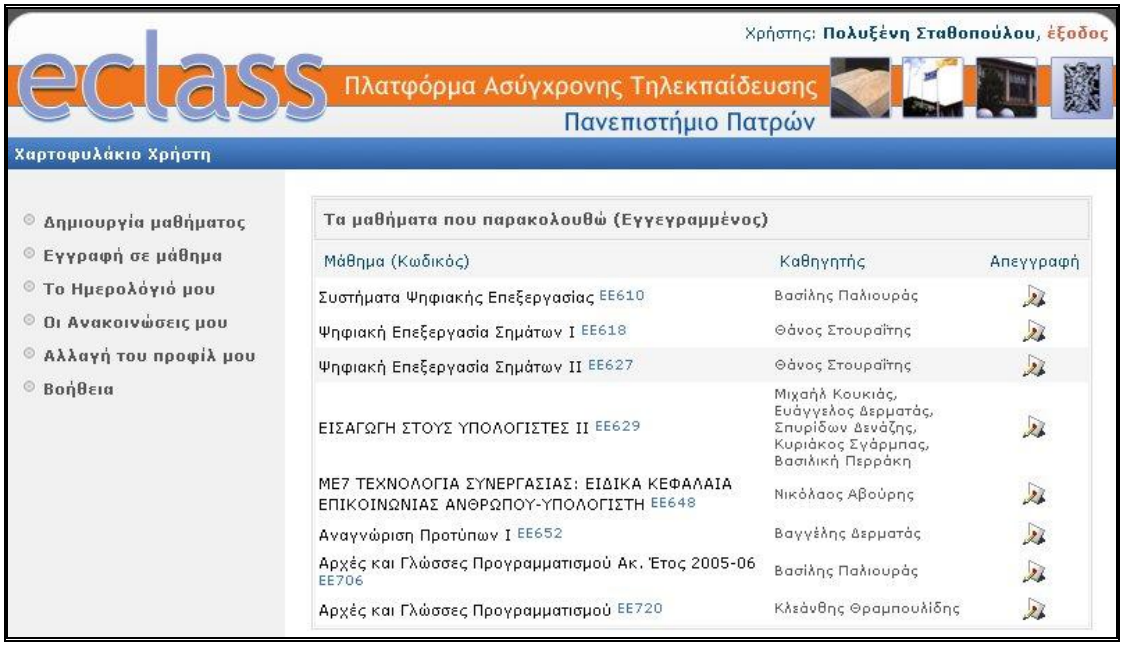
- **ανοιχτά**
- **απαιτούν εγγραφή**
- **κλειστά**

Σημειώνοντας στην αριστερή πλευρά της οθόνης του σχήματος 5 τα επιλεγόμενα μαθήματα στην συνέχεια μετακινούμαστε στο κάτω μέρος της σελίδας (σχήμα 6) για να ενεργοποιήσουμε, πιέζοντας το πλήκτρο **υποβολής αλλαγών**, οριστικά την προεπιλεγόμενη εγγραφή μαθημάτων. Η τελευταία επιλογή μας μεταφέρει αυτόματα στην οθόνη του σχήματος 7, όπου μας επιβεβαιώνει την υιοθέτηση των επιλογών μας και παράλληλα μας οδηγεί μέσω της επιλογής **επιστροφή στην αρχική σελίδα** στην οθόνη του σχήματος 8 η οποία αποτελεί το χαρτοφυλάκιο του κάθε χρήστη της πλατφόρμας **eClass**.

Σχήμα 6. Τμήμα Οθόνης ενεργοποίησης εγγραφής.



Σχήμα 7. Οθόνη επιβεβαίωση εγγραφής.



Σχήμα 8. Τμήμα Οθόνης χαρτοφυλακίου χρήστη πλατφόρμας **eClass**.

Αυτή ήταν η διαδικασία εγγραφής για πρώτη φορά στην πλατφόρμα **eClass** και παράλληλα στο μάθημα. «**22Υ103 Εισαγωγή στους Υπολογιστές Ι**». Από τις επόμενες φορές τόσο για την διαχείριση του συγκεκριμένου μαθήματος όσο και για

την εγγραφή σας σε άλλα ηλεκτρονικά μαθήματα του τμήματός σας ή άλλων τμημάτων θα πρέπει να ακολουθήσετε την διαδικασία εισόδου του χρήστη.

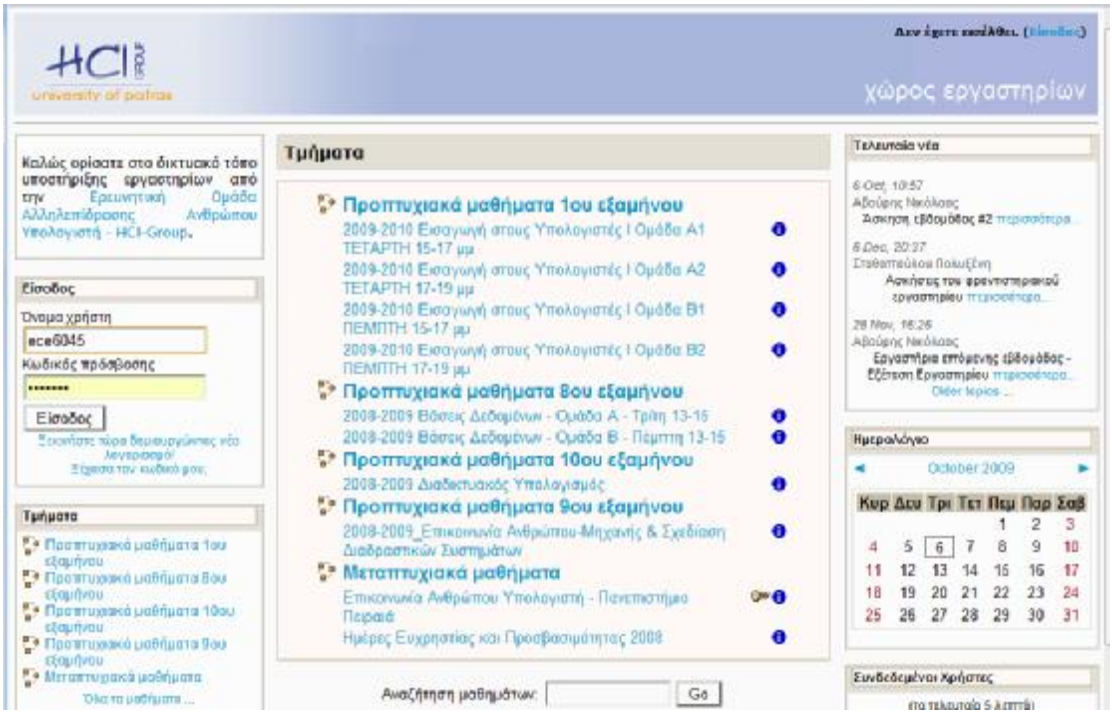
Προσοχή**!!!** ως **username** χρησιμοποιούμε την διεύθυνση του **e-mail (**δηλ. **eexxxx@upnet.gr)** και ως **password** το συνθηματικό**,** που καταχωρίσατε κατά την πρώτη σας εγγραφή στην πλατφόρμα **eClass.**

Για περισσότερες πληροφορίες και οδηγίες χρήσης πλατφόρμας **eClass** μπορείτε να ανατρέξετε στην ιστοδιεύθυνση http://eclass.upatras.gr/manuals/manual.php

1.3 Γνωριμία με το Σύστημα Ηλεκτρονικής Υποστήριξης των Εργαστηρίων (http://hci.ece.upatras.gr/labs)

Εισαγωγικά
Οι φοιτητές που παρακολουθούν το εργαστήριο του μαθήματος **«22Υ103 Εισαγωγή τους Υπολογιστές Ι»** θα πρέπει να εγγραφούν και να χρησιμοποιήσουν τον ιστότοπο υποστήριξης του Εργαστηρίου του μαθήματος. Αυτός βρίσκεται στην ηλεκτρονική διεύθυνση http://hci.ece.upatras.gr/labs. Το σύστημα αυτό έχει αναπτυχθεί και συντηρείται από το Εργαστήριο της Ερευνητικής Ομάδας Αλληλεπίδρασης Ανθρώπου-Υπολογιστή του Τμήματος μας. Το σύστημα αυτό έχει βασιστεί στην πλατφόρμα Τηλεκπαίδευσης ανοικτού κώδικα **Moodle** και περιέχει υλικό για όλα τα εργαστηριακά μαθήματα που υποστηρίζονται από τη συγκεκριμένη Ερευνητική Ομάδα. Η ανάπτυξη της πλατφόρμας αυτής όπως και του υλικού του εργαστηρίου έγινε αρχικά και υποστηρίχτηκε από το έργο ΕΠΕΑΕΚ ΙΙ του Υπουργείου Παιδείας κατά το ακαδ. έτος 2005-2006, αν και το υλικό των εργαστηριακών ασκήσεων και των οδηγιών ανανεώνεται κάθε χρόνο.

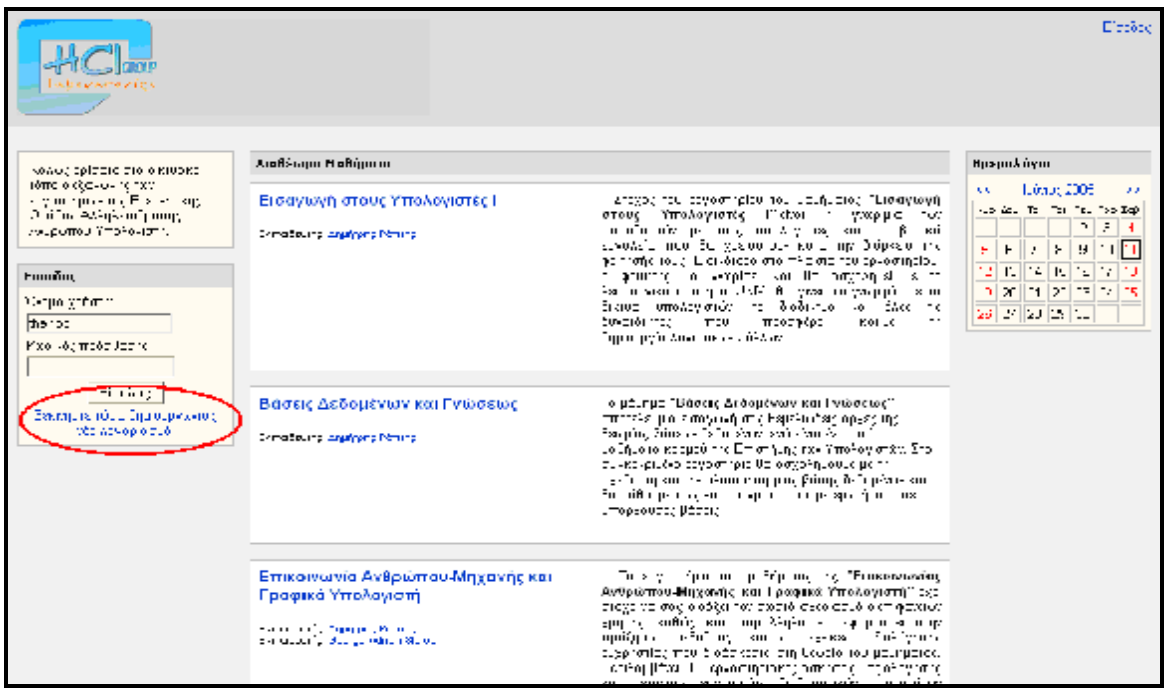
Από το σύστημα αυτό οι φοιτητές θα μπορούν να ενημερωθούν για σχετικά γεγονότα και ανακοινώσεις, να έχουν πρόσβαση σε λογισμικά εργαλεία, φυλλάδια και οδηγίες χρήσεως σε ηλεκτρονική μορφή και να συζητήσουν με άλλους σπουδαστές. Κατά την είσοδό του στο σύστημα ο φοιτητής βλέπει την οθόνη με όλα τα εργαστήρια που υποστηρίζονται από την Ερευνητική Ομάδα Αλληλεπίδρασης Ανθρώπου Υπολογιστή του Τμήματος.



Οθόνη υποδοχής συστήματος Ηλεκτρονικής Υποστήριξης των Εργαστηρίων

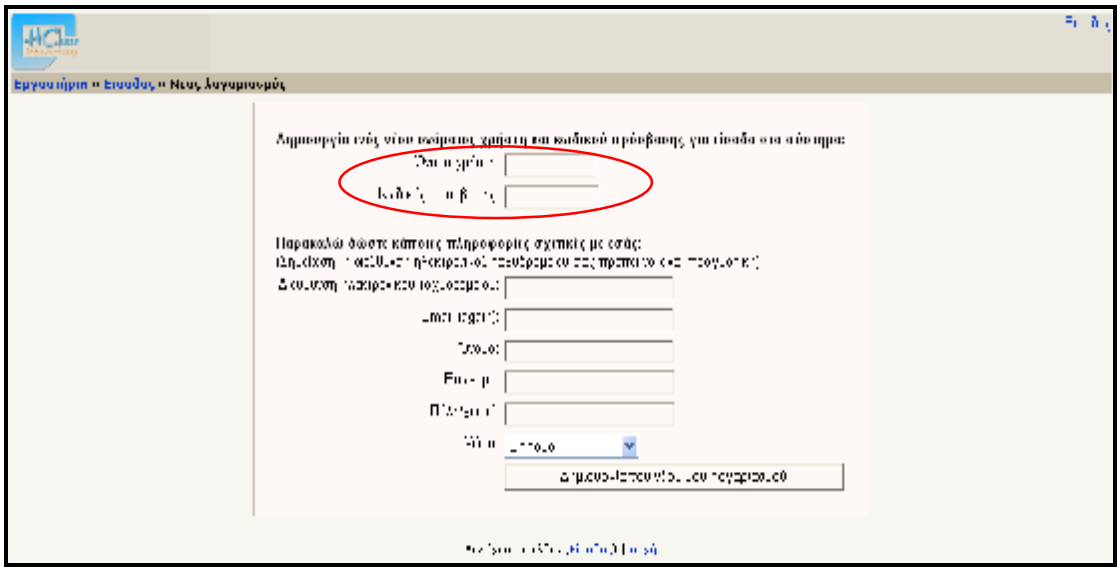
Εγγραφή Φοιτητή - Δημιουργία λογαριασμού

Για να χρησιμοποιήσει ο φοιτητής το σύστημα αυτό θα πρέπει πρώτα να δημιουργήσει έναν νέο λογαριασμό. Αυτό γίνεται εφικτό μέσω της επιλογής **«Ξεκινήστε τώρα δημιουργώντας νέο λογαριασμό»** που βρίσκεται στην οθόνη υποδοχής του συστήματος ηλεκτρονικής υποστήριξης εργαστηριών (βλ. παρακάτω εικόνα):



Υπερσύνδεσμος που οδηγεί στην οθόνη δημιουργίας νέου λογαριασμού

Ακολουθώντας αυτόν τον υπερσύνδεσμο εμφανίζεται η οθόνη δημιουργίας νέου λογαριασμού η οποία και φαίνεται παρακάτω.



Οθόνη δημιουργίας νέου λογαριασμού

Ο φοιτητής υποχρεούται να συμπληρώσει όλα τα στοιχεία που η συγκεκριμένη φόρμα εγγραφής του υπαγορεύει και με συγκεκριμένο τρόπο. Πρέπει λοιπόν καταρχήν να πληκτρολογήσει το **Όνομα Χρήστη (login)** και τον **Κωδικό Πρόσβασης (password)** τα οποία θα χρησιμοποιεί κάθε φορά που επιθυμεί να συνδεθεί με το σύστημα. Ο τρόπος δήλωσης των δύο αυτών στοιχείων υποδεικνύεται στον παρακάτω πίνακα.

Όνομα Χρήστη:	**aYY-XXXX**
Κωδικός Πρόσβασης:	*Κάποιο συνδυασμό χαρακτήρων που να μην τον ξεχάσετε ποτέ!*

Όπου:
YY= ο αριθμός έτους εγγραφής στη σχολή (π.χ. για το 2008 έχουμε YY=08) και XXXX = τα τέσσερα τελευταία ψηφία του αριθμού μητρώου σας (π.χ. 4548)

Στην συνέχεια θα συνεχίσετε την συμπλήρωση των επόμενων στοιχείων της φόρμας, **Email, Όνομα, Επώνυμο**.

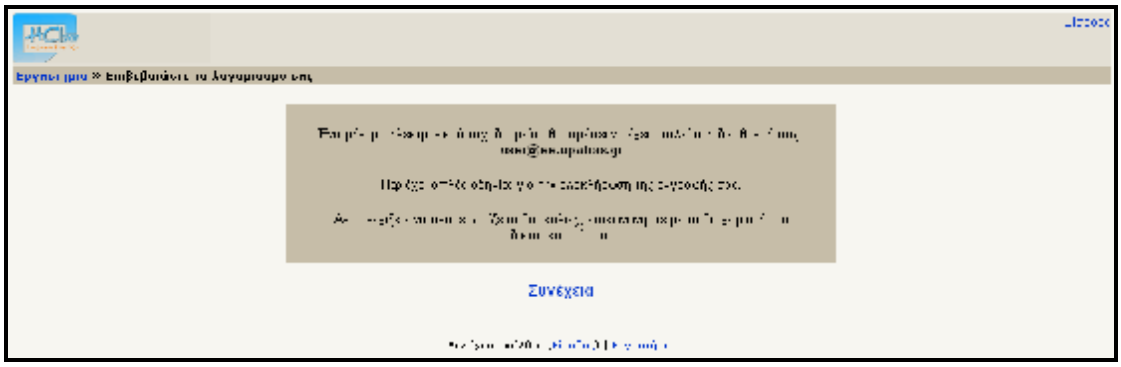
- **Προσοχή:**

το **email** θα πρέπει να είναι έγκυρο γιατί το σύστημα απαιτεί επιβεβαίωση εγγραφής του χρήστη μέσω ενός μηνύματος που στέλνει στην συγκεκριμένη ηλεκτρονική διεύθυνση. Κατά προτίμηση χρησιμοποιείται το **email** σας στο υπολογιστικό κέντρο του τμήματος στο **upnet (όνομα@upnet.gr)**

Email:	**eeXXXX@upnet.gr**
Όνομα:	*Το πραγματικό σας όνομα με **κεφαλαίους ελληνικούς χαρακτήρες. ΟΧΙ** ψευδώνυμο*
Επώνυμο:	*Το πραγματικό σας επώνυμο με **κεφαλαίους ελληνικούς χαρακτήρες.***

	ΟΧΙ *ψευδώνυμο*

Αφού συμπληρωθεί η φόρμα ο φοιτητής συνεχίζει πατώντας «*Δημιουργία του νέου λογαριασμού*» όπου και μεταβαίνει στην οθόνη επιβεβαίωσης του λογαριασμού.



Οθόνη επιβεβαίωσης λογαριασμού

Στο σημείο αυτό ο φοιτητής θα πρέπει να περιμένει μέχρι να λάβει μήνυμα όπως φαίνεται παρακάτω και να κάνει την επιβεβαίωση.

From: Διαχειριστής Χρήστης [mailto:support@rheya.csl.ee.upatras.gr]
Sent: Monday, June 06, 2005 2:33 PM
To: student@ee.upatras.gr
Subject: Εργαστήρια ερευνητικής ομάδας αλληλεπίδρασης ανθρώπου υπολογιστή: επιβεβαίωση λογαριασμού

Γειά σας,
Έχει γίνει αίτηση ενός νέου λογαριασμού στο 'Εργαστήρια ερευνητικής ομάδας αλληλεπίδρασης ανθρώπου υπολογιστή' χρησιμοποιώντας τη δική σας διεύθυνση ηλεκτρονικού ταχυδρομείου.
Για να επιβεβαιώσετε το νέο λογαρισμό σας, παρακαλώ επισκεφθείτε την παρακάτω διεύθυνση:

http://rheya.csl.ee.upatras.gr/moodle/login/confirm.php?p=g7kPfYVvlyJjUTn&s=student2

Στα περισσότερα προγράμματα ηλεκτρονικού ταχυδρομείου αυτό θα πρέπει να εμφανίζεται σαν μπλε σύνδεσμος στο οποίο μπορείτε να πατήσετε με το ποντίκι. Αν αυτό δε λειτουργεί τότε αντιγράψτε και επικολλήστε αυτή τη διεύθυνση στη γραμμή διευθύνσεων στο πάνω μέρος του φυλλομετρητή.

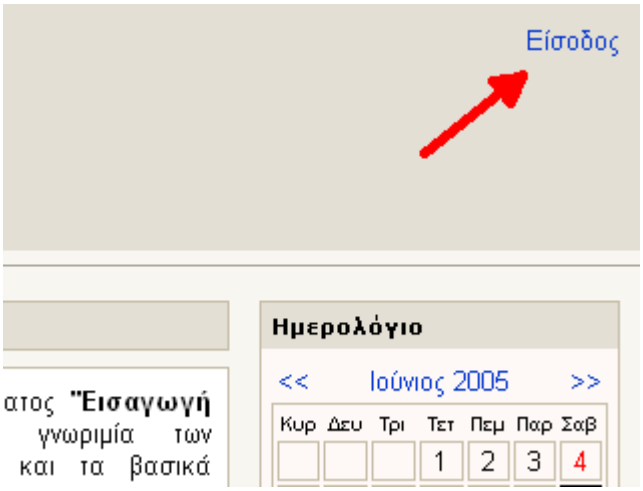
Φιλικά,
ο διαχειριστής των 'Εργαστήρια ερευνητικής ομάδας αλληλεπίδρασης ανθρώπου υπολογιστή',

Διαχειριστής Χρήστης (support@rheya.csl.ee.upatras.gr)

Ενδεικτικό Email Επιβεβαίωσης της Εγγραφής

Σύνδεση στο σύστημα ηλεκτρονικής υποστήριξης εργαστηρίων

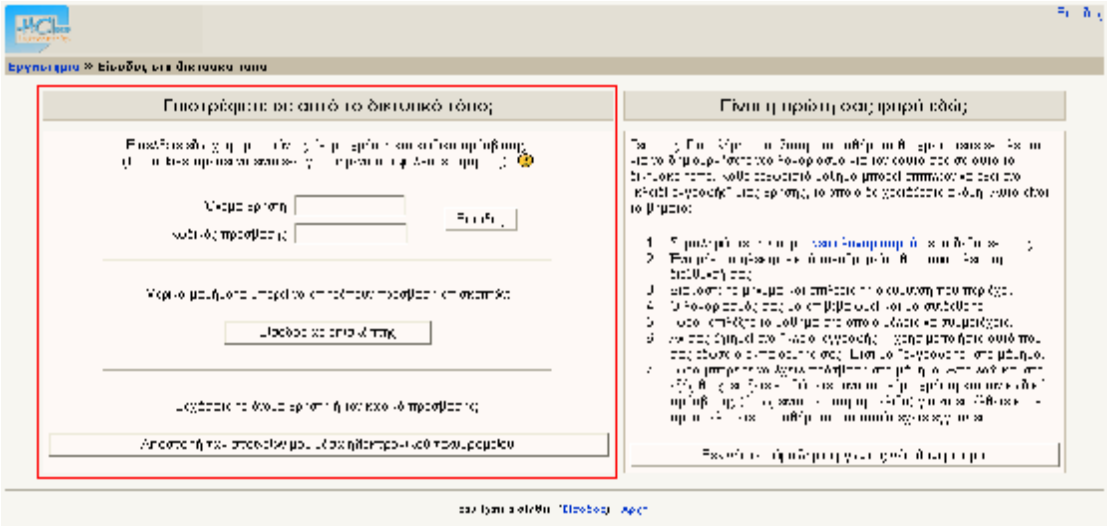
Στο σημείο αυτό ο φοιτητής μπορεί πλέον να συνδεθεί με το σύστημα δίνοντας το όνομα χρήστη (login) και τον κωδικό (password) του λογαριασμού που δημιούργησε. Για να το κάνει αυτό θα πρέπει να πατήσει στην επιλογή **Είσοδος** που βρίσκεται πάνω δεξιά (ή εναλλακτικά μπορεί να βάλει τα στοιχεία του στο κουτάκι είσοδος αριστερά) .



Υπενθυμίζουμε ότι ο λογαριασμός σας θα έχει την εξής μορφή:

Όνομα Χρήστη	**aYY-XXX***
Κωδικός Πρόσβασης	**Ο κωδικός σας**

Αφού ο φοιτητής έχει μεταφερθεί στο οθόνη εισόδου με το σύστημα υποστήριξης και εφόσον υπάρχει λογαριασμός τότε γράφει τα στοιχεία του (login και password) στην φόρμα του αριστερού πλαισίου, όπως φαίνεται παρακάτω.

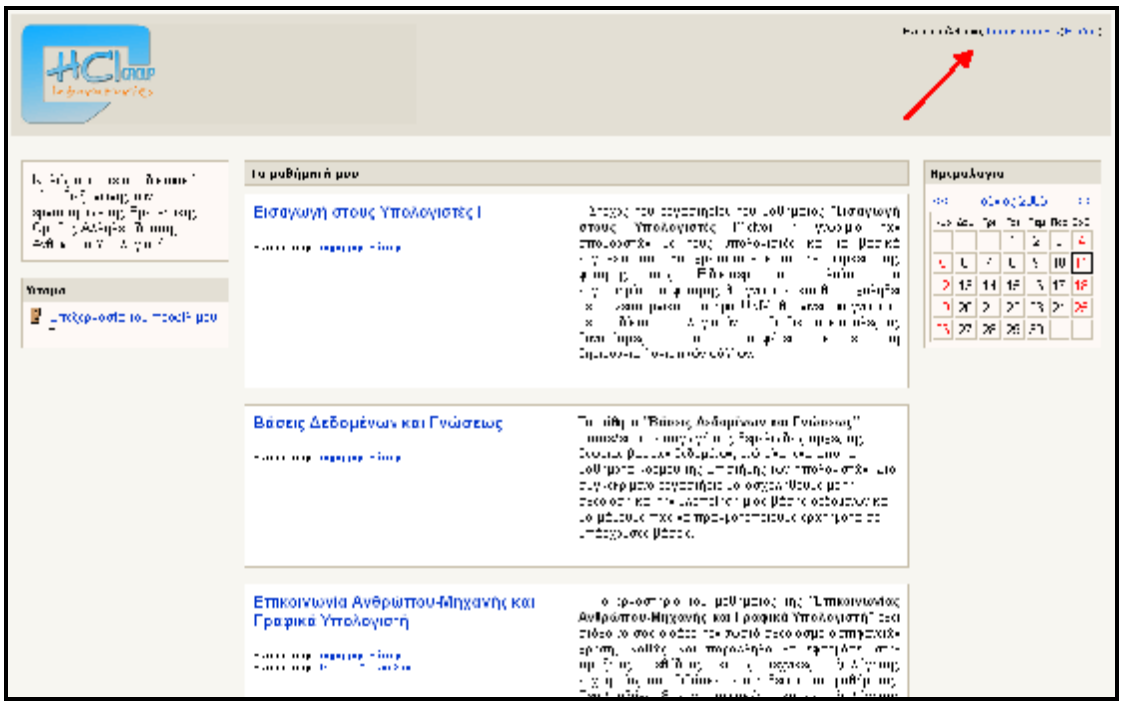


Οθόνη εισόδου

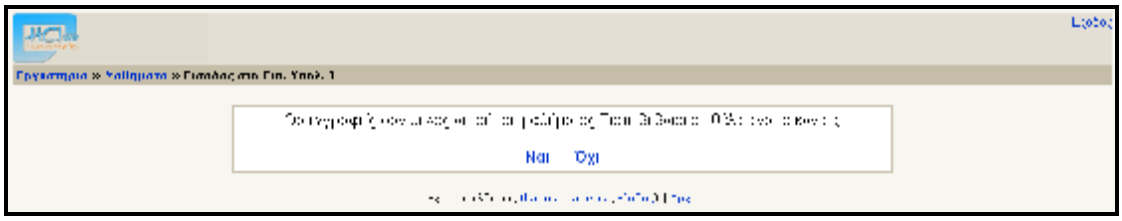
Εγγραφή στο μάθημα «Εισαγωγή στους Υπολογιστές Ι»

Από την στιγμή που ο φοιτητής έχει συνδεθεί στο σύστημα, έχοντας δώσει login και password, βρίσκεται στην οθόνη που φαίνεται παρακάτω.

* YY= ο αριμός έτους (π.χ. για το 2006 έχουμε YY=06) και XXXX = αριθμός μητρώου σας (π.χ. 4548)



Χαρακτηριστικό γνώρισμα της επιτυχούς σύνδεσής του είναι η ένδειξη στο πάνω μέρος δεξιά που αναγράφει «Έχετε συνδεθεί ως …». Στο σημείο αυτό ο φοιτητής χρειάζεται να εγγραφεί στο μάθημα που θέλει να παρακολουθήσει. Για να το κάνει αυτό θα πρέπει να πατήσει στο αντίστοιχο μάθημα. Λόγω του ότι είναι η πρώτη φορά που ο φοιτητής εγγράφεται στο μάθημα, το σύστημα ζητάει επιβεβαίωση εγγραφής στο μάθημα όπως φαίνεται στην παρακάτω οθόνη.



Οθόνη επιβεβαίωσης εγγραφής στο μάθημα

Σε περίπτωση που ζητείται κλειδί εγγραφής, θα σας δοθεί στο μάθημα ή μπορείτε να ρωτήσετε τον βοηθό του εργαστηρίου σας. Αφού ο φοιτητής πατήσει «Ναι» μεταφέρεται στην βασική οθόνη του μαθήματος που έχει επιλέξει.

Η πρώτη επαφή του φοιτητή με το σύστημα ηλεκτρονικής υποστήριξης εργαστηρίων

Αφού δώσει το όνομα χρήστη και τον κωδικό πρόσβασης, ο φοιτητής εισέρχεται σε μια σελίδα όπου βλέπει τα μαθήματα που είναι εγγεγραμμένος.

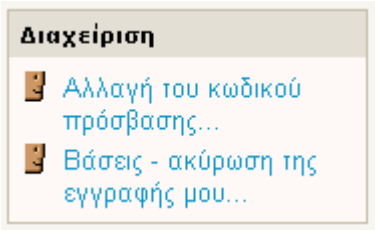
Εισαγωγική σελίδα για το φοιτητή

Στη συγκεκριμένη περίπτωση ο φοιτητής έχει γραφτεί σε τρία μαθήματα. Επιλέγουμε για παράδειγμα το μάθημα «Εισαγωγή στους Υπολογιστές Ι» και στη συνέχεια στον φοιτητή εμφανίζεται η αρχική σελίδα του αντίστοιχου εργαστηρίου.

Εισαγωγική σελίδα μαθήματος «Εισαγωγή στους Υπολογιστές Ι»

Ο φοιτητής έχει τις εξής δυνατότητες από το σύστημα :

Α) Διαχείριση

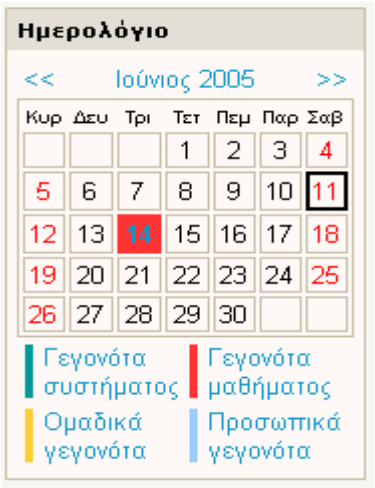


Ο φοιτητής μπορεί να αλλάξει τον κωδικό πρόσβασής του, να επεξεργαστεί γενικά το προφίλ του και να διαγραφεί από το μάθημα «Εισαγωγή στους Υπολογιστές Ι».

Β) Ανακοινώσεις

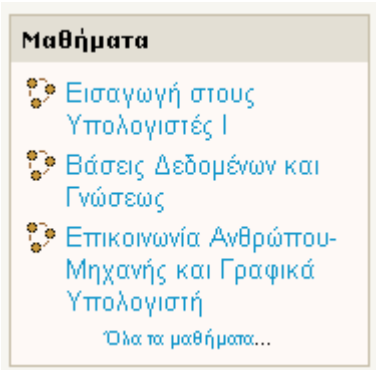
Ο φοιτητής μπορεί να δει τις ανακοινώσεις που έχουν αναρτήσει οι καθηγητές του εργαστηρίου.

Γ) Ημερολόγιο



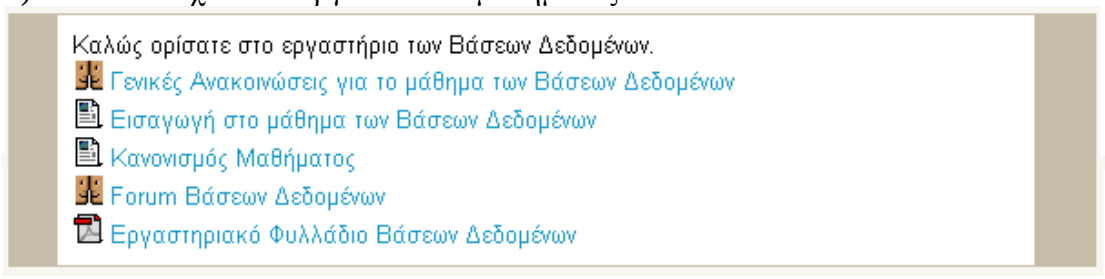
Στο ημερολόγιο ο φοιτητής μπορεί να ενημερώνεται για τα επικείμενα γεγονότα που αφορούν το μάθημα. Στη συγκεκριμένη περίπτωση ο φοιτητής πρέπει να παραδώσει στις 14 Ιουνίου την 1η εργαστηριακή άσκηση.

Δ) Μαθήματα



Στο συγκεκριμένο μενού ο φοιτητής μπορεί να μεταβεί στα υπόλοιπα μαθήματα στα οποία έχει εγγραφεί.

Ε) Γενικά στοιχεία και εργαλεία του μαθήματος



Και το συγκεκριμένο μενού, όπως και τα προηγούμενα, βρίσκεται μόνιμα στη σελίδα του μαθήματος «Εισαγωγή στους Υπολογιστές Ι». Αναλυτικά οι λειτουργίες του συγκεκριμένου μενού είναι:

- «**Γενικές Ανακοινώσεις για το μάθημα Εισαγωγή στους Υπολογιστές Ι**»

Στο συγκεκριμένο σύνδεσμο αναρτώνται οι ανακοινώσεις από τους καθηγητές του μαθήματος «Εισαγωγή στους Υπολογιστές Ι» και ταυτόχρονα αποστέλλονται με email στην διεύθυνση που έχει δώσει ο χρήστης (π..χ στο aYY-XXXX@students.ee.upatras.gr).

- «**Εισαγωγή στο μάθημα Εισαγωγή στους Υπολογιστές Ι**»

Ο συγκεκριμένος σύνδεσμος οδηγεί σε μια σελίδα όπου γίνεται μια εισαγωγή για το εργαστήριο του μαθήματος «Εισαγωγή στους Υπολογιστές Ι».

- «**Κανονισμός Μαθήματος**»

Ο συγκεκριμένος σύνδεσμος οδηγεί σε μια σελίδα όπου ο φοιτητής ενημερώνεται για τον κανονισμό του μαθήματος.

- «**Forum Εισαγωγή στους Υπολογιστές Ι**»

Στο συγκεκριμένο forum οι φοιτητές έχουν ελεύθερη πρόσβαση και μπορούν να συζητήσουν οτιδήποτε αφορά το μάθημα «Εισαγωγή στους Υπολογιστές Ι». Στο συγκεκριμένο παράδειγμα το forum είναι άδειο και ο φοιτητής μπορεί να θέση το δικό του θέμα συζήτησης και να δεχθεί απαντήσεις από τους συμφοιτητές του και τους βοηθούς του εργαστηρίου.

- «**Εργαστηριακό Φυλλάδιο μαθήματος «Εισαγωγή στους Υπολογιστές Ι**»

Σε αυτό το σύνδεσμο βρίσκεται σε μορφή pdf ολόκληρο το εργαστηριακό φυλλάδιο του μαθήματος «Εισαγωγή στους Υπολογιστές Ι». Σε περίπτωση που θέλετε να αποθηκεύσετε ένα pdf αρχείο από τον ηλεκτρονικό χώρο του εργαστηρίου στον υπολογιστή σας απλά **ανοίγετε το επιθυμητό αρχείο** (με αριστερό κλικ) και χρησιμοποείται το **save a copy.**

Στην συνέχεια παρατίθενται ορισμένες συχνές ερωτήσεις και οι απαντήσεις του που αποτελούν προϊόν συσσωρευμένης διδακτικής εμπειρίας και θα πρέπει να τις οπωσδήποτε να τις διαβάσετε.

Εργαστήριο 2: Υπηρεσίες Διαδικτύου: FTP, Email, Στρατηγικές πλοήγησης και αναζήτησης στο Διαδίκτυο, HTML

2.1 Γενικά

Στόχος του εργαστηρίου αυτού είναι η εξάσκηση στη χρήση των υπηρεσιών του ηλεκτρονικού ταχυδρομείου του Πανεπιστημίου Πατρών, και στην πλοήγηση και αναζήτηση επιστημονικής πληροφορίας στο διαδίκτυο καθώς και τη δημιουργία απλών ιστοσελίδων. Για το εργαστήριο αυτό θα πρέπει να μελετήσετε τις αντίστοιχες ενότητες του βιβλίου θεωρίας (σελ. 219-249).

2.2 Ηλεκτρονικό Ταχυδρομείο

Το ηλεκτρονικό ταχυδρομείο είναι η πιο διαδεδομένη και παλιά υπηρεσία του διαδικτύου. Είναι ασύγχρονο μέσο επικοινωνίας που μεταφέρει τη φιλοσοφία ανταλλαγής επιστολών μεταξύ χρηστών υπολογιστών.
Σε περιβάλλον Windows και Linux υπάρχει πληθώρα εφαρμογών διαχείρισης ηλεκτρονικού ταχυδρομείου όπως Mozila Thunderbird, Microsoft Outlook, Eudora, Netscape Messanger κλπ, ενώ είναι συνηθισμένο να χρησιμοποιείται υπηρεσία ηλεκτρονικού ταχυδρομείου μέσα από τα πρωτόκολλα του παγκόσμιου ιστού, δηλαδή με πρόσβαση στα μηνύματα μέσα από το φυλλομετρητή του ιστού μετά από εγγραφή και δημιουργία σχετικού λογαριασμού στην αντίστοιχη ιστοσελίδα. Η υπηρεσία αυτή συνήθως παρέχεται δωρεάν, όπως η υπηρεσία gmail.com, hotmail.com κλπ.

Ωστόσο στο εργαστήριο θα μάθουμε να χρησιμοποιούμε την υπηρεσία **webmail** του Πανεπιστημίου Πατρών. Η υπηρεσία webmail δίνει την δυνατότητα να διαχειριζόμαστε την αλληλογραφία μας χωρίς να απαιτείται στον υπολογιστή που χρησιμοποιούμε να υπάρχει κάποια εφαρμογή ηλεκτρονικού ταχυδρομείου την οποία θα πρέπει να έχουμε ρυθμίσει κατάλληλα ώστε να διαχειρίζεται τα ηλεκτρονικά μας μηνύματα. Η υπηρεσία αυτή λειτουργεί όπως και άλλες υπηρεσίες διαδικτυακού ηλεκτρονικού ταχυδρομείου (gmail, hotmail, yahoomail κλπ): απαιτείται σύνδεση σε ιστοσελίδες στις οποίες θα πρέπει να δώσουμε login και password προκειμένου να έχουμε πρόσβαση στα προσωπικά μας μηνύματα.

Πληροφορίες για την υπηρεσία αυτή μπορείτε να βρείτε στη διεύθυνση http://www.upnet.gr/email.php ενώ η σύνδεση στην υπηρεσία αυτή γίνεται μέσω της διεύθυνσης https://mail.upnet.gr Δίνουμε το προσωπικό μας όνομα και κωδικό που μας έχει δοθεί κατά την εγγραφή μας στη Σχολή.

Στο αριστερό πλαίσιο φαίνεται μία λίστα από φακέλους οι οποίοι μας βοηθούν να οργανώσουμε την αλληλογραφία μας. Οι φάκελοι αυτοί είναι οι εξής:

- **Εισερχόμενα**: Είναι ο φάκελος τον οποίο χρησιμοποιούμε πιο πολύ. Στο φάκελο αυτό εισέρχονται τα νέα μηνύματα και παραμένουν εκεί μέχρι να τα σβήσουμε ή να τα μεταφέρουμε σε άλλο φάκελο.
- **Drafts**: Στο φάκελο αυτό αποθηκεύουμε τα μηνύματα τα οποία προετοιμάζουμε αλλά δεν επιθυμούμε να στείλουμε πριν αποσυνδεθούμε από το σύστημα. Έτσι τα αποθηκεύουμε εκεί ώστε να συνεχίσουμε την συγγραφή τους όταν ξανά συνδεθούμε στο σύστημα.
- **Sent**: Στο φάκελο αυτό αποθηκεύονται τα μηνύματα τα οποία έχουμε στείλει στο παρελθόν.
- **Trash**: Στο φάκελο αυτό αποθηκεύονται τα μηνύματα τα οποία έχουμε διαγράψει.

Στο πάνω μέρος της οθόνης υπάρχει ένα σύνολο από βασικές επιλογές της υπηρεσίας διαχείρισης ηλεκτρονικού ταχυδρομείου. Οι επιλογές αυτές είναι οι εξής:

- **Σύνθεση**: Μεταφερόμαστε σε μία φόρμα στην οποία δημιουργούμε και αποστέλλουμε ένα νέο μήνυμα.
- **Διευθύνσεις**: Μεταφερόμαστε σε μία φόρμα στην οποία εισάγουμε νέα διεύθυνση mail φίλου ή γνωστού και την προσθέτουμε σε μία λίστα ώστε να μην χρειάζεται να την απομνημονεύουμε.
- **Φάκελοι**: Δημιουργούμε ένα νέο φάκελο όπως οι φάκελοι «Εισερχόμενα», «Drafts» κτλ.
- **Επιλογές**: Ρυθμίζουμε παραμέτρους του συστήματος ηλεκτρονικού ταχυδρομείου.
- **Αναζήτηση**: Αναζητούμε ένα μήνυμα με βάση κάποια κριτήρια. Η αναζήτηση είναι χρήσιμη όταν τα μηνύματα είναι πάρα πολλά και δεν βολεύει να τα ψάξουμε ένα προς ένα.
- **Βοήθεια**: Λαμβάνουμε οδηγίες χρήσης της υπηρεσίας ηλεκτρονικού ταχυδρομείου.

Τα μηνύματα σε κάθε φάκελο εμφανίζονται ανά ομάδες ώστε το σύστημα να μην καθυστερεί μεταφέροντας μη χρήσιμη πληροφορία. Η λίστα των μηνυμάτων μπορεί να είναι ταξινομημένη με βάση τον αποστολέα, με βάση την ημερομηνία αποστολής του μηνύματος ή με βάση το θέμα. Ο χρήστης μπορεί να μετακινηθεί στις ομάδες μηνυμάτων πατώντας «Προηγούμενη», «Επόμενη» ή απευθείας τον αριθμό της ομάδας την οποία επιθυμεί να δει.

Για να διαχειριστούμε ένα ή περισσότερα μηνύματα (π.χ. να τα διαγράψουμε, να τα μεταφέρουμε σε άλλο φάκελο, να τα δηλώσουμε ως αναγνωσμένα κτλ.) θα πρέπει να μαρκάρουμε το κουτάκι που βρίσκεται αριστερά από μήνυμα και μετά να επιλέξουμε το κουμπί που αντιστοιχεί στην επιθυμητή ενέργεια.

Στο σημείο αυτό θα πρέπει να εξηγήσουμε την έννοια **προώθηση μηνύματος**. Με την προώθηση μηνυμάτων (forward) μπορούμε να στέλνουμε μηνύματα που έχουμε ήδη λάβει από κάποιον αποστολέα Α σε έναν άλλο παραλήπτη Β, χωρίς να χρειάζεται να ξαναγράφουμε το περιεχόμενο του μηνύματος. Επίσης μας δίνεται η δυνατότητα να προσθέσουμε νέο περιεχόμενο αν το επιθυμούμε.

Τέλος όταν λαμβάνουμε μηνύματα έχουμε την δυνατότητα να απαντήσουμε εύκολα και γρήγορα στον αποστολέα πατώντας «Απάντηση» (**reply**). Τα πλεονεκτήματα αυτής της εκδοχής σε σύγκριση με την δημιουργία ενός νέου μηνύματος είναι ότι δεν χρειάζεται να ξαναγράψουμε τον αποστολέα. Επίσης το περιεχόμενο του μηνύματος διατηρείται ενώ μπορούμε να προσθέσουμε νέο περιεχόμενο. Με την διαδικασία αυτή μπορεί να αναπτυχθεί ένας διάλογος με κάποιο πρόσωπο όπου η ιστορία του να σώζεται και να περιέχεται σε κάθε νέο μήνυμα.

Για να κάνουμε reply σε ένα μήνυμα θα πρέπει πρώτα να επιλέξουμε το μήνυμα αυτό από την λίστα στην κύρια οθόνη διαχείρισης ηλεκτρονική μηνυμάτων και στην συνέχεια αφού μετακινηθούμε στην οθόνη στην οποία μπορούμε να διαβάσουμε αναλυτικά το περιεχόμενό του, να πατήσουμε «Απάντηση» (βρίσκεται πάνω δεξιά στο δεξιό πλαίσιο).

2.2 Υπηρεσία FTP

Με την υπηρεσία FTP (File Transfer Protocol) αναφερόμαστε σε ένα πρωτόκολλο επικοινωνίας μεταξύ δύο υπολογιστών το οποίο επιτρέπει την ανταλλαγή αρχείων μεταξύ τους. Στην επικοινωνία αυτή ο ένας από τους δύο υπολογιστές λειτουργεί ως πελάτης (client) όπου θέλει να λάβει ή να αποθηκεύσει κάποια αρχεία από ή σε έναν υπολογιστή, ενώ ο δεύτερος υπολογιστής λειτουργεί ως εξυπηρετητής (server) και περιέχει τα επιθυμητά αρχεία προς λήψη ή τα αρχεία προς αποθήκευση από τον πελάτη.

Ένας πελάτης μπορεί να συνδεθεί σε έναν εξυπηρετητή αρκεί να γνωρίζει την διεύθυνση του εξυπηρετητή, ένα όνομα χρήστη (login name) και τον κατάλληλο κωδικό (password). Επίσης ο πελάτης μπορεί να συνδεθεί ανώνυμα με ένα εξυπηρετητή (εφόσον ο δεύτερος το επιτρέπει) χρησιμοποιώντας την διεύθυνση του εξυπηρετητή, για όνομα χρήστη (login name) την λέξη anonymous και για κωδικό (password) την διεύθυνση του ηλεκτρονικού του ταχυδρομείου. Η διαφορά με την ανώνυμη σύνδεση είναι ότι έχουμε ελεύθερη πρόσβαση σε κάποιους καταλόγους του εξυπηρετητή χωρίς να χρειάζεται κάποιο συγκεκριμένο login name και password. Ωστόσο οι δυνατότητες και τα δικαιώματα που δίνονται σε μία ανώνυμη σύνδεση είναι συνήθως πολύ λιγότερα από μία επώνυμη. Το Υπολογιστικό Κέντρο του ΕΜΠ διαθέτει FTP εξυπηρετητή (ftp.ntua.gr) με δυνατότητα ανώνυμης σύνδεσης. Σε αυτόν θα βρείτε μια μεγάλη συλλογή από προγράμματα.

Από την στιγμή που ένας πελάτης συνδεθεί σε έναν ftp εξυπηρετητή τότε μπορούμε να πούμε ότι ο πελάτης διαχειρίζεται δύο ‘χώρους’. Ο ένας είναι ο τοπικός δηλαδή οι κατάλογοι και τα αρχεία που βρίσκονται στο δικό του υπολογιστή και ο δεύτερος είναι ο απομακρυσμένος δηλαδή οι κατάλογοι και τα αρχεία που βρίσκονται στον ftp εξυπηρετητή. Γι’ αυτό το λόγο θα πρέπει να προσέχουμε σε ποιο κατάλογο είμαστε προτού ενεργοποιήσουμε την υπηρεσία ftp γιατί αυτός θα είναι ο κατάλογος που θα τοποθετηθούν τα αρχεία που θα λάβουμε, εφόσον βέβαια δεν ορίσουμε εμείς κάποιο άλλο συγκεκριμένο κατάλογο προς τοποθέτηση.

Για να ενεργοποιήσουμε την υπηρεσία FTP εκτελούμε στην θέση του prompt την εντολή “**ftp**” (χωρίς τα εισαγωγικά). Πλέον το prompt του συστήματος αντικαθίσταται από το prompt της υπηρεσίας FTP δηλαδή το “ftp>”, γεγονός που μας βοηθάει να καταλάβουμε σε ποια κατάσταση είμαστε. Η υπηρεσία FTP περιλαμβάνει ένα μεγάλο αριθμό από εντολές τις οποίες μπορούμε να δούμε εκτελώντας την εντολή “**?**”, ενώ μπορούμε να δούμε πληροφορίες για την κάθε εντολή εκτελώντας την εντολή “**? [εντολή]**” όπου εντολή βάζουμε το όνομα της εντολής που επιθυμούμε π.χ. “? get”.

Οι πιο βασικές εντολές από τις οποίες μερικές θα χρειαστούμε και για την διεξαγωγή του εργαστηρίου, είναι οι παρακάτω:

Εντολή	Ενέργεια
open	Σύνδεση στον απομακρυσμένο ftp εξυπηρετητή.
close	Τερματισμός σύνδεσης.
bye	Έξοδος από την υπηρεσία ftp.
pwd	Εμφάνιση του καταλόγου στον οποίο δουλεύουμε στον απομακρυσμένο υπολογιστή.
ls	Εμφάνιση των περιεχομένων του τρέχοντος απομακρυσμένου καταλόγου.
cd	Αλλαγή απομακρυσμένου καταλόγου.
get	Λήψη απομακρυσμένου αρχείου.
mget	Λήψη πολλών απομακρυσμένων αρχείων.
put	Αποστολή τοπικού αρχείου.
mput	Αποστολή πολλών τοπικών αρχείων.

Παρακάτω παρουσιάζονται αναλυτικά οι εξηγήσεις για τις βασικές λειτουργίες της υπηρεσίας FTP:
open: Από την στιγμή που θα ενεργοποιήσουμε την υπηρεσία ftp θα πρέπει να δημιουργήσουμε μία σύνδεση με τον απομακρυσμένο ftp εξυπηρετητή. Αυτό επιτυγχάνεται με την εντολή open. Μόλις εκτελέσουμε την εντολή open, το σύστημα μας ζητά διαδοχικά την διεύθυνση του ftp εξυπηρετητή, μετά ένα όνομα χρήστη (login name) και τέλος το κατάλληλο κωδικό (password).

cd: Μπορούμε να αλλάξουμε κατάλογο εκτελώντας την εντολή “**cd [όνομα καταλόγου]**” όπου [όνομα καταλόγου] το όνομα του επιθυμητού καταλόγου.

get: Με την εντολή get μπορούμε να ‘κατεβάσουμε’ ένα απομακρυσμένο αρχείο στο δικό μας υπολογιστή. Η σύνταξη της εντολής get είναι ίδια με αυτή της cp του συστήματος Unix. Συγκεκριμένα είναι:

get s_path/s_file d_path/d_file

όπου:

- **s_path** είναι το μονοπάτι για τον απομακρυσμένο κατάλογο που βρίσκεται το αρχείο προς λήψη,
- **s_file** είναι το απομακρυσμένο αρχείο που θέλουμε να λάβουμε,
- **d_path** είναι το μονοπάτι για το τοπικό κατάλογο που θέλουμε να τοποθετήσουμε το απομακρυσμένο αρχείο και
- **d_file** είναι το όνομα που θέλουμε να έχει το απομακρυσμένο αρχείο όταν τοποθετηθεί στον τοπικό κατάλογο.

Παράδειγμα αποτελεί η εντολή "**get notes**". Σε αυτήν οι παράμετροι s_path, d_path και d_file δεν έχουν οριστεί πράγμα που σημαίνει ότι το αρχείο notes περιέχεται μέσα στον τρέχοντα απομακρυσμένο κατάλογο, ότι θα τοποθετηθεί στο τρέχοντα τοπικό κατάλογο δηλαδή το κατάλογο που βρισκόμασταν πριν ενεργοποιήσουμε την υπηρεσία ftp και τέλος ότι το όνομα του αρχείου notes θα παραμείνει το ίδιο.

mget: Η εντολή mget κάνει ότι και η get με την διαφορά ότι μπορούμε να 'κατεβάσουμε' με μία εντολή πολλά αρχεία. Παράδειγμα αποτελεί η εντολή "**mget file1 file2 file3**" όπου file1, file2, file3 κτλ είναι τα αρχεία που θέλουμε να κατεβάσουμε από τον τρέχοντα απομακρυσμένο κατάλογο στο τοπικό τρέχοντα κατάλογο. Ένας πιο εύκολος τρόπος για να κατεβάσουμε όλα τα αρχεία του τρέχοντα απομακρυσμένου καταλόγου στο τρέχοντα τοπικό κατάλογο είναι με την εντολή: "**mget ***".

Σημείωση: Ένας εύκολος τρόπος για να ελέγξουμε ότι όντως τα αρχεία κατέβηκαν στον τρέχοντα τοπικό κατάλογο, είναι μετά από την εκτέλεση της εντολής "**get ...**" ή "**mget ...**" να εκτελέσουμε την εντολή "**!ls**".

2.4 Ιστοσελίδες και Χρήση Φυλλομετρητών

Σήμερα το διαδίκτυο (internet) έχει εξαπλωθεί και αναπτυχθεί σε πολύ μεγάλο βαθμό, ώστε να μπορεί να βρει κανείς σε αυτό ένα πολύ μεγάλο όγκο και ποικιλία υλικού, όπως εικόνες, μουσική, αρχεία, κείμενα, video κ.α. Ωστόσο, ο κύριος λόγος εμφάνισής του ήταν η απομακρυσμένη πρόσβαση σε απλά κείμενα τα οποία ονομάστηκαν ιστοσελίδες (ή απλά σελίδες) και τα οποία είχαν την δυνατότητα να περιέχουν συνδέσμους (links) μεταξύ τους έτσι, ώστε από την μία ιστοσελίδα να μπορούμε να περάσουμε σε μία άλλη. Σήμερα οι ιστοσελίδες είναι πάρα πολύ σύνθετες και εμπλουτισμένες και δεν περιέχουν απλό κείμενο αλλά συνδυασμό κειμένου, ήχου, εικόνας κτλ, ενώ το περιεχόμενο τους παράγεται συχνά δυναμικά μετά από αναζήτηση πληροφορίας σε συνδεμένες βάσεις δεδομένων.

Οι ιστοσελίδες βρίσκονται αποθηκευμένες σε κάποιον υπολογιστή, ο οποίος παίζει το ρόλο του εξυπηρετητή (web server) και ο ενδιαφερόμενος για τις ιστοσελίδες αυτές, ο οποίος παίζει το ρόλο του πελάτη (client), θα πρέπει να συνδεθεί στον εξυπηρετητή και να ζητήσει την ιστοσελίδα που επιθυμεί. Στην επικοινωνία μεταξύ εξυπηρετητή και πελάτη έρχονται να δώσουν λύση οι φυλλομετρητές ή πλοηγοί (browsers) οι οποίοι βοηθούν στην σύνδεση με τον εξυπηρετητή, στην κλήση ιστοσελίδων και εμφάνιση αυτών.

Οι πιο δημοφιλείς και διαδεδομένοι φυλλομετρητές σήμερα είναι οι Microsoft Internet Explorer, Mozila FireFox, Google Chrome, Safari, Opera κλπ. Υπάρχουν εκδόσεις τους για όλα τα γνωστά λειτουργικά συστήματα. Οι φυλλομετρητές αυτοί

έχουν τη δυνατότητα άμεσης παρουσίασης κειμένου, εικόνων και γραφικών, ήχων και video. Επίσης μπορεί κάποιος να ενσωματώσει "φίλτρα" για άμεση πρόσβαση σε αρχεία δημιουργημένα από γνωστές εφαρμογές λογισμικού (MS Word κλπ.). Πρόκειται για φυλλομετρητές πλήρως "παραθυρικούς", που εκμεταλλεύονται όλες τις δυνατότητες των παραθυρικών περιβαλλόντων.
Η χρήση ενός φυλλομετρητή είναι πολύ απλή. Στο πλαίσιο Διεύθυνσης ιστοσελίδας γράφουμε το επιθυμητό URL (π.χ. www.ece.upatras.gr) και απλά ακολουθούμε τους συνδέσμους (links). Για να ψάξουμε σε ένα κείμενο πληκτρολογούμε Ctrl-F ή πατάμε το πλήκτρο Find. Για να σημειώσουμε κάποια σύνδεση στο αντίστοιχο ειδικό αρχείο Bookmark πληκτρολογούμε Ctrl-D. Για να δούμε όλες τις σημειωμένες συνδέσεις πληκτρολογούμε Ctrl-B. Για να εκτυπώσουμε πληκτρολογούμε Ctrl-P (ή πατάμε το κουμπί Print), ενώ μπορούμε να επιστρέψουμε στην προηγούμενη σύνδεση με Alt-< και να ξανά επιστρέψουμε στην επόμενη με Alt-> (ή πλήκτρα Back και Forward, αντίστοιχα).

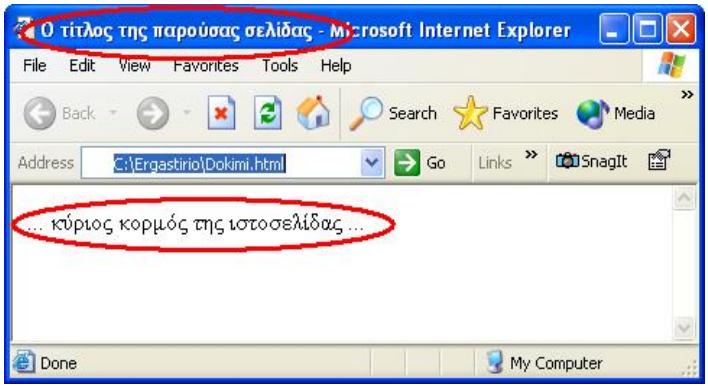
2.5 HTML**

Στο τμήμα αυτό του εργαστήριου θα εξασκηθούμε στην κατασκευή ιστοσελίδων οι οποίες θα περιέχουν τα βασικά χαρακτηριστικά με αυτά των ιστοσελίδων που επισκεπτόμαστε κατά την πλοήγησή μας στο διαδίκτυο. Θα γνωρίσουμε το βασικό σκελετό μιας ιστοσελίδας, πως μπορούμε να μορφοποιούμε ένα κείμενο (έντονα-πλάγια γράμματα, μέγεθος και τύπο γραμματοσειράς), πώς να εμφανίζουμε εικόνες, να τοποθετούμε δεσμούς με άλλες ιστοσελίδες και πώς να χρησιμοποιούμε πίνακες. Μελετήστε τις σελίδες 250-266 του βιβλίου και άλλες πηγές στο διαδίκτυο (HTML tutorial), Ενδεικτικά αναφέρονται σελίδες με χρήσιμες πληροφορίες για την HTML: http://www.w3schools.com/html/
Για την άσκηση αυτή να προετοιμαστείτε φέρνοντας μαζί σας σε φορητή συσκευή αποθήκευσης (πχ USB flash disk) φωτογραφίες που θα θέλατε να ενσωματώσετε στην ιστοσελίδα σας.
Η κατασκευή μίας ιστοσελίδας βασίζεται στην γλώσσα HTML (HyperText Markup Language). Για να μπορέσει κάποιος να δημοσιεύσει ένα κομμάτι πληροφορίας (π.χ. ένα κείμενο) απαιτείται ένας παγκοσμίως γνωστός τρόπος απεικόνισης πληροφορίας όπου όλοι οι υπολογιστές θα μπορούν να καταλάβουν. Την δυνατότητα αυτή προσφέρει η γλώσσα HTML.
Ένας HTML κώδικας μπορεί να γραφεί σε έναν απλό κειμενογράφο (όπως το vi στο Unix και το Notepad στα Windows). Οι εντολές της HTML μοιάζουν με ετικέτες (tags) που περιβάλλουν το κομμάτι εκείνο της ιστοσελίδας που θέλουμε να πάρει μία συγκεκριμένη μορφή ή να συμπεριφέρεται με ένα συγκεκριμένο τρόπο. Παρακάτω παρουσιάζεται ο βασικός σκελετός του κώδικα μίας ιστοσελίδας.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
        <HEAD>
                <TITLE>Ο τίτλος της παρούσας σελίδας</TITLE>
        </HEAD>
        <BODY>
                … κύριος κορμός της ιστοσελίδας …
        </BODY>
</HTML>
```

Σχήμα 2.1 Βασικός σκελετός ιστοσελίδας.

Το αποτέλεσμα του παραπάνω κώδικα σε έναν φυλλομετρητή είναι το εξής:



Σχήμα 2.2 Βασικός σκελετός ιστοσελίδας.

Οι χαρακτήρες < και > καθώς και το κείμενο που περικλείουν (π.χ. <HTML>, </BODY>, <TITLE> κτλ.) αποτελούν μία ετικέτα (tag). Οι ετικέτες της μορφής <*> όπου * μία οποιαδήποτε σειρά χαρακτήρων, αποτελούν ετικέτες έναρξης μίας συγκεκριμένης μορφοποίησης ή συμπεριφοράς ενώ οι ετικέτες της μορφής </*> αποτελούν ετικέτες τέλους μίας μορφοποίησης ή συμπεριφοράς. Υπάρχουν ετικέτες έναρξης μορφοποίησης ή συμπεριφοράς οι οποίες πρέπει απαραίτητα να συνοδεύονται από τις αντίστοιχες ετικέτες τέλους μορφοποίησης ή συμπεριφοράς και άλλες όχι. Μία ετικέτα έναρξης μορφοποίησης ή συμπεριφοράς μπορεί να παίρνει και παραμέτρους και στην περίπτωση αυτή η μορφή της είναι η εξής:

<* παράμετρος1=τιμή1 παράμετρος2=τιμή2 …>

Αντίθετα μία ετικέτα τέλους μορφοποίησης ή συμπεριφοράς δεν παίρνει ποτέ παραμέτρους.

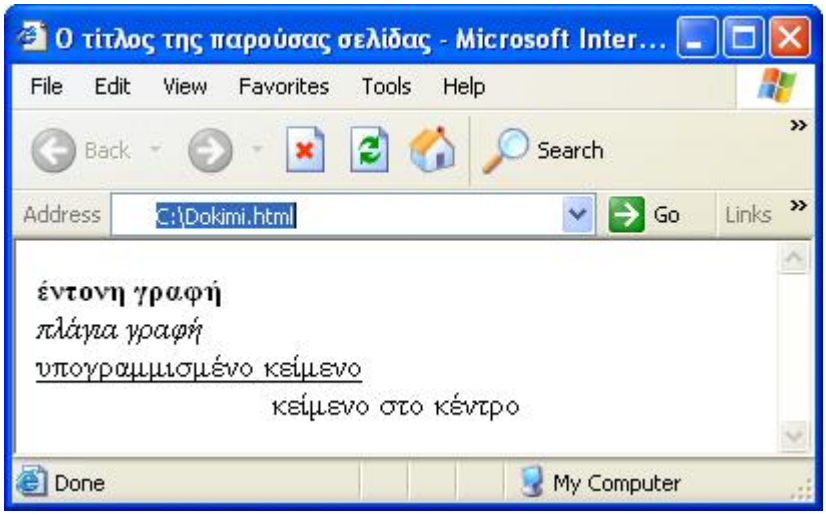
Μερικές από τις σημαντικότερες και πιο χρήσιμες ετικέτες είναι οι εξής:

Ετικέτα	Επεξήγηση
<B>	Το κείμενο που περικλείεται μεταξύ των ετικετών <B> και </B> εμφανίζεται με πιο έντονη γραφή. π.χ. <B>έντονη γραφή</B>
<I>	Το κείμενο που περικλείεται μεταξύ των ετικετών <I> και </I> εμφανίζεται με πλάγια γραφή. π.χ. <I>πλάγια γραφή</I>
<U>	Το κείμενο που περικλείεται μεταξύ των ετικετών <U> και </U> εμφανίζεται με υπογράμμιση. π.χ. <U>υπογραμμισμένο κείμενο</U>
<CENTER>	Το κείμενο που περικλείεται από τις ετικέτες <CENTER> και </CENTER> εμφανίζεται στο κέντρο της γραμμής. Η ετικέτα <CENTER> δεν δέχεται παραμέτρους. π.χ. <CENTER>κείμενο στο κέντρο</CENTER>

Παρακάτω παρουσιάζεται ένα παράδειγμα μίας ιστοσελίδας με κώδικα που περιέχει τις παραπάνω ετικέτες καθώς επίσης παρουσιάζεται και η όψη της ιστοσελίδας αυτής σε ένα φυλλομετρητή. (Το ρόλο της ετικέτας
 θα τον δούμε παρακάτω.)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
    <HEAD>
        <TITLE>Ο τίτλος της παρούσας σελίδας</TITLE>
    </HEAD>
    <BODY>
        <B>έντονη γραφή</B><BR>
        <I>πλάγια γραφή</I><BR>
        <U>υπογραμμισμένο κείμενο</U><BR>
        <CENTER>κείμενο στο κέντρο</CENTER>
    </BODY>
</HTML>
```

Σχήμα 2.3 Παράδειγμα βασικών ετικετών.



Σχήμα 2.4 Παράδειγμα βασικών ετικετών.

Άλλες χρήσιμες ετικέτες είναι οι εξής:

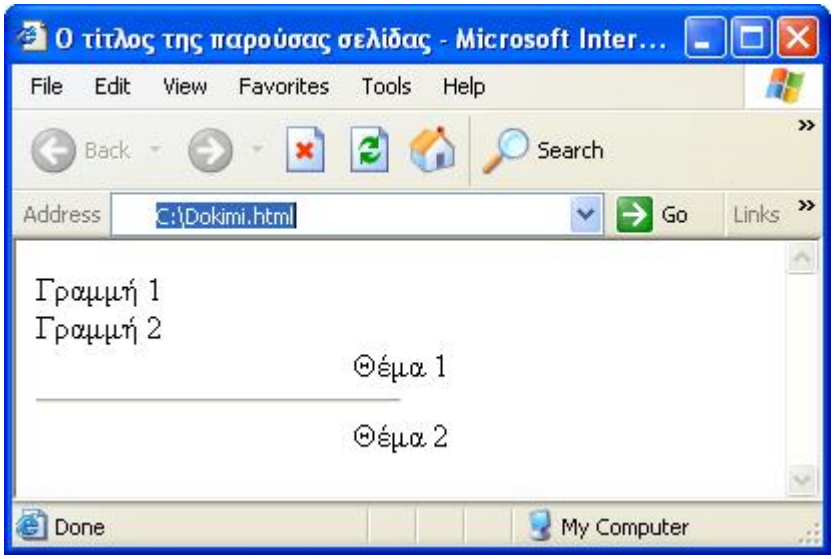
Ετικέτα	Επεξήγηση
 	Η ετικέτα προξενεί αλλαγή γραμμής. Δεν ακολουθεί ετικέτα </BR>. π.χ. Γραμμή 1 Γραμμή 2
<HR>	Η ετικέτα <HR> (HR = Horizontal Rule) δημιουργεί μία οριζόντια γραμμή. Δεν ακολουθεί ετικέτα </HR>. π.χ. Θέμα 1<HR WIDTH="90%" SIZE="2" ALIGN="left">Θέμα 2

Παρακάτω παρουσιάζεται ένα παράδειγμα μίας ιστοσελίδας με κώδικα που περιέχει τις δύο παραπάνω ετικέτες καθώς επίσης παρουσιάζεται και η όψη της ιστοσελίδας αυτής σε ένα φυλλομετρητή.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
    <HEAD>
        <TITLE>Ο τίτλος της παρούσας σελίδας</TITLE>
    </HEAD>
    <BODY>
```

```
        Γραμμή 1<BR>Γραμμή 2<BR>
        <CENTER>
                Θέμα 1<HR WIDTH="50%" SIZE="2" ALIGN="Left">Θέμα 2
        </CENTER>
    </BODY>
</HTML>
```

Σχήμα 2.5 Παράδειγμα βασικών ετικετών.



Σχήμα 2.6 Παράδειγμα βασικών ετικετών.

Στην παραπάνω εικόνα παρατηρούμε ότι οι φράσεις «Θέμα 1» και «Θέμα 2» βρίσκονται στο κέντρο και αυτό λόγο του ότι περιέχονται στις ετικέτες <CENTER> και </CENTER>. Όμως δεν συμβαίνει το ίδιο για την οριζόντια γραμμή που προξενείται από την ετικέτα <HR> γιατί η μορφοποίηση που προκαλεί η ετικέτα <CENTER> παρακάμπτεται από την παράμετρο ALIGN=”Left” που περιέχει η ετικέτα <HR> και επιβάλλει στην οριζόντια γραμμή να τοποθετηθεί αριστερά.

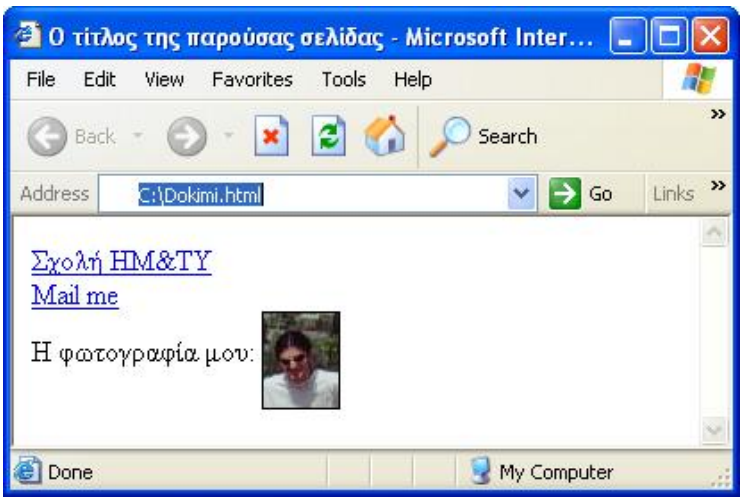
Μερικές πιο πολύπλοκες ετικέτες είναι οι εξής:

Ετικέτα	Επεξήγηση
<A>	Το κείμενο μεταξύ των ετικετών <A> και </A> μετατρέπεται σε ένα δεσμό υπερκειμένου. π.χ. <A HREF=“www.ee.upatras.gr”>Σχολή ΗΜ&ΤΥ</A> π.χ. <A HREF=“mailto:a04-4176@amalia.ee.upatras.gr>Mail me</A>
<IMG>	Η ετικέτα <IMG> προσφέρει την δυνατότητα εισαγωγής εικόνας στην ιστοσελίδα. Δεν ακολουθεί ετικέτα </IMG>. π.χ. Η φωτογραφία μου: <IMG SRC=“photo.jpg” BORDER=“1” ALT=“Νίκος” ALIGN=“Middle”>

Παρακάτω παρουσιάζεται ένα παράδειγμα μίας ιστοσελίδας με κώδικα που περιέχει τις δύο παραπάνω ετικέτες καθώς επίσης παρουσιάζεται και η όψη της ιστοσελίδας αυτής σε ένα φυλλομετρητή.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
    <HEAD>
        <TITLE>Ο τίτλος της παρούσας σελίδας</TITLE>
    </HEAD>
    <BODY>
        <A HREF="www.ee.upatras.gr">Σχολή ΗΜ&ΤΥ</A><BR>
        <A HREF="mailto:a04-4176@amalia.ee.upatras.gr">Mail me</A><BR>
        Η φωτογραφία μου: <IMG SRC="photo.jpg" BORDER="1" ALT="Νίκος"
ALIGN="Middle">
    </BODY>
</HTML>
```

Σχήμα 2.7 Παράδειγμα πιο πολύπλοκων ετικετών.



Σχήμα 2.8 Παράδειγμα πιο πολύπλοκων ετικετών.

Στην παραπάνω εικόνα παρατηρούμε ότι οι δύο δεσμοί υπερκειμένου αυτόματα υπογραμμίζονται και παίρνουν χρώμα μπλε χωρίς να έχουμε κάνει κάποια σχετική ρύθμιση. Αυτό είναι κάτι που συμβαίνει για όλους του δεσμούς υπερκειμένου και αποτελεί το χαρακτηριστικό τους γνώρισμα. Γι' αυτό πολύ σπάνια χρησιμοποιούμε την ετικέτα <U> η οποία υπογραμμίζει ένα κείμενο, αφού έτσι δίνεται η εντύπωση δεσμού υπερκειμένου και παρασύρει τον χρήστη στο να κάνει click στο δεσμό χωρίς όμως αποτέλεσμα.

Ασκήσεις

Να εκτελέστε τις παρακάτω ασκήσεις και να παραδώσετε Έκθεση Εργαστηρίου, σύμφωνα με τις οδηγίες, απαντώντας σε όλα τα ερωτήματα.

Μέρος 2.1 Ηλεκτρονικό ταχυδρομείο (webmail):

1. Να στείλετε με χρήση της υπηρεσίας mail.upnet.gr ένα μήνυμα στον εαυτό σας και σε κάποιους συμφοιτητές σας με θέμα τις μέχρι τώρα αντυπώσεις σας από το Πανεπιστήμιο και το μάθημα.
   Μόλις λάβετε το παραπάνω μήνυμα (το οποίο μόλις στείλατε στον εαυτό σας) προωθήστε το (forward) στον υπεύθυνο του εργαστηρίου. (Το email του θα

σας δοθεί κατά την διάρκεια του εργαστηρίου.). Αυτό το τμήμα της άσκησης να ολοκληρωθεί κατά τη διάρκεια του εργαστηρίου.

2. Χρησιμοποιείστε την υπηρεσία google groups ή το facebook για τον ίδιο σκοπό (το τμήμα αυτό της άσκησης να γίνει από το σπίτι και όχι κατά τη διάρκεια του εργαστηρίου).

Μέρος 2.2 Αναζήτηση στο διαδίκτυο και στρατηγικές αναζήτησης:

1. Αναζητήστε πανεπιστημιακούς κόμβους που περιέχουν πληροφορίες για τα προγράμματα σπουδών στα τμήματα Ηλεκτρολόγων Μηχανικών & Υπολογιστών ή Μηχανικών Υπολογιστών (2 από την Ελλάδα, 2 από την Ευρώπη και 2 από τις Η.Π.Α.).
   a) Ποιες διευθύνσεις βρήκατε;
   b) Με ποιον τρόπο φθάσατε γρηγορότερα στο επιθυμητό αποτέλεσμα.
   c) Με ποιον τρόπο βρήκατε τις κατάλληλες πληροφορίες.

   Τα κύρια αποτελέσματα της αναζήτησής σας να τα φυλάξετε στο αρχείο **universities**
2. Κάποιος φίλος σας ζητά πληροφορίες για σπουδές στο Ελληνικό Ανοικτό Πανεπιστήμιο. Σε ποιον κόμβο πιστεύετε ότι μπορεί να ανατρέξετε για να βρείτε πληροφορίες; Πως τον βρήκατε; Αποθηκεύστε τις βασικές πληροφορίες που βρήκατε στο αρχείο **openuniv**
3. Έστω ότι θέλετε να πάτε ταξίδι 3 ημερών στο Παρίσι. Οργανώστε το ταξίδι σας με κάθε δυνατή λεπτομέρεια στηριζόμενοι σε πληροφορίες που θα βρείτε στο διαδίκτυο (μετακινήσεις, ξενοδοχεία, αξιοθέατα). Συντάξετε σχετικό κείμενο με όνομα **trip**.

Ενσωματώστε όλες τις πληροφορίες στα αρχεία (universities, dvd, openuniv, course, trip) που έχετε αποθηκεύσει μαζί τα ζητούμενα σχόλια στην Έκθεση σας. Αυτό το τμήμα της έκθεσης δεν πρέπει να έχει μέγεθος μεγαλύτερο από 2 σελίδες Α4.

Μέρος 2.3 Δημιουργία προσωπικής ιστοσελίδας **

Να παραδώσετε το .html αρχείο που δημιουργήσατε μαζί (συμπιεσμένα σε ένα αρχείο) με όλα τα απαραίτητα αρχεία (π.χ. εικόνες) ώστε η σελίδα σας να εμφανίζετε όπως ακριβώς την δημιουργήσατε. Για περισσότερη βοήθεια σχετικά με τις HTML ετικέτες μπορείτε να ανατρέξετε στο σύγγραμμα του μαθήματος ή να ψάξετε στον παγκόσμιο ιστό.

Αντικείμενο της άσκησης αυτής είναι να δημιουργήσετε μία προσωπική ιστοσελίδα με όνομα **AM.html,** όπου ΑΜ ο αριθμός μητρώου σας, και η οποία θα πρέπει να περιέχει τουλάχιστον τις παρακάτω πληροφορίες: α) το ονοματεπώνυμό σας β) τον αριθμό μητρώου σας γ) πληροφορίες για τον εαυτό σας (ενδιαφέροντα, γνώσεις, χόμπι κτλ.) δ) συνδέσμους σε άλλες σελίδες στο διαδίκτυο που θεωρείται ως σημαντικές. Για την κατασκευή της προσωπικής ιστοσελίδας **χρησιμοποιήστε τουλάχιστον μία φορά όλες τις ετικέτες που παρουσιάζονται παραπάνω στην θεωρία**. Ωστόσο μην αρκεστείτε μόνο σε αυτό και προσπαθήστε να μας εντυπωσιάσετε! (ΠΡΟΣΟΧΗ: Παράδοση του Τμήματος αυτού της άσκησης κατά την ώρα του εργαστηρίου)

Εργαστήριο 3: Σχεδίαση διαγραμμάτων ροής αλγορίθμων: Βασικές Έννοιες

3.1 Γενικά

Σκοπός του εργαστηρίου είναι η εξάσκηση με τις βασικές έννοιες των αλγορίθμων και δομών και η αναπαράσταση αλγορίθμων μέσω διαγραμμάτων ροής.
Στα πλαίσια αυτής θα δημιουργηθεί ένα διάγραμμα ροής που υλοποιεί ένα δοθέντα αλγόριθμο. Σύμφωνα με τις οδηγίες που θα δοθούν την ώρα του εργαστηρίου θα δημιουργήσετε ατομικά ή σε συνεργασία με κάποιο άλλο συνάδελφό σας διαγράμματα ροής με τη χρήση του εργαλείου συνεργατικής σχεδίασης Synergo.
Θα βαθμολογηθείτε για τη λύση και για τη συνεργασία που θα έχετε με το συνάδελφό σας. Η συνολική δραστηριότητα σας στην οποία φαίνεται η συμμετοχή σας καταγράφεται από το ίδιο το εργαλείο Synergo.

Η θεωρία που αφορά τους Αλγόριθμους περιέχεται στο κεφαλαίο 2 του βιβλίου «Εισαγωγή στους υπολογιστές» (Ν. Αβούρης, Ο. Κουφοπαύλου, Δ. Σερπάνος)

*Σημ :Το Περιβάλλον Συνεργασίας Synergo (*http:/hci.ece.upatras.gr/synergo*), έχει αναπτυχθεί στο Πανεπιστήμιο Πατρών από την ερευνητική ομάδα Αλληλεπίδρασης Ανθρώπου Υπολογιστή (HCI group). Το SYNERGO χρησιμοποιείται στις εργαστηριακές ασκήσεις που αφορούν τους αλγόριθμους, στο παρόν εργαστήριο, καθώς και σε εργαστηριακές ασκήσεις, σε επόμενα έτη των σπουδών σας.*

3.2 Ασκήσεις

Αλγόριθμος : Εύρεση Μεγίστου, Ελαχίστου και Μέσου όρου .

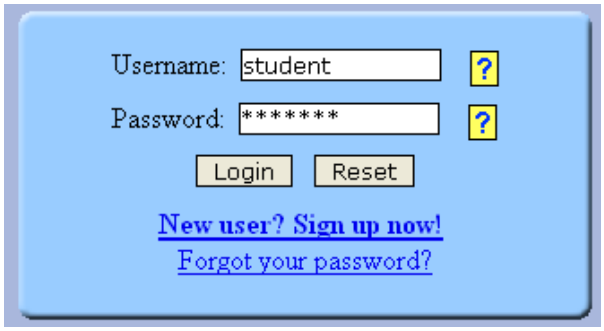
Α) Δίνονται διαδοχικά Ν ακέραιοι θετικοί αριθμοί. Το Ν είναι δεδομένο. **Ζητείται να γραφεί το διάγραμμα ροής του αλγορίθμου που υπολογίζει το μέγιστο max, τον ελάχιστο min και το μέσο όρο ΜΟ και να τους εμφανίζει στην οθόνη.**

Υποδείξεις:
1)Θα πρέπει να διερευνήσετε την ορθότητα του αλγορίθμου που απεικονίζει το διάγραμμα ροής. Στην περίπτωση που εργάζεστε συνεργατικά προτείνουμε την ακόλουθη διαδικασία : ο ένας από τους δυο συνεργάτες να δείχνει ή να δηλώνει στο chat ποια εντολή του διαγράμματος ροής που δημιουργήσατε εκτελείται (να παίζει τον ρόλο του επεξεργαστή δηλαδή) και ο συνεργάτης του να δηλώνει στο chat αν και πως αλλάζουν οι τιμές των μεταβλητών και το περιεχόμενο της οθόνης.
2) ΔΕΝ απαιτείται η χρήση πίνακα. Ο αλγόριθμος πρέπει να τερματίζει μετά την εισαγωγή του νιοστού αριθμού.
3) Θα πρέπει να τεκμηριώσετε τον αλγόριθμο σας χρησιμοποιώντας sticky notes με σχετικό περιεχόμενο (δείτε οδηγίες παρακάτω)

Εκκίνηση:

Πηγαίνετε στην διεύθυνση http:/hci.ece.upatras.gr/synergo και περιμένετε μέχρι να φορτώσει η σελίδα. Χρησιμοποιείστε σαν username και password τα εξής

Username: student
Password: student



Στην συνέχεια επιλέξτε από το πλαίσιο Options (πάνω αριστερά) τον σύνδεσμο **Start Synergo** και ακολουθήστε τις οδηγίες που αναγράφονται στην οθόνη. Μόλις ολοκληρώσετε τα παραπάνω περιμένετε λίγο χρόνο μέχρι να ξεκινήσει η εφαρμογή. Σε αυτό το σημείο θα πρέπει να εμφανιστεί το παρακάτω παράθυρο:

Όταν δείτε το παραπάνω παράθυρο, εισάγετε το ΑΜ σας σαν username και password το ίδιο και θα εκκινήσει η εφαρμογή. Δηλαδή για παράδειγμα:

Username: a07-5634
Password: a07-5634

Αποθήκευση εργασίας

Αν δουλεύετε ατομικά χωρίς συνεργασία, αποθηκεύστε το τρέχον μοντέλο, ως εξής: ΑΜ-όνομα άσκησης (π.χ. a07-5242-maxminmo). Καθώς εργάζεστε, να αποθηκεύετε τακτικά το αρχείο σας.

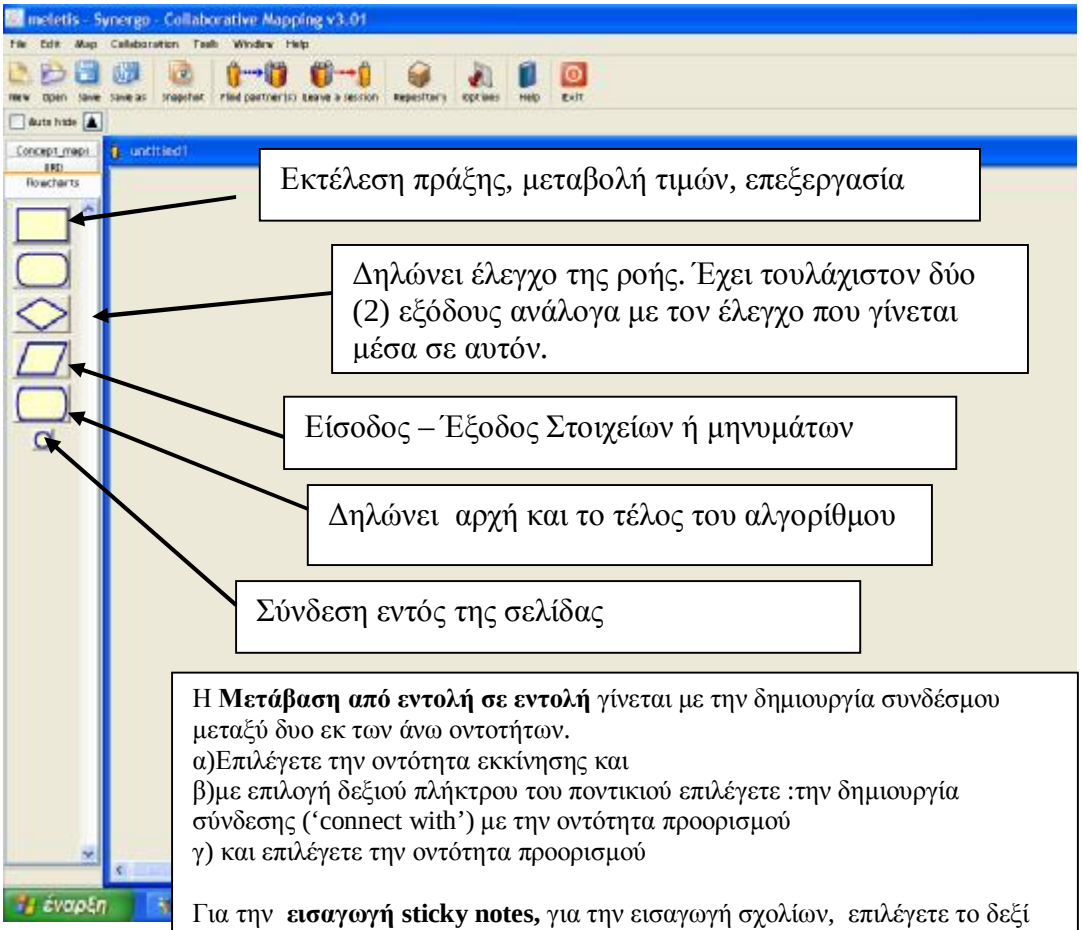
Αν δουλεύετε συνεργατικά, οι επιβλέποντες του εργαστηρίου θα ενεργοποιήσουν μια συνεργασία με έναν συνάδελφό σας και ακολούθως αποθηκεύστε το τρέχον μοντέλο, ως εξής: ΑΜ1-ΑΜ2-όνομα άσκησης (π.χ. a07-5242-a07-5642- maxminmo). Καθώς εργάζεστε, να αποθηκεύετε τακτικά το αρχείο σας.

Ολοκλήρωση άσκησης

Παραδίδετε το αρχείο που έχετε δημιουργήσει (με επέκταση *.synergo) που βρίσκεται στο φάκελο “My Documents”, στον υποφάκελο “My models”. Όλα τα αρχεία που υπάρχουν στον φάκελο “My models” θα πρέπει να υποβληθούν μέσω του **eclass** ή να αποσταλούν με email στη διεύθυνση του βοηθού του εργαστηρίου που θα σας δοθεί την ώρα του εργαστηρίου.
Επίσης θα πρέπει να παραδώσετε μία αναφορά που θα περιγράφει τον τρόπο με τον οποίο υλοποιήσατε τα διαγράμματα ροής (είτε συνεργατικά είτε ατομικά), τις εντυπώσεις σας ή τα πιθανά προβλήματα που αντιμετωπίσατε κατά την συνεργατική αλληλεπίδραση και ότι άλλο θεωρείται ότι χρειάζεται να σχολιάσετε.

Αν δεν υπάρχει αρχείο με επέκταση *.synergo , τότε συμπιέστε τα αρχεία με επέκταση *.model και *,history σε ένα ZIP πριν τα αποστείλετε.

Οι σχηματικές αναπαραστάσεις των Διαγραμμάτων Ροής στο Synergo



Η **Μετάβαση από εντολή σε εντολή** γίνεται με την δημιουργία συνδέσμου μεταξύ δυο εκ των άνω οντοτήτων.
α)Επιλέγετε την οντότητα εκκίνησης και
β)με επιλογή δεξιού πλήκτρου του ποντικιού επιλέγετε :την δημιουργία σύνδεσης (‘connect with’) με την οντότητα προορισμού
γ) και επιλέγετε την οντότητα προορισμού

Για την **εισαγωγή sticky notes,** για την εισαγωγή σχολίων, επιλέγετε το δεξί πλήκτρο του ποντικιού πάνω στην επιφάνεια σχεδίασης και ακολούθως επιλέγετε: ‘insert sticky note’

The Process Symbol represents any process, function, or action and is the most frequently used symbol in flowcharting.

The Input/Output Symbol represents data that is available for input or resulting from processing (i.e. customer database records).

The Decision Symbol is a junction where a decision must be made. A single entry may have any number of alternative solutions, but only one can be chosen.

The Connector Symbol represents the exit to, or entry from, another part of the same flow chart. It is usually used to break a flow line that will be continued elsewhere. It's a good idea to reference page numbers for easy location of connectors.

Εργαστήριο 4: 2η Άσκηση σχεδίασης διαγραμμάτων ροής αλγορίθμων :

4.1 Γενικά

Σκοπός της άσκησης είναι η εξάσκηση με πιο προχωρημένες (συγκριτικά με το εργαστήριο 3) αλγοριθμικές έννοιες και δομές και η αναπαράσταση αλγορίθμων μέσω διαγραμμάτων ροής.
Στα πλαίσια αυτής της άσκησης θα δημιουργηθεί ένα διάγραμμα ροής που υλοποιεί ένα δοθέντα αλγόριθμο. Σύμφωνα με τις οδηγίες που θα δοθούν την ώρα του εργαστηρίου θα δημιουργήσετε ατομικά ή σε συνεργασία με κάποιο άλλο συνάδελφό σας **διαγράμματα ροής με τη χρήση του εργαλείου συνεργατικής σχεδίασης Synergo**. Θα βαθμολογηθείτε **για τη λύση και για τη συνεργασία** που θα έχετε με το συνάδελφό σας. Η συνολική δραστηριότητα σας στην οποία φαίνεται η συμμετοχή σας καταγράφεται από το ίδιο το εργαλείο Synergo.

Η θεωρία που αφορά τους Αλγόριθμους περιέχεται στο κεφαλαίο 2 του βιβλίου «Εισαγωγή στους υπολογιστές» (Ν. Αβούρης, Ο. Κουφοπαύλου, Δ. Σερπάνος)

Σημ :Το Περιβάλλον Συνεργασίας Synergo (http:/hci.ece.upatras.gr/synergo), έχει αναπτυχθεί στο Πανεπιστήμιο Πατρών από την ερευνητική ομάδα Αλληλεπίδρασης Ανθρώπου Υπολογιστή (HCI group). Το SYNERGO χρησιμοποιείται στις εργαστηριακές ασκήσεις που αφορούν τους αλγόριθμους, στο παρόν εργαστήριο, καθώς και σε εργαστηριακές ασκήσεις, σε επόμενα έτη των σπουδών σας.

4.2 Ασκήσεις

Αλγόριθμος : Δυαδική Αναζήτηση

Δίνεται ένας πίνακας **n** θέσεων ταξινομημένος κατά αύξουσα σειρά που περιέχει ακέραιους αριθμούς. Να βρεθεί αν σε αυτούς υπάρχει ένας δεδομένος αριθμός **K** και η θέση **N** στην οποία βρίσκεται και να εμφανιστεί στην οθόνη .Για την εύρεση του στοιχείου να χρησιμοποιηθεί ο αλγόριθμος Δυαδικής Αναζήτησης ο οποίος περιγράφεται παρακάτω (Binary Search). Ο αλγόριθμος θα πρέπει να αναπαρασταθεί με διάγραμμα ροής. Αν επιθυμείτε μπορείτε να γράψετε και τον αντίστοιχο ψευδοκώδικα.

Σημαντικές Υποδείξεις

1) Ο αλγόριθμος Δυαδικής αναζήτησης λειτουργεί ως εξής σε έναν ταξινομημένο κατά αύξουσα σειρά πίνακα n θέσεων ως εξής:

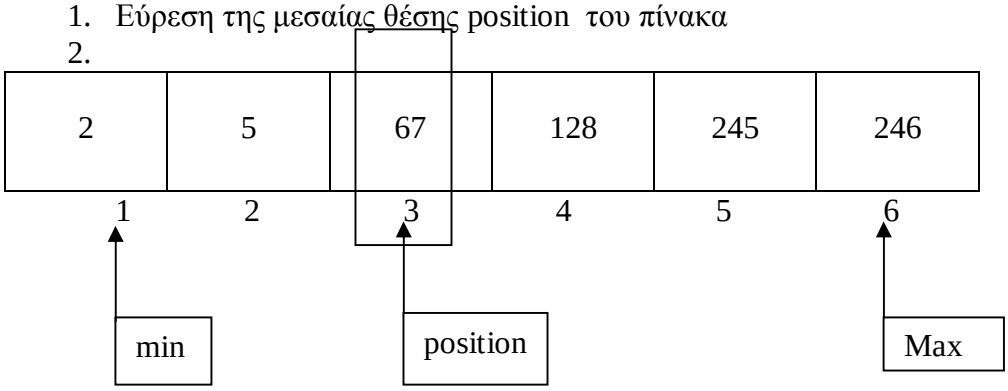
- Βρίσκουμε το μεσαίο στοιχείο του πίνακα.
- Αν το περιεχόμενο της θέσης είναι ίσο με αυτό που ζητάμε , τότε βρέθηκε ο αριθμός και η θέση του
- Αν ο αριθμός που αναζητούμε είναι μικρότερος προχωράμε την αναζήτηση στο πρώτο μισό

- Αν ο αριθμός που αναζητούμε είναι μεγαλύτερος προχωράμε την αναζήτηση στο δεύτερο μισό

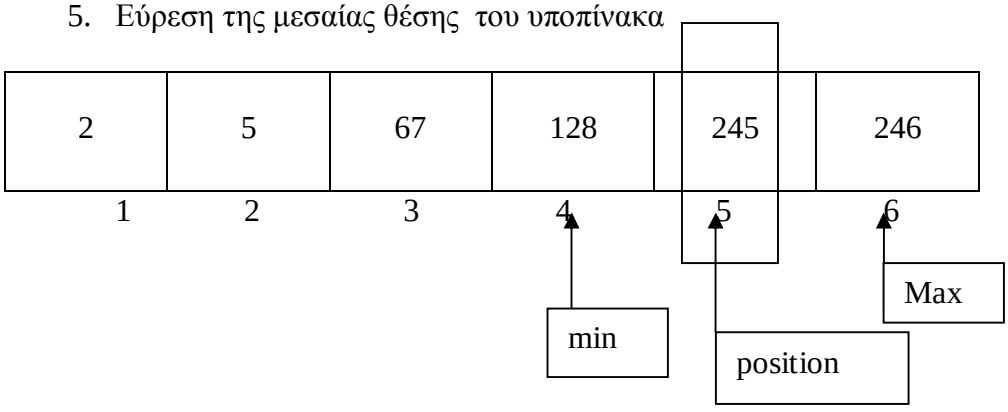
Ο αλγόριθμος τερματίζει με την εύρεση του στοιχείου και εκτύπωση της θέσης του στοιχείου, ή εκτύπωση μηνύματος για τη μη εύρεση του.
2) Το Ακέραιο μέρος διαίρεσης του x/y υπολογίζεται με x**DIV**y
3) Θα πρέπει να διερευνήσετε την ορθότητα του αλγορίθμου που απεικονίζει το διάγραμμα ροής. Στην περίπτωση που εργάζεστε συνεργατικά προτείνουμε την ακόλουθη διαδικασία : ο ένας από τους δυο συνεργάτες να δείχνει ή να δηλώνει στο chat ποια εντολή του διαγράμματος ροής που δημιουργήσατε εκτελείται (να παίζει τον ρόλο του επεξεργαστή δηλαδή) και ο συνεργάτης του να δηλώνει στο chat αν και πως αλλάζουν οι τιμές των μεταβλητών και το περιεχόμενο της οθόνης.
4) Θα πρέπει να τεκμηριώσετε τον αλγόριθμο σας χρησιμοποιώντας sticky notes με σχετικό περιεχόμενο (δείτε οδηγίες προηγούμενης άσκησης)
5) Προτείνεται να χρησιμοποιήσετε τα ονόματα των μεταβλητών που δίνονται στο παράδειγμα.

Παράδειγμα του αλγορίθμου δυαδικής αναζήτησης

Σε ένα πίνακα ταξινομημένο n=6 θέσεων γίνεται δυαδική αναζήτηση για την εύρεση του K=246

1. Εύρεση της μεσαίας θέσης position του πίνακα
2. 

3. Είναι το περιεχόμενο της θέσης position μικρότερο του 246? (67< 246)
4. Αναζήτηση στο δεύτερο μισό του πίνακα
5. Εύρεση της μεσαίας θέσης του υποπίνακα

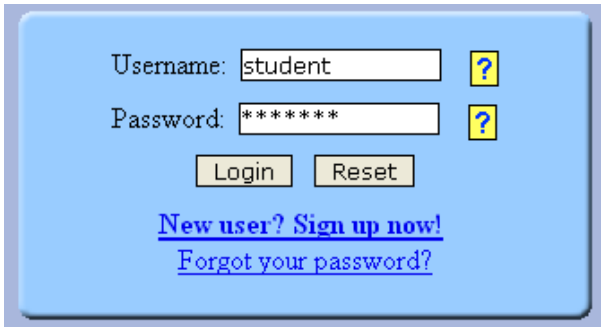


6. Είναι το περιεχόμενο της θέσης position μικρότερο του 246? (245< 246)
7. Αναζήτηση στο δεύτερο μισό του πίνακα
8. Και ο Αλγόριθμος συνεχίζεται μέχρι την εύρεση του αριθμού ή μέχρι... (**βρείτε τη συνθήκη τερματισμού.)**

Εκκίνηση:

Πηγαίνετε στην διεύθυνση http:/hci.ece.upatras.gr/synergo και περιμένετε μέχρι να φορτώσει η σελίδα. Χρησιμοποιείστε σαν username και password τα εξής

Username: student
Password: student



Στην συνέχεια επιλέξτε από το πλαίσιο Options (πάνω αριστερά) τον σύνδεσμο **Start Synergo** και ακολουθήστε τις οδηγίες που αναγράφονται στην οθόνη. Μόλις ολοκληρώσετε τα παραπάνω περιμένετε λίγο χρόνο μέχρι να ξεκινήσει η εφαρμογή. Σε αυτό το σημείο θα πρέπει να εμφανιστεί το παρακάτω παράθυρο:

Όταν δείτε το παραπάνω παράθυρο, εισάγετε το ΑΜ σας σαν username και password το ίδιο και θα εκκινήσει η εφαρμογή. Δηλαδή για παράδειγμα:

Username: a07-5634
Password: a07-5634

Αποθήκευση εργασίας

Αν δουλεύετε ατομικά χωρίς συνεργασία, αποθηκεύστε το τρέχον μοντέλο, ως εξής: ΑΜ-όνομα άσκησης (π.χ. a07-5242-binary). Καθώς εργάζεστε, να αποθηκεύετε τακτικά το αρχείο σας.

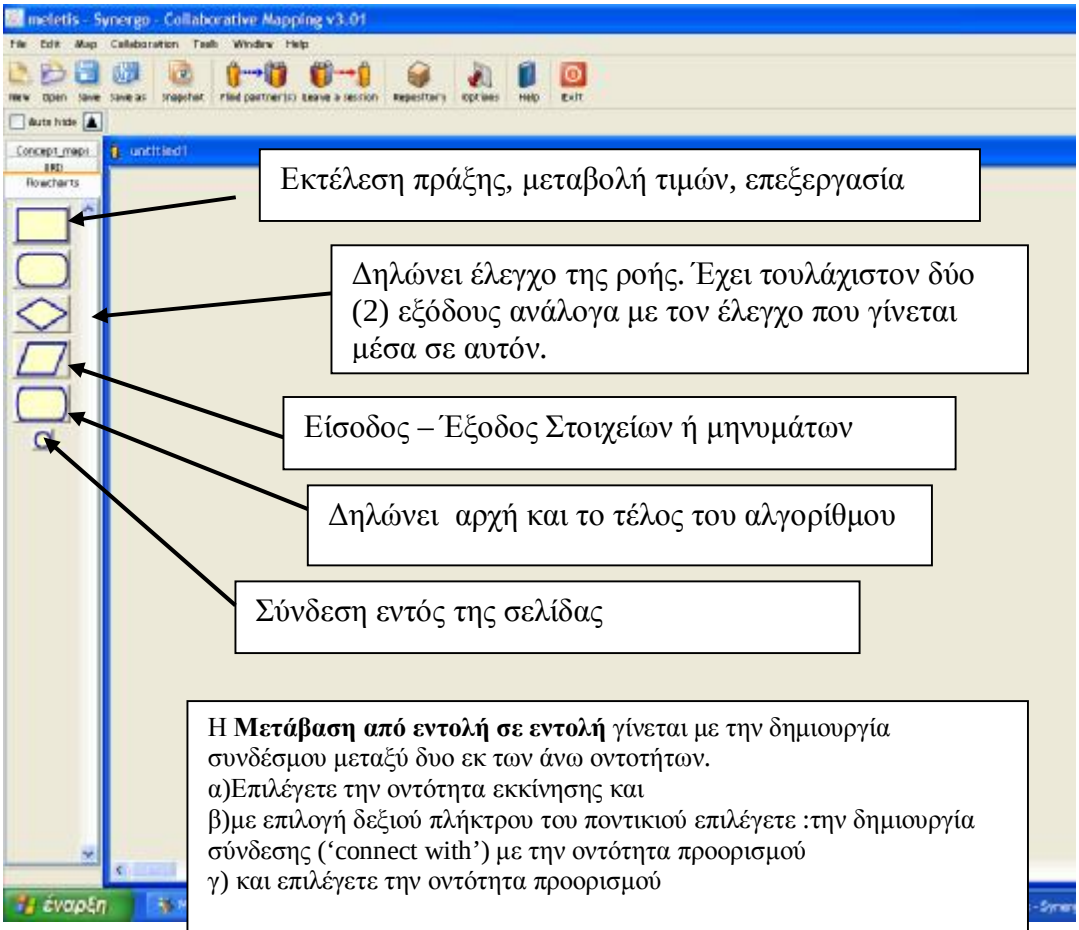
Αν δουλεύετε συνεργατικά, οι επιβλέποντες του εργαστηρίου θα ενεργοποιήσουν μια συνεργασία με έναν συνάδελφό σας και ακολούθως αποθηκεύστε το τρέχον μοντέλο, ως εξής: ΑΜ1-ΑΜ2-όνομα άσκησης (π.χ. a07-5242-a07-5642-binary). Καθώς εργάζεστε, να αποθηκεύετε τακτικά το αρχείο σας.

Ολοκλήρωση άσκησης

Παραδίδετε το αρχείο που έχετε δημιουργήσει (με επέκταση *.synergo) που βρίσκεται στο φάκελο “My Documents”, στον υποφάκελο “My models”. Όλα τα αρχεία που υπάρχουν στον φάκελο “My models” θα πρέπει να υποβληθούν μέσω του **eclass** ή να αποσταλούν με email στη διεύθυνση του βοηθού του εργαστηρίου που θα σας δοθεί την ώρα του εργαστηρίου.

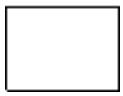
Επίσης θα πρέπει να παραδώσετε μία αναφορά που θα περιγράφει τον τρόπο με τον οποίο υλοποιήσατε τα διαγράμματα ροής (είτε συνεργατικά είτε ατομικά), τις εντυπώσεις σας ή τα πιθανά προβλήματα που αντιμετωπίσατε κατά την συνεργατική αλληλεπίδραση και ότι άλλο θεωρείται ότι χρειάζεται να σχολιάσετε.

Αν δεν υπάρχει αρχείο με επέκταση *.synergo , τότε συμπιέστε τα αρχεία με επέκταση *.model και *,history σε ένα ZIP πριν τα αποστείλετε.



Η **Μετάβαση από εντολή σε εντολή** γίνεται με την δημιουργία συνδέσμου μεταξύ δυο εκ των άνω οντοτήτων.
α)Επιλέγετε την οντότητα εκκίνησης και
β)με επιλογή δεξιού πλήκτρου του ποντικιού επιλέγετε :την δημιουργία σύνδεσης (‘connect with’) με την οντότητα προορισμού
γ) και επιλέγετε την οντότητα προορισμού

Για την **εισαγωγή sticky notes,** για την εισαγωγή σχολίων, επιλέγετε το δεξί πλήκτρο του ποντικιού πάνω στην επιφάνεια σχεδίασης και ακολούθως επιλέγετε: ‘insert sticky note’



The Process Symbol represents any process, function, or action and is the most frequently used symbol in flowcharting.

The Input/Output Symbol represents data that is available for input or resulting from processing (i.e. customer database records).

The Decision Symbol is a junction where a decision must be made. A single entry may have any number of alternative solutions, but only one can be chosen.

The Connector Symbol represents the exit to, or entry from, another part of the same flow chart. It is usually used to break a flow line that will be continued elsewhere. It's a good idea to reference page numbers for easy location of connectors.

Εργαστήριο 5: Πρώτη Άσκηση Προγραμματισμού Python

5.1 Γενικά

Στην άσκηση αυτή γίνεται εισαγωγή στον προγραμματισμό με χρήση της γλώσσας Python. Η Python είναι ίσως η απλούστερη γλώσσα με την οποία θα μπορούσε να ξεκινήσει κάποιος να μαθαίνει προγραμματισμό αλλά δεν υστερεί καθόλου σε δυνατότητες. Είναι σχετικά πρόσφατη γλώσσα. Ξεκίνησε να αναπτύσσεται στις αρχές τις δεκαετίας του 1990 από τον μαθηματικό Guido van Rossum. Η γλώσσα έχει γίνει πλέον ιδιαίτερα δημοφιλής στον ακαδημαϊκό και στον επιστημονικό χώρο για τους εξής λόγους:

- Μπορεί κανείς να την χρησιμοποιήσει διαδραστικά (interactively) με τον ίδιο τρόπο που χρησιμοποιεί μια αριθμομηχανή. Κι αυτός είναι ο πιο εύκολος τρόπος να αρχίσει κανείς να μαθαίνει προγραμματισμό.

- Αν και είναι γλώσσα υψηλού επιπέδου, διαθέτει πολύ απλή σύνταξη. Έτσι επιτρέπει στον προγραμματιστή να ασχοληθεί με την επίλυση του προβλήματος κι όχι με τις ιδιαιτερότητές της. Αν και φτάνει σε πολύ υψηλό επίπεδο δυνατοτήτων (πχ. ενσωματώνει συναρτησιακό και αντικειμενοστρεφή προγραμματισμό), η απλή της σύνταξη κρατά κρυμμένες αυτές τις δυνατότητες από τους αρχάριους προγραμματιστές και δεν τους αναγκάζει να προγραμματίσουν με κάποιο συγκεκριμένο μοντέλο.

- Μπορεί να συνδυαστεί με άλλες δημοφιλείς γλώσσες όπως C, C++ και Java.

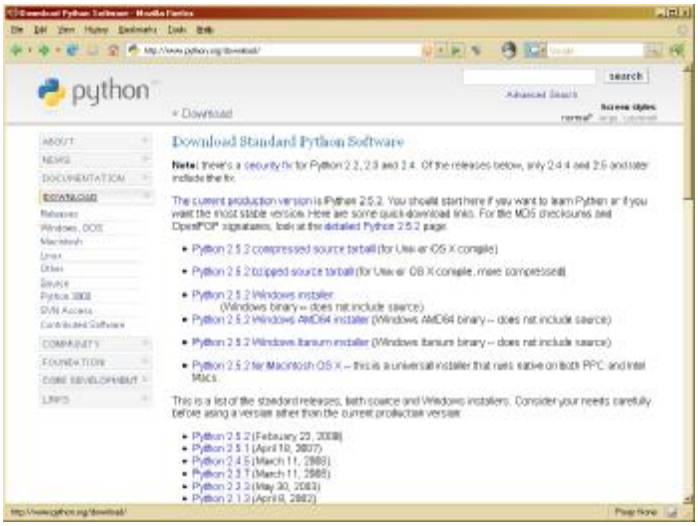
- Αποτελεί ελεύθερο λογισμικό και διατίθεται δωρεάν από το επίσημο site http://www.python.org Υπάρχουν εκδόσεις για κάθε λειτουργικό σύστημα (ακόμα και για κινητά τηλέφωνα). Στο διαδίκτυο μπορεί κανείς να βρεί επίσης πληθώρα βιβλιοθηκών και υποστηρικτικού λογισμικού για την Python που έχουν κατασκευαστεί από διάφορους προγραμματιστές και διατίθενται επίσης ελεύθερα.

5.2 Εγκατάσταση και Εκτέλεση

Κατά πάσα πιθανότητα θα βρείτε την Python ήδη εγκατεστημένη σε κάθε υπολογιστή του υπολογιστικού κέντρου. Όμως για κάθε ενδεχόμενο (πχ. σε περίπτωση που θέλετε να την εγκαταστήσετε στον προσωπικό σας υπολογιστή) η διαδικασία έχει ως εξής:

- Συνδεθείτε στην ιστοσελίδα http://www.python.org

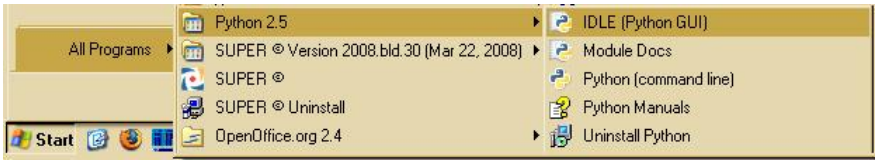
- Επιλέξτε “DOWNLOAD” (από το μενού στα αριστερά της σελίδας) και κατεβάστε το αρχείο που αντιστοιχεί στο λειτουργικό σας σύστημα. Προσέξτε ότι μπορεί να υπάρχει νεώτερη έκδοση[3] και το αρχείο να έχει λίγο διαφορετικό όνομα από αυτό που φαίνεται στην εικόνα.



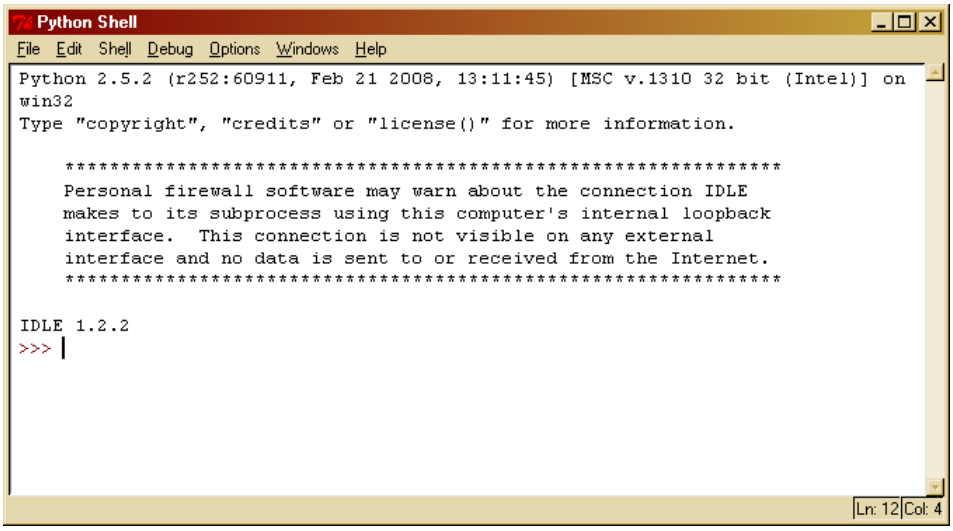
- Τρέξτε το εκτελέσιμο αρχείο που κατεβάσατε και ακολουθήστε τις οδηγίες. Το εκτελέσιμο εγκαθιστά και οδηγίες βοήθειας (on-line help), αλλά μπορείτε να κατεβάσετε από το ίδιο site αρκετούς οδηγούς μελέτης και εκμάθησης σε εκτυπώσιμη μορφή.

Αμέσως μετά είστε σε θέση να τρέξετε το περιβάλλον της γλώσσας. Από το μενού του λειτουργικού συστήματος επιλέξτε την Python. Υπάρχουν δυο περιβάλοντα. Η **κονσόλα** (Python Command Line) και το **Integrated Development Environment** (IDLE Python GUI). Επιλέξτε το IDLE.

[3] Περί εκδόσεων: Στο site της Python θα δείτε ότι συνυπάρχουν δυο παράλληλες γραμμές εκδόσεων, η 2.x και η 3.x. Προτιμήστε να κατεβάσετε και να εγκαταστήσετε την έκδοση 2 κι όχι την 3 (οι ακριβείς λόγοι θα εξηγηθούν στο μάθημα). Η πιο πρόσφατη (2010) έκδοση είναι η 2.7 κι από ότι έχει ανακοινωθεί η έκδοση 2 θα σταματήσει εκεί. Όμως όταν πρωτογράφτηκαν (2008) αυτές οι ασκήσεις ήταν η πιο πρόσφατη έκδοση ήταν η 2.5 και γιαυτό θα δείτε τα περισσότερα σχήματα να αναφέρουν αυτήν.



Θα δείτε μια εικόνα όπως η ακόλουθη:



Είστε έτοιμοι να προγραμματίσετε σε Python.

5.3 Χρήση της Python ως Αριθμομηχανής

Μπορούμε να φανταστούμε τη γλώσσα ως μια αριθμομηχανή όπου οι τέσσερεις βασικές πράξεις λειτουργούν με τα σύμβολα +, -, *, /. Κάθε φορά που θέλουμε να τυπώσουμε το αποτέλεσμα μιας πράξης γράφουμε την εντολή print. Για παράδειγμα:

```
>>> print 5+3
8
>>> print (7-2)*4
20
```

Η γλώσσα επιτρέπει τη χρήση παρενθέσεων. Επίσης μπορούμε να τυπώνουμε πολλά αποτελέσματα με την ίδια εντολή, αρκεί να να χωρίζουμε με κόμματα:

```
>>> print 7+2, 7-2, 7*2, 7/2
9 5 14 3
```

Μια μικρή προσοχή χρειάζεται στην πράξη της διαίρεσης. Η Python εκτελεί ακέραια διαίρεση, εκτός αν δηλώσουμε ότι θέλουμε να κάνει διαίρεση πραγματικών αριθμών. Ο πιο απλός τρόπος να το δηλώσουμε αυτό είναι να γράψουμε τον διαιρετέο ή τον διαιρέτη (ή και τους δυο) ως πραγματικούς. Πχ.:

```
>>> print 7/2, 7/2.0, 7.0/2, 7.0/2.0, 7./2, 7/2., 7./2.
3 3.5 3.5 3.5 3.5 3.5 3.5
```

Και για να συμπληρώσουμε τις βασικές πράξεις, η ύψωση σε δύναμη δηλώνεται ως **. Πχ.:

```
>>> print 9**2, 9**0.5
81 3.0
```

Αξίζει να σημειώσουμε ότι οι ακέραιοι αριθμοί στην Python έχουν απεριόριστο μέγεθος. Πχ.:

```
>>> print 2**1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542167660429831652624386837205668069376
```

Για να φυλάξουμε ενδιάμεσα αποτελέσματα, οι περισσότερες αριθμομηχανές διαθέτουν έναν ή περισσότερους **καταχωρητές μνήμης** (Memory Register = MR). Η Python επιτρέπει πρακτικά απεριόριστο πλήθος καταχωρητών. Τους ονομάζει **μεταβλητές** (variables) και ο προγραμματιστής τους δίνει ένα όνομα (με έναν ή περισσότερους χαρακτήρες) και μια τιμή ως εξής:

```
>>> a=5
```

Δηλώθηκε η μεταβλητή a με τιμή 5.

```
>>> metavliti2=34.45
```

Ομοίως, δηλώθηκε η metavliti2 με τιμή 34.45. Και τώρα μπορούμε να τις χρησιμοποιήσουμε σε εκφράσεις σα να ήταν αριθμοί:

```
>>> print a, metavliti2, metavliti2/a
>>> 5 34.45 6.89
```

Αυτό που αξίζει να αναφέρουμε τώρα είναι ότι η Python εκτός από ακέραιους και πραγματικους αριθμούς, χειρίζεται εξίσου καλά και μιγαδικούς. Οι μιγαδικοί δηλώνονται με τη μορφή a+bj όπου το j δηλώνει τη φανταστική μονάδα. Πχ.:

```
>>> n=5+2j
>>> m=2+3j
>>> print n, m, n+m, n-m, n*m, n/m
(5+2j) (2+3j) (7+5j) (3-1j) (4+19j) (1.23076923077-0.846153846154j)
```

Προσέξτε ότι στη διαίρεση μιγαδικών αριθμών η μετατροπή των συνιστωσών σε πραγματικά μεγέθη γίνεται αυτόματα. Όμως δεν γίνεται αυτόματα η μετατροπή πραγματικών σε μιγαδικούς όπου αυτό απαιτείται. Για παράδειγμα, αν θέλουμε να υπολογίσουμε την τετραγωνική ρίζα του –1, αυτό:

```
>>> print (-1)**0.5
Traceback (most recent call last):
  File "<pyshell#44>", line 1, in <module>
    (-1)**0.5
ValueError: negative number cannot be raised to a fractional power
```

...βγάζει λάθος. Όμως αυτό:

```
>>> print (-1+0j)**0.5
(6.12323399574e-17+1j)
```

...δίνει μια πολύ καλή προσέγγιση[4] της φανταστικής μονάδας (το πραγματικό μέρος είναι 6.123×10^{-17}).

Τέλος αξίζει να επισημάνουμε ότι η Python διαθέτει μια πληθώρα συναρτήσεων για πράξεις με διανύσματα και πίνακες, τριγωνομετρικούς υπολογισμούς, γραφικές παραστάσεις, κλπ, και ακόμα διαθέτει συναρτήσεις όχι μόνο για μαθηματικές λειτουργίες αλλά για ο,τιδήποτε μπορεί να κάνει ένας υπολογιστής (πχ. συναρτήσεις για αναπαραγωγή μουσικής και βίντεο, για κρυπτογράφηση αρχείων, για απομακρυσμένη σύνδεση με υπολογιστές στο διαδίκτυο κλπ). Οι συναρτήσεις ομαδοποιούνται σε βιβλιοθήκες ομοειδών συναρτήσεων και μπορείτε ψάχνοντας το on-line help ή σε ένα εγχειρίδιο της Python να ενημερωθείτε για αυτές. Από τη στιγμή που θα εντοπίσετε τη συνάρτηση που σας ενδιαφέρει και τη βιβλιοθήκη που την περιέχει, την εισάγετε στην Python με την εντολή import. Για παράδειγμα, για να χρησιμοποιήσουμε τη συνάρτηση του συνημιτόνου (cos) που βρίσκεται στην βιβλιοθήκη math, δίνουμε πρώτα την εντολή:

```
>>> from math import cos
```

Και στη συνέχεια χρησιμοποιούμε τη συνάρτηση:

```
>>> print cos(0.5)
0.87758256189
```

Αρκεί να εισάγουμε μόνο μια φορά τη συνάρτηση για να τη χρησιμοποιούμε συνέχεια όσες φορές θέλουμε, μέχρι να κλείσουμε το παράθυρο της Python. Αν θέλουμε να εισάγουμε όλες τις συναρτήσεις μιας βιβλιοθήκης βάζουμε το σύμβολο * στη θέση του ονόματος της συνάρτησης. Πχ.:

```
>>> from math import *
>>> print cos(0.5), sin(0.5), tan(0.5)
0.87758256189 0.479425538604 0.546302489844
```

Και βέβαια, εκτός από τις ενσωματωμένες βιβλιοθήκες και συναρτήσεις, ο κάθε προγραμματιστής μπορεί να φτιάξει δικές του. Στο διαδίκτυο υπάρχει πληθώρα βιβλιοθηκών σε Python για οποιαδήποτε λειτουργία φανταστεί κανείς. Πολύ γρήγορα θα μάθετε να φτιάχνετε κι εσείς τις δικές σας.

[4] Σχετικά με τις προσεγγίσεις: Συχνά το αποτέλεσμα που δίνει η Python για τα ψηφία μετά την υποδιαστολή δεν είναι ακριβώς αυτό που θα περιμέναμε. Αυτό συμβαίνει επειδή η Python (όπως και οι περισσότερες γλώσσες προγραμματισμού) αποθηκεύει τους πραγματικούς αριθμούς στο δυαδικό σύστημα (με αυστηρά καθορισμένο πλήθος bits) κι αυτή η μετατροπή οδηγεί σε μικρή απώλεια ακριβείας. Φυσικά υπάρχουν αρκετοί τρόποι για να παρακαμφθεί αυτό το πρόβλημα, όμως δεν είναι αντικείμενο του παρόντος να τους συζητήσουμε.

5.4 Το Πρώτο μας Πρόγραμμα

Τώρα γνωρίζουμε αρκετά για να επιχειρήσουμε ένα απλό πρόγραμμα. Έστω ότι θέλουμε να φτιάξουμε ένα πρόγραμμα που θα υπολογίζει τις λύσεις της δευτεροβάθμιας εξίσωσης $2x^2$-5x+2=0. Ας ξεκινήσουμε όπως γνωρίζουμε, δίνοντας τις εντολές μια-μια όπως σε μια αριθμομηχανή, κι ας υπολογίσουμε πρώτα τη διακρίνουσα:

```
>>> a=2
>>> b=-5
>>> c=2
>>> diak=b*b-4*a*c
>>> print diak
9
```

Βλέπουμε ότι η διακρίνουσα είναι θετική, επομένως μπορούμε να υπολογίσουμε την τετραγωνική της ρίζα:

```
>>> rd=diak**0.5
```

…και να συνεχίσουμε κατά τα γνωστά:

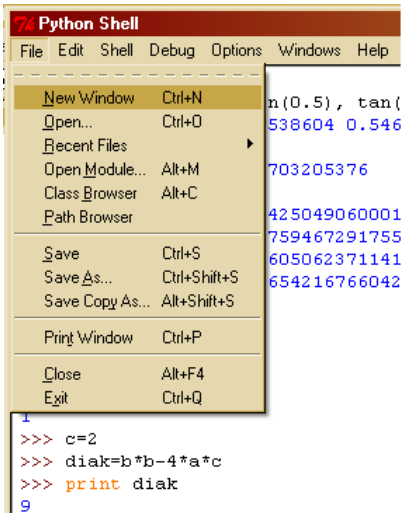
```
>>> x1=(-b-rd)/(2*a)
>>> x2=(-b+rd)/(2*a)
>>> print x1, x2
0.5 2.0
```

Αυτές είναι οι λύσεις. Δείτε τώρα έναν πιο ωραίο τρόπο να τις τυπώσουμε:

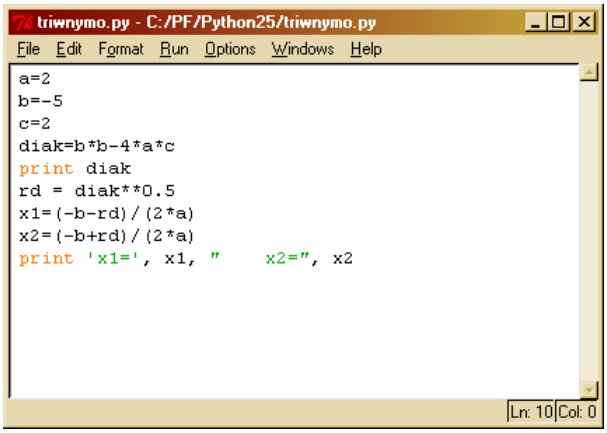
```
>>> print 'x1=', x1, "    x2=", x2
x1= 0.5     x2= 2.0
```

Η εντολή print δεν τυπώνει μόνο αριθμούς και περιεχόμενα μεταβλητών. Τυπώνει και ο,τιδήποτε βάλουμε μέσα σε (απλά ή διπλά) εισαγωγικά ως κείμενο. Ακόμα και τα κενά.

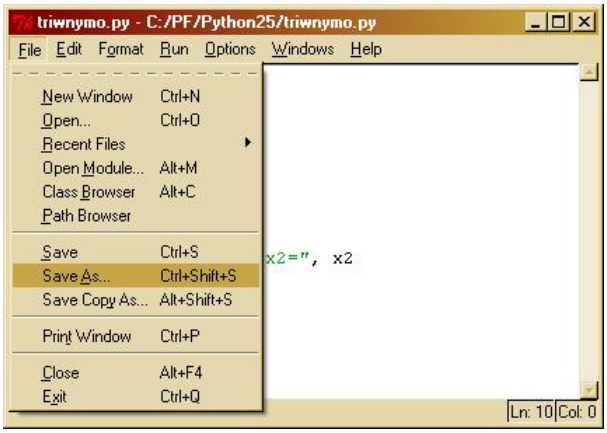
Και τώρα που λύσαμε την εξίσωση $2x^2$-5x+2=0 ας προσπαθήσουμε να λύσουμε άλλη μια παρόμοια, ας πούμε την $2x^2$-5x+3=0. Προσέξτε ότι δεν αρκεί να αλλάξουμε την τιμή της μεταβλητής c από 2 σε 3. Θα πρέπει να ξαναδώσουμε όλες τις επόμενες εντολές μια-μια, κι ας είναι ακριβώς ίδιες! Σκεφτείτε πόσο βολικό θα ήταν να λέγαμε στην Python να χρησιμοποιήσει τις ίδιες εντολές χωρίς να τις ξαναγράψουμε. Για να το κάνουμε αυτό θα πρέπει να έχουμε γράψει τις εντολές σε ένα αρχείο και να πούμε στη γλώσσα να τις εκτελέσει. Ο πιο απλός τρόπος είναι επιλέγοντας File>New Window (Ctrl+N) από το μενού του IDLE...



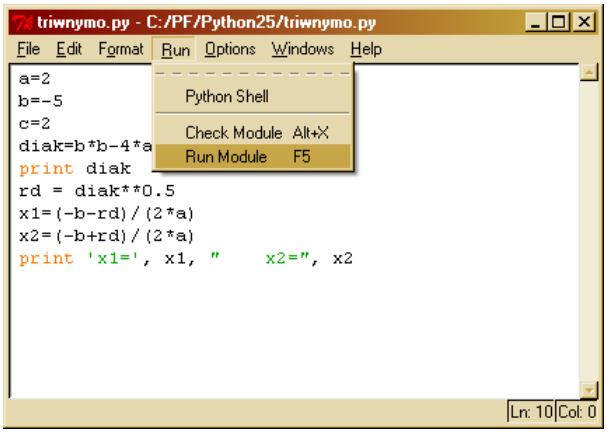
... και στη συνέχεια στο νέο παράθυρο που θα ανοίξει γράφουμε τις εντολές (ή τις κάνουμε copy-paste από το αρχικό, χωρίς όμως τους χαρακτήρες '>>>' στην αρχή κάθε γραμμής)...



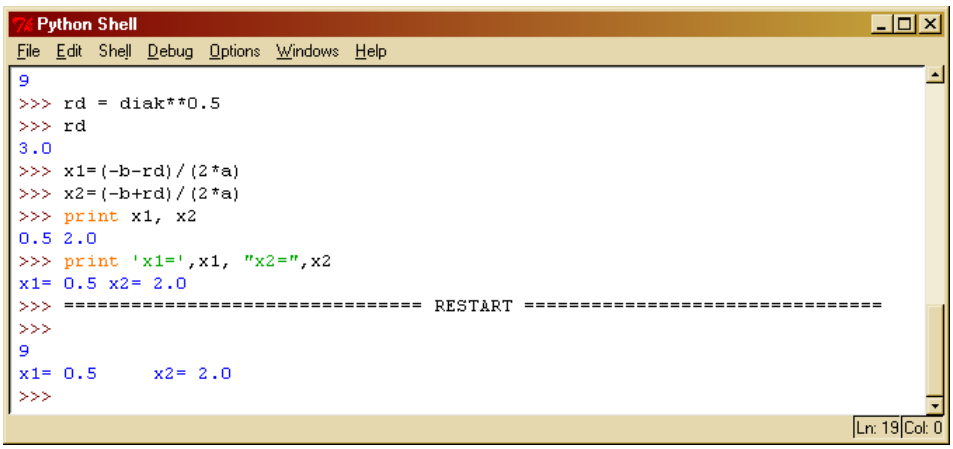
... και το σώζουμε με Save As… (Ctrl+Shift+S). Για όνομα αρχείου βάζουμε ό,τι θέλουμε, όμως η προέκταση πρέπει να είναι .py. Πχ. «triwnymo.py».



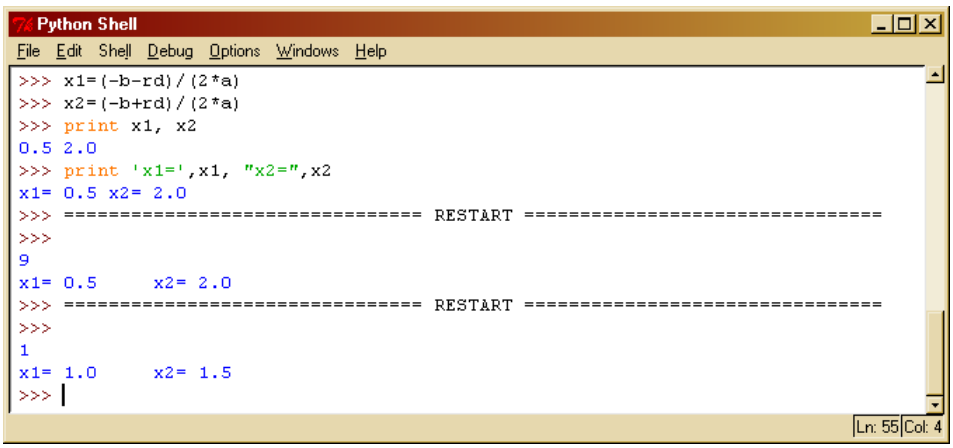
Και τώρα αρκεί να επιλέξουμε Run>Run Module (F5)...



... και θα δούμε όλες τις εντολές να εκτελούνται στο αρχικό παράθυρο:



Αν τώρα αλλάξουμε το c=3, ξανασώσουμε το αρχείο και το ξανατρέξουμε, θα έχουμε βρει τις λύσεις της 2^{ης} εξίσωσης $2x^2-5x+3=0$.



Λέμε ότι οι εντολές που έχουμε γράψει στο αρχείο triwnymo.py αποτελούν ένα πρόγραμμα. Φυσικά αν θέλουμε να λύσουμε ένα και μόνο πρόβλημα, δεν υπάρχει λόγος να γράψουμε πρόγραμμα. Αν όμως θέλουμε να λύσουμε μια κατηγορία ομοειδών προβλημάτων (πχ. να λύσουμε πολλές εξισώσεις 2^{ου} βαθμού), τότε κερδίζουμε πολύ χρόνο φτιάχνοντας ένα κατάλληλο πρόγραμμα. Όπως είδαμε, αρκεί να αλλάξουμε μόνο τους συντελεστές a,b,c για να λύσουμε μια διαφορετική εξίσωση με το ίδιο πρόγραμμα.

Όμως για να το πετύχουμε αυτό θα πρέπει να είμαστε πολύ προνοητικοί κατά την κατασκευή του προγράμματος, ώστε να ανταποκρίνεται σωστά σε κάθε δυνατό συνδυασμό παραμέτρων. Το πρόγραμμα που έχουμε φτιάξει δεν δουλεύει σωστά αν τύχει η διακρίνουσα να είναι αρνητική (δοκιμάστε το). Ένα πιο προνοητικά κατασκευασμένο πρόγραμμα μόλις εντόπιζε αρνητική διακρίνουσα θα τύπωνε κατάλληλο μήνυμα και δεν θα προχωρούσε στον υπολογισμό της τετραγωνικής ρίζας. Ή θα ρωτούσε αν θέλουμε να υπολογίσει μιγαδικές λύσεις και θα προχωρούσε διαφορετικά.

Μια άλλη παράμετρος στην κατασκευή προγραμμάτων είναι ότι πολύ συχνά θα χρειαστεί να το χρησιμοποιήσουν και άλλοι άνθρωποι (συνήθως τους λέμε χρήστες) εκτός από τον προγραμματιστή. Οι χρήστες δεν έχουν καμμία υποχρέωση να γνωρίζουν πώς δουλεύει το πρόγραμμα, τί κάνει η κάθε μεταβλητή και ποιές ακριβώς τιμές θα πρέπει να αλλάξουν στον κώδικα για να λυθεί σωστά το πρόβλημά τους. Για αυτό το λόγο οι καλοί προγραμματιστές φτιάχνουν τα προγράμματά τους με τέτοιο τρόπο ώστε να μη χρειάζονται αλλαγές στον κώδικα κάθε φορά που χρησιμοποιούνται για την επίλυση ενός νέου προβλήματος. Για παράδειγμα αντί να αλλάζουν τις τιμές των μεταβλητών στον κώδικα, βάζουν το πρόγραμμα στην αρχή να κάνει τις κατάλληλες ερωτήσεις στον χρήστη και από τις απαντήσεις του διαμορφώνουν τις τιμές στις μεταβλητές. Όμως για να κάνουμε τέτοιες βελτιώσεις στα προγράμματά μας χρειάζεται να μάθουμε μερικές εντολές ακόμη, που είναι το αντικείμενο της επόμενης ενότητας.

5.5 Εντολές input, if, else

Είδαμε προηγουμένως την εντολή print που χρησιμοποιείται για να τυπώσουμε δεδομένα στην οθόνη του υπολογιστή μας. Την αντίθετη λειτουργία πραγματοποιεί μια συνάρτηση με το όνομα input. Χρησιμοποιείται ως εξής:

```
>>> a=input()
4
>>> print a
4
```

Εδώ αντί να δώσουμε μια τιμή στη μεταβλητή a της λέμε να πάρει την τιμή που θα της δώσει η συνάρτηση input. Στη συνέχεια, ό,τι γράψουμε στο πληκτρολόγιο κατά την εκτέλεση της input περνάει ως τιμή στην a, κι όταν τυπώσουμε την a βλέπουμε την τιμή που δώσαμε.

Μέσα στην παρένθεση της input μπορούμε προαιρετικά να βάλουμε ένα μήνυμα που θα τυπώνεται τη στιγμή που εκτελείται:

```
>>> b=input('Δώσε έναν αριθμό:')
Δώσε έναν αριθμό:12
>>> print b
12
```

Συχνά κάποια εντολή χρειάζεται να εκτελεστεί υπό συνθήκη. Αν η συνθήκη ισχύει τότε η εντολή (ή οι εντολές – μπορεί να είναι και περισσότερες από μια) πρέπει να

εκτελεστεί, διαφορετικά όχι. Αυτό το πετυχαίνει η εντολή if. Δείτε το παρακάτω τμήμα προγράμματος:

```
...
a=input('Δώσε τον διαιρετέο:')
b=input('Δώσε τον διαιρέτη:')
if b==0:
      print 'Ο διαιρέτης δεν πρέπει να είναι μηδέν.'
      b= input('Ξαναδώσε τον διαιρέτη:')
print 'Το πηλίκο της διαίρεσης είναι:', a/b
...
```

Αυτό το τμήμα κώδικα ζητά από τον χρήστη να πληκτρολογήσει δυο αριθμούς, τον a (διαιρετέο) και τον b (διαιρέτη) με σκοπό να τους διαιρέσει και να τυπώσει το αποτέλεσμα. Όμως πριν γίνει η διαίρεση πρέπει να ελεγχθεί μήπως ο διαιρέτης είναι μηδέν. Αυτό κάνει η εντολή if. Ελέγχει τη συνθήκη b==0 (δηλαδή αν το b είναι μηδέν) και αν αυτή ισχύει, τότε εκτελεί τις δυο εντολές που είναι γραμμένες μια στήλη (TAB) δεξιώτερα, κάτω από την if (λέγονται εσωτερικές εντολές της if) και αμέσως μετά συνεχίζει στην εκτέλεση της print. Αν όμως η συνθήκη δεν ισχύει, οι δυο εσωτερικές εντολές δεν εκτελούνται και πηγαίνουμε κατευθείαν στην print. (Σημείωση: Στην Python τα TABs έχουν συντακτικό ρόλο. Με αυτόν τον τρόπο αποφεύγεται η χρήση συμβόλων για την αρχή και το τέλος ομάδων εντολών και ο κώδικας διατηρείται απλοποιημένος κι ευανάγνωστος.)

Προσέξτε ότι ο έλεγχος ισότητας συμβολίζεται με δυο σύμβολα '='. Το ένα όπως ξέρουμε σημαίνει καταχώρηση τιμής σε μεταβλητή, κι όχι έλεγχο ισότητας. Το διπλό '==' λέγεται τελεστής. Οι τελεστές που μπορούν να χρησιμοποιηθούν στην if είναι:

<	μικρότερο
>	μεγαλύτερο
<=	μικρότερο ή ίσο
>=	μεγαλύτερο ή ίσο
==	ίσο
!=	διαφορετικό

Μια παραλλαγή της if χρησιμοποιεί την λέξη else για να ορίσει δυο ομάδες εσωτερικών εντολών, μια που θα εκτελείται όταν ισχύει η συνθήκη και μια που θα εκτελείται όταν δεν ισχύει. Δείτε ένα παράδειγμα:

```
...
a=input('Δώσε έναν αριθμό:')
if a>=0:
      print 'Ο αριθμός είναι θετικός (ή μηδέν).'
      b=a
else:
      print 'Ο αριθμός είναι αρνητικός.'
      b=-a
print 'Η απόλυτη τιμή του αριθμού είναι:', b
...
```

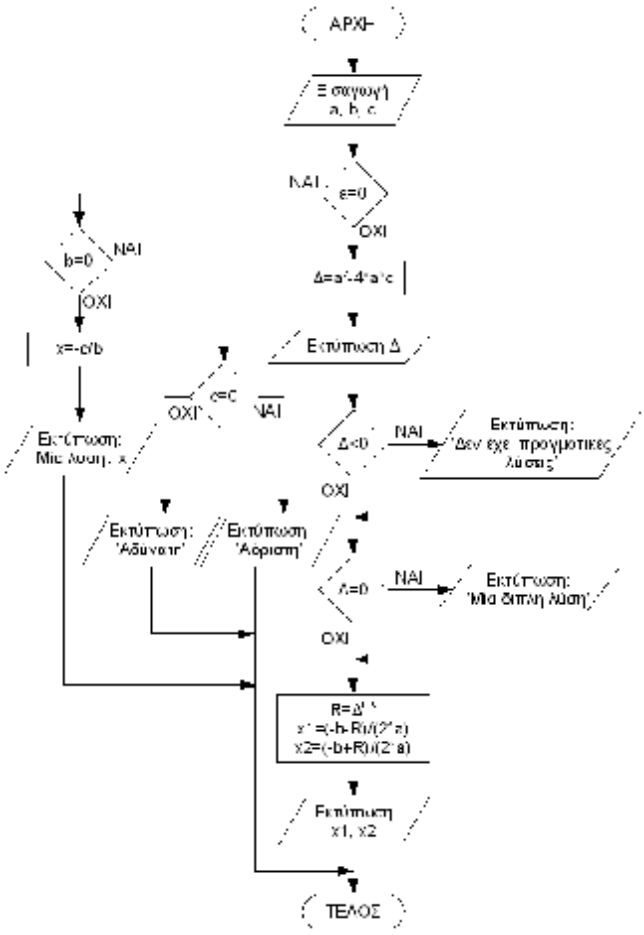
Αυτό το τμήμα κώδικα διαβάζει έναν αριθμό από το πληκτρολόγιο, ελέγχει αν είναι θετικός ή αρνητικός και τυπώνει την απόλυτη τιμή του. Οι δυο εσωτερικές εντολές που βρίσκονται κάτω από την if εκτελούνται μόνο όταν η συνθήκη a>=0 ισχύει.

Αντίθετα, οι δυο εσωτερικές εντολές κάτω από την else εκτελούνται μόνο όταν η συνθήκη a>=0 δεν ισχύει. Η print εκτελείται και στις δυο περιπτώσεις, αφού δεν είναι εσωτερική εντολή.

Επίσης, αξίζει να σημειώσουμε ότι στις εσωτερικές εντολές μιας if μπορούν να βρίσκονται οποιεσδήποτε εντολές, ακόμα κι άλλες if, όπως θα δούμε στην επόμενη ενότητα.

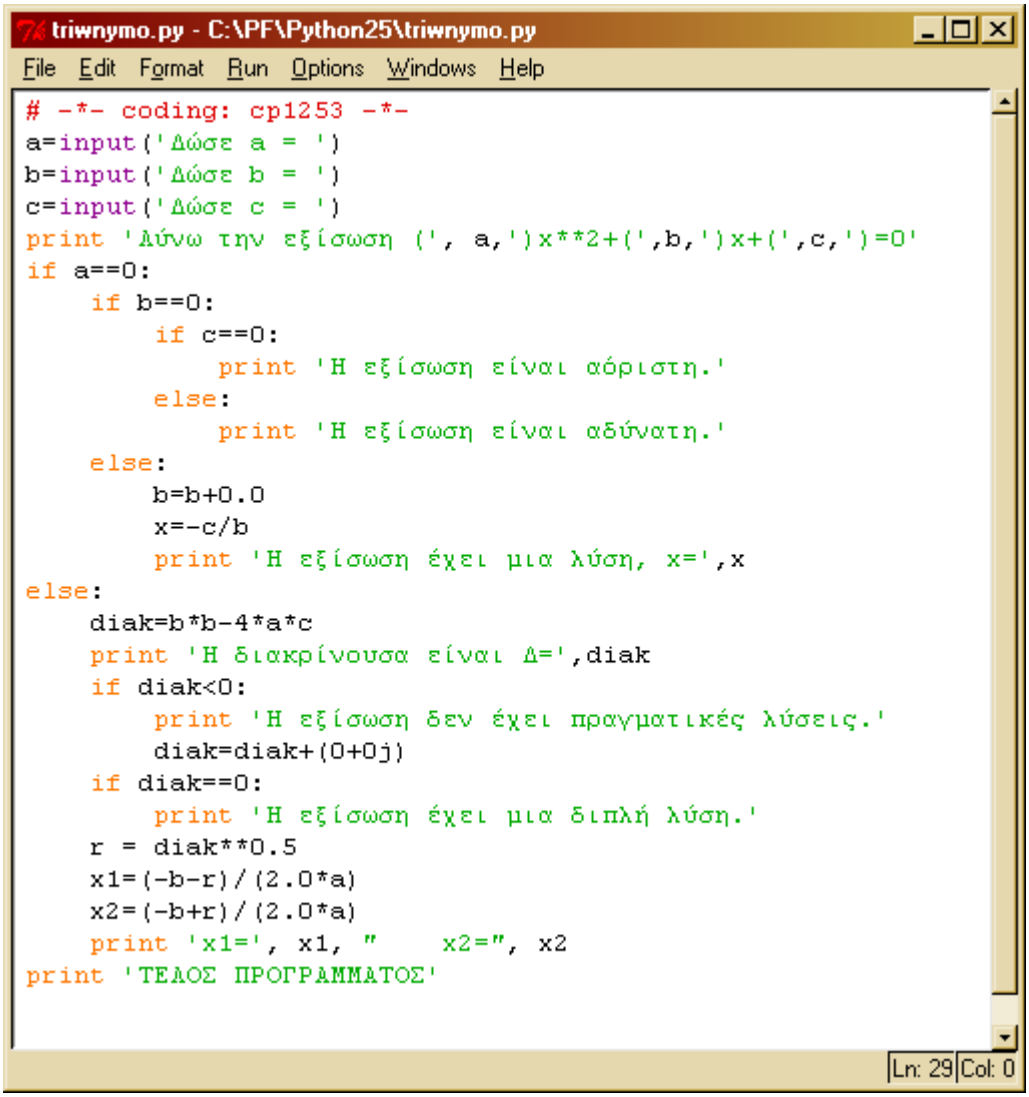
5.6 Το Πλήρες Πρόγραμμα για την Επίλυση Εξισώσεων 2^{ου} Βαθμού

Τώρα γνωρίζουμε αρκετά για να φτιάξουμε ένα πλήρες πρόγραμμα για την επίλυση εξισώσεων 2^{ου} βαθμού. Αλλά πριν αρχίσουμε να γράφουμε τον κώδικα ενός σύνθετου προγράμματος, μια καλή πρακτική είναι να σχεδιάζουμε πρώτα το διάγραμμα ροής. Έτσι αποφεύγουμε να ξεχάσουμε κάποια βασική λειτουργία του προγράμματος και γλιτώνουμε από πολλές διορθώσεις στον κώδικα. Να πώς θα μπορούσαμε να σχεδιάσουμε το διάγραμμα ροής για την επίλυση της εξίσωσης 2^{ου} βαθμού:



Και τώρα είναι πολύ πιο εύκολο να γράψουμε τον κώδικα του προγράμματος. Τα καίρια σημεία του διαγράμματος είναι οι ρόμβοι. Κάθε ρόμβος αντιστοιχεί σε μια εντολή if με τον έλεγχο που περιέχει. Ό,τι βρίσκεται στον κλάδο ΝΑΙ, στον κώδικα θα εμφανιστεί στο πεδίο των εσωτερικών εντολών κάτω από την if. Ό,τι βρίσκεται

στον κλάδο ΟΧΙ, στον κώδικα θα εμφανιστεί κάτω από την else. Ακολουθεί ο κώδικας[5]:



Παρατηρήστε ότι το διάγραμμα ροής καθορίζει τη δομή του προγράμματος κι όχι τις λεπτομέρειες του κώδικα. Δηλαδή δεν μας δεσμεύει στα ακριβή ονόματα των μεταβλητών που θα χρησιμοποιήσουμε τελικά (αν και καλό είναι να υπάρχει μια εμφανής αντιστοιχία), ούτε εξαντλεί όλα τα ενημερωτικά μηνύματα που τελικά αποφασίζουμε να εντάξουμε στο πρόγραμμά μας. Επίσης (και αυτό είναι το πιο σημαντικό) δεν ασχολείται καθόλου με τις προγραμματιστικές λεπτομέρειες που εξαρτώνται από τη γλώσσα και αφορούν τον τρόπο υπολογισμού των τιμών των μεταβλητών και τις ιδιαιτερότητες των επιμέρους εντολών και συναρτήσεων.

[5] Η πρώτη γραμμή (`# -*- coding: cp1253 -*-`) δηλώνει ότι στο αρχείο χρησιμοποιούνται ελληνικοί χαρακτήρες (codepage 1253). Δεν είναι απαραίτητη όταν δουλεύουμε με ένα μόνο αρχείο σε έναν υπολογιστή, όμως αν το πρόγραμμά μας αποτελείται από πολλά αρχεία ή πρόκειται να το μεταφέρουμε σε άλλον υπολογιστή, γλιτώνουμε από πολλές ασυμβατότητες αν δηλώσουμε την κωδικοποίηση με την οποία γράφτηκε, βάζοντας στην 1η ή στην 2η γραμμή του κάθε αρχείου αυτή τη δήλωση. Οι συνηθέστεροι κωδικοί είναι cp1253 (Windows Ελληνικά) και utf-8 (για Unicode).

Για παράδειγμα, προσέξτε ότι στον υπολογισμό των x1 και x2 οι παρονομαστές στον κώδικα είναι (2.0*a) αντί (2*a) που φαίνεται στο διάγραμμα ροής. Όταν φτιάχνουμε το διάγραμμα ροής δεν σκεφτόμαστε τις τεχνικές λεπτομέρειες του τρόπου που κάνει διαίρεση η Python, όμως στον κώδικα πρέπει να τις σκεφτούμε και να προνοήσουμε ώστε να δουλέψει σωστά. Για τον ίδιο λόγο, στην περίπτωση που a=0 και b δεν είναι μηδέν, στον κώδικα έχουμε βάλει την εντολή:

```
b=b+0.0
```

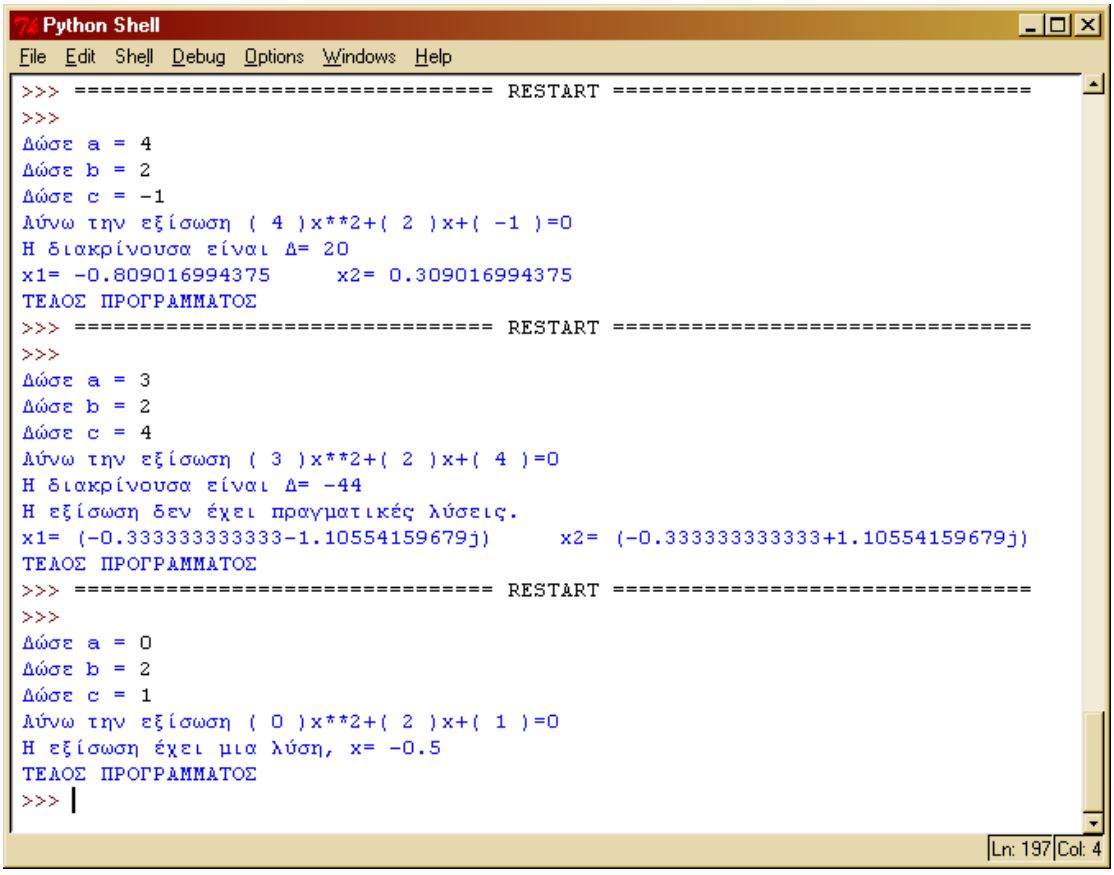
Ουσιαστικά δεν αλλάζουμε την τιμή της b, αφού προσθέτουμε μηδέν, όμως προσθέσαμε το μηδέν με πραγματική μορφή. Αυτός είναι ένας τρόπος να πούμε στην Python να θεωρεί στο εξής την b ως πραγματική μεταβλητή, ώστε να αποφευχθεί η ακέραια διαίρεση.

Στο ίδιο πνεύμα, προσέξτε ότι στο διάγραμμα ροής υπολογίζεται η τετραγωνική ρίζα του Δ ακόμα κι αν αυτό είναι αρνητικό. Όμως στον κώδικα η διακρίνουσα χρειάζεται πρώτα να μετατραπεί σε μιγαδικό αριθμό για να λειτουργήσει σωστά η διαδικασία. Αυτή η μετατροπή γίνεται με την εντολή:

```
diak=diak+(0+0j)
```

Η εντολή αυτή δεν αλλάζει την τιμή της μεταβλητής diak αφού προσθέτει μηδέν, όμως προσθέτει το μηδέν με μιγαδική μορφή κι έτσι αναγκάζει την diak στο εξής να είναι μιγαδική μεταβλητή.

Ακολουθούν μερικά παραδείγματα από την εκτέλεση του προγράμματος:



5.7 Ασκήσεις

Εκπονήστε τις παρακάτω ασκήσεις. Ομαδοποιήστε τα αρχεία με τις απαντήσεις σας σας σε ένα ενιαίο συμπιεσμένο αρχείο και αναρτήστε το στο Moodle σύμφωνα με τις οδηγίες που θα σας δοθούν στο εργαστήριο. Δώστε ιδιαίτερη προσοχή στην πληρότητα και την εμφάνιση των απαντήσεών σας.

Άσκηση #1

Πληκτρολογήστε τον κώδικα της ενότητας 5.6 και τρέξτε τον. Πειραματιστείτε με τη λειτουργία του και αν θέλετε βελτιώστε τον κατά βούληση.

Άσκηση #2

Παρακάτω περιγράφεται η διαδικασία[6] επίλυσης της πολυωνυμικής εξίσωσης 3ου βαθμού $Ax^3+Bx^2+Cx+D=0$. Χρησιμοποιήστε το Synergo και αποτυπώστε την σε ένα διάγραμμα ροής.

Πώς λύνουμε την εξίσωση $Ax^3+Bx^2+Cx+D=0$

Αν A=0 τότε η εξίσωση εκφυλίζεται σε 2ου βαθμού και λύνεται σύμφωνα με τον κώδικα της Άσκησης #1. Διαφορετικά διαιρούμε όλους τους συντελεστές δια A και η εξίσωση γίνεται: $x^3 + ax^2 + bx + c = 0$, όπου $a = \frac{B}{A},\ b = \frac{C}{A},\ c = \frac{D}{A}$.

Η διακρίνουσα υπολογίζεται ως: $\Delta = 18abc - 4a^3c + a^2b^2 - 4b^3 - 27c^2$

Αν Δ>0, η εξίσωση έχει 3 πραγματικές λύσεις.
Αν Δ=0, έχει λιγώτερες από 3 πραγματικές λύσεις (κάποιες συμπίπτουν).
Αν Δ<0, η εξίσωση έχει 1 πραγματική λύση και 2 μιγαδικές.

Για να βρούμε τις λύσεις, υπολογίζουμε πρώτα τις ενδιάμεσες τιμές:

$$p = b - \frac{a^2}{3} \qquad q = c + \frac{2a^3 - 9ab}{27} \qquad u = \sqrt[3]{-\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

Προσοχή, η τελευταία σχέση μάς δίνει 6 τιμές για το u, όχι μόνο 2 (για κάθε τιμή του προσήμου της τετραγωνικής ρίζας, η κυβική ρίζα δίνει 3 τιμές με διαφορά φάσης 270o στο μιγαδικό επίπεδο).

Οι λύσεις της εξίσωσης υπολογίζονται τελικά από τη σχέση: $x = -\frac{p}{3u} + u - \frac{a}{3}$

Βάζοντας τις 6 τιμές του u στην παραπάνω σχέση, κάποιες τιμές συμπίπτουν και τελικά εμφανίζονται το πολύ 3 διαφορετικές τιμές για το x.

Άσκηση #3

Με τη βοήθεια του διαγράμματος ροής που φτιάξατε στην Άσκηση #2, γράψτε ένα πρόγραμμα στη γλώσσα Python που θα λύνει την εξίσωση 3ου βαθμού σύμφωνα με τους συντελεστές A, B, C, D που θα δίνει ο χρήστης[7]. Γράψτε όσο το δυνατόν πληρέστερο πρόγραμμα.

[6] Για περισσότερες λεπτομέρειες, δείτε: http://en.wikipedia.org/wiki/Cubic_polynomial

[7] Αν επιθυμείτε να ελέγξετε τη λύση σας μπορείτε να χρησιμοποιήσετε το εξής πρόγραμμα επίλυσης εξισώσεων (μέχρι και 4ου βαθμού): http://www.freewebs.com/brianjs/ultimateequationsolver.htm

Εργαστήριο 6: Δεύτερη Άσκηση Προγραμματισμού Python

6.1 Γενικά

Στην άσκηση αυτή θα εξεταστούν μερικά πιο εξελιγμένα χαρακτηριστικά της γλώσσας Python, όπως είναι οι επαναλήψεις, ο ορισμός συναρτήσεων από τον προγραμματιστή, καθώς και μερικές συναρτήσεις γραφικών.

6.2 Ορισμός Συναρτήσεων

Στην προηγούμενη άσκηση είδαμε συνοπτικά τον τρόπο χρήσης συναρτήσεων από τις βιβλιοθήκες της Python με την εντολή import. Τώρα θα δούμε πώς μπορούμε να ορίσουμε δικές μας συναρτήσεις. Έστω η συνάρτηση $f(x)= 0.0001*x^3+0.015*x^2-1.14*x-2.8$. Στην Python την ορίζουμε με την εντολή def ως εξής:

```
>>> def f(x):
      return 0.0001*x**3+0.015*x**2-1.14*x-2.8
```

Δηλαδή, μετά την def βάζουμε το όνομα της συνάρτησης και σε παρένθεση την ανεξάρτητη μεταβλητή. Η εντολή return στην επόμενη γραμμή καθορίζει την τιμή που θα 'επιστρέψει' η συνάρτηση μόλις την καλέσουμε (για παράδειγμα, η τιμή που θα τυπώσει αν την εκτελέσουμε με την print). Τώρα μπορούμε να την καλέσουμε ακριβώς όπως τις ενσωματωμένες συναρτήσεις της γλώσσας:

```
>>> print f(3.5)
-6.6019625
>>> print f(-12), f(111.11)
12.8672 192.886177563
```

Αν στη συνάρτηση υπάρχουν περισσότερες από μια ανεξάρτητες μεταβλητές (λέγονται και **ορίσματα**), τις χωρίζουμε με κόμματα. Για παράδειγμα, να μια συνάρτηση που υπολογίζει τη διακρίνουσα ενός τριωνύμου από τους συντελεστές a, b και c:

```
>>> def diakrinousa(a,b,c):
      return b*b-4*a*c
>>> print diakrinousa(4,2,-1)
20
```

Επιπλέον, οι συναρτήσεις μπορούν να αποτελούνται από πολλές γραμμές κώδικα, να καλούν άλλες συναρτήσεις και δεν είναι απαραίτητο να επιστρέφουν μια αριθμητική τιμή αλλά ο,τιδήποτε. Δείτε το παρακάτω παράδειγμα:

```
>>> def remainder(x,y):
      z=x/y
```

```
      r=x-z*y
      return r

>>> def check_odd_or_even(x):
      y=remainder(x,2)
      if y==0:
            return 'even'
      else:
            return 'odd'
```

Η συνάρτηση remainder υπολογίζει και επιστρέφει το υπόλοιπο της διαίρεσης δυο ακεραίων αριθμών. Στη συνέχεια η check_odd_or_even καλεί την remainder για να διαπιστώσει αν κάποιος ακέραιος είναι περιττός (odd) ή άρτιος (even).

```
>>> print remainder(8,3)
2
>>> print check_odd_or_even(13)
odd
>>> print check_odd_or_even(14)
even
```

Προσέξτε ότι όπως γράψαμε την check_odd_or_even επιστρέφει αλφαριθμητικά (λέξεις).

6.3 Επαναλήψεις με for και Λίστες τιμών

Έστω ότι θέλουμε να τυπώσουμε τις τιμές της συνάρτησης f(x) που ορίσαμε προηγουμένως για πολλές τιμές του x. Για να αποφύγουμε να δώσουμε πολλές εντολές print, η Python επιτρέπει να γράψουμε κάτι τέτοιο:

```
>>> for x in [0, 1, 2, 3, 4]:
      print 'Για x=', x, '   f(x)=',f(x)

Για x= 0    f(x)= -2.8
Για x= 1    f(x)= -3.9249
Για x= 2    f(x)= -5.0192
Για x= 3    f(x)= -6.0823
Για x= 4    f(x)= -7.1136
```

Το [0, 1, 2, 3, 4] λέγεται **λίστα**. Περιέχει πέντε αριθμούς και η εντολή for θα δώσει διαδοχικά στη μεταβλητή x κάθε τιμή που βρίσκεται μέσα στη λίστα και θα επαναλάβει τις εσωτερικές της εντολές (εδώ έχει μόνο την print) για κάθε τιμή του x. Το χρήσιμο με αυτή τη σύνταξη είναι ότι η Python διαθέτει την ενσωματωμένη συνάρτηση range που δημιουργεί λίστες αριθμών και δεν χρειάζεται να τους πληκτρολογούμε εκ των προτέρων:

```
>>> print range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Η range όταν καλείται με ένα όρισμα επιστρέφει μια λίστα με όλους τους ακεραίους που δεν ξεπερνούν το όρισμα, ξεκινώντας από το 0.

```
>>> print range(-3,4)
```

```
[-3, -2, -1, 0, 1, 2, 3]
```

Αν την καλέσουμε με δυο ορίσματα καθορίζουμε εμείς από που θα ξεκινήσει.

```
>>> print range(1,15,2)
[1, 3, 5, 7, 9, 11, 13]
```

Με τρία ορίσματα καθορίζουμε και το βήμα με το οποίο θα μετράει. Εδώ επέστρεψε όλους τους ακεραίους που είναι μικρότεροι από 15, ξεκινώντας από το 1 και μετρώντας ανά 2.

Έτσι, αν θέλουμε να γίνουν πολλές επαναλήψεις, μπορούμε να συνδυάσουμε την συνάρτηση range() με την εντολή for, γράφοντας κάτι τέτοιο:

```
>>> for x in range(100):
      print 'Για x=', x, '   f(x)=',f(x)

Για x= 0    f(x)= -2.8
Για x= 1    f(x)= -3.9249
Για x= 2    f(x)= -5.0192
Για x= 3    f(x)= -6.0823
...
Για x= 97    f(x)= 119.0223
Για x= 98    f(x)= 123.6592
Για x= 99    f(x)= 128.3849
```

Μπορούμε να έχουμε επαναλήψεις ως εσωτερικές εντολές σε άλλες επαναλήψεις. Σε αυτή την περίπτωση οι επαναλήψεις λέγονται **εμφωλιασμένες** (**nested**). Εμφωλιασμένες επαναλήψεις χρησιμοποιούνται συχνά για να αναζητήσουμε λύσεις σε κάποιο πρόβλημα με εξαντλητικό τρόπο, αρκεί να γνωρίζουμε τη συνθήκη που πρέπει να ικανοποιηθεί και το πεδίο ορισμού του είναι αρκετά περιορισμένο. Για παράδειγμα:

<u>Πρόβλημα 1:</u> Να βρεθούν όλες οι πυθαγόρειες τριάδες ακεραίων (a, b, c) με 0<a<b<c<50.

Πυθαγόρειες λέγονται οι τριάδες ακεραίων που ικανοποιούν την συνθήκη του πυθαγόρειου θεωρήματος, δηλαδή $a^2+b^2=c^2$.

Αφού οι αριθμοί πρέπει να είναι ακέραιοι στο διάστημα (0,50) το πεδίο ορισμού είναι αρκετά περιορισμένο και μπορούμε να κάνουμε εξαντλητική αναζήτηση μέσα σε αυτό (δηλαδή να εξετάσουμε όλους τους δυνατούς συνδυασμούς) γράφοντας κάτι τέτοιο:

```
>>> for a in range(1,50):
      for b in range(a+1,50):
            for c in range(b+1,50):
                  if a*a+b*b==c*c:
                        print a, b, c

3 4 5
5 12 13
6 8 10
7 24 25
```

```
8 15 17
9 12 15
9 40 41
10 24 26
12 16 20
12 35 37
15 20 25
15 36 39
16 30 34
18 24 30
20 21 29
21 28 35
24 32 40
27 36 45
```

Οι εμφωλιασμένες εντολές for εξασφαλίζουν ότι συνδυάζουμε κάθε τιμή του a με όλες τις τιμές του b και κάθε τιμή του b με όλες τις τιμές του c. Προσέξτε επίσης ότι ξεκινάμε το range της κάθε μιας έναν ακέραιο μετά την τιμή της προηγούμενης μεταβλητής ώστε να εξασφαλίσουμε ότι a<b<c.

Επαναλήψεις πολύ συχνά χρησιμοποιούνται στο εσωτερικό συναρτήσεων, όπως στο επόμενο παράδειγμα:

<u>Πρόβλημα 2:</u> Να γραφτεί συνάρτηση που θα δέχεται έναν ακέραιο αριθμό και θα εξετάζει αν είναι πρώτος ή όχι.

Πρώτος (**prime**) λέγεται ένας θετικός ακέραιος αν δεν διαιρείται ακριβώς με κανέναν άλλον (θετικό) ακέραιο (εκτός από τον εαυτό του και τη μονάδα). Ένας ισοδύναμος ορισμός λέει ότι πρώτος είναι ένας ακέραιος όταν έχει ακριβώς δυο διαιρέτες (για αυτό το λόγο ο 1 δεν θεωρείται πρώτος).

Απευθείας από τον πρώτο ορισμό μπορούμε να γράψουμε:

```
>>> def is_prime(n):
      if n<2:
            return('no')
      for x in range(2,n-1):
            if remainder(n,x)==0:
                  return('no')
      return('yes')
```

Εδώ χρησιμοποιήθηκε επαναληπτικά η συνάρτηση remainder() για να υπολογίσει τα υπόλοιπα όλων των διαιρέσεων του δοσμένου αριθμού n με κάθε αριθμό από 2 μέχρι n-1. Αν κάποιο από αυτά είναι μηδέν τότε ο αριθμός δεν είναι πρώτος και η συνάρτηση επιστρέφει ‘no’. Αν το for τελειώσει χωρίς να μηδενιστεί κανένα υπόλοιπο τότε επιστρέφει ‘yes’. Τώρα μπορούμε να τη χρησιμοποιήσουμε:

```
>>> for t in range(20):
      print t, is_prime(t)

0 no
1 no
2 yes
3 yes
4 no
```

```
5 yes
6 no
7 yes
8 no
9 no
10 no
11 yes
12 no
13 yes
14 no
15 no
16 no
17 yes
18 no
19 yes
```

Σημειωτέον ότι ο παραπάνω αλγόριθμος υπολογισμού δεν είναι ιδιαίτερα γρήγορος. Η καθυστέρησή του είναι εμφανής όσο μεγαλώνουν οι αριθμοί που ελέγχει. Πιο γρήγοροι αλγόριθμοι μπορούν να γραφτούν αν λάβουμε υπόψιν ότι οι διαιρέτες των αριθμών εμφανίζονται κατά ζεύγη (δηλαδή αν ο a είναι διαιρέτης του n, τότε και ο n/a θα είναι διαιρέτης του n) και ότι αν ένας αριθμός δεν είναι διαιρέτης τότε κανένα πολλαπλάσιό του δεν θα είναι διαιρέτης.

6.4 Συναρτήσεις Γραφικών

Η Python διαθέτει πολλές βιβλιοθήκες γραφικών. Ίσως η πιο απλή στη χρήση είναι η βιβλιοθήκη **γραφικών χελώνας** (**turtle graphics**). Πήρε το όνομά της από τη γλώσσα προγραμματισμού Logo, που στις αρχές της δεκαετίας του 1980 εισήγαγε ένα παρόμοιο σύστημα γραφικών που στη θέση του κέρσορα είχε σχεδιασμένη μια χελώνα στην οποία ο προγραμματιστής έδινε εντολές να μετακινηθεί και να σχεδιάσει γραμμές πάνω στην οθόνη.

Για να φορτώσουμε τη βιβλιοθήκη δίνουμε την εντολή:

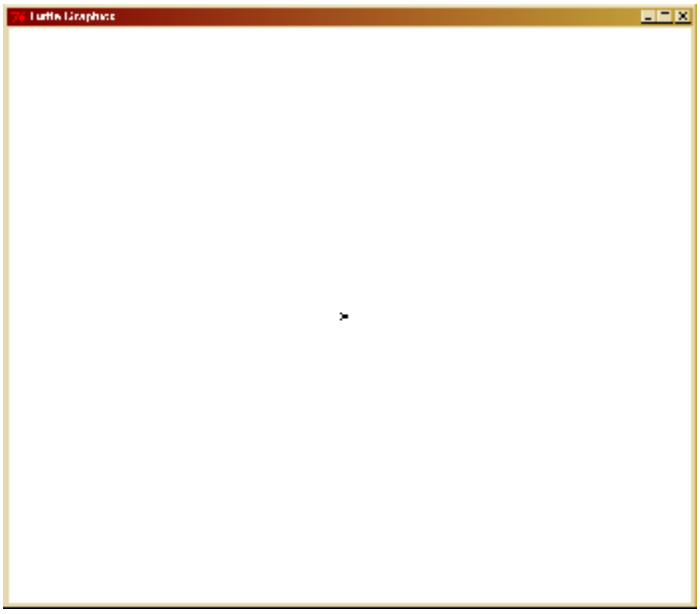
```
>>> from turtle import *
```

Και αμέσως μετά, αν δώσουμε κάποια εντολή[8] γραφικών όπως:

```
>>> reset()
```

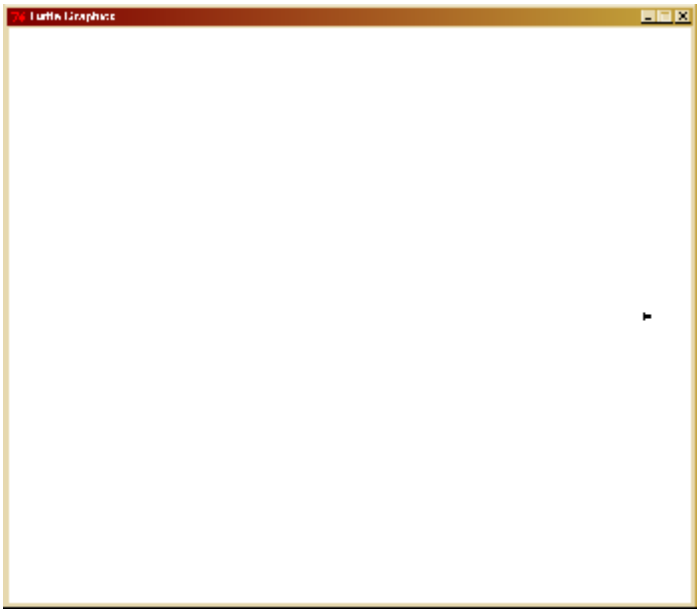
Εμφανίζεται ένα νέο παράθυρο με όνομα «Turtle Graphics»:

[8] Αν και αναφερόμαστε σε «εντολές γραφικών» καλό είναι να θυμόμαστε ότι στην πραγματικότητα πρόκειται για συναρτήσεις. Γιαυτό και ακόμα κι όσες δεν δέχονται κανένα όρισμα τις γράφουμε με παρενθέσεις στο τέλος του ονόματός τους.



Το παράθυρο είναι άδειο αλλά ακριβώς στο κέντρο του έχει σχεδιασμένη μια **γραφίδα** (ή **cursor**, ή «**χελώνα**») η οποία περιμένει τις εντολές μας για να σχεδιάσει γραφικά πάνω στο νέο παράθυρο. Η εντολή reset σβήνει ό,τι τυχόν έχουμε σχεδιάσει προηγουμένως στο παράθυρο και τοποθετεί τη γραφίδα στο κέντρο με προσανατολισμό προς τα δεξιά. Το κέντρο έχει συντεταγμένες (0,0). Μπορούμε να μετακινήσουμε τη γραφίδα σε κάποιες άλλες συντεταγμένες με την εντολή goto. Πχ.:

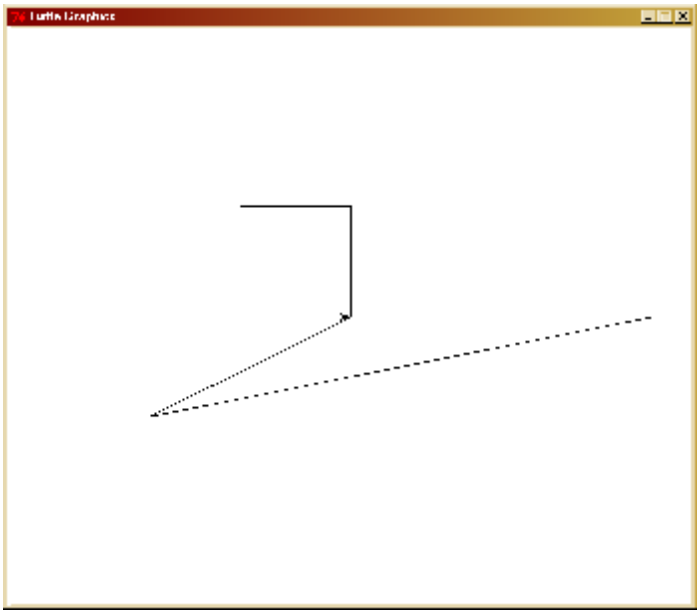
```
>>> goto(300,0)
```



Η γραφίδα μετακινήθηκε κατά 300 **pixels** (**εικονοστοιχεία** ή **ψηφίδες**) προς τα δεξιά αφήνοντας ως ίχνος μια ευθεία γραμμή. Μπορούμε να δώσουμε οποιοδήποτε σημείο ως παράμετρο στην εντολή goto (ακόμα κι αν αυτό βρίσκεται εκτός του παραθύρου). Με διαδοχικές εντολές goto μπορούμε να σχεδιάσουμε απλά σχήματα:

```
>>> goto(-200,-100)
>>> goto(0,0)
```

```
>>> goto(0,110)
>>> goto(-110,110)
>>> goto(-110,0)
>>> goto(0,0)
```

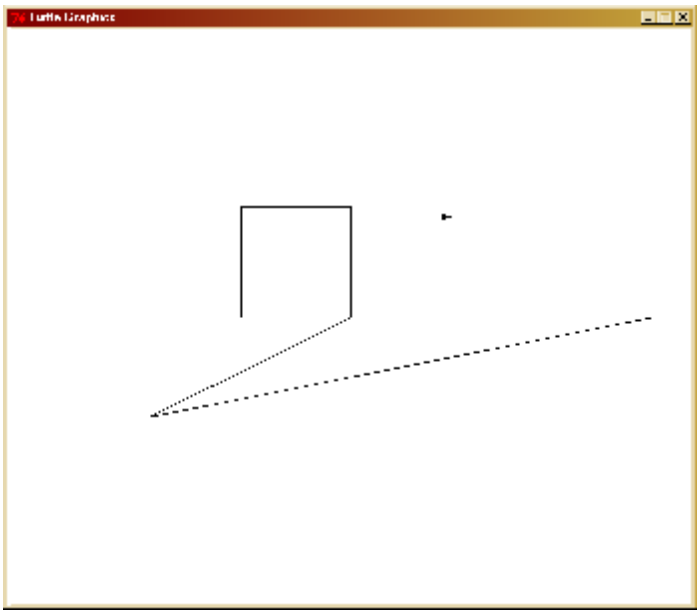


Αν η ταχύτητα με την οποία σχεδιάζονται οι γραμμές είναι αργή μπορείτε να δώσετε την εντολή:

```
>>> speed('fastest')
```
... και στο εξής όλες οι γραμμές θα σχεδιάζονται στη μέγιστη ταχύτητα.

Αν θέλουμε να μεταβούμε σε κάποιο σημείο χωρίς να αφήσουμε ίχνος, δίνουμε πρώτα την εντολή up, πχ:

```
>>> up()
>>> goto(100,100)
```



Και μόλις πάμε στο σημείο που θέλουμε δίνουμε:

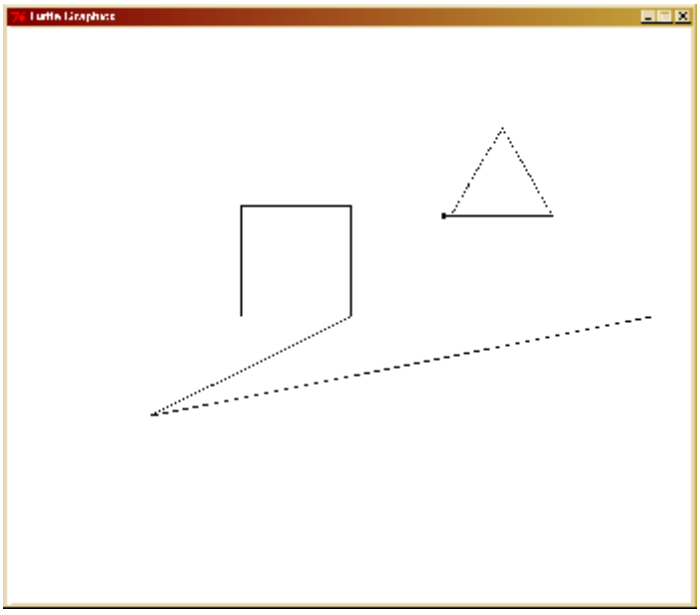
```
>>> down()
```

... για να αφήνουμε στο εξής ίχνος. Φανταστείτε ότι σηκώνουμε (up) το στυλό από το χαρτί μας, τον μετακινούμε (goto) σε ένα άλλο σημείο και μετά τον κατεβάζουμε (down) στο χαρτί για να συνεχίσουμε να σχεδιάζουμε.

Στον μέχρι τώρα σχεδιασμό χρησιμοποιήσαμε καρτεσιανές συντεταγμένες. Όμως η βιβλιοθήκη διαθέτει και συναρτήσεις για σχεδιασμό με πολικές συντεταγμένες (στις οποίες δίνουμε γωνία και απόσταση. Για παράδειγμα, να πώς σχεδιάζουμε ένα ισόπλευρο τρίγωνο πλευράς 100 pixels με πολικές συντεταγμένες:

```
>>> forward(100)
>>> left(120)
>>> forward(100)
>>> left(120)
>>> forward(100)
>>> left(120)
```

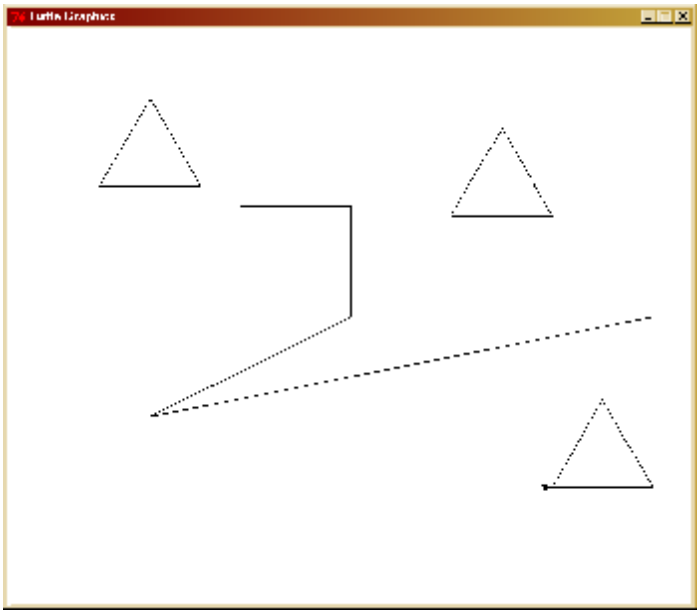
Η εντολή forward(100) μετακινεί τη γραφίδα 100 pixels προς την κατεύθυνση που δείχνει. Η εντολή left(120) στρίβει τη γραφίδα 120 μοίρες προς τα αριστερά. Δίνοντας 3 φορές τις ίδιες εντολές έχουμε φτιάξει ένα ισόπλευρο τρίγωνο και έχουμε καταλήξει στο σημείο από όπου ξεκινήσαμε, με τον αρχικό προσανατολισμό:



Το καλό όταν σχεδιάζουμε σε πολικές συντεταγμένες είναι ότι με τις ίδιες εντολές φτιάχνουμε ακριβώς τα ίδια σχήματα σε οποιοδήποτε σημείο της οθόνης κι αν βρισκόμαστε:

```
>>> up()
>>> goto(-250,130)
>>> down()
>>> forward(100)
>>> left(120)
```

```
>>> forward(100)
>>> left(120)
>>> forward(100)
>>> left(120)
>>> up()
>>> goto(200,-170)
>>> down()
>>> forward(100)
>>> left(120)
>>> forward(100)
>>> left(120)
>>> forward(100)
>>> left(120)
```



Για να κάναμε το ίδιο με καρτεσιανές συντεταγμένες (δηλαδή με εντολές goto) θα ήταν αρκετά πιο δύσκολο αφού θα έπρεπε να υπολογίσουμε τις ακριβείς συντεταγμένες των κορυφών των τριγώνων.

Η βιβλιοθήκη turtle περιέχει και άλλες χρήσιμες εντολές που μπορούμε να δούμε στο on-line help[9] της Python:

[9] Όπως μπορείτε να δείτε στο on-line help, κάποιες συναρτήσεις έχουν διπλά ονόματα, πχ. αντί για forward(100) μπορούμε να γράψουμε fd(100). Προς το παρόν αγνοήστε το πρόθεμα “turtle.” που εμφανίζεται στο help μπροστά από τα ονόματα των συναρτήσεων. Αυτό χρειάζεται αν καλέσουμε με διαφορετικό τρόπο την import και θα το εξηγήσουμε σε επόμενο εργαστήριο.

Κάποιες από αυτές θα χρησιμοποιήσουμε στη συνέχεια. Προς το παρόν ξέρουμε αρκετές εντολές για να κάνουμε κάτι χρήσιμο όπως τον σχεδιασμό γραφικών παραστάσεων μαθηματικών συναρτήσεων.

6.5 Γραφικές Παραστάσεις

Στην ενότητα 6.2 είδαμε πώς ορίζουμε μαθηματικές συναρτήσεις με το παράδειγμα της $f(x)= 0.0001*x^3+0.015*x^2-1.14*x-2.8$:

```
>>> def f(x):
      return 0.0001*x**3+0.015*x**2-1.14*x-2.8
```

Τώρα θα δούμε πώς να κάνουμε γραφικές παραστάσεις συναρτήσεων με τις εντολές που μάθαμε. Δίνουμε τις εντολές:

```
>>> reset()
>>> speed('fastest')
>>> forward(300)
>>> goto(0,0)
>>> left(90)
>>> forward(300)
>>> goto(0,0)
>>> left(90)
>>> forward(300)
>>> goto(0,0)
>>> left(90)
>>> forward(300)
>>> up()
>>> goto(-300,f(-300))
>>> down()
```

Οι παραπάνω εντολές σχεδιάζουν τους δυο άξονες και μετακινούν τη γραφίδα στο σημείο (-300,f(-300)) από όπου θα ξεκινήσουμε τη σχεδίαση. Ας χρησιμοποιήσουμε κόκκινο χρώμα για τη σχεδίαση. Γράφουμε:

```
>>> color('red')
```

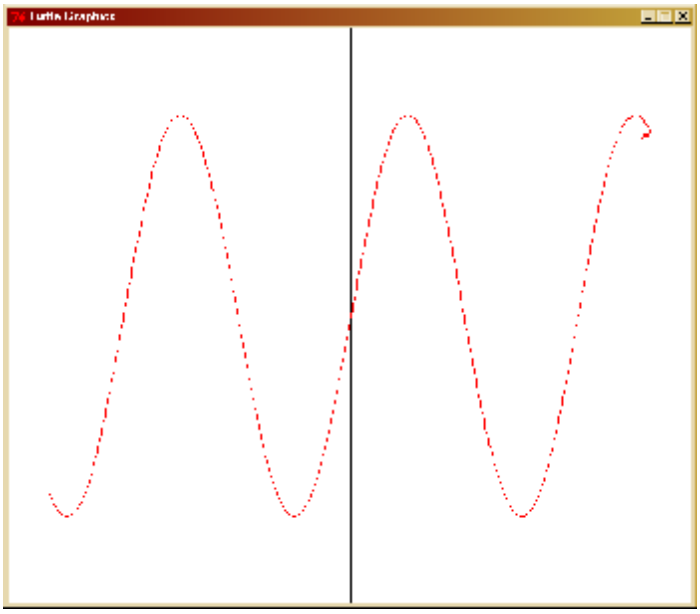
… και στη συνέχεια:

```
>>> for x in range(-300,300):
      goto(x,f(x))
```

Και αυτή είναι η γραφική παράσταση της f(x) στο διάστημα [-300,300].

Όμως η συγκεκριμένη συνάρτηση «έτυχε» να φαίνεται καλά στο παράθυρο. Πολύ συχνά η συνάρτηση χρειάζεται αλλαγή κλίμακας είτε στον άξονα x είτε στον άξονα y για να εμφανιστεί σωστά. Φανταστείτε για παράδειγμα να προσπαθούσαμε να απεικονίσουμε τη συνάρτηση του ημιτόνου, sin(x). Μια που το πλάτος της θα ήταν μόλις ένα pixel και η περίοδος 6 pixels, δεν θα βλέπαμε τίποτε. Χρησιμοποιώντας αλλαγή κλίμακας όμως, μπορούμε να γράψουμε κάτι τέτοιο:

```
>>> from math import *
>>> cy=200
>>> cx=36.
>>> x=-300
>>> up()
>>> goto(x,cy*sin(x/cx))
>>> down()
>>> for x in range(-300,300):
      goto(x,cy*sin(x/cx))
```



Τα cx, cy είναι οι μεγεθύνσεις κατά τους άξονες των x και y αντίστοιχα. Πλέον το πλάτος είναι 200 pixels και η περίοδος 2π*36=226 pixels και η γραφική παράσταση της συνάρτησης είναι ευδιάκριτη στο παράθυρο.

6.6 Αλγοριθμική Ζωγραφική

Η βιβλιοθήκη turtle διαθέτει τη συνάρτηση circle(r) η οποία σχεδιάζει έναν κύκλο ακτίνας r. Ας φτιάξουμε ένα πρόγραμμα που σχεδιάζει 100 κύκλους τυχαίων ακτίνων σε τυχαία σημεία με τυχαία χρώματα κι ας δούμε πόσο ωραίο θα φανεί.

Θα χρειαστούμε τη βιβλιοθήκη random που περιέχει διάφορες συναρτήσεις για την παραγωγή (ψευδο)τυχαίων αριθμών:

```
>>> from random import *
```

Η βιλιοθήκη random περιέχει μια συνάρτηση random() η οποία κάθε φορά που καλείται επιστρέφει έναν τυχαίο πραγματικό αριθμό μεταξύ 0 και 1:

```
>>> print random()
0.178361201055
>>> print random()
0.671443452792
```

Περιέχει επίσης και τη συνάρτηση randint(low,high) η οποία κάθε φορά που καλείται επιστρέφει έναν ακέραιο αριθμό στο διάστημα [low,high]:

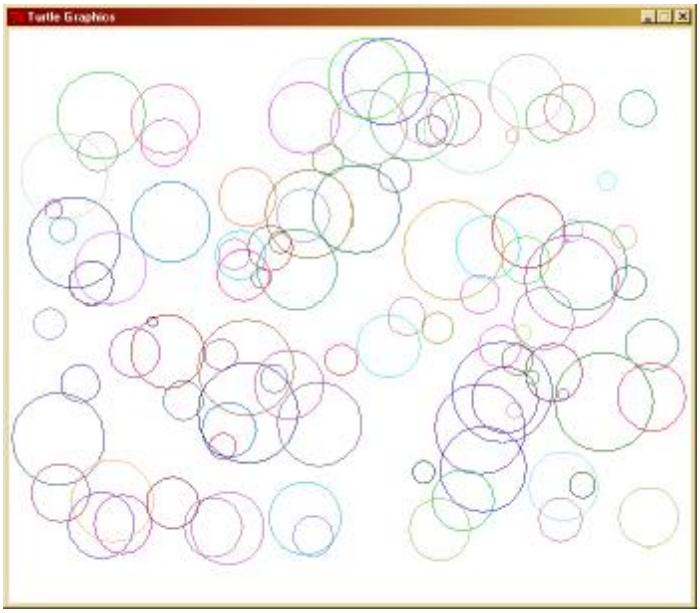
```
>>> print randint(10,100)
47
```

Να πώς τις χρησιμοποιούμε:

```
>>> for i in range(100):
```

```
up()
x=randint(-300,300)
y=randint(-250,200)
goto(x,y)
down()
color(random(),random(),random())
r=randint(5,50)
circle(r)
```

Στο παραπάνω τμήμα κώδικα μετακινούμε τη γραφίδα σε ένα τυχαίο σημείο (x,y) όπου το x παίρνει τιμές στο διάστημα [-300,300] και το y παίρνει τιμές στο διάστημα [-250,200]. Στη συνέχεια επιλέγουμε ένα τυχαίο χρώμα μέσω της συνάρτησης color(R,G,B). Η συνάρτηση αυτή είναι διαφορετική από την color που χρησιμοποιήσαμε στις γραφικές παραστάσεις. Αντί να καθορίσουμε το χρώμα με το όνομά του, δίνουμε τρία ορίσματα που αντιστοιχούν στις ποσότητες των τριών βασικών χρωμάτων (κόκκινο, πράσινο, μπλε) που θα αναμιχθούν για να προκύψει το χρώμα του κύκλου. Οι αριθμοί είναι πραγματικοί στο διάστημα [0,1] και οι συνδυασμοί τους παράγουν όλα τα δυνατά χρώματα. Τέλος καθορίζεται η ακτίνα του κύκλου r ως τυχαίος ακέραιος στο διάστημα [5,50] και δίνεται εντολή να σχεδιαστεί ο κύκλος. Η διαδικασία επαναλαμβάνεται 100 φορές αφού βρίσκεται στο εσωτερικό της for. Να το αποτέλεσμα:



Τι θα αλλάζαμε στον κώδικα αν θέλαμε αντί για κύκλους να σχεδιάσουμε τρίγωνα; Προφανώς θα έπρεπε να αντικαταστήσουμε την συνάρτηση circle(r) με κάποια άλλη που σχεδιάζει τρίγωνα. Αλλά αν ψάξουμε στο help της βιβλιοθήκης turtle, δεν θα βρούμε τέτοια συνάρτηση. Μπορούμε όμως να ορίσουμε εμείς μια. Ήδη στην ενότητα 6.4 είδαμε πώς να φτιάχνουμε ισόπλευρα τρίγωνα. Χρησιμοποιώντας τις ίδιες εντολές να μια συνάρτηση που σχεδιάζει ένα ισόπλευρο τρίγωνο πλευράς r:

```
>>> def triangle(r):
      forward(r)
      left(120)
      forward(r)
      left(120)
      forward(r)
```

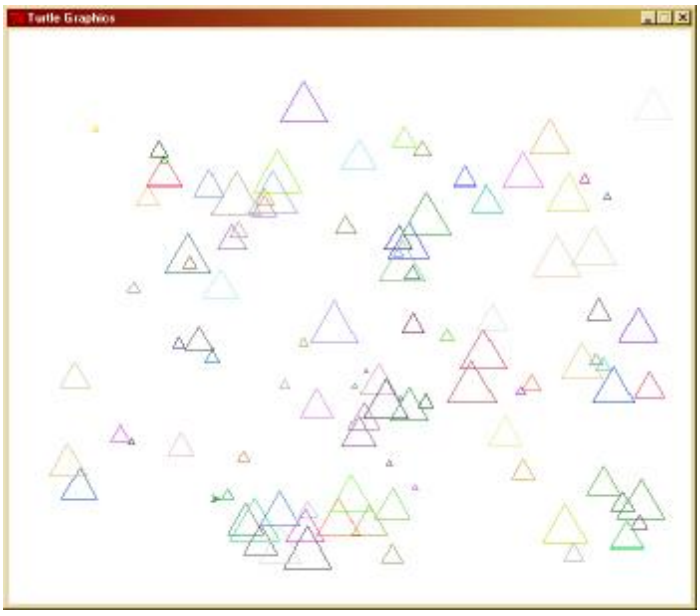
```
      left(120)
```

Και τώρα:

```
>>> clear()
```

Καθαρίζουμε πρώτα την οθόνη για να μη ζωγραφίσουμε πάνω στους κύκλους. Η συνάρτηση clear() αντίθετα με τη reset() καθαρίζει την οθόνη χωρίς να αλλάξει τη θέση της γραφίδας, την ταχύτητά της ή το χρώμα.

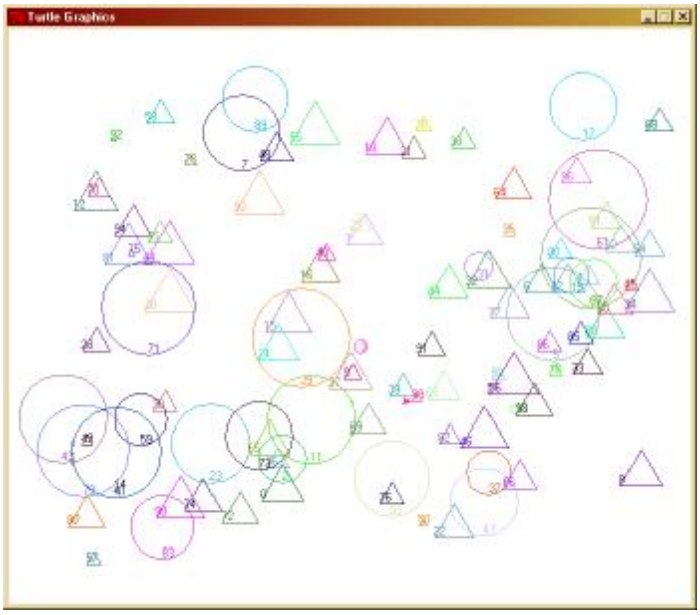
```
>>> for i in range(100):
      up()
      x=randint(-300,300)
      y=randint(-250,200)
      goto(x,y)
      down()
      color(random(),random(),random())
      r=randint(5,50)
      triangle(r)
```



Και τώρα ας τα συνδυάσουμε. Έστω ότι θέλουμε να ζωγραφίσουμε 100 σχήματα που θα αριθμούνται από 0 έως 99. Αν ο αύξων αριθμός είναι πρώτος (θα ελέγχεται με τη συνάρτηση is_prime που ορίσαμε νωρίτερα) το σχήμα που θα σχεδιάζεται θα είναι κύκλος. Διαφορετικά θα σχεδιάζεται τρίγωνο. Για να ελέγξουμε την ορθότητα του αλγορίθμου θα γράφουμε δίπλα σε κάθε σχήμα τον αύξοντα αριθμό του. Αυτό το κάνει η συνάρτηση write(). Ακολουθεί ο κώδικας:

```
>>> clear()
>>> for i in range(100):
      up()
      x=randint(-300,300)
      y=randint(-250,200)
      goto(x,y)
      down()
      color(random(),random(),random())
      r=randint(5,50)
```

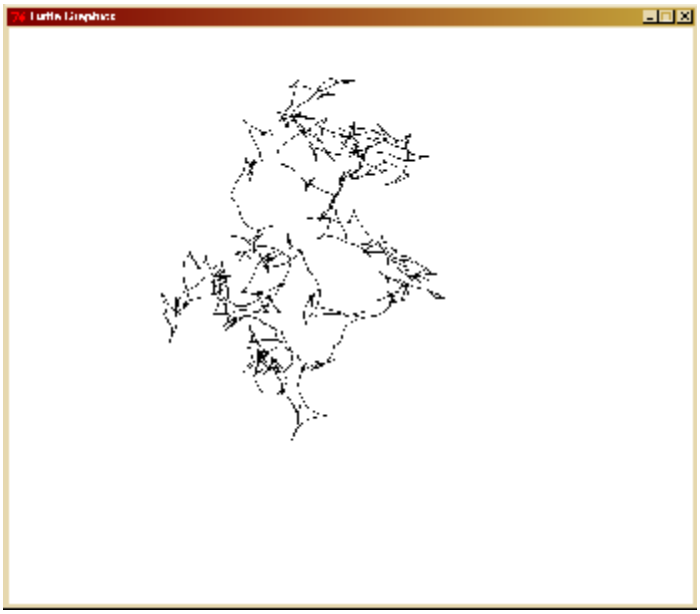
```
if is_prime(i)=='yes':
      circle(r)
else:
      triangle(r)
write(i)
```



Αν παρακολουθήσατε τον τρόπο με τον οποίο σχεδιάζονται οι κύκλοι, ίσως παρατηρήσατε ότι σχεδιάζονται πάντα με ανθωρολογιακή φορά, εφαπτόμενοι στο διάνυσμα που ορίζει η γραφίδα στο σημείο που βρίσκεται. Η συνάρτηση circle μπορεί να δεχτεί και δεύτερο όρισμα που καθορίζει το τόξο σε μοίρες που θα σχεδιαστεί. Για παράδειγμα, η συνάρτηση:
>>> circle(50,360)
… θα σχεδιάσει έναν πλήρη κύκλο (τόξο 360 μοιρών) με ακτίνα 50, όμως η:
>>> circle(50,180)
... θα σχεδιάσει ένα ημικύκλιο (τόξο 180 μοιρών). Επιπλέον, η circle μπορεί να δεχτεί και αρνητικούς αριθμούς. Αρνητική γωνία σημαίνει ότι θα σχεδιάσει το τόξο με ωρολογιακή φορά. Αρνητική ακτίνα σημαίνει ότι θα φτιάξει το κατοπτρικό τόξο (υπάρχουν πάντα δυο εφαπτόμενα τόξα στο δεδομένο σημείο της γραφίδας). Δείτε για παράδειγμα αυτό το τμήμα κώδικα:

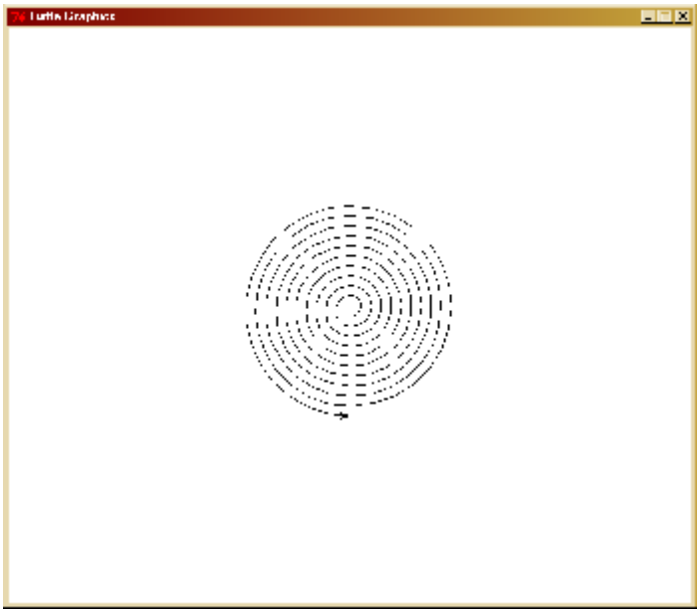
```
>>> reset()
>>> speed('fastest')
>>> for i in range(1000):
      circle(randint(-40,40),randint(-45,45))
```

Σχεδιάζει 1000 διαδοχικά τυχαία τόξα, με ακτίνες που δεν ξεπερνούν τα 40 pixels και γωνίες που δεν ξεπερνούν τις 45 μοίρες:



Μπορούμε να χρησιμοποιήσουμε διαδοχικά τόξα για να φτιάξουμε σπείρες. Για παράδειγμα:

```
>>> reset()
>>> for i in range(20):
      circle(10+5*i,180)
```



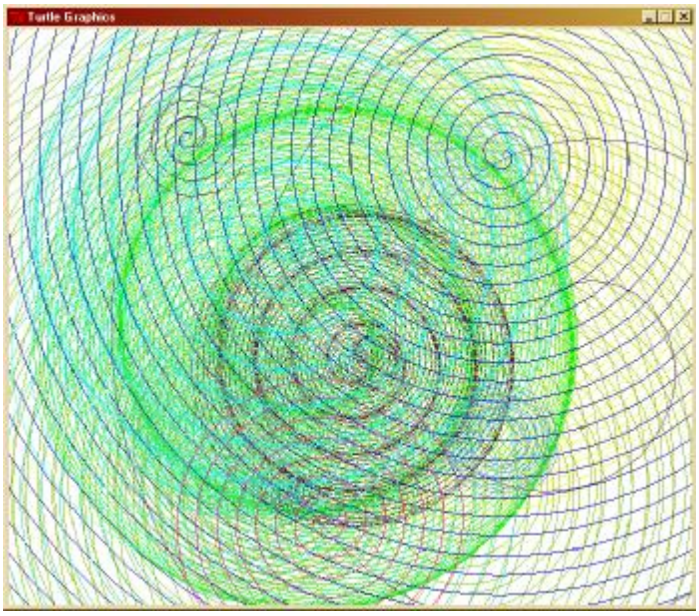
Εδώ σχεδιάσαμε 20 ημικύκλια. Το εσωτερικό έχει ακτίνα 10 pixels και κάθε επόμενο έχει ακτίνα 5 pixels μεγαλύτερη από ότι το προηγούμενο.

Αν θέλουμε να παίξουμε με τις σπείρες, βολεύει να φτιάξουμε μια συνάρτηση:

```
>>> def spiral(x,y,arc,times,inner,increment):
      up()
      goto(x,y)
      down()
      for i in range(times):
            circle(inner+increment*i,arc)
```

Η συνάρτηση spiral σχεδιάζει μια σπείρα ξεκινώντας από το σημείο (x,y) και φτιάχνοντας times το πλήθος τόξα, καθένα γωνίας arc μοιρών. Το πρώτο θα έχει ακτίνα inner και η ακτίνα κάθε επόμενου θα αυξάνεται κατά increment. Και τώρα μπορούμε να φτιάξουμε π.χ. 20 τυχαίες σπείρες:

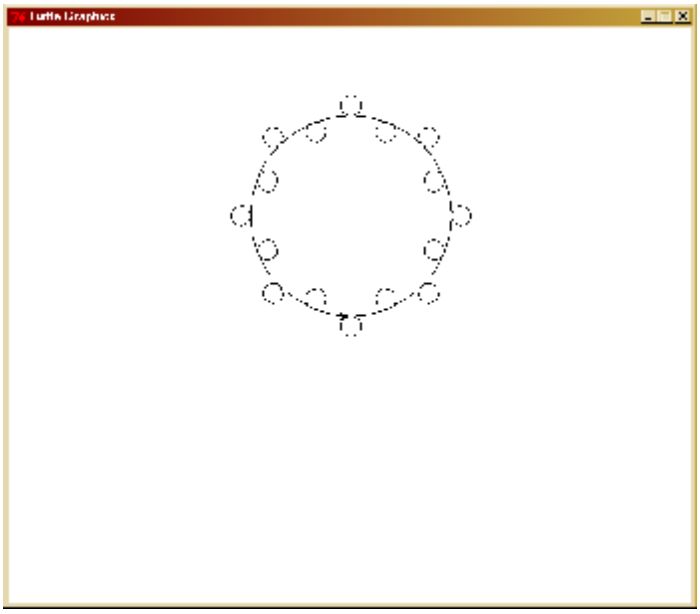
```
>>> reset()
>>> speed('fastest')
>>> for i in range(20):
      color(random(),random(),random())
      x=randint(-200,200)
      y=randint(-200,200)
      arc=randint(-360,360)
      spiral(x,y,arc,randint(10,100),randint(-30,30),randint(-10,10))
```



Με τα τόξα και λίγη προσοχή στον προγραμματισμό μπορούμε να φτιάξουμε αρκετά ωραία γραφικά. Για παράδειγμα, προσπαθήστε να καταλάβετε πώς ο παρακάτω κώδικας:

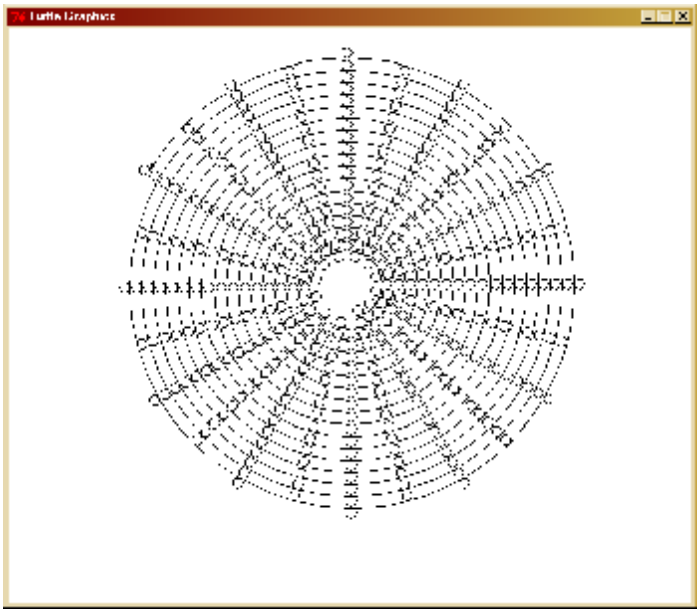
```
>>> reset()
>>> for i in range(8):
      circle(100,22.5)
      circle(10)
      circle(100,22.5)
      circle(-10)
```

... σχεδιάζει το ακόλουθο σχήμα:



Και μετά συνδυάστε τον με μια σπείρα όπως:

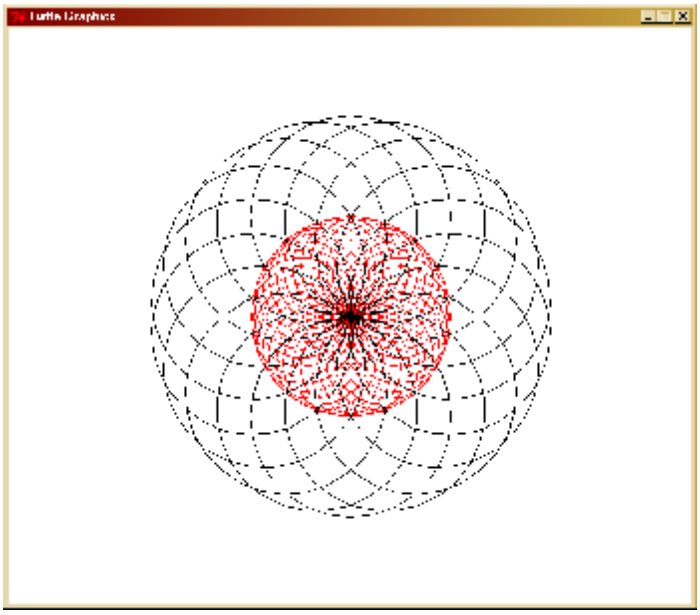
```
>>> reset()
>>> for i in range(200):
      circle(30+i,15)
      circle(5)
      circle(30+i,15)
      circle(-5)
```



Δείτε επίσης έναν άλλον σχετικά απλό τρόπο να φτιάχνουμε εντυπωσιακά σχήματα, χρησιμοποιώντας μόνο κύκλους:

```
>>> reset()
>>> speed('fastest')
>>> color('red')
>>> for i in range(36):
      circle(50)
      left(10)
```

```
>>> color('black')
>>> for i in range(18):
      circle(100)
      left(20)
```

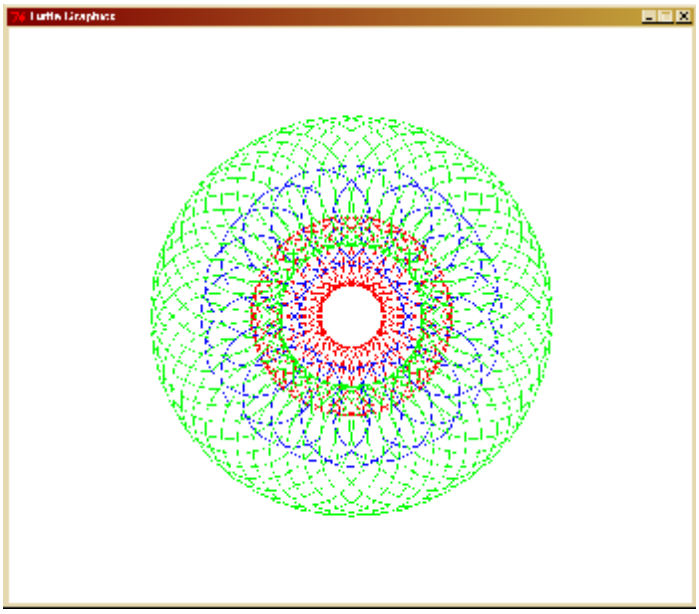


Και πάλι, μπορούμε να εξελίξουμε τον αλγόριθμο σε συνάρτηση που θα φτιάχνει «μαργαρίτες» με τις παραμέτρους που θα δίνουμε:

```
>>> def daisy(inner, outer, number, colour):
      angle=360.0/number
      radius=-(outer-inner)/2.
      color(colour)
      up()
      goto(0,0)
      right(90)
      forward(inner)
      left(90)
      for i in range(number):
            down()
            circle(radius)
            up()
            circle(inner,angle)
```

Η ιδέα είναι η εξής: δυο ομόκεντροι κύκλοι με ακτίνες inner και outer ορίζουν ένα «δακτυλίδι» μέσα στο οποίο η συνάρτηση σχεδιάζει number το πλήθος ισαπέχοντες κύκλους χρώματος colour. Και τώρα μπορούμε να γράψουμε, πχ.:

```
>>> reset()
>>> speed('fastest')
>>> daisy(30,100,36,'red')
>>> daisy(50,150,20,'blue')
>>> daisy(70,200,50,'green')
```



6.7 Ασκήσεις

Αφού μελετήσετε τα παραδείγματα των ενοτήτων που προηγήθηκαν, εκπονήστε τις παρακάτω ασκήσεις. Αναρτήστε τα αρχεία σας ως ένα ενιαίο συμπιεσμένο αρχείο. Δώστε ιδιαίτερη προσοχή στην πληρότητα και την εμφάνιση του κώδικά σας.

Άσκηση #1

Επισκεφθείτε την ιστοσελίδα του Project Euler (http://projecteuler.net).

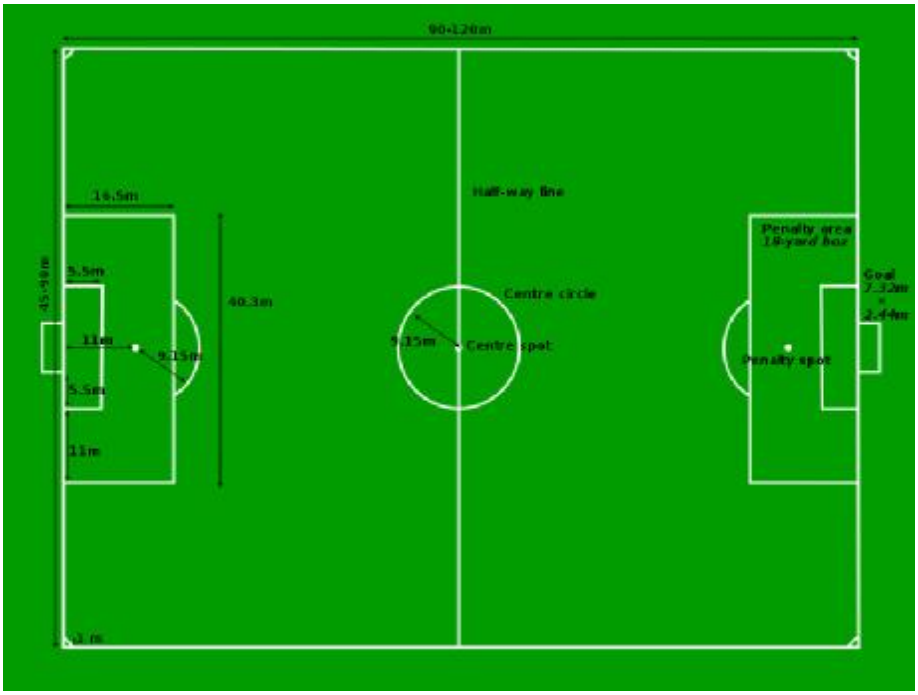
Επιλέξτε δύο προβλήματα[10] από εκεί και λύστε τα με τη βοήθεια της Python. Χρησιμοποιήστε ξεχωριστό αρχείο κώδικα για κάθε πρόβλημα και μέσα σε αυτό γράψτε με σχόλια τον αριθμό του προβλήματος, την εκφώνησή του (copy-paste από το site) και το αποτέλεσμα[11] που βρήκατε.

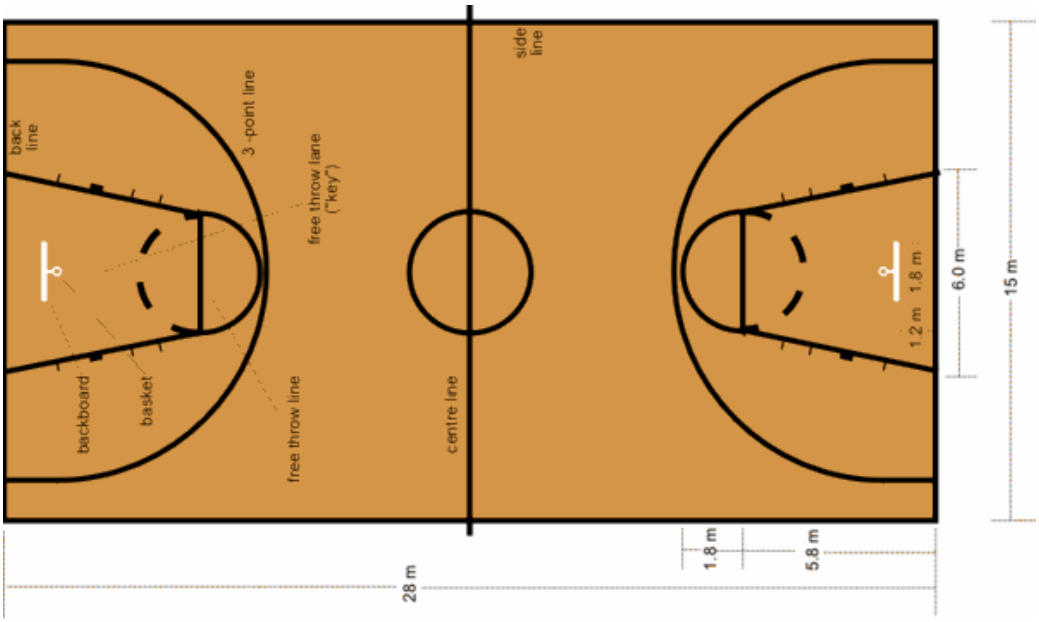
[10] Τα προβλήματα με μικρό αύξοντα αριθμό είναι τα πιο εύκολα.

[11] Μπορείτε αν θέλετε να ελέγξετε αν η απάντηση που βρήκατε είναι σωστή, μέσω του ίδιου site.

Άσκηση #2

Χρησιμοποιήστε τη βιβλιοθήκη turtle της Python για να σχεδιάσετε την κάτοψη ενός γηπέδου ποδοσφαίρου ή μπάσκετ. Δεν μας ενδιαφέρουν οι ακριβείς αναλογίες, ούτε τα χρώματα, αλλά προσπαθήστε να πετύχετε σωστά τα σχήματα (παραλληλόγραμμα, ημικύκλια, κλπ). Μαζί με τον κώδικα δώστε και την εικόνα (screen capture) του γηπέδου όπως το σχεδίασε το πρόγραμμά σας.





Εργαστήριο 7: Τρίτη Άσκηση Προγραμματισμού Python

7.1 Γενικά

Στην άσκηση αυτή θα εξεταστούν μερικές πιο **σύνθετες δομές δεδομένων** της Python, συγκεκριμένα οι **λίστες** (**lists**) και τα **λεξικά** (**dictionaries**) και θα χρησιμοποιηθούν σε παραδείγματα παραγωγής ήχων και μουσικής. Θα γίνει επίσης αναφορά στις **μεθόδους** (**methods**) και θα εξηγηθεί ο συσχετισμός τους με τις συναρτήσεις.

7.2 Λίστες

Στην προηγούμενη άσκηση είδαμε για πρώτη φορά τη χρήση λιστών. Οι λίστες είναι **δομές δεδομένων** οι οποίες μπορούν να περιέχουν οποιοδήποτε πλήθος στοιχείων, για παράδειγμα:

```
>>> a=[14,17,11,22,19,11,33]
>>> print a
[14, 17, 11, 22, 19, 11, 33]
```

Από το παραπάνω παράδειγμα είναι εμφανές το πλεονέκτημα που αποκομίζουμε όταν τοποθετούμε τα δεδομένα μας σε δομές δεδομένων όπως οι λίστες: μπορούμε να τα φυλάξουμε όλα μαζί σε μια μεταβλητή και μετά να τα χειριστούμε κατά βούληση.

Προσέξτε ότι σε μια λίστα μπορεί να επαναλαμβάνονται κάποια στοιχεία. Επιπλέον, τα στοιχεία μιας λίστας δεν είναι απαραίτητο να είναι του ίδιου τύπου. Μπορεί σε μια λίστα κάποια στοιχεία να είναι αριθμοί, κάποια αλφαριθμητικά, κάποια άλλες λίστες, κλπ. Πχ.:

```
>>> q=[34, 'hello', -12.17, ['a', 'b']]
>>> print q
[34, 'hello', -12.17, ['a', 'b']]
```

Η λίστα q περιέχει τέσσερα στοιχεία: τον ακέραιο 34, το αλφαριθμητικό ‘hello’, τον πραγματικό αριθμό –12.17 και τη λίστα ['a', 'b'].

Το μήκος μιας λίστας επιστρέφει η συνάρτηση len():

```
>>> print len(a), len(q)
7 4
```

Μπορούμε να επιλέξουμε κάποιο συγκεκριμένο στοιχείο από μια λίστα δίνοντας το όνομα της λίστας και τη θέση του στοιχείου με τον εξής τρόπο:

```
>>> print a[0], q[1]
14 hello
```

Το 1ο στοιχείο μιας λίστας θεωρείται ότι βρίσκεται στη θέση 0, το 2ο στη θέση 1, κ.ο.κ. Μπορούμε να επιλέξουμε και αρίθμηση από το τέλος, αν χρησιμοποιήσουμε αρνητικούς αριθμούς: το –1 αντιστοιχεί στην τελευταία θέση, το –2 στην προτελευταία, κ.ο.κ. Π.χ.:

```
>>> print a[-2], q[-1]
11 ['a', 'b']
```

Μπορούμε ακόμα να επιλέξουμε τμήμα λίστας (υπολίστα) δίνοντας τις θέσεις των ακραίων στοιχείων της ως εξής:

```
>>> print a, a[2:5]
[14, 17, 11, 22, 19, 11, 33] [11, 22, 19]
```

Ενώνουμε δυο λίστες με το σύμβολο (τελεστή) +:

```
>>> a=[1,2]
>>> b=[34,56]
>>> c=a+b
>>> print c
[1, 2, 34, 56]
```

Ενώ ο τελεστής * επαναλαμβάνει μια λίστα όσες φορές ορίσουμε (εδώ 3):

```
>>> d=a*3
>>> print d
[1, 2, 1, 2, 1, 2]
```

Οι λίστες έχουν κι άλλες πολλές δυνατότητες, κάποιες από τις οποίες θα δούμε αργότερα, όμως προς το παρόν μπορούμε να εφαρμόσουμε όσα γνωρίζουμε για τις λίστες στην παραγωγή ήχων με τον υπολογιστή μας.

7.3 Μουσική με Λίστες

Όπως και στα γραφικά, έτσι και στη μουσική χρειάζεται να φορτώσουμε μια βιβλιοθήκη (module) συναρτήσεων. Και πάλι, υπάρχουν πολλές σχετικές βιβλιοθήκες και συναρτήσεις, όμως εμείς θα ασχοληθούμε μόνο με την απλούστερη συνάρτηση (λέγεται Beep()) από τη βιβλιοθήκη winsound η οποία είναι άμεσα διαθέσιμη από την βασική εγκατάσταση της Python στα Windows:

```
>>> from winsound import Beep
```

Μια που χρειαστήκαμε όχι όλη τη βιβλιοθήκη αλλά μόνο μια συνάρτηση, με την εντολή import εισάγαμε μόνο αυτή τη συνάρτηση, που έχει το όνομα Beep. Προσέξτε ότι τη γράφουμε με Β κεφαλαίο και όχι πεζό. Η Python κάνει διάκριση μεταξύ πεζών και κεφαλαίων χαρακτήρων και γιαυτό προκειμένου να αναφερθούμε σωστά σε μια συνάρτηση (ή σε μια μεταβλητή) πρέπει να γράφουμε το όνομά της ακριβώς όπως έχει οριστεί. Επίσης πρέπει να τονίσουμε ότι η βιβλιοθήκη winsound είναι διαθέσιμη

μόνο σε συστήματα Windows και μάλιστα δεν δουλεύει το ίδιο καλά σε όλες τις εκδόσεις των Windows. Αν τυχόν δεν δουλέψει στο σύστημά σας δεν θα ακούτε τους ήχους της άσκησης, όμως θα καταλάβετε πώς λειτουργεί ο κώδικας των παραδειγμάτων γιατί εκτός από τους ήχους θα φροντίσουμε να τυπώνουμε κατάλληλα μηνύματα στην οθόνη. Ας δοκιμάσουμε αν δουλεύει σωστά:

```
>>> Beep(1200,500)
```

Αν όλα πήγαν καλά θα πρέπει να ακούσουμε έναν ήχο από το μεγάφωνο του υπολογιστή μας. Ο ήχος θα έχει συχνότητα 1200Hz και διάρκεια μισό δευτερόλεπτο (500msec). Το πρώτο όρισμα στη συνάρτηση Beep είναι ένας ακέραιος στο διάστημα [37, 32767] που δηλώνει τη συχνότητα σε Hertz του ήχου που θα παραχθεί. Το δεύτερο όρισμα δηλώνει τη χρονική διάρκεια του ήχου σε milliseconds. Δυνατότητα να καθορίσουμε την ένταση του ήχου ή να συνδυάσουμε ήχους σε συγχορδίες δεν παρέχει η συνάρτηση Beep, αλλά ακόμα κι έτσι μπορούμε να κάνουμε αρκετά πράγματα. Πχ.:

```
>>> from random import *
>>> for i in range(20):
      freq=randint(1200,20000)
      print freq,"Hz"
      Beep(freq,200)

10594 Hz
4543 Hz
15006 Hz
...
14551 Hz
6327 Hz
9590 Hz
```

Βέβαια αυτό απέχει πολύ από το να μπορεί να χαρακτηριστεί «μουσική». Είναι απλώς τυχαίες συχνότητες ίδιας διάρκειας που διαδέχονται η μία την άλλη. Για να πετύχουμε κάτι πλησιέστερο στη μουσική, θα πρέπει τουλάχιστον να επιλέξουμε εκείνες τις συχνότητες που αντιστοιχούν στις νότες της μουσικής. Ο τρόπος επιλογής είναι αλγοριθμικός:

Ως βασική συχνότητα στη μουσική έχει οριστεί η συχνότητα των 440 Hz. Αντιστοιχεί στη νότα Λα που βρίσκεται περίπου στη μέση ενός πιάνου. Από αυτή τη συχνότητα προκύπτουν όλες οι υπόλοιπες. Το αμέσως επόμενο Λα έχει ακριβώς διπλάσια συχνότητα (880 Hz) και λέμε ότι βρίσκεται μια οκτάβα ψηλότερα. Το αμέσως προηγούμενο έχει ακριβώς μισή συχνότητα (220 Hz) και λέμε ότι βρίσκεται μια οκτάβα χαμηλότερα. Να πώς ακούγονται:

```
>>> List=[220,440,880]
>>> for freq in List:
      print freq,"Hz"
      Beep(freq,500)
```

Προσέξτε ότι αν και η απόσταση της $3^{ης}$ νότας από τη $2^{η}$ είναι διπλάσια (880-440=440 Hz) από ότι η απόσταση της $1^{ης}$ νότας από την $2^{η}$ (440-220=220 Hz), εντούτοις οι δυο ακραίες νότες ακούγονται σα να «ισαπέχουν» από τη μεσαία. Αυτό

συμβαίνει επειδή η ανθρώπινη αντίληψη για τους ήχους είναι λογαριθμική κι όχι γραμμική. Πράγματι, προσέξτε ότι οι λογαριθμικές τους αποστάσεις είναι ίδιες:

```
>>> from math import log
>>> print log(List[2])-log(List[1])
0.69314718056
>>> print log(List[1])-log(List[0])
0.69314718056
```

Και με την ίδια λογική, συνεχώς διπλασιάζοντας τη συχνότητα πηγαίνουμε σε όλο και υψηλότερες νότες Λα, συνεχώς υποδιπλασιάζοντας πηγαίνουμε σε νότες Λα σε όλο και χαμηλότερες οκτάβες. Να πώς μπορούμε να βάλουμε σε μια λίστα όλες τις νότες Λα που μπορεί να παράγει η συνάρτηση Beep():

```
>>> La_List=[]
>>> F=440
>>> while F<=32767:
      La_List=La_List+[F]
      F=F*2
```

Ξεκινήσαμε με μια λίστα (La_List) αρχικά κενή και μια μεταβλητή (F) με αρχική τιμή 440. Στη συνέχεια χρησιμοποιήσαμε μια επαναληπτική διαδικασία με την εντολή while. Την εντολή while είναι η πρώτη φορά που τη χρησιμοποιούμε. Εκτελεί επαναλήψεις όπως η for, μόνο που οι επαναλήψεις της διαρκούν όσο ισχύει η συνθήκη που γράφουμε δίπλα στη while. Εδώ γράψαμε F<=32767, επομένως όσο η F (που τώρα έχει τιμή 440 αλλά προφανώς στη συνέχεια θα αλλάξει) έχει τιμή μικρότερη ή ίση από 32767 οι εσωτερικές εντολές της while θα επαναλαμβάνονται. Και οι εσωτερικές εντολές λένε να προστεθεί η F στο τέλος της λίστας και στη συνέχεια να διπλασιαστεί. Αν τυπώσουμε τη λίστα θα δούμε:

```
>>> print La_List
[440, 880, 1760, 3520, 7040, 14080, 28160]
```

Αυτές είναι οι συχνότητες Λα από 440 Hz και πάνω. Με παρόμοιο τρόπο προσθέτουμε και τις χαμηλότερες οκτάβες:

```
>>> F=440
>>> while F>=37:
      if not(F in La_List):
            La_List=[F]+La_List
      F=F/2
```

Εδώ για να μην προστεθεί δυο φορές στη λίστα η συχνότητα 440, γίνεται πρώτα ένας έλεγχος if. Αν η F δεν περιέχεται ήδη στη λίστα, τότε μόνο προστίθεται. Προσέξτε επίσης ότι, αντίθετα από ότι προηγουμένως, η F προστίθεται στην αρχή της λίστας. Και τελικά:

```
>>> print La_List
[55, 110, 220, 440, 880, 1760, 3520, 7040, 14080, 28160]
```

Αυτές είναι όλες οι νότες Λα που μπορεί να παράγει η Beep(). Ας τις ακούσουμε:

```
>>> def play_list(L,Duration):
      for freq in L:
```

```
        print freq, "Hz"
        Beep(freq,Duration)
```

Επειδή στη συνέχεια θα ακούσουμε πολλές λίστες συχνοτήτων, για να διευκολυνθούμε ορίσαμε μια συνάρτηση που δέχεται ως ορίσματα μια λίστα συχνοτήτων L και μια μεταβλητή Duration και παίζει τις νότες της λίστας με διάρκεια Duration msec την κάθε μια. Και τώρα αρκεί να γράψουμε:

```
>>> play_list(La_List,500)
55 Hz
110 Hz
220 Hz
440 Hz
880 Hz
1760 Hz
3520 Hz
7040 Hz
14080 Hz
28160 Hz
```

Κατά πάσα πιθανότητα την τελευταία νότα δεν θα μπορέσετε να την ακούσετε. Το ανθρώπινο αυτί ακούει ήχους περίπου στην περιοχή 20–22000 Hz. Ήχοι με συχνότητα πάνω από 22000 Hz λέγονται **υπέρηχοι**. Ήχοι κάτω από 20 Hz λέγονται **υπόηχοι**.

Βέβαια, η μουσική δεν αποτελείται μόνο από νότες Λα. Ανάμεσα σε δυο διαδοχικά πλήκτρα Λα στο πιάνο υπάρχουν άλλα 11 πλήκτρα, δηλαδή κάθε οκτάβα έχει συνολικά 12 διαφορετικές νότες. Με δεδομένο ότι κι αυτές ισαπέχουν μεταξύ τους με λογαριθμικό τρόπο, μπορούμε να υπολογίσουμε τη συχνότητα κάθε επόμενου πλήκτρου από κάθε προηγούμενο όχι διπλασιάζοντας αλλά πολλαπλασιάζοντας με τον συντελεστή $\delta = 2^{(1/12)} = 1.05946309436$.

Ο συντελεστής αυτός προκύπτει πολύ εύκολα, αν σκεφτούμε ως εξής: Ξεκινώντας από τη νότα Λα = 440 Hz, πολλαπλασιάζοντας με δ, πηγαίνουμε στην επόμενη (Λα# = δ*440). Ξαναπολλαπλασιάζοντας με δ, πηγαίνουμε στη μεθεπόμενη (Σολ = δ*δ*440) κ.ο.κ. Αν πολλαπλασιάσουμε ακριβώς 12 φορές θα έχουμε πέσει πάνω στη Λα της επόμενης οκτάβας που έχει τη διπλάσια συχνότητα από την αρχική. Επομένως $\delta^{12}*440=2*440$ και $\delta = 2^{(1/12)}$. Να η επαλήθευση με κώδικα Python:

```
>>> A=440.
>>> d=2.**(1./12.)
>>> print d
1.05946309436

>>> L=[A]
>>> for i in range(12):
      A=A*d
      L=L+[A]

>>> print L
[440.0, 466.1637615180899, 493.8833012561241, 523.2511306011974,
554.3652619537443, 587.3295358348153, 622.253967444162,
659.2551138257401, 698.456462866008, 739.988845423269,
783.9908719634989, 830.6093951598906, 880.0000000000003]
```

Η λίστα L περιέχει όλες τις συχνότητες μεταξύ 440 Hz – 880 Hz που αντιστοιχούν σε μουσικές νότες, υπολογισμένες με αρκετά καλή ακρίβεια (προσέξτε το τελευταίο στοιχείο που αντιστοιχεί στα 880 Hz). Όμως δεν μπορούμε να τις τροφοδοτήσουμε ακόμα στην Beep(). Η Beep() περιμένει ακέραιους αριθμούς ενώ η λίστα περιέχει πραγματικούς. Θα πρέπει να τους μετατρέψουμε.

Η συνάρτηση που μετατρέπει έναν πραγματικό αριθμό σε ακέραιο λέγεται int():

```
>>> print int(2.4), int(2.5), int(2.6)
2 2 2
```

Η int() ουσιαστικά «κόβει» το δεκαδικό μέρος από τον πραγματικό αριθμό και επιστρέφει μόνο το ακέραιο μέρος του. Έτσι, και το 2.4 και το 2.5 και το 2.6 στρογγυλοποιούνται στο 2. Αν θέλουμε να πετύχουμε στρογγυλοποίηση στον πλησιέστερο ακέραιο, ένα απλό κόλπο είναι να προσθέσουμε 0.5 στον αριθμό που μετατρέπουμε. Πχ.:

```
>>> print int(0.5+2.4), int(0.5+2.5), int(0.5+2.6)
2 3 3
```

Και τώρα ξέρουμε πώς να φτιάξουμε από την L μια λίστα ακεραίων:

```
>>> K=[]
>>> for f in L:
      K=K+[int(0.5+f)]

>>> print K
[440, 466, 494, 523, 554, 587, 622, 659, 698, 740, 784, 831, 880]
```

Σημειωτέον ότι μπορούμε να κάνουμε ακριβώς το ίδιο και με έναν ακόμα πιο απλό τρόπο. Η Python επιτρέπει να ορίζουμε λίστες περιγραφικά. Πχ.:

```
>>> M=[int(0.5+x) for x in L]
```

Αυτό λέει ότι η M είναι μια λίστα με στοιχεία που προκύπτουν από τον υπολογισμό int(0.5+x), για κάθε x που βρίσκεται στη λίστα L. Το αποτέλεσμα είναι το ίδιο:

```
>>> print M
[440, 466, 494, 523, 554, 587, 622, 659, 698, 740, 784, 831, 880]
```

Και τώρα μπορούμε να ακούσουμε όλες τις νότες:

```
>>> play_list(M,300)
440 Hz
466 Hz
494 Hz
523 Hz
554 Hz
587 Hz
622 Hz
659 Hz
698 Hz
740 Hz
784 Hz
831 Hz
```

```
880 Hz
```

Με παρόμοιο τρόπο μπορούμε να παράγουμε τις νότες και για τις υπόλοιπες οκτάβες. Ή, εναλλακτικά και πιο εύκολα μπορούμε να χρησιμοποιήσουμε τον περιγραφικό τρόπο ορισμού των λιστών για να πάρουμε την επόμενη ή την προηγούμενη οκτάβα με διπλασιασμό ή υποδιπλασιασμό όλων των στοιχείων:

```
>>> M2=[2*x for x in M]
>>> print M2
[880, 932, 988, 1046, 1108, 1174, 1244, 1318, 1396, 1480, 1568, 1662,
1760]
>>> M0=[x/2 for x in M]
>>> print M0
[220, 233, 247, 261, 277, 293, 311, 329, 349, 370, 392, 415, 440]
```

Και τώρα έχουμε τις νότες από 3 ολόκληρες οκτάβες και μπορούμε αν θέλουμε να τις ενώσουμε όλες σε μια μεγάλη λίστα:

```
>>> Mall=M0+M+M2
>>> print Mall
[220, 233, 247, 261, 277, 293, 311, 329, 349, 370, 392, 415, 440,
440, 466, 494, 523, 554, 587, 622, 659, 698, 740, 784, 831, 880, 880,
932, 988, 1046, 1108, 1174, 1244, 1318, 1396, 1480, 1568, 1662, 1760]
```

Αρκετά καλό αποτέλεσμα, όμως όχι τέλειο. Προσέξτε ότι οι συχνότητες 440 και 880 επαναλαμβάνονται 2 φορές στη λίστα Mall, επειδή κάθε λίστα που συνθέσαμε περιείχε 13 στοιχεία (περιείχε και τη νότα Λα της επόμενης οκτάβας). Αυτό το διορθώνουμε εύκολα, αν θυμηθούμε ότι μπορούμε να επιλέξουμε τμήματα λιστών, όπως είδαμε στην ενότητα 7.2:

```
>>> Mall=M0[:12]+M+M2[1:]
>>> print Mall
[220, 233, 247, 261, 277, 293, 311, 329, 349, 370, 392, 415, 440,
466, 494, 523, 554, 587, 622, 659, 698, 740, 784, 831, 880, 932, 988,
1046, 1108, 1174, 1244, 1318, 1396, 1480, 1568, 1662, 1760]
```

Το M0[:12] είναι συντομογραφία του M0[0:12] και σημαίνει «τα πρώτα 12 στοιχεία της M0». Παρομοίως, η συντομογραφία M2[1:] σημαίνει «τα στοιχεία της M2 από το δεύτερο μέχρι το τελευταίο». Και τώρα μπορούμε να ξαναδοκιμάσουμε την τυχαία σύνθεση μουσικής γράφοντας κάτι τέτοιο:

```
>>> for i in range(20):
      note=randint(0,36)
      freq=Mall[note]
      print "Note=", note, "   Freq=", freq, "Hz"
      Beep(freq,200)

Note= 13    Freq= 466 Hz
Note= 15    Freq= 523 Hz
Note= 21    Freq= 740 Hz
...
Note= 9    Freq= 370 Hz
Note= 24    Freq= 880 Hz
Note= 31    Freq= 1318 Hz
```

Το παραπάνω πρόγραμμα επιλέγει τυχαία νότες από τη λίστα Mall και τις παίζει με τη συνάρτηση Beep(). Και πάλι δεν πρόκειται για ποιοτική μουσική, αλλά ακούγεται κάπως καλύτερα από τον θόρυβο των τελείως τυχαίων συχνοτήτων. Για να πετύχουμε κάτι καλύτερο χρειάζεται να δούμε μια ακόμα δομή δεδομένων που λέγεται **Λεξικό**.

7.4 Λεξικά

Το **Λεξικό** (**Dictionary**) είναι μια δομή δεδομένων στην οποία καταχωρούμε δεδομένα τα οποία μπορούμε να ανακτήσουμε με το όνομά τους που λέγεται **κλειδί** (**key**). Να πώς ορίζεται ένα λεξικό:

```
>>> a={'La':440, 'La#':466, 'Si':494, 'Do':523}
```

Εδώ ορίσαμε ένα λεξικό που αποτελείται από 4 στοιχεία, αυτά που βρίσκονται μεταξύ των αγκυλών { }. Κάθε στοιχείο αποτελείται από ένα ζεύγος «κλειδί : τιμή». Το κλειδί είναι η ετικέτα ή το όνομα του στοιχείου και χωρίζεται από την τιμή με το σύμβολο «:». Το λεξικό καταχωρείται στη μεταβλητή a. Τώρα μπορούμε να ανακτήσουμε τις τιμές των στοιχείων του λεξικού ως εξής:

```
>>> print a['La#']
466
>>> print a['Si'], a['Do']
494 523
```

Αντίθετα με τις λίστες, δε χρειάζεται να γνωρίζουμε τη θέση του στοιχείου για να τυπώσουμε την τιμή του. Αρκεί να γνωρίζουμε το όνομά του. Εξ'άλλου, η σειρά των στοιχείων δεν διατηρείται μέσα σε ένα λεξικό, όπως φαίνεται από την πλήρη εκτύπωσή του:

```
>>> print a
{'Do': 523, 'Si': 494, 'La#': 466, 'La': 440}
```

Νέα στοιχεία μπορούν να προστεθούν στο λεξικό με τον εξής τρόπο:

```
>>> a['Sol']=392
>>> a['Sol#']=415
>>> print a
{'Do': 523, 'La': 440, 'Sol#': 415, 'La#': 466, 'Si': 494, 'Sol':
392}
```

Αν προσθέσουμε ένα στοιχείο με όνομα που ήδη υπάρχει, τότε γίνεται αντικατάσταση:

```
>>> a['Do']=261
>>> print a
{'Do': 261, 'La': 440, 'Sol#': 415, 'La#': 466, 'Si': 494, 'Sol':
392}
```

Για να σβήσουμε ένα στοιχείο χρησιμοποιούμε την εντολή del:

```
>>> del a['Do']
```

```
>>> print a
{'La': 440, 'Sol#': 415, 'La#': 466, 'Si': 494, 'Sol': 392}
```

Τελικά, αν βάλουμε στο ένα λεξικό όλες τις νότες μιας οκτάβας:

```
>>> a={'Do':261, 'Do#':277, 'Re':293, 'Re#':311, 'Mi':329, 'Fa':349,
'Fa#':370, 'Sol':392, 'Sol#':415, 'La':440, 'La#':466, 'Si':494}

>>> print a
{'Do': 261, 'Re': 293, 'Do#': 277, 'La': 440, 'Mi': 329, 'Sol': 392,
'Sol#': 415, 'Fa': 349, 'Si': 494, 'Fa#': 370, 'Re#': 311, 'La#':
466}
```

... μπορούμε να προγραμματίσουμε μουσική στον υπολογιστή μας δίνοντας απευθείας τα ονόματα των νοτών. Πχ. να μια μουσική φράση:

```
>>> p=['Do','Mi','Sol','La','La#','La','Sol','Mi']
```

Και να πώς μετατρέπεται αυτόματα σε λίστα συχνοτήτων με τη βοήθεια του λεξικού:

```
>>> f=[a[x] for x in p]
>>> print f
[261, 329, 392, 440, 466, 440, 392, 329]
```

Και τώρα μπορούμε να την ακούσουμε:

```
>>> play_list(f,250)
261 Hz
329 Hz
392 Hz
440 Hz
466 Hz
440 Hz
392 Hz
329 Hz
```

... και πολλές φορές αν θέλουμε:

```
>>> play_list(f*4,250)
261 Hz
329 Hz
...
392 Hz
329 Hz
```

Μπορούμε ακόμα να «μετατοπίσουμε» τη φράση κατά πχ. 5 νότες ψηλότερα:

```
>>> f5=[int(0.5+x*d**5) for x in f]
>>> print f5
[348, 439, 523, 587, 622, 587, 523, 439]
>>> play_list(f5,250)
```

... ή 7 νότες ψηλότερα από την αρχική φράση:

```
>>> f7=[int(0.5+x*d**7) for x in f]
>>> print f7
```

```
[391, 493, 587, 659, 698, 659, 587, 493]
>>> play_list(f7,250)
```

Μπορούμε επίσης να συνδυάσουμε την αρχική φράση με τις μετατοπίσεις της γράφοντας μια σύνθεση λιστών ως εξής:

```
>>> play_list(f*2+f5+f+f7+f5+f*2,250)
261 Hz
329 Hz
392 Hz
...
440 Hz
392 Hz
329 Hz
```

Κι όσο για τη σύνθεση τυχαίας μουσικής, ακούεται καλύτερα αν περιορίσουμε τη λίστα επιλογών σε κάποιο μικρό υποσύνολο των νοτών της οκτάβας (στη μουσική λέγεται **κλίμακα**). Να μερικά παραδείγματα:

```
>>> p1=['Do','Re','Mi','Fa','Sol','La','Si']
>>> f1=[a[x] for x in p1]
>>> for i in range(20):
      Beep(f1[randint(0,6)],250)

>>> p2=['Do','Mi','Sol','La','La#']
>>> f2=[a[x] for x in p2]
>>> for i in range(20):
      Beep(f2[randint(0,4)],200)

>>> p3=['Do#','Re#','Fa#','Sol#','La#']
>>> f3=[a[x] for x in p3]
>>> for i in range(20):
      Beep(f3[randint(0,4)],200)
```

Ουσιαστικά πρόκειται για τον ίδιο κώδικα που τρέχει σε διαφορετικές κλίμακες και κάθε μια ακούγεται λίγο διαφορετικά. Θα μπορούσαμε να τον εξελίξουμε και να τον γενικεύσουμε σε συνάρτηση, ως εξής:

```
>>> def rand_music(KLIM,OCT,Notes,Speed):
      f=[a[x] for x in KLIM]
      q=f
      while OCT>1:
            OCT=OCT-1
            r=[x*2 for x in f]
            q=q+r
      L=len(q)-1
      for i in range(Notes):
            note=randint(0,L)
            freq=q[note]
            Beep(freq,Speed)
```

KLIM είναι η λίστα με την κλίμακα, OCT το πλήθος των οκτάβων στις οποίες θα επαναλαμβάνεται η κλίμακα, Notes το πλήθος των τυχαίων νοτών που θα παιχτούν και Speed η διάρκεια της κάθε νότας. Η κλήση γίνεται ως εξής:

```
>>> rand_music(['Do#','Re#','Fa#','Sol#','La#'],2,20,250)
```

Μέχρι τώρα έχουμε κρατήσει σταθερές τις χρονικές διάρκειες των νοτών. Αν θέλουμε όμως να παίξουμε ακριβέστερη μουσική θα πρέπει για κάθε νότα να προσδιορίσουμε και τη διάρκειά της. Στη μουσική οι χρονικές διάρκειες είναι κβαντισμένες. Αφού καθοριστεί η διάρκεια μιας ολόκληρης νότας σε msec, όλες οι άλλες νότες έχουν διάρκειες υποπολλαπλάσιες δυνάμεις του 2 ως προς την αρχική: 1/2, 1/4, 1/8, 1/16, 1/32 και 1/64. Βολεύει λοιπόν να ορίσουμε τη διάρκεια της ολόκληρης νότας ως ακέραιο πολλαπλάσιο του 64, πχ. 30*64=1920 msec.

Επίσης χρειαζόμαστε και έναν ήχο παύσης (ησυχίας). Αρκεί να προσθέσουμε στο λεξικό έναν υπέρηχο:

```
>>> a['P']=28000
```

Η κωδικοποίηση μιας μουσικής φράσης θα είναι τώρα λίγο πιο σύνθετη:

```
>>> music=[['La',16],['Sol',16],['La',2],['P',8],['Sol',16],
['Fa',16],['Mi',16],['Re',16],['Do#',4],['Re',2]]
```

Κάθε στοιχείο της λίστας είναι μια άλλη λίστα που περιέχει δυο στοιχεία. Το πρώτο είναι το όνομα της νότας, το δεύτερο η χρονική διάρκεια. Να μια συνάρτηση που μετατρέπει σε μουσική την παραπάνω λίστα:

```
>>> def play_musiclist(L):
      for x in L:
            [note,time]=x
            freq=a[note]
            duration=1920/time
            print "Note=", note, "(", freq, "Hz)   Dur= 1/",
            print time, "(", duration, "msec)"
            Beep(freq,duration)

>>> play_musiclist(music)
Note= La ( 440 Hz)   Dur= 1/ 16 ( 120 msec)
Note= Sol ( 392 Hz)   Dur= 1/ 16 ( 120 msec)
Note= La ( 440 Hz)   Dur= 1/ 2 ( 960 msec)
Note= P ( 28000 Hz)   Dur= 1/ 8 ( 240 msec)
Note= Sol ( 392 Hz)   Dur= 1/ 16 ( 120 msec)
Note= Fa ( 349 Hz)   Dur= 1/ 16 ( 120 msec)
Note= Mi ( 329 Hz)   Dur= 1/ 16 ( 120 msec)
Note= Re ( 293 Hz)   Dur= 1/ 16 ( 120 msec)
Note= Do# ( 277 Hz)   Dur= 1/ 4 ( 480 msec)
Note= Re ( 293 Hz)   Dur= 1/ 2 ( 960 msec)
```

7.5 Μέθοδοι

Πριν αρχίσουμε να γράφουμε τον κώδικα ενός προγράμματος, μια καλή τακτική είναι να ψάχνουμε στο on-line help της Python (ή στο εγχειρίδιο αναφοράς το οποίο κατεβάζουμε από το επίσημο web-site) για τυχόν έτοιμες βιβλιοθήκες (modules) ή συναρτήσεις που μπορούμε να χρησιμοποιήσουμε. Οι έτοιμες συναρτήσεις εμφανίζονται με τρεις διαφορετικές συντακτικές μορφές κλήσης:

Κάποιες καλούνται όπως οι συναρτήσεις που ορίζουμε με την εντολή def. Τα ορίσματά τους τα γράφουμε μέσα σε παρένθεση. Μπορεί να επιστρέφουν τιμή ή όχι. Τέτοιες συναρτήσεις είναι για παράδειγμα η sin() και η Beep().

Κάποιες άλλες καλούνται με τη βοήθεια τελεστών. Για παράδειγμα η συνάρτηση που ενώνει δυο λίστες ενεργοποιείται με τον τελεστή + :

```
>>> print [1,2]+[3,4]
[1, 2, 3, 4]
```

Αν δεν μας αρέσει αυτός ο τρόπος κλήσης μπορούμε πάντα να ορίσουμε μια δική μας συνάρτηση όπως:

```
>>> def connect(L1,L2):
      return L1+L2
```

Και τώρα κάθε φορά που θέλουμε να ενώσουμε δυο λίστες θα καλούμε τη συνάρτηση connect():

```
>>> print connect([1,2],[3,4])
[1, 2, 3, 4]
```

Τέλος υπάρχει και η τρίτη κατηγορία συναρτήσεων που ονομάζονται **μέθοδοι** και καλούνται ως εξής:

```
>>> L=['a','b','a','c','a','a','d']
>>> print L.count('a')
4
```

Η count() είναι μια μέθοδος που μετράει πόσα από τα στοιχεία της λίστας είναι ίδια με την παράμετρο που δίνουμε. Η ίδια η λίστα δεν μπαίνει ως παράμετρος στην παρένθεση. Γράφουμε απλώς το όνομα της λίστας, μια τελεία και το όνομα της μεθόδου. Αυτός ο τρόπος σύνταξης λέγεται **dot notation** (σημειογραφία με τελεία).

Και πάλι, αν δεν μας αρέσει αυτή η σύνταξη κλήσης μπορούμε να ορίσουμε κάτι τέτοιο:

```
>>> def my_count(A,x):
      return A.count(x)
```

… και στη συνέχεια να γράφουμε:

```
>>> print my_count(L,'a')
4
```

Οι λόγοι για τους οποίους υπάρχουν αυτές οι τρεις διαφορετικές κατηγορίες συναρτήσεων με τους διαφορετικούς τρόπους κλήσης αρχίζουν να φαίνονται όταν φτιάχνει κανείς μεγάλα και σύνθετα προγράμματα. Προς το παρόν αρκεί να εξηγήσουμε ότι κάποιες λειτουργίες είναι βολικότερο να συντάσσονται ως τελεστές, άλλες ως μέθοδοι και άλλες ως συναρτήσεις.

Μέθοδοι υπάρχουν όχι μόνο για λίστες αλλά για οποιοδήποτε τύπο δομών δεδομένων που χρησιμοποιεί η Python. Για παράδειγμα η upper() είναι μια μέθοδος που επιδρά σε αλφαριθμητικά (strings) και επιστρέφει το ίδιο αλφαριθμητικό αλλά με κεφαλαία γράμματα.

```
>>> v='αβγδε'
>>> print v.upper()
ΑΒΓΔΕ
```

Οι keys() και values() είναι μέθοδοι που επιδρούν πάνω σε λεξικά και επιστρέφουν λίστες με όλα τα κλειδιά και όλες τις τιμές των στοιχείων του λεξικού αντίστοιχα. Πχ.:

```
>>> a={'Do':261, 'Do#':277, 'Re':293, 'Re#':311, 'Mi':329, 'Fa':349,
'Fa#':370, 'Sol':392, 'Sol#':415, 'La':440, 'La#':466, 'Si':494}
>>> print a.keys()
['Do', 'Re', 'Do#', 'La', 'Mi', 'Sol', 'Sol#', 'Fa', 'Si', 'Fa#',
'Re#', 'La#']
>>> print a.values()
[261, 293, 277, 440, 329, 392, 415, 349, 494, 370, 311, 466]
```

Προσοχή χρειάζεται μόνο στην περίπτωση που κάποια μέθοδος (ή συνάρτηση) συμβαίνει να αλλάζει το στοιχείο στο οποίο επιδρά. Για παράδειγμα η reverse() είναι μια μέθοδος που αντιστρέφει τη σειρά των στοιχείων μιας λίστας. Η ίδια η μέθοδος δεν επιστρέφει κάποια τιμή. Απλώς κάθε φορά που την καλούμε κάνει αντιστροφή:

```
>>> L=[1,2,3,4]
>>> L.reverse()
>>> print L
[4, 3, 2, 1]
>>> L.reverse()
>>> print L
[1, 2, 3, 4]
```

Και τώρα δείτε το εξής (εκ πρώτης όψεως παράξενο):

```
>>> L=[1,2,3,4]
>>> M=L
>>> print L,M
[1, 2, 3, 4] [1, 2, 3, 4]
>>> M.reverse()
>>> print L,M
[4, 3, 2, 1] [4, 3, 2, 1]
```

Εμείς είπαμε να αντιστρέψει μόνο τη M. Γιατί αντέστρεψε και τις δυο;

Η απάντηση έγκειται στον τρόπο με τον οποίο η Python χειρίζεται τον τελεστή ανάθεσης τιμής '='. Όταν γράψαμε L=[1,2,3,4] η Python πρώτα δημιούργησε τη λίστα [1,2,3,4] και στη συνέχεια της έδωσε το όνομα L. Με την δεύτερη εντολή M=L δεν είπαμε να δημιουργήσει μια δεύτερη λίστα ίδια με την L, είπαμε απλώς ότι η L θα έχει και ένα δεύτερο όνομα M. Τελικά δημιουργήσαμε μία λίστα με δυο ονόματα. Αν θέλαμε να δημιουργήσουμε δυο λίστες θα έπρεπε να γράψουμε κάτι τέτοιο:

```
>>> L=[1,2,3,4]
>>> M=[1,2,3,4]
```

... ή κάτι τέτοιο για να μην ξαναγράφουμε τα στοιχεία της λίστας:

```
>>> L=[1,2,3,4]
>>> M=L[:]
```

Το L[:] είναι συντομογραφία του L[0:len(L)] και δηλώνει το τμήμα της λίστας από την αρχή μέχρι το τέλος της. Είναι ένας τρόπος για να πούμε στην Python ότι θέλουμε να δημιουργηθεί μια νέα λίστα ίδια με την L. Όχι όμως κι ο μοναδικός. Θα μπορούσαμε να γράψουμε και κάτι τέτοιο:

```
>>> L=[1,2,3,4]
>>> M=L+[]
```

... που σημαίνει «δημιούργησε μια νέα λίστα που θα είναι η σύνδεση της L με την κενή λίστα ([]) και δώσε της το όνομα M».

Με όποιον τρόπο κι αν το κάνουμε, δημιουργούμε δυο ξεχωριστές ίδιες λίστες και τώρα η reverse() αλλάζει μόνο τη μία:

```
>>> print L,M
[1, 2, 3, 4] [1, 2, 3, 4]
>>> M.reverse()
>>> print L,M
[1, 2, 3, 4] [4, 3, 2, 1]
```

Σημειωτέον ότι με τον ίδιο τρόπο συμπεριφέρεται η Python με όλους τους τύπους δεδομένων, όμως δεν έχουν όλοι οι τύποι μεθόδους που τους αλλάζουν κι έτσι δεν γίνεται αντιληπτό πάντα αυτό το ζήτημα. Για να γίνει κατανοητό αυτό ας δούμε το ίδιο παράδειγμα με ακεραίους αντί για λίστες:

```
>>> L=5
>>> M=L
>>> print L,M
5 5
```

Στην πραγματικότητα δεν υπάρχουν δυο αντίγραφα του αριθμού 5 στη μνήμη του υπολογιστή. Υπάρχει μόνο ένα 5 με δυο «ετικέττες» L και M. Αν μπορούσαμε να αλλάξουμε την τιμή αυτή απευθείας στη μνήμη τότε θα άλλαζαν και οι δυο μεταβλητές. Όμως ο συνήθης τρόπος αλλαγής τιμής είναι γράφοντας κάτι τέτοιο:

```
 >>> M=M+1
```

Αυτό δεν αλλάζει το 5 σε 6. Δημιουργεί ένα 6, σε κάποιο άλλο σημείο στη μνήμη του υπολογιστή και μεταφέρει την «ετικέττα» M εκεί. Η L παραμένει στο 5 το οποίο δεν έχει αλλάξει. Κι έτσι:

```
>>> print L,M
5 6
```

Κάποιος λιγώτερο «υποψιασμένος» πτογραμματιστής θα μπορούσε να υποθέσει ότι είχαμε δυο μεταβλητές, στην αρχή βάλαμε την ίδια τιμή και στις δυο και μετά αλλάξαμε την τιμή στη μία. Το αποτέλεσμα (φαινομενικά) είναι το ίδιο. Όμως εμείς τώρα ξέρουμε τί έγινε πραγματικά.

Και πάλι, αν δεν μας βολεύει ο τρόπος με τον οποίο λειτουργεί μια συνάρτηση ή μια μέθοδος μπορούμε να φτιαξουμε μια δική μας παραλλαγή. Για παράδειγμα, να μια συνάρτηση που δέχεται μια λίστα και επιστρέφει την αντίστροφή της ως τιμή, χωρίς να αλλάζει την αρχική:

```
>>> def rev(L):
      M=L+[]
      M.reverse()
      return M

>>> J=[1,2,3,4]
>>> print J, rev(J), J
[1, 2, 3, 4] [4, 3, 2, 1] [1, 2, 3, 4]
```

Να και μια παραλλαγή που κάνει το ίδιο χωρίς να χρησιμοποιεί τη μέθοδο reverse():

```
>>> def rev2(L):
      if L==[]:
            return L
      H=L[0]
      T=L[1:]
      RT=rev2(T)
      return RT+[H]

>>> J=[1,2,3,4]
>>> print J, rev2(J), J
[1, 2, 3, 4] [4, 3, 2, 1] [1, 2, 3, 4]
```

Διαβάζοντας τον κώδικα της rev2() βλέπουμε ότι κατ'αρχήν ελέγχει αν η λίστα που δίνουμε ως παράμετρο είναι κενή και τότε την επιστρέφει ως τιμή και σταματά. Αν η λίστα δεν είναι κενή, τότε διαβάζει το πρώτο στοιχείο της (L[0]), δημιουργεί μια νέα υπολίστα με όλα τα υπόλοιπα στοιχεία πλην του πρώτου (L[1:]) και τέλος επιστρέφει μια λίστα που είναι η σύνδεση της αντεστραμμένης υπολίστας με το L[0] στο τέλος. Το ενδιαφέρον είναι ότι για να υπολογίσει την αντεστραμμένη υπολίστα η rev2() καλεί τον εαυτό της! Αυτό είναι κάτι απόλυτα επιτρεπτό στις περισσότερες γλώσσες προγραμματισμού και δεν είναι περισσότερο παράξενο από μια έκφραση όπως η M=M+1 όπου μια μεταβλητή αναφέρεται στον εαυτό της. Πρέπει μόνο να προσέχουμε ώστε οι διαδοχικές κλήσεις κάποτε να σταματούν και να μη συνεχίζουν επ'αόριστον. Εδώ είμαστε σίγουροι ότι οι κλήσεις κάποια στιγμή θα σταματήσουν επειδή (α) σε κάθε κλήση το μήκος της υπολίστας μικραίνει, (β) σε περίπτωση κενής λίστας η εκτέλεση σταματά, και (γ) ο έλεγχος για κενή λίστα γίνεται πριν την κλήση στον εαυτό της. Αν κάποιο από τα (α), (β) και (γ) δεν ίσχυε τότε η rev2() δεν θα σταματούσε ποτέ. Συναρτήσεις που καλούν τον εαυτό τους λέγονται **αναδρομικές**.

7.6 Ασκήσεις

Αφού τρέξετε τα παραδείγματα των ενοτήτων που προηγήθηκαν, εκπονήστε τις παρακάτω ασκήσεις. Αναρτήστε τα αρχεία σας ως ένα ενιαίο συμπιεσμένο αρχείο.

Άσκηση #1

Φτιάξτε μια λίστα που θα περιέχει όλες τις συχνότητες των νοτών στο ακουστικό φάσμα 20-22000 Hz καθώς και τα ονόματά τους. Οι συχνότητες να είναι στρογγυλοποιημένες στον πλησιέστερο ακέραιο.

Άσκηση #2

Δίδεται η εξής εντολή εκχώρησης τιμής:

```
mysterysong=[['Do',3,'Σ1'],['Fa',2,'Σ2'],['Fa',1,'Σ3'],['Fa',4,'Σ4'],
['Sol',3,'Σ5'],['La',2,'Σ6'],['La',1,'Σ7'],['La',4,'Σ8'],['La',1,'Σ9'
],['Sol',1,'Σ10'],['La',1,'Σ11'],['La#',2,'Σ12'],['Mi',2,'Σ13'],['Sol
',2,'Σ14'],['Fa',4,'Σ15']]
```

Κάθε στοιχείο της παραπάνω λίστας αποτελείται από μια τριάδα (πάλι λίστα) που ως $1^{ο}$ στοιχείο έχει το όνομα μιας νότας, ως $2^{ο}$ στοιχείο μια χρονική διάρκεια και ως $3^{ο}$ στοιχείο μια συλλαβή (ή λέξη) η οποία θα πρέπει να τυπώνεται σε συγχρονισμό με τη νότα. Η χρονική διάρκεια είναι εκφρασμένη ως πολλαπλάσιο των 240 msec. Δηλαδή:

1 **à** 1x240 = 240 msec
2 **à** 2x240 = 480 msec
3 **à** 3x240 = 720 msec, κλπ

Να κατασκευάσετε ένα πρόγραμμα Python που περιέχει τις κατάλληλες δομές dictionary και εντολές εκτέλεσης, οι οποίες με χρήση της winsound θα παίζουν το mysterysong. Μόλις αναγνωρίσετε το τραγούδι αλλάξτε τις συλλαβές Σ1-Σ15 ώστε να εμφανίζουν συγχρονισμένα τα λόγια του τραγουδιού.

Άσκηση #3

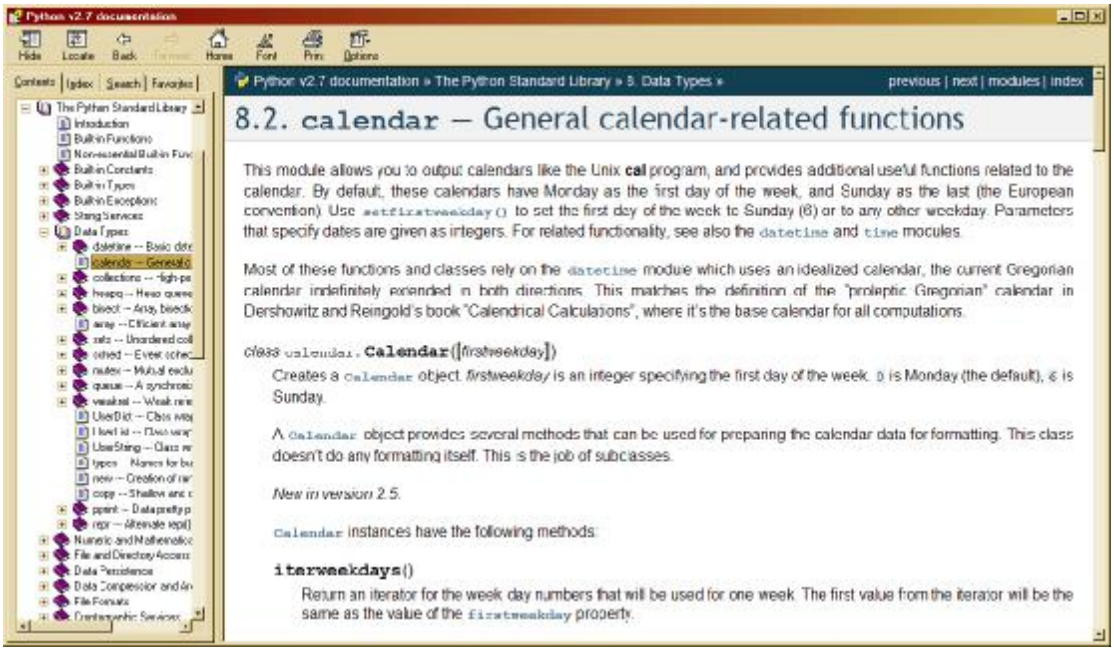
Φτιάξτε κώδικα Python που θα δημιουργεί μια τυχαία μουσική φράση με μήκος 5-10 νότες και στη συνέχεια θα παίζει τη φράση και την αντίστροφή της.

8.1 Γενικά

Η άσκηση αυτή αφορά **δημιουργία και διαχείριση αρχείων** στη γλώσσα Python, **κλήσεις στο λειτουργικό σύστημα** και **αποστολή e-mails**, με χρήση κατάλληλων βιβλιοθηκών συναρτήσεων.

8.2 Ημερολόγια και Αρχεία

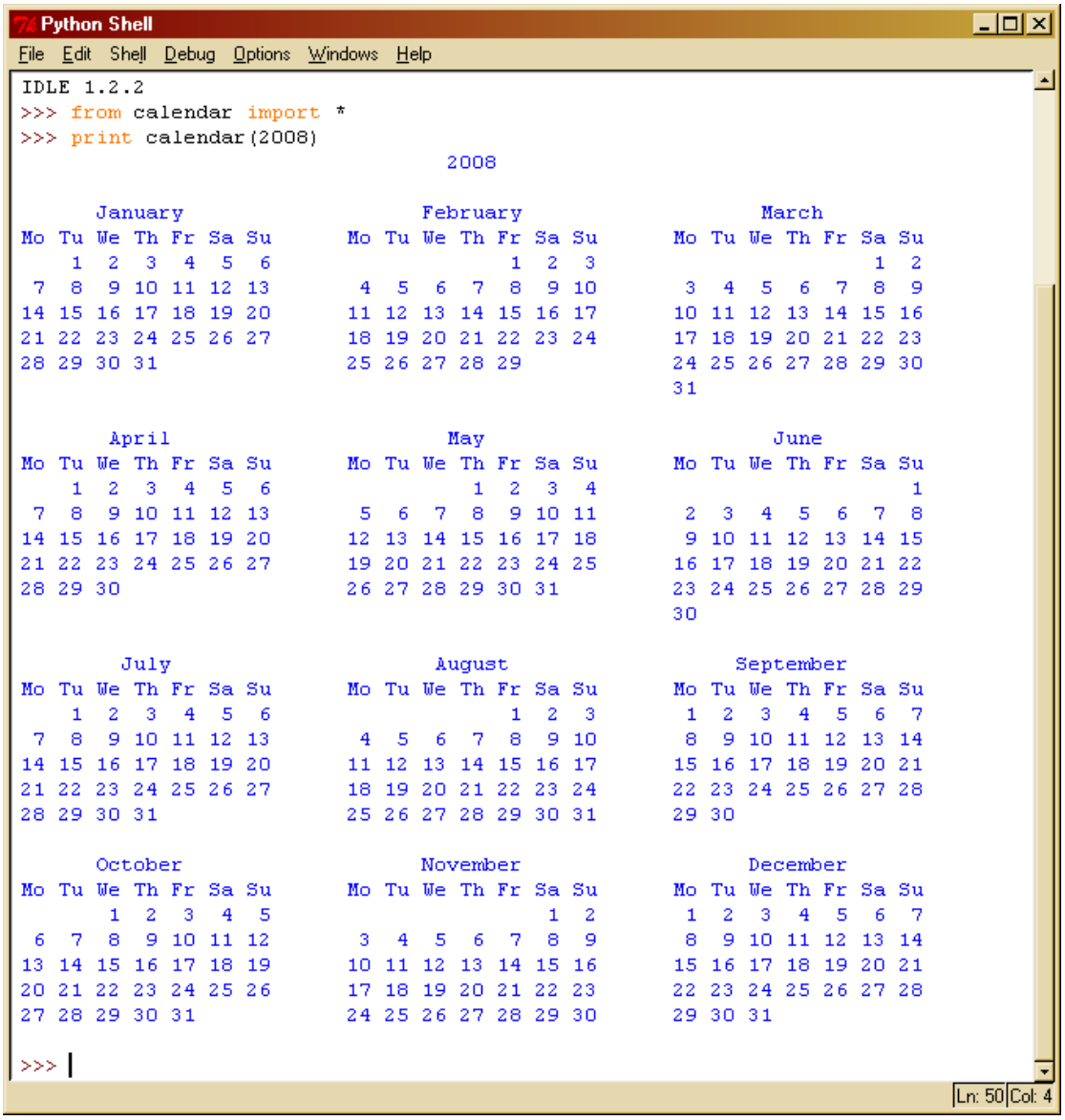
Έστω ότι θέλουμε να τυπώσουμε ένα ημερολόγιο (π.χ. του έτους 2008). Κανονικά θα έπρεπε να υπολογίσουμε πόσες ημέρες έχει κάθε μήνας, αν το έτος είναι δίσεκτο, ποιό όνομα ημέρας αντιστοιχεί σε κάθε ημερομηνία κλπ. Όμως η Python διαθέτει τη βιβλιοθήκη calendar η οποία περιέχει συναρτήσεις για όλα τα παραπάνω, όπως μπορείτε να δείτε στο on-line Help (πατώντας F1 και μετά επιλέγοντας "Library Reference > Data Types > calendar"):



Επιπλέον, η βιβλιοθήκη διαθέτει και την ομώνυμη συνάρτηση calendar() η οποία εμφανίζει ένα πλήρες ημερολόγιο μόνο με μια εντολή. Δηλαδή αν γράψουμε:

```
>>> from calendar import *
>>> print calendar(2008)
```

... θα δούμε κάτι τέτοιο:



Αξίζει να επισημάνουμε ότι αυτό που επιστρέφει η συνάρτηση calendar() είναι ένα αλφαριθμητικό (string). Εμείς επιλέξαμε να το τυπώσουμε με την print. Θα μπορούσαμε να το βάλουμε σε μια μεταβλητή, πχ.:

```
>>> a=calendar(2008)
>>> print a
```

… ή να το σώσουμε σε ένα αρχείο και να το τυπώσουμε στον εκτυπωτή ή να το στείλουμε με e-mail σε κάποιο φίλο μας. Όλα αυτά (και πολλά περισσότερα) μπορούμε να τα κάνουμε μέσα από την Python. Στη συνέχεια θα δούμε πώς γράφουμε πληροφορίες σε αρχεία, τα οποία στη συνέχεια μπορούν να διαβαστούν από άλλες εφαρμογές.

Για να δημιουργήσουμε ένα αρχείο θα πρέπει πρώτα να γνωρίζουμε τον τρόπο με τον οποίο κωδικοποιούνται τα δεδομένα σε αυτό. Για παράδειγμα, ένα αρχείο PDF έχει μια επικεφαλίδα στην οποία έχει καταγραφεί η εφαρμογή με την οποία δημιουργήθηκε, η ημερομηνία, το όνομα του χρήστη, και διάφορα χαρακτηριστικά

(πχ. αν επιτρέπεται η εκτύπωση του αρχείου ή όχι, κλπ). Στη συνέχεια υπάρχει μια σειρά από bytes στα οποία καταγράφονται οι γραμματοσειρές (fonts) που χρησιμοποιούνται στο μέσα στο αρχείο, μετά ακολουθούν οι σελίδες με πληροφορίες για τις διαστάσεις (και ενδεχομένως το χρώμα) της κάθε μιας, το κείμενο της κάθε σελίδας κι αν μέσα στο κείμενο περιέχονται και εικόνες, υπάρχει επιπλέον κωδικοποιημένη πληροφορία για την κάθε μια. Συνεπώς, για να δημιουργήσουμε ένα αρχείο κάποιας σύνθετης εφαρμογής χρειάζεται να γνωρίζουμε όλες αυτές τις λεπτομέρειες (οι οποίες για τα ανοικτού τύπου αρχεία είναι ελεύθερα διαθέσιμες στο διαδίκτυο) ή να φορτώσουμε κάποια κατάλληλη βιβλιοθήκη η οποία θα αναλάβει να απλοποιήσει τη διαδικασία για μας.

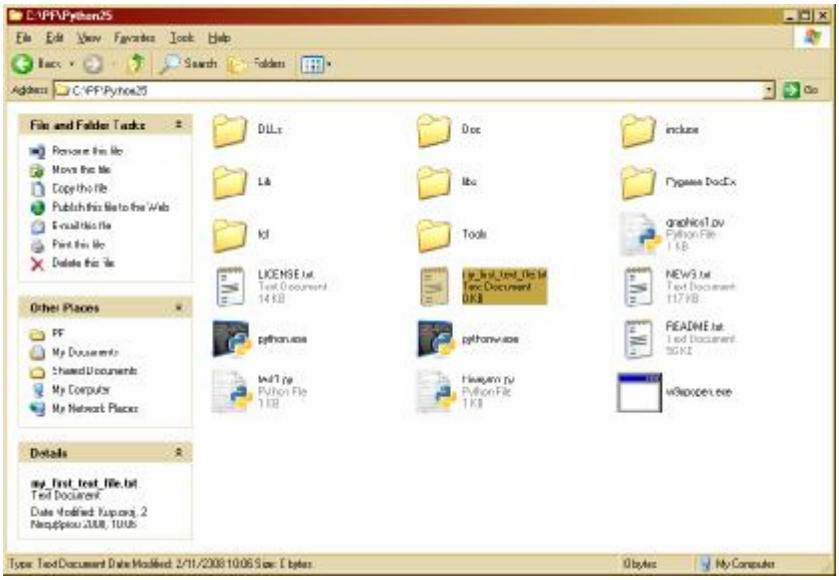
Τα αρχεία με την πιο απλή κωδικοποίηση είναι τα αρχεία κειμένου (.txt), τα οποία δημιουργούνται και διαβάζονται από διορθωτές κειμένου (text editors). Τα αρχεία αυτά δεν έχουν επικεφαλίδα και κάθε byte τους αντιστοιχεί σε έναν ASCII χαρακτήρα. Για να δημιουργήσουμε ένα αρχείο κειμένου με την Python χρησιμοποιούμε τη συνάρτηση open(), ως εξής:

```
>>> my_file=open("my_first_text_file.txt","w")
```

Κατ'αρχήν η συνάρτηση open() είναι ενσωματωμένη στην Python και δε χρειάζεται να φορτώσουμε κάποια βιβλιοθήκη. Δέχεται δυο παραμέτρους:

- Το όνομα του αρχείου ("my_first_text_file.txt") είναι το όνομα με το οποίο αναγνωρίζει το λειτουργικό σύστημα το αρχείο μας.
- Έναν κωδικό που καθορίζει αν το αρχείο πρόκειται να το γρησιμοποιήσουμε για εγγραφή ή για ανάγνωση ("w" για εγγραφή, "r" για ανάγνωση).

Η μεταβλητή my_file είναι το όνομα με το οποίο θα αναγνωρίζει η Python το αρχείο. Επειδή τα πραγματικά ονόματα των αρχείων συχνά είναι μεγάλα και δύσχρηστα, συνηθίζουμε να χρησιμοποιούμε απλούστερα μνημονικά ονόματα μέσα στον κώδικά μας. Μόλις εκτελεστεί η open() δημιουργείται το αρχείο και μπορούμε να το δούμε από το λειτουργικό σύστημα:



Εδώ βλέπουμε δυο πράγματα:

- Το αρχείο έχει δημιουργηθεί στον κατάλογο (φάκελο) που είναι εγκατεστημένη η Python. Αν θέλουμε να δημιουργηθεί κάπου αλλού θα

πρέπει να δώσουμε το πλήρες path μαζί με το όνομα του αρχείου στην πρώτη παράμετρο της open().

- Το αρχείο φαίνεται ότι έχει μέγεθος 0 bytes. Απόλυτα λογικό, αφού δεν έχουμε γράψει τίποτα ακόμη σε αυτό.

Για να γράψουμε κάτι, χρησιμοποιούμε τη μέθοδο write(). Ως μέθοδος, η write() επιδρά στη μεταβλητή του αρχείου και δέχεται μια παράμετρο (πχ. ένα αλφαριθμητικό, μια μεταβλητή, κλπ), τα περιεχόμενα της οποίας θα γραφτούν στο αρχείο. Μπορούμε να τη χρησιμοποιήσουμε όσες φορές θέλουμε, διαδοχικά:

```
>>> my_file.write("Το παρακάτω ημερολόγιο έχει δημιουργηθεί με τη
γλώσσα Python:\n\n")
>>> a=calendar(2008)
>>> my_file.write(a)
>>> my_file.write("\n\nPython: Κάνει τα πάντα...\n... αρκεί να ξέρεις
πώς.\n(Read the Manual!!!)")
```

Το "\n" μέσα στα αλφαριθμητικά προκαλεί αλλαγή γραμμής (σα να έχουμε πατήσει το πλήκτρο "Enter/Return").

Εναλλακτικά, αντί για τη μέθοδο write() μπορούμε να χρησιμοποιήσουμε την εντολή print, με τη σύνταξη:

```
>>> print >>my_file, a
```

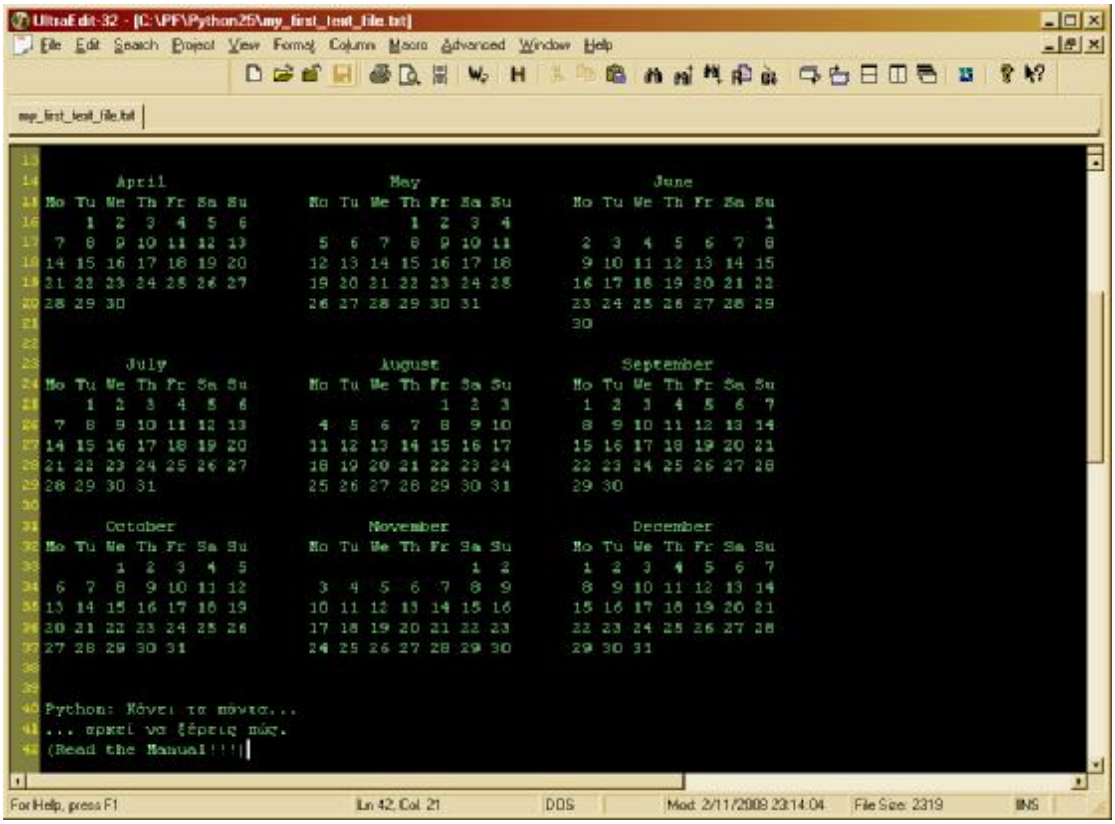
Δουλεύει όπως η write() με τη διαφορά ότι δεν χρειάζεται να βάζουμε τους χαρακτήρες «\n» για να τυπωθούν οι αλλαγές γραμμής στο τέλος των strings. Για παράδειγμα, τα περιεχόμενα του ίδιου αρχείου με εντολές print μπορούν να γραφτούν ως εξής:

```
>>> print >>my_file, "Το παρακάτω ημερολόγιο έχει δημιουργηθεί με τη
γλώσσα Python:"
>>> print >>my_file, ""
>>> a=calendar(2008)
>>> print >>my_file, a
>>> print >>my_file, ""
>>> print >>my_file, "Python: Κάνει τα πάντα... "
>>> print >>my_file, "... αρκεί να ξέρεις πώς. "
>>> print >>my_file, "Read the Manual!!!)"
```

Με όποιον τρόπο κι αν γράψουμε το αρχείο, μόλις τελειώσουμε τις εγγραφές χρειάζεται να καλέσουμε τη μέθοδο close() ώστε να κλείσει το αρχείο, δηλαδή να γραφτούν όλα τα περιεχόμενα της RAM στο σκληρό δίσκο.

```
>>> my_file.close()
```

Και τώρα μπορούμε να το διαβάσουμε με οποιονδήποτε editor:



Φυσικά, μπορούμε να το διαβάσουμε και μέσα από την Python, ως εξής:

```
>>> my_file=open("my_first_text_file.txt","r")
```

Τώρα η open() ανοίγει το αρχείο με παράμετρο "r", που σημαίνει ότι επιτρέπει μόνο την ανάγνωσή του. Στη συνέχεια η μέθοδος read() μεταφέρει όλα τα περιεχόμενα στη μεταβλητή contents, την οποία μετά χειριζόμαστε όπως θέλουμε:

```
>>> contents=my_file.read()
>>> print contents
```

Προσοχή, αυτή η διαδικασία λειτουργεί σωστά μόνο για αρχεία κειμένου. Αν το αρχείο περιέχει άλλα δεδομένα χρειαζόμαστε κατάλληλη βιβλιοθήκη για να το διαβάσουμε. Μόλις πάψουμε να χρειαζόμαστε άλλο το αρχείο, το κλείνουμε με τη μέθοδο close():

```
>>> my_file.close()
```

8.3 Κλήσεις στο Λειτουργικό Σύστημα

Σύμφωνα με όσα είπαμε παραπάνω, για να αντιγράψουμε ένα αρχείο μπορούμε να κάνουμε κάτι τέτοιο:

```
>>> file1=open("my_first_text_file.txt","r")
>>> file2=open("my_second_text_file.txt","w")
>>> contents=file1.read()
>>> file2.write(contents)
```

```
>>> file2.close()
>>> file1.close()
```

Όμως είναι ακόμα καλύτερα να χρησιμοποιήσουμε τις συναρτήσεις της βιβλιοθήκης os, που δίνει πρόσβαση σε κλήσεις του λειτουργικού συστήματος:

```
>>> from os import *
```

Η βιβλιοθήκη os περιέχει χρήσιμες συναρτήσεις, όπως η getcwd() με την οποία βλέπουμε σε ποιόν κατάλογο (φάκελο) δουλεύουμε:

```
>>> print getcwd()
C:\PF\Python25
```

... η listdir() που μας δείχνει σε μια λίστα τα περιεχόμενα ενός καταλόγου:

```
>>> d=getcwd()
>>> print listdir(d)
['DLLs', 'Doc', 'graphics1.py', 'include', 'Lib', 'libs',
'LICENSE.txt', 'my_first_text_file.txt', 'my_second_text_file.txt',
'NEWS.txt', 'Pygame DocEx', 'python.exe', 'pythonw.exe',
'README.txt', 'tcl', 'test1.py', 'Tools', 'triwnymo.py',
'w9xpopen.exe']
```

... όμως ίσως η πιο χρήσιμη συνάρτηση στον os είναι η system(), η οποία μας δίνει πρόσβαση σε όλες τις εντολές του Command Line (DOS prompt). Οι εντολές δίνονται σε ένα αλφαριθμητικό ως παράμτερος στην system(). Πχ:

Αντιγραφές αρχείων:
```
>>> system("copy my_first_text_file.txt my_third_text_file.txt")
0
>>> system("copy my_third_text_file.txt my_forth_text_file.txt")
0
```

Μετονομασία:
```
>>> system("ren my_forth_text_file.txt my_fourth_text_file.txt")
0
```

Διαγραφή:
```
>>> system("del my_third_text_file.txt")
0
```

Όλες αυτές οι συναρτήσεις επιστρέφουν 0 όταν έχουν εκτελεστεί επιτυχώς.

8.4 Συμπίεση Αρχείων

Με άλλες βιβλιοθήκες της Python μπορούμε να χειριστούμε αρχεία ειδικών τύπων. Για παράδειγμα, οι βιβλιοθήκες zipfile και zlib επιτρέπουν τη δημιουργία και ανάγνωση συμπιεσμένων αρχείων. Να πώς συμπιέζουμε αρχεία στα προγράμματά μας:

```
>>> from zipfile import *
```

```
>>> from zlib import *
>>> fz=ZipFile("AllFiles.zip","w",ZIP_DEFLATED)
```

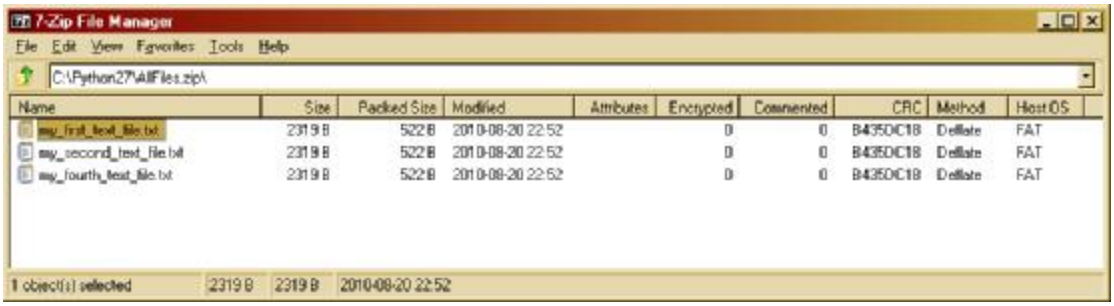
Η μέθοδος ZipFile() λειτουργεί όπως η open(). Εδώ ανοίξαμε ένα αρχείο με όνομα "AllFiles.zip" για εγγραφή ("w"). Η σταθερά ZIP_DEFLATED δηλώνει τον αλγόριθμο συμπίεσης που θα χρησιμοποιηθεί. Αν την παραλείψουμε, θα γίνει καταχώρηση των αρχείων χωρίς συμπίεση. Στη συνέχεια γράφουμε μέσα στο fz όσα αρχεία θέλουμε να περιέχει:

```
>>> fz.write("my_first_text_file.txt")
>>> fz.write("my_second_text_file.txt")
>>> fz.write("my_fourth_text_file.txt")
```

... και κλείνουμε το αρχείο:

```
>>> fz.close()
```

Τώρα το συμπιεσμένο αρχείο AllFiles.zip μπορεί να διαβαστεί από οποιαδήποτε εφαρμογή διαχειρίζεται συμπιεσμένα αρχεία:



8.5 Αποστολή e-mail

Τα αρχεία κειμένου που δημιουργούμε μπορούμε να τα στέλνουμε με e-mail μέσα από την Python, χωρίς να μεσολαβεί κάποια εφαρμογή αποστολής e-mail. Η βιβλιοθήκη που χρησιμοποιούμε λέγεται smtplib και όλες οι δυνατότητές της εξηγούνται στο help της Python. Εδώ χρησιμοποιούμε μόνο όσες μεθόδους χρειάζονται για να στείλουμε ένα απλό e-mail:

```
>>> from smtplib import *
>>> server=SMTP("patreas.upatras.gr")
```

Η μέθοδος SMTP() παίρνει ως παράμετρο το όνομα του server ο οποίος θα στείλει το e-mail μας. Στη συνέχεια, βάζουμε σε μια μεταβλητή το μήνυμά μας. Πχ. Αν έχει γραφτεί σε κάποιο αρχείο:

```
>>> file=open("my_first_text_file.txt","r")
>>> message=file.read()
>>> file.close()
```

... ή, αν το γράφουμε κατευθείαν:

```
>>> message="""Αυτό είναι ένα e-mail που θα σταλεί
από τον: ecexxxx@ece.upatras.gr
στον: ecexxxx@ece.upatras.gr
μέσω της γλώσσας Python
με τη βιβλιοθήκη smtplib."""
```

Προσέξτε ότι μπορούμε να γράψουμε ένα αλφαριθμητικό που επεκτείνεται σε πολλές γραμμές, αν το κλείσουμε σε τριπλά εισαγωγικά. Η Python θα διατηρήσει τη μορφοποίησή του.

Και μετά η μέθοδος sendmail() αναλαμβάνει να στείλει το e-mail:

```
>>> server.sendmail("ecexxxx@ece.upatras.gr",
"ecexxxx@ece.upatras.gr", message)
```

Η πρώτη παράμετρος είναι το e-mail του αποστολέα, η δεύτερη του παραλήπτη και η τρίτη το μήνυμα. Με την εκτέλεση δεν θα δούμε κάποιο μήνυμα (εκτός αν γράψουμε κάτι λάθος). Αν θέλουμε να διαβάσουμε τα ενδιάμεσα ενημερωτικά μηνύματα του server, πριν την sendmail() πρέπει να δώσουμε:

```
>>> server.set_debuglevel(1)
```

Και τότε μόλις δώσουμε την sendmail() θα δούμε κάτι τέτοιο:

```
send: 'ehlo think.wcl.ee.upatras.gr\r\n'
reply: '250-patreas.upatras.gr patreas.upatras.gr Universty of
Patras\r\n'
reply: '250-PIPELINING\r\n'
reply: '250-SIZE 10000000\r\n'
reply: '250 8BITMIME\r\n'
reply: retcode (250); Msg: patreas.upatras.gr patreas.upatras.gr
Universty of Patras
PIPELINING
SIZE 10000000
8BITMIME
send: 'mail FROM:<sender@upatras.gr> size=400\r\n'
reply: '250 ok\r\n'
reply: retcode (250); Msg: ok
send: 'rcpt TO:<recipient@upatras.gr>\r\n'
reply: '250 ok\r\n'
reply: retcode (250); Msg: ok
send: 'data\r\n'
reply: '354 Please start mail input.\r\n'
reply: retcode (354); Msg: Please start mail input.
data: (354, 'Please start mail input.')
send: 'test\r\n.\r\n'
reply: '250 Mail queued for delivery.\r\n'
reply: retcode (250); Msg: Mail queued for delivery.
data: (250, 'Mail queued for delivery.')
{}
```

Σε κάθε περίπτωση,μόλις τελειώσουμε την αποστολή των e-mail θα πρέπει να κλείσουμε τη σύνδεσή μας με την εντολή:

```
>>> server.quit()
send: 'quit\r\n'
reply: '221 Closing connection. Good bye.\r\n'
```

```
reply: retcode (221); Msg: Closing connection. Good bye.
```

8.6 Ασκήσεις

Αφού τρέξετε τα παραδείγματα των ενοτήτων που προηγήθηκαν, εκπονήστε τις παρακάτω ασκήσεις. Στείλτε τα αρχεία σας (προγράμματα και αποτελέσματα) ως ένα ενιαίο συμπιεσμένο αρχείο.

Άσκηση #1

Γράψτε πρόγραμμα σε Python που θα αποθηκεύει όλα τα ετήσια ημερολόγια του 21ου αιώνα (δηλαδή από το έτος 2001 έως το 2100) σε ξεχωριστά αρχεία με ονόματα της μορφής xxxx.txt, όπου xxxx το έτος (πχ. 2001.txt, 2002.txt κλπ). Στη συνέχεια, τα 100 αρχεία που θα δημιουργηθούν θα τα συμπιέζει σε ένα αρχείο ZIP με όνομα 21century.zip και μετά θα τα σβήνει από τον σκληρό δίσκο.

Άσκηση #2

Η ημερομηνία της Κυριακής του Πάσχα για τους ορθόδοξους είναι 3 Απριλίου + p ημέρες, όπου το p εξαρτάται από το έτος και υπολογίζεται από την εξής διαδικασία:

$p = v_1+v_2$
$v_1 = (6*v_2+m_4+m_2) \bmod 7$
$v_2 = (16+m_{19}) \bmod 30$
$m_2 = 2*(\text{ΕΤΟΣ} \bmod 4)$
$m_4 = 4*(\text{ΕΤΟΣ} \bmod 7)$
$m_{19} = 19*(\text{ΕΤΟΣ} \bmod 19)$

Με (x mod y) συμβολίζεται το υπόλοιπο της ακέραιας διαίρεσης x/y. Με βάση τα παραπάνω γράψτε πρόγραμμα σε Python που θα υπολογίζει τις ημερομηνίες του Πάσχα για κάθε έτος του 21ου αιώνα και θα τις στέλνει με ένα e-mail στον λογαριασμό σας. Το κείμενο του e-mail θα είναι της μορφής:

```
Η ημερομηνία του Πάσχα για το 2001 είναι 15 Απριλίου.
Η ημερομηνία του Πάσχα για το 2002 είναι 5 Μαίου.
Η ημερομηνία του Πάσχα για το 2003 είναι 27 Απριλίου.
...
```

Υπόδειξη: Αν στα προγράμματά σας φορτώνετε πολλές βιβλιοθήκες, ενδεχομένως κάποια ονόματα συναρτήσεων/μεθόδων να συμπίπτουν. Αν διαπιστώσετε κάτι τέτοιο τότε φορτώστε τις βιβλιοθήκες με την εντολή «import libraryname», αντί για «from libraryname import *». Στη συνέχεια, για να προσδιορίσετε τη συνάρτηση, αντί να γράφετε σκέτο το όνομά της, θα γράφετε (με dot notation) libraryname.function()

Εργαστήριο 9: Πέμπτη Άσκηση Προγραμματισμού Python

9.1 Γενικά

Η άσκηση αυτή επιχειρεί μια εισαγωγή στον αντικειμενοστραφή προγραμματισμό. Αφορά **αντικείμενα**, **κλάσεις**, **μεθόδους** και **κληρονομικότητα**.

9.2 Αντικείμενα και Κλάσεις

Θυμάστε σε προηγούμενη άσκηση που χρειάστηκε να κάνουμε υπολογισμούς με μιγαδικούς αριθμούς πόσο βολικό ήταν που η Python διέθετε αυτή τη δυνατότητα; Βέβαια, ακόμα κι αν δεν την διέθετε, πάλι θα μπορούσαμε να κάνουμε τις ίδιες πράξεις αναπαριστώντας τους μιγαδικούς με κάποιον άλλο τρόπο (πχ. ως λίστες ή ως λεξικά δύο στοιχείων) απλώς το πρόγραμμα θα ήταν λίγο πιο περίπλοκο. Συχνά τα προγράμματά μας απλοποιούνται πολύ όταν ορίζουμε κατάλληλες σύνθετες δομές (συνήθως αρκετά πιο σύνθετες από έναν μιγαδικό αριθμό) οι οποίες λέγονται **αντικείμενα** (**objects**). Γύρω από τη χρήση των αντικειμένων υπάρχει μια ολόκληρη σχεδιαστική φιλοσοφία που λέγεται **αντικειμενοστρεφής προγραμματισμός** (**object oriented programming**).

Ας δούμε ένα παράδειγμα. Έστω ότι θέλουμε να προγραμματίσουμε ένα παιχνίδι με τραπουλόχαρτα[12]. Για καθένα θα χρειαστεί να καταγράψουμε την αξία του (πχ. 5, 8, Q, A, κλπ), το σύμβολό του (μπαστούνι, κούπα, σπαθί, καρώ) και ίσως ακόμα κάποια επιπλέον στοιχεία που χρειάζονται για κάποια παιχνίδια, όπως το χρώμα (μαύρο ή κόκκινο) και το αν είναι φιγούρα ή όχι. Όλες αυτές τις πληροφορίες μπορούμε να τις διαχειριστούμε με πολλούς τρόπους, όμως φανταστείτε πόσο θα διευκολυνόμασταν αν ορίζαμε μεταβλητές που δεν θα ήταν ακέραιες, πραγματικές ή μιγαδικές, αλλά τραπουλόχαρτα (αντικείμενα), το καθένα με όλες τις πληροφορίες του.

Για να δημιουργήσουμε αντικείμενα πρέπει πρώτα να προγραμματίσουμε μια «μηχανή» που θα τα δημιουργεί. Αυτή λέγεται **κλάση** (**class**) και στην πιο απλή μορφή της είναι κάτι τέτοιο:

```
>>> class card():
      pass
```

Η συντακτική της δομή είναι παρόμοια με τη δομή της def. Βάζουμε πρώτα τη λέξη class, μετά το όνομα της κλάσης με παρενθέσεις, άνω-κάτω τελεία και από την

[12] Εξυπακούεται ότι οι διδάσκοντες σε καμμία περίπτωση δεν ενθαρρύνουν την χαρτοπαιξία. Τα τραπουλόχαρτα και οι τράπουλες χρησιμοποιούνται εδώ επειδή αποτελούν ιδιαίτερα επιτυχημένα παραδείγματα για την κατανόηση των εξεταζόμενων εννοιών, όπως άλλωστε συμβαίνει και σε πλείστα επιστημονικά συγγράμματα θεωρίας πιθανοτήτων και στατιστικής.

επόμενη γραμμή αρχίζουμε τις εντολές της κλάσης. Εδώ γράψαμε μόνο την εντολή pass. Στην Python, η pass είναι μια εντολή η οποία δεν κάνει τίποτα. Χρησιμοποιείται όπου είναι απαραίτητο να γράψουμε κάτι για συντακτικούς λόγους κι εμείς δεν θέλουμε να βάλουμε τίποτε άλλο.

Η παραπάνω εντολή λοιπόν δημιούργησε απλώς μια κλάση με όνομα card και τίποτε άλλο. Αυτό και μόνο αρκεί για να δημιουργήσουμε μερικά αντικείμενα (τραπουλόχαρτα):

```
>>> c1=card()
>>> c2=card()
```

Και πάλι, η ομοιότητα με τις συναρτήσεις είναι εμφανής. Τα c1 και c2 είναι μεταβλητές που παίρνουν ως τιμή αυτό που επιστρέφει η card(). Η card() συμπεριφέρεται σα μια μηχανή που φτιάχνει τραπουλόχαρτα. Όμως αυτά τα τραπουλόχαρτα προς το παρόν είναι άδεια. Για να γράψουμε πάνω τους χρησιμοποιούμε dot notation και ορίζουμε χαρακτηριστικά (ιδιότητες) και τιμές:

```
>>> c1.value='5'
>>> c1.symbol='d'
>>> c1.color='R'
>>> c1.fig=False

>>> c2.value='K'
>>> c2.symbol='c'
>>> c2.color='B'
>>> c2.fig=True
```

Έτσι ορίσαμε ότι το τραπουλόχαρτο c1 είναι 5 καρώ (d=diamond), κόκκινο (R=red) και όχι φιγούρα (c1.fig=False), ενώ το c2 είναι ρήγας (K=King) σπαθί (c=club), μαύρο (B=black) και φιγούρα (c2.fig=True).

Και φυσικά μπορούμε να δούμε και να χρησιμοποιήσουμε πλέον αυτές τις τιμές:

```
>>> print c1.value, c1.symbol, c2.value, c2.symbol
5 d K c
```

Αυτή είναι η βασική ιδέα. Και τώρα αρχίζει μια σειρά βελτιώσεων. Για παράδειγμα, παρατηρήστε ότι σύμφωνα με τα παραπάνω για κάθε νέο τραπουλόχαρτο που δημιουργούμε θα πρέπει να δίνουμε πέντε εντολές. Μία για να δημιουργηθεί το αντικείμενο και τέσσερεις για να καταχωρθούν οι τιμές. Δεν θα ήταν καλύτερο οι τιμές να πέρναγαν ως παράμετροι στην πρώτη εντολή; Ή κάποιες από αυτές (color, fig) να υπολογίζονταν αυτόματα από άλλες (symbol, value); Να πώς μπορούμε να το πετύχουμε:

```
>>> class card():
    def __init__(self,val,sym):
        self.value=val
        self.symbol=sym
        if self.symbol in "sc": self.color='B'
        else: self.color='R'
        if self.value in "JQK": self.fig=True
        else: self.fig=False
```

```
>>> c1=card('5','d')
>>> print c1.value, c1.symbol, c1.color, c1.fig
5 d R False
```

Αλλάξαμε τον ορισμό της κλάσης αντικαθιστώντας την pass με τον ορισμό μιας συνάρτησης. Και δείτε το αποτέλεσμα: με μία μόνο εντολή δημιουργήσαμε το 4 καρώ δύο από τις τιμές υπολογίστηκαν αυτόματα.

Ας εξετάσουμε τη συνάρτηση που τα έκανε όλα αυτά. Το όνομά της είναι __init__ (δύο underscores, init, δυο underscores) και ανήκει σε μια ομάδα ειδικών συναρτήσεων με παρόμοια ονόματα[13], οι οποίες καθορίζουν πώς θα συμπεριφέρονται τα αντικείμενα που θα δημιουργεί η κλάση. Η __init__ καθορίζει τί θα γίνεται μόλις δημιουργείται ένα αντικείμενο. Επειδή δεν μπορεί να γνωρίζει το όνομα που θα διαλέξουμε για το αντικείμενο (πχ. αν το ονομάσουμε c1 ή c2 ή κάπως αλλιώς), αναφέρεται σε αυτό με την προεπιλεγμένη ονομασία “self”. Το “self” μπαίνει πάντα ως πρώτο όρισμα στην παρένθεση της __init__. Στην περίπτωσή μας η παρένθεση περιέχει και άλλα δυο ορίσματα, τα val και sym. Με αυτόν τον τρόπο, όταν εμείς δώσουμε την εντολή c1=card(‘5’,’d’), το self θα γίνει c1, το val θα γίνει ‘5’ και το sym θα γίνει ‘d’. Στη συνέχεια θα εκτελεστούν οι δυο πρώτες εντολές της __init__ και θα πάρουν τιμή τα c1.value και c1.symbol. Και ανάλογα με αυτές τις τιμές, οι εντολές if – else που ακολουθούν υπολογίζουν τις τιμές των c1.color και c1.fig.

Σημαντική λεπτομέρεια: Προσέξτε ότι ενώ κατά την κλήση c1=card(‘5’,’d’) οι τιμές val και sym γράφονται στην παρένθεση της card, κατά τον ορισμό της κλάσης αυτές οι παράμετροι τοποθετούνται στην παρένθεση της __init__, ενώ η παρένθεση της κλάσης card μένει κενή! Αργότερα θα δούμε ότι η παρένθεση των κλάσεων χρησιμοποιείται για άλλο σκοπό (για να ορίσει κληρονομικότητα από άλλες κλάσεις).

Προσέξτε κάτι άλλο τώρα. Ενώ μπορούμε να τυπώνουμε τις τιμές όλων των ιδιοτήτων των αντικειμένων που ορίζουμε:

```
>>> print c1.value, c1.symbol, c1.color, c1.fig
5 d R False
```

... αν προσπαθήσουμε να τυπώσουμε τα ίδια τα αντικείμενα, παίρνουμε μόνο αναφορές στις διευθύνσεις μνήμης στις οποίες βρίσκονται:

```
>>> print c1
<__main__.card instance at 0x011CAAF8>
```

Κάθε αντικείμενο μπορούμε να το φανταστούμε σαν ένα κουτί που περιέχει τιμές για διάφορες ιδιότητες, ενδεχομένως συναρτήσεις/μεθόδους, ακόμα και άλλα αντικείμενα/κουτιά. Συνεπώς αν δεν ορίσουμε με κάποιον τρόπο τί ακριβώς σημαίνει η εκτύπωση ενός αντικειμένου (πχ. εκτύπωση όλων ή κάποιων από τα περιεχόμενά του, και με ποιά σειρά) προφανώς δεν μπορούμε να έχουμε την απαίτηση από την Python να το τυπώσει. Αυτός ο ορισμός γίνεται με την ειδική συνάρτηση __str__. Δείτε τον νέο ορισμό της κλάσης που πλέον έχουμε προσθέσει και την __str__ κάτω από την __init__:

```
>>> class card():
```

[13] __str__, __del__, __new__, __cmp__, κλπ. Δείτε στο on-line help για τον πλήρη κατάλογο.

```
    def __init__(self,val,sym):
        self.value=val
        self.symbol=sym
        if self.symbol in "sc": self.color='B'
        else: self.color='R'
        if self.value in "JQK": self.fig=True
        else: self.fig=False
    def __str__(self):
        return self.value+self.symbol
```

Η __str__ δεν χρειάζεται άλλη παράμετρο παρά μόνο το αντικείμενο (self). Πρέπει πάντα να επιστρέφει ένα αλφαριθμητικό (string). Εδώ την βάλαμε να επιστρέφει το self.value+self.symbol. Συνεπώς μπορούμε πλέον να δώσουμε:

```
>>> c1=card('5','d')
>>> print c1
5d
```

Και φυσικά, εκτός από τις ειδικές συναρτήσεις μπορούμε να προσθέσουμε μέσα στην κλάση και δικές μας οι οποίες θα καλούνται ως μέθοδοι με dot notation. Για παράδειγμα:

```
>>> class card():
    def __init__(self,val,sym):
        self.value=val
        self.symbol=sym
        if self.symbol in "sc": self.color='B'
        else: self.color='R'
        if self.value in "JQK": self.fig=True
        else: self.fig=False
    def __str__(self):
        return self.value+self.symbol
    def detailed_info(self):
        print 'Αξία=',self.value, '    Σύμβολο=',self.symbol
        print 'Χρώμα=',self.color, '    Φιγούρα=',self.fig

>>> c1=card('5','d')
>>> c1.detailed_info()
Αξία= 5     Σύμβολο= d
Χρώμα= R     Φιγούρα= False
```

Και τώρα μπορούμε να δημιουργήσουμε 52 τραπουλόχαρτα για να τα χρησιμοποιήσουμε στο παιχνίδι μας, ή να κάνουμε κάτι πολύ καλύτερο. Τα τραπουλόχαρτα περιέχονται σε τράπουλες. Ας φτιάξουμε λοιπόν μια νέα κλάση που θα δημιουργεί πλήρεις τράπουλες με όλα τα τραπουλόχαρτα. Θα δούμε έτσι πώς μια κλάση μπορεί να καλέσει αντικείμενα άλλης κλάσης. Δείτε εδώ τον πλήρη κώδικα της και στη συνέχεια θα εξηγήσουμε ένα-ένα τα μέρη του:

```
import random

class deck():
    values="A23456789TJQK"  # Όλες οι αξίες
    symbols="shcd"          # Όλα τα σύμβολα
    content=[]    # Χαρτιά που βρίσκονται στην τράπουλα
    pile=[]       # Χαρτιά που βγήκαν από την τράπουλα
    def __init__(self):
        self.content=[]
```

```
        self.pile=[]
        for s in self.symbols:
            for v in self.values:
                c=card(v,s)
                self.content.append(c)
    def __str__(self):
        s=""
        cntr=0
        for i in self.content:
            s=s+str(i)+" "
            cntr=cntr+1
            if cntr%13==0: s=s+'\n'
        if s[-1]<>'\n': s=s+'\n'
        s=s+str(len(self.content))+"-"+str(len(self.pile))
        return s
    def shuffle(self):
        random.shuffle(self.content)
    def draw(self):
        if len(self.content)<1: return "empty"
        c=self.content[0]
        self.content=self.content[1:]
        self.pile.append(c)
        return c
    def collect(self):
        self.content=self.content+self.pile
        self.pile=[]
```

Στην αρχή του κώδικα, αμέσως μετά την import και το όνομα της κλάσης deck βλέπουμε τέσσερεις μεταβλητές. Η values περιέχει σε ένα αλφαριθμητικό όλες (13) τις πιθανές αξίες ενός τραπουλόχαρτου (για να έχουμε ομοιομορφία αργότερα στην εκτύπωση, χρησιμοποιούμε για κάθε αξία μόνο ένα χαρακτήρα, έτσι το 10 γράφτηκε ως T). Η symbols περιέχει με τον ίδιο τρόπο τα 4 σύμβολα (s=spade/μπαστούνι, h=heart/κούπα, c=club/σπαθί, d=diamond/καρώ). Αργότερα (στην __init__) θα συνδυάσουμε τις 13 αξίες με τα 4 σύμβολα για να φτιάξουμε όλα τα 13x4=52 τραπουλόχαρτα. Και θα χρειαστεί να τα αποθηκεύσουμε σε μία λίστα. Αυτή είναι η content που προς το παρόν είναι κενή. Γεμίζει κι αυτή στην __init__. Τέλος, έχουμε ορίσει μια άλλη λίστα, την pile. Αυτή χρησιμοποιείται ως μνήμη για να «θυμάται» η τράπουλα ποιά φύλλα έχει μοιράσει. Αυτή η μνήμη δεν είναι απαραίτητη, όμως διευκολύνει όταν τελειώσει ένα παιχνίδι και θα πρέπει να ξανασυγκεντρωθούν όλα τα χαρτιά στην τράπουλα για να αρχίσει το επόμενο. Αντί να τα μαζεύει από όλους τους παίκτες, αρκεί να τα μαζέψει από την pile. Αυτό το κάνει η μέθοδος collect(). Παρατηρήστε τώρα ότι αυτές οι τέσσερεις μεταβλητές είναι μέσα στην κλάση deck αλλά έξω από τις μεθόδους. Αυτό σημαίνει ότι για να τις προσπελάσει μια μέθοδος θα πρέπει να τις καλέσει με dot notation από το αντικείμενο self, πχ. self.values.

Πάμε τώρα να δούμε την __init__. Έχει μόνο ένα όρισμα, το self. Αυτό σημαίνει ότι για να δημιουργήσουμε μια τράπουλα δεν δίνουμε καμμία παράμετρο στην deck. Γράφουμε απλώς κάτι τέτοιο:

```
>>> d=deck()
```

Το πρώτο for περνάει από όλα τα σύμβολα, το δεύτερο από όλες τις αξίες, και προσέξτε πώς η c=card(v,s) δημιουργεί ένα τραπουλόχαρτο το οποίο μπαίνει στο τέλος της content με την κλήση self.content.append(c). Όταν τελειώσουν οι δύο for

όλα τα τραπουλόχαρτα θα είναι με τη σειρά στη λίστα content. Πράγματι, αν τυπώσουμε την d θα δούμε:

```
>>> print d
As 2s 3s 4s 5s 6s 7s 8s 9s Ts Js Qs Ks
Ah 2h 3h 4h 5h 6h 7h 8h 9h Th Jh Qh Kh
Ac 2c 3c 4c 5c 6c 7c 8c 9c Tc Jc Qc Kc
Ad 2d 3d 4d 5d 6d 7d 8d 9d Td Jd Qd Kd
52-0
```

Τον τρόπο εκτύπωσης καθορίζει η __str__. Επιστρέφει ένα αλφαριθμητικό s στο οποίο έχουμε βάλει όλα τα στοιχεία της self.content, 13 ανά γραμμή, και στο τέλος το μέγεθος της content και το μέγεθος της pile. Εδώ αξίζει προσοχή η s=s+str(i)+“ ”. Το i είναι τραπουλόχαρτο, επομένως το str(i) είναι το string που θα επέστρεφε η print i, δηλαδή αυτό που έχουμε ορίσει στην __str__ της κλάσης card.

Και βέβαια, δεν παίζουμε αν πρώτα δεν ανακατέψουμε την τράπουλα. Αυτό το κάνει η μέθοδος shuffle, η οποία καλεί μια μέθοδο με όμοιο όνομα από το module random (για αυτό το λόγο κάναμε import την random στην αρχή του κώδικά μας). Προσέξτε ότι έχουμε δυο μεθόδους με το ίδιο όνομα. Η μία είναι η self.shuffle() και η άλλη είναι η random.shuffle(). Τις ξεχωρίζουμε με dot notation. Τώρα αρκεί να γράψουμε:

```
>>> d.shuffle()
```

... και πλέον έχουμε ανακατέψει την τράπουλα:

```
>>> print d
8d 3c 2c Ts Ks 9c Jc 5h 2h Kd 3h Td 5c
5d 7s 8h 9d Ad 5s Tc 7c 4c 6c 7d Qh Qs
3d Ah Ac Js Qc 4s 3s 6d Jd As Kh 4h 9h
Th 9s Jh 6h 4d 2s 7h 2d Qd Kc 8c 8s 6s
52-0
```

Τί άλλο θέλουμε να κάνουμε με μια τράπουλα; Να μοιράσουμε φύλλα. Αυτό κάνει η επόμενη μέθοδος draw() που έχει προγραμματιστεί στην κλάση. Κάθε φορά που καλείται, η draw() επιστρέφει το φύλλο που βρίσκεται πρώτο (self.content[0]) στην τράπουλα, εκτός αν είναι άδεια, οπότε επιστρέφει «empty». Το φύλλο αφαιρείται από τη λίστα content και ένα αντίγραφό του αποθηκεύεται στην pile.

```
>>> h=d.draw()
>>> print h
8d
>>> print d
3c 2c Ts Ks 9c Jc 5h 2h Kd 3h Td 5c 5d
7s 8h 9d Ad 5s Tc 7c 4c 6c 7d Qh Qs 3d
Ah Ac Js Qc 4s 3s 6d Jd As Kh 4h 9h Th
9s Jh 6h 4d 2s 7h 2d Qd Kc 8c 8s 6s
51-1
```

Η τελευταία μέθοδος, η collect() είναι πολύ απλή. Προσθέτει την pile στο τέλος της content και μετά αδειάζει την pile. Με αυτόν τον τρόπο ξανασυγκεντρώνει όλα τα φύλλα της τράπουλας για να ξεκινήσει νέο παιχνίδι.

```
>>> for i in range(10): h=d.draw()
```

```
>>> print d
Td 5c 5d 7s 8h 9d Ad 5s Tc 7c 4c 6c 7d
Qh Qs 3d Ah Ac Js Qc 4s 3s 6d Jd As Kh
4h 9h Th 9s Jh 6h 4d 2s 7h 2d Qd Kc 8c
8s 6s
41-11
>>> d.collect()
>>> print d
Td 5c 5d 7s 8h 9d Ad 5s Tc 7c 4c 6c 7d
Qh Qs 3d Ah Ac Js Qc 4s 3s 6d Jd As Kh
4h 9h Th 9s Jh 6h 4d 2s 7h 2d Qd Kc 8c
8s 6s 8d 3c 2c Ts Ks 9c Jc 5h 2h Kd 3h
52-0
```

Όμως χρειάζεται λίγη προσοχή κατά τη χρήση. Επειδή η collect() από μόνη της δεν αφαιρεί τα φύλλα από τους παίκτες, θα πρέπει να θυμηθούμε να αδειάσουμε τα χέρια των παικτών με ξεχωριστό κώδικα.

Μέχρι τώρα είπαμε πολλά για χαρτιά και τράπουλες, όμως δεν είπαμε τίποτα για το παιχνίδι που θα προγραμματίσουμε. Αυτό δείχνει ότι οι κλάσεις που φτιάξαμε είναι ανεξάρτητες από το παιχνίδι και φυσικά θα μπορούσαν να χρησιμοποιηθούν σε οποιοδήποτε παιχνίδι με τράπουλα. Αυτό σημαίνει ότι αξίζει να σώσουμε τις κλάσεις μας σε ξεχωριστό αρχείο και να τις συμπεριλάβουμε με import ως module[14] από το αρχείο του κυρίως προγράμματος που θα φτιάξουμε. Βάλτε λοιπόν τις δύο κλάσεις σε ένα αρχείο, πρώτα την card(), μετά την deck() και μην ξεχάσετε την εντολή import random στην αρχή. Σώστε το αρχείο με το όνομα playing_cards.py στον κατάλογο που σώζετε όλα σας τα προγράμματα Python, και οι κλάσεις θα είναι πάντα διεθέσιμες μόλις γράψουμε “from playing_cards import *”.

9.3 Χρήση Αντικειμένων (Ο Κώδικας του Παιχνιδιού)

Το παιχνίδι που θα προγραμματίσουμε έχει τους εξής κανόνες:

1. Παίζεται με δυο παίκτες (ο άνθρωπος εναντίον του υπολογιστή) και μία τράπουλα.
2. Στην αρχή του παιχνιδιού μοιράζονται 7 φύλλα σε κάθε παίκτη και 1 φύλλο ανοικτό στο τραπέζι.
3. Με τυχαίο τρόπο καθορίζεται ποιός παίκτης θα παίξει πρώτος.
4. Ο κάθε παίκτης, όταν είναι η σειρά του να παίξει θα πρέπει να ρίξει στο τραπέζι ένα από τα φύλλα του, αρκεί να συμφωνεί σε αξία ή σύμβολο με το προηγούμενο φύλλο που έπεσε στο τραπέζι (πχ. αν στο τραπέζι είναι το 5 κούπα, ο παίκτης θα πρέπει να ρίξει ή 5 ή κούπα). Αν δεν έχει κατάλληλο φύλλο θα πρέπει να τραβήξει από την τράπουλα (όσες φορές χρειαστεί) μέχρι να βρει.
5. Νικητής του παιχνιδιού είναι όποιος πετάξει πρώτος όλα τα φύλλα του.
6. Αν κάποιος παίκτης προσπαθήσει να τραβήξει φύλλο αλλά η τράπουλα έχει τελειώσει, το παιχνίδι σταματά και νικητής ανακηρύσσεται όποιος κρατάει τα λιγώτερα φύλλα.

[14] Αυτό άλλωστε είναι όλα τα modules που χρησιμοποιήσαμε ως τώρα. Συλλογές κλάσεων που περιέχουν μεθόδους και δεδομένα. Δείτε για παράδειγμα με έναν text editor τα περιεχόμενα του αρχείου random.py στον φάκελο Lib της Python.

Για τον προγραμματισμό του παιχνιδιού θα χρησιμοποιήσουμε το module playing_cards που φτιάξαμε προηγουμένως, 5 συναρτήσεις και λίγες γραμμές κώδικα για το κυρίως πρόγραμμα. Επίσης θα έχουμε την ευκαιρία να γνωρίσουμε την έννοια των **καθολικών μεταβλητών** (**global variables**) αφού θα χρησιμοποιήσουμε τέσσερεις καθολικές μεταβλητές στο πρόγραμμά μας: την d που θα είναι μια τράπουλα, και τρείς λίστες, τις table, computer_hand, human_hand, που θα περιέχουν τα τραπουλόχαρτα που βρίσκονται στο τραπέζι, στα «χέρια» του υπολογιστή, και στα χέρια του ανθρώπου-παίκτη, αντίστοιχα. Θα εξηγήσουμε τη λειτουργία όλων των συνιστωσών του προγράμματος βήμα-βήμα, καθώς θα συμπληρώνουμε τον κώδικα στο αρχείο.

Κατ'αρχήν, ανοίγουμε ένα νέο αρχείο και γράφουμε στην αρχή του την εντολή:

```
from playing_cards import *
```

Αυτό θα μας δώσει τη δυνατότητα να κάνουμε χρήση[15] των κλάσεων card και deck.

Η πρώτη συνάρτηση που θα προσθέσουμε καθορίζει τον τρόπο που θα «σκέφτεται» και θα παίζει ο υπολογιστής:

```
def computer_plays():
    global d, table, computer_hand
    target_val=table[-1].value
    target_sym=table[-1].symbol
    for c in computer_hand:
        if c.value==target_val or c.symbol==target_sym:
            print "Ο Η/Υ ρίχνει",c
            computer_hand.remove(c)
            table.append(c)
            if len(computer_hand)<1: return "computer wins"
            else: return "continue"
    new_card=d.draw()
    if new_card=="empty": return "count"
    else:
        print "Ο Η/Υ τραβάει φύλλο."
        computer_hand.append(new_card)
        return computer_plays()
```

Στην πρώτη γραμμή μετά το όνομα της συνάρτησης βλέπουμε την εντολή global και τα ονόματα των μεταβλητών d, table και computer_hand. Αυτό σημαίνει ότι οι μεταβλητές αυτές είναι καθολικές. Δηλαδή είναι ορισμένες στο κυρίως πρόγραμμα και η συνάρτηση τις διαβάζει από εκεί. Αυτό σημαίνει επίσης ότι αν η συνάρτηση αλλάξει την τιμή τους αυτή η αλλαγή θα μεταφερθεί και στο κυρίως πρόγραμμα και σε κάθε άλλη συνάρτηση που τις χρησιμοποιεί. Για να πετύχουμε παρόμοια συμπεριφορά με συνηθισμένες (τοπικές) μεταβλητές θα έπρεπε να τις περνάμε ως ορίσματα στη συνάρτηση (για να τις διαβάσει) και μετά να επιστρέφουμε τις τιμές τους με το όνομα της συνάρτησης (για να περάσει η νέα τιμή στο κυρίως πρόγραμμα). Έτσι, τώρα το τελευταίο φύλλο στη λίστα table (το table[-1]) στο οποίο αναφέρονται οι δυο επόμενες εντολές είναι το ίδιο και κοινό παντού κι όχι κάποιο αντίγραφο ή συνωνυμία.

[15] Αν τυχόν η Python δεν μπορέσει να διαβάσει το αρχείο προσθέστε στην πρώτη γραμμή του τη δήλωση κωδικοσελίδας (πχ. `# -*- coding: cp1253 -*-`).

Ως προς τη λογική της συνάρτησης τώρα, στις μεταβλητές target_val και target_sym αποθηκεύονται η αξία και το σύμβολο αντίστοιχα του τελευταίου φύλλου που έπεσε στο τραπέζι. Στη συνέχεια, η for ελέγχει με τη σειρά ένα-ένα τα χαρτιά στο computer_hand και αν κάποιο βρεθεί να έχει ίδια αξία με το target_val ή ίδιο σύμβολο με το target_sym τότε εκτυπώνεται το μήνυμα και το χαρτί φεύγει από το computer_hand με την remove() και μεταφέρεται στο table με την append(). Μετά γίνεται ένας έλεγχος μήπως ο Η/Υ έχει περάξει όλα τα χαρτιά του και αν αυτό συμβαίνει η συνάρτηση επιστρέφει με τιμή το string "computer wins". Διαφορετικά επιστρέφει με την τιμή "continue".

Αν τελειώσει η for και δεν βρεθεί κατάλληλο φύλλο, τότε η d.draw() επιχειρεί να τραβήξει ένα νέο φύλλο από την τράπουλα. Μια if ελέγχει μήπως η τράπουλα ήταν άδεια και δεν επέστρεψε φύλλο και σε αυτήν την περίπτωση η συνάρτηση επιστρέφει ως τιμή τη λέξη "count". Σε αντίθετη περίπτωση η else τυπώνει το μήνυμα ότι ο Η/Υ τραβάει φύλλο, προσθέτει το φύλλο στο computer_hand και τέλος εκτελείται η return computer_plays(). Αυτό το τελευταίο θέλει λίγη προσοχή. Σημαίνει «επέστρεψε την τιμή που θα σου δώσει μια νέα κλήση της computer_plays()». Βλέπουμε λοιπόν ότι η συνάρτηση καλεί τον εαυτό της, δηλαδή είναι όπως λέμε **αναδρομική** (**recursive**). Αυτός είναι ένας βολικός[16] τρόπος να ξεκινήσει η διαδικασία από την αρχή ώστε να ελεγχθεί το νέο computer_hand.

Τελικά η computer_plays() επιστρέφει ένα αλφαριθμητικό με την κατάσταση της παρτίδας μέχρι εκείνη τη στιγμή. Το αλφαριθμητικό θα είναι ή "continue", ή "computer wins", ή "count", σηματοδοτώντας αν το παιχνίδι συνεχίζεται ή τελείωσε και με ποιόν τρόπο. Η συνάρτηση πάντα θα επιστρέφει κάποια από αυτές τις τιμές, κι αν αυτό δεν συμβεί με την πρώτη κλήση, μετά καλεί τον εαυτό της για κάθε νέο φύλλο που τραβάει ώστε να επαναληφθεί η διαδικασία.

Ακολουθεί η συνάρτηση που εκτελείται όταν είναι η σειρά του ανθρώπου να παίξει. Αν και θα περιμέναμε να είναι πιο απλή από την προηγούμενη (πχ. να διαβάζει την επιλογή του παίκτη και να την εκτελεί) ο κώδικας είναι λίγο μεγαλύτερος για δυο λόγους: (α) τυπώνει όλες τις πληροφορίες που χρειάζονται για να αποφασίσει ο παίκτης το φύλλο που θα πετάξει, και (β) προφυλάσσει το πρόγραμμα από τυχόν ανθρώπινα σφάλματα κατά την είσοδο:

```
def human_plays():
    global d, table, human_hand
    print
    print "Η τράπουλα έχει", len(d.content), "φύλλα"
    print "Ο Η/Υ έχει", len(computer_hand), "φύλλα"
    print "Στο τραπέζι έχουν πέσει", len(table), "φύλλα"
    t=table[-1]
    print "Το πάνω φύλλο είναι",t
    print "Τα φύλλα σου είναι",len(human_hand)
    HHS=[str(x) for x in human_hand]
    print HHS
    print
```

[16] Αλλά όχι ιδιαίτερα αποτελεσματικός αφού ξαναελέγχει όλα τα προηγούμενα φύλλα από την αρχή. Θα μπορούσαμε να βελτιώσουμε αυτό το σημείο με λίγο περισσότερο κώδικα, όμως θα περιέπλαμε τη συνάρτηση αρκετά περισσότερο χωρίς λόγο, αφού προς το παρόν δεν μας ενδιαφέρει η βελτίωση της ταχύτητας του κώδικα.

```
    print "Διάλεξε ποιό θα πετάξεις"
    print "ή πάτα σκέτο ENTER για να τραβήξεις"
    sel=raw_input()
    if sel=="":
        new_card=d.draw()
        if new_card=="empty": return "count"
        else:
            human_hand.append(new_card)
            return human_plays()
    else:
        if not(sel in HHS):
            print sel, ";;;   Δεν έχεις τέτοιο φύλλο."
            return human_plays()
        target_val=t.value
        target_sym=t.symbol
        if sel[0]<>target_val and sel[1]<>target_sym:
            print "Δεν επιτρέπεται να ρίξεις το φύλλο ",sel
            return human_plays()
        print "Ρίχνεις το",sel
        ind=HHS.index(sel)
        selc=human_hand[ind]
        human_hand.remove(selc)
        table.append(selc)
        if len(human_hand)<1: return "human wins"
        else: return "continue"
```

Η αρχή του κώδικα της human_plays() δεν έχει εκπλήξεις. Πάλι η global δηλώνει τις τρεις καθολικές μεταβλητές που χρησιμοποιεί η συνάρτηση και μετά μια σειρά από εντολές print τυπώνουν διάφορες βοηθητικές πληροφορίες. Εξήγηση χρειάζεται μόνο η χρήση της βοηθητικής μεταβλητής που ορίζεται με την εντολή HHS=[str(x) for x in human_hand]: επειδή δεν έχουμε ορίσει πώς τυπώνεται μια λίστα από τραπουλόχαρτα, αν τυπώναμε απ'ευθείας την human_hand θα παίρναμε μια σειρά από διευθύνσεις μνήμης. Αντιθέτως, η νέα λίστα που δημιουργούμε από τις εκτυπώσεις του κάθε στοιχείου της αρχικής είναι επίσης εκτυπώσιμη.

Για την καταγραφή της επιλογής του παίκτη δεν χρησιμοποιήθηκε η raw_input(). Λειτουργεί όπως η input() αλλά θεωρεί αυτομάτως ως αλφαριθμητικό την είσοδο του χρήστη, πράγμα που στη συγκεκριμένη περίπτωση μας εξυπηρετεί. Η if που ακολουθεί ελέγχει μήπως η παίκτης πατήσει απλώς ENTER χωρίς να γράψει κάποιο φύλλο. Αυτό σημαίνει ότι ο παίκτης θέλει να τραβήξει από την τράπουλα. Μετά από έναν έλεγχο μήπως η τράπουλα έχει αδειάσει, το νέο φύλλο προστίθεται στην human_hand και η συνάρτηση καλεί αναδρομικά τον εαυτό της για να επαναλάβει τη διαδικασία. Σε αυτά τα σημεία η human_plays() μοιάζει πολύ με την computer_plays() που είδαμε νωρίτερα.

Ακολουθούν δυο έλεγχοι, πρώτα αν το φύλλο που έγραψε ο παίκτης πράγματι το έχει στα χέρια του και ύστερα αν το φύλλο ταιριάζει με αυτό που είναι πάνω στο τραπέζι. Και στις δυο περιπτώσεις η συνάρτηση ξανακαλεί τον εαυτό της, διαφορετικά μεταφέρει το φύλλο από την human_hand στην table. Και πάλι παρομοίως με την computer_plays(), η human_plays() επιστρέφει ένα κατάλληλο αλφαριθμητικό ("human wins", "continue", ή "count") για να δηλώσει το αποτέλεσμα.

Κατά τη διάρκεια του παιχνιδιού η εκτέλεση των συναρτήσεων computer_plays() και human_plays() θα εναλάσσεται συνεχώς. Αυτό μπορεί να επιτευχθεί με την επαναληπτική κλήση της συνάρτησης next_turn() που ακολουθεί. Η συνάρτηση

δέχεται ως όρισμα έναν αριθμό που δηλώνει τον παίκτη που έχει σειρά να παίξει (+1 ο άνθρωπος, -1 ο Η/Υ) και επιστρέφει (για την ακρίβεια μεταφέρει) την έξοδο της αντίστοιχης συνάρτησης human_plays() ή computer_plays().

```
def next_turn(who_plays):
    if who_plays==1:
        print
        print "------------------- ΣΕΙΡΑ ΠΑΙΚΤΗ ------------------"
        return human_plays()
    else:
        print
        print "-------------------- ΣΕΙΡΑ Η/Υ --------------------"
        print
        return computer_plays()
```

Αν η παρτίδα τελειώσει, η ακόλουθη συνάρτηση evaluate() δέχεται ως όρισμα ένα από τα αλφαριθμητικά “human_wins”, “computer_wins”, ή “count” και τυπώνει τα κατάλληλα μηνύματα. Ειδικά στην περίπτωση του “count” μετράει και τυπώνει το πλήθος των φύλλων που έχει ο κάθε παίκτης.

```
def evaluate(result):
    global computer_hand, human_hand
    if result=="count":
        ch=len(computer_hand)
        hh=len(human_hand)
        print "Η τράπουλα τελείωσε, ο Η/Υ έχει",ch, "φύλλα και ο
παίκτης", hh
        if ch>hh: result="human wins"
        if ch<hh: result="computer wins"
        if ch==hh: print "ΙΣΟΠΑΛΙΑ!!!"
    if result=="human wins": print "ΣΥΓΧΑΡΗΤΗΡΙΑ. ΚΕΡΔΙΣΕΣ!!!"
    if result=="computer wins": print "Ο Η/Υ ΚΕΡΔΙΣΕ."
    print
```

Η συνάρτηση initial() που ακολουθεί αναλαμβάνει να εκτελέσει όλες τις λειτουργίες που γίνονται στην αρχή μιας παρτίδας. Η d.collect() μαζεύει τα χαρτιά στην τράπουλα, καθαρίζονται το τραπέζι και τα «χέρια» των παικτών, μετά η d.shuffle() ανακατεύει την τράπουλα, και η d.draw() καλείται για να μοιράσει επτά φύλλα στους παίκτες και ένα στο τραπέζι. Τέλος η random.random() καλείται και χρησιμοποιείται για να καθορίσει ποιός παίκτης θα παίξει πρώτος. Το αποτέλεσμα (+1 ή –1) επιστρέφεται ως τιμή της initial() για να χρησιμοποιηθεί στο κυρίως πρόγραμμα.

```
def initial():
    global d, table, computer_hand, human_hand
    print "Μαζεύω τα χαρτιά..."
    d.collect()
    table=[]
    computer_hand=[]
    human_hand=[]

    print "Ανακατεύω την τράπουλα..."
    d.shuffle()

    print "Μοιράζω τα φύλλα..."
    table.append(d.draw())
    print "Στο τραπέζι έπεσε",table[-1]
    for i in range(7):
```

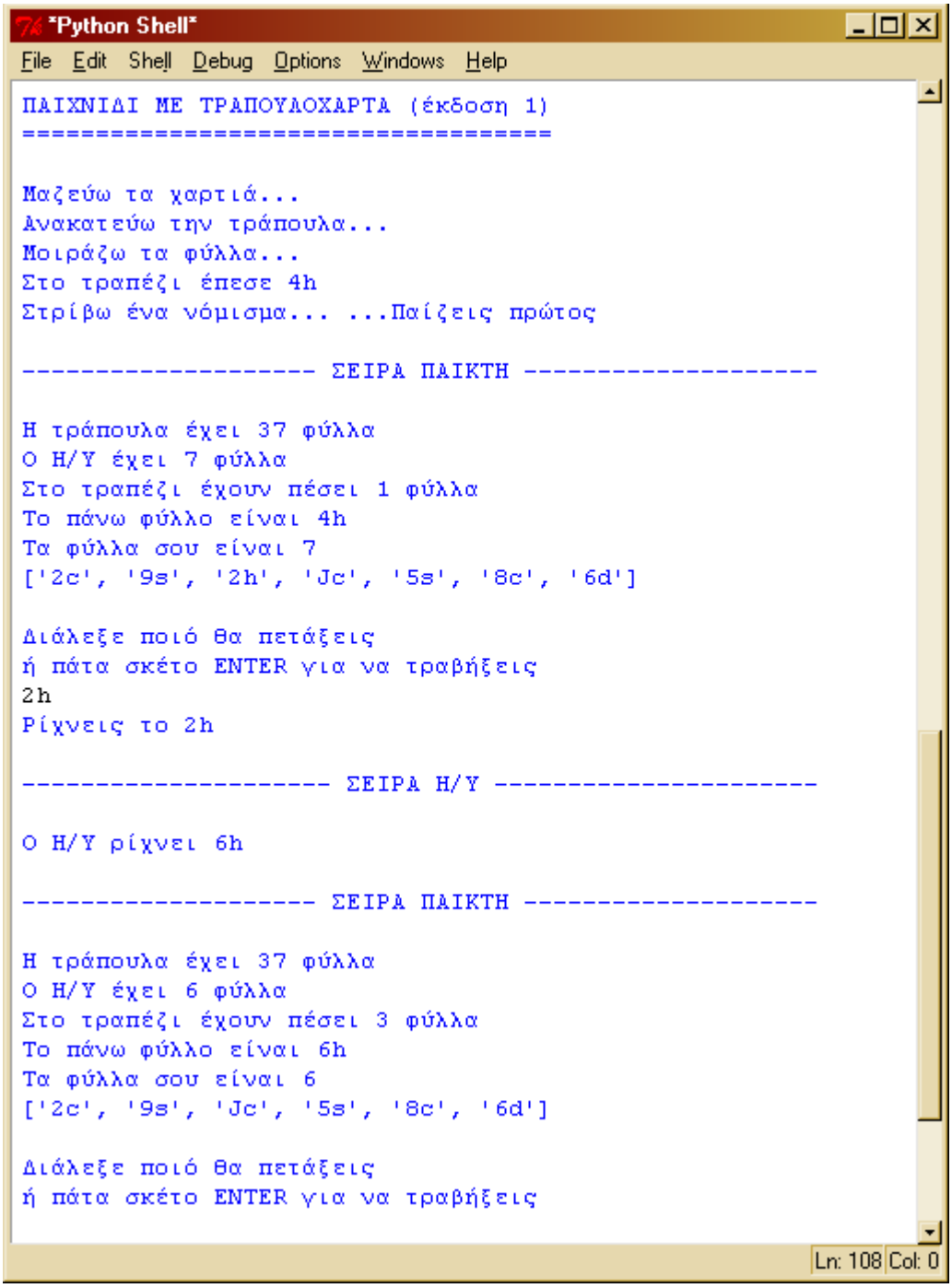
```
        human_hand.append(d.draw())
        computer_hand.append(d.draw())

    print "Στρίβω ένα νόμισμα...",
    if random.random()<0.5:
        who=-1
        print "...Παίζεις πρώτος"
    else:
        who=1
        print "...Ο Η/Υ παιζει πρώτος"
    return who
```

Και στο τέλος του αρχείου προσθέτουμε τον ακόλουθο κώδικα ως κυρίως πρόγραμμα:

```
# ΚΥΡΙΩΣ ΠΡΟΓΡΑΜΜΑ
print "ΠΑΙΧΝΙΔΙ ΜΕ ΤΡΑΠΟΥΛΟΧΑΡΤΑ (έκδοση 1)"
print "====================================="
print

# ΚΑΘΟΛΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ
d=deck()
table=[]
computer_hand=[]
human_hand=[]

again="y"
while again=="y":
    who=initial()

    result="continue"
    while result=="continue":
        who=-who
        result=next_turn(who)
    evaluate(result)

    print "Θέλεις να ξαναπαίξουμε;"
    again=raw_input("γράψε y ή n: ")
print "Τέλος Προγράμματος"
```

Μετά την εκτύπωση των αρχικών μηνυμάτων και την αρχικοποίηση των καθολικών μεταβλητών, ακολουθούν δυο while loops. Το εξωτερικό χρησιμοποιείται για να παίζουμε πολλές παρτίδες. Κάθε φορά που τελειώνει, ρωτά τον χρήστη αν θέλει να ξαναπαίξει. Κάθε φορά που ο χρήστης απαντά “y” καλείται η initial() και ξεκινά μια καινούργια παρτίδα. Το εσωτερικό while είναι υπεύθυνο για την εναλλαγή της σειράς των παικτών. Όσο το result της next_turn() είναι “continue” αλλάζει το πρόσημο που καθορίζει τη σειρά του παίκτη (who = -who) και ξαναεκτελείται η next_turn(). Μόλις το result πάψει να είναι “continue” το παιχνίδι έχει τελειώσει και καλείται η evaluate(result) για να γράψει τα κατάλληλα μηνύματα.

Σώστε το αρχείο με ένα κατάλληλο όνομα (πχ. card_game.py) και τρέξτε το. Μόλις το τρέξετε θα δείτε μια οθόνη σαν την ακόλουθη. Εδώ ο παίκτης έτυχε να παίζει πρώτος και έριξε το 2h πάνω στο 4h που ήταν στο τραπέζι. Ο Η/Υ συνέχισε με 6h και τώρα είναι πάλι η σειρά του παίκτη:



9.4 Κληρονομικότητα Κλάσεων

Ας υποθέσουμε ότι θέλουμε να φτιάξουμε μια νέα έκδοση του παιχνιδιού που να παίζεται με 2, 3 ή περισσότερες τράπουλες. Στην αρχή του παιχνιδιού ας ρωτάει ο Η/Υ τον παίκτη με πόσες τράπουλες θέλει να παίξει, κι ανάλογα με την απάντηση του παίκτη να προχωράει το παιχνίδι. Και πάλι, αυτή η αλλαγή μπορεί να γίνει με πολλούς τρόπους. Εδώ θα δείξουμε πώς γίνεται, προσθέτοντας πολύ λίγες γραμμές κώδικα, εκμεταλλευόμενοι μια ιδιότητα των κλάσεων που λέγεται **κληρονομικότητα**. Ανοίξτε το αρχείο playing_cards.py και προσθέστε στο τέλος του τον παρακάτω νέο ορισμό κλάσης:

```
class pack(deck):
    def __init__(self,number_of_decks=2):
        d=deck()
        self.content=d.content*number_of_decks
```

Αυτή η νέα κλάση μας δίνει τη δυνατότητα να δημιουργούμε «πακέττα» από τράπουλες. Πριν εξηγήσουμε πώς δουλεύει ας δούμε μερικά παραδείγματα. Σώστε και τρέξτε το αρχείο ώστε να μάθει η Python τη νέα κλάση και γράψτε:

```
>>> p=pack(3)
>>> print p
As 2s 3s 4s 5s 6s 7s 8s 9s Ts Js Qs Ks
Ah 2h 3h 4h 5h 6h 7h 8h 9h Th Jh Qh Kh
Ac 2c 3c 4c 5c 6c 7c 8c 9c Tc Jc Qc Kc
Ad 2d 3d 4d 5d 6d 7d 8d 9d Td Jd Qd Kd
As 2s 3s 4s 5s 6s 7s 8s 9s Ts Js Qs Ks
Ah 2h 3h 4h 5h 6h 7h 8h 9h Th Jh Qh Kh
Ac 2c 3c 4c 5c 6c 7c 8c 9c Tc Jc Qc Kc
Ad 2d 3d 4d 5d 6d 7d 8d 9d Td Jd Qd Kd
As 2s 3s 4s 5s 6s 7s 8s 9s Ts Js Qs Ks
Ah 2h 3h 4h 5h 6h 7h 8h 9h Th Jh Qh Kh
Ac 2c 3c 4c 5c 6c 7c 8c 9c Tc Jc Qc Kc
Ad 2d 3d 4d 5d 6d 7d 8d 9d Td Jd Qd Kd
156-0
```

Το p είναι ένα αντικείμενο τύπου pack το οποίο αποτελείται από 3 τράπουλες, επειδή γράψαμε pack(3). Πράγματι, βλέπουμε στην εκτύπωση ότι κάθε φύλλο εμφανίζεται 3 φορές και έχουμε σύνολο 156 (=3x52) φύλλα.

```
>>> p.shuffle()
>>> print p
2c 5s 3c Qd 8h Jd Js 8d Ac 7s 9c 5s 9d
Ad 5h 4c 6d 9h Jc 5h 4d 8c Qs 6d 4h Qh
4h Ks As Td 5c Jh 7s 6h 2s 3s Kh Kh 3d
Qc 8d 9c 9d Kh 9s 3d 9h 6h 7s Kc 5h Ah
Tc 6c 8c 9h Tc Kd Kc 2d 7c Qc 4c 2d 9d
2h Jh 5c 3h 4h Ac Ah 2s Ks Th Th 7d 7d
Kd 6s Kc As Jc Qs 6s 6d 2d 4s 7c 5d 2h
5d 6h 3c Tc 2s 8d 9c 5d 2c Js 6c Qd 5c
4s Kd 3h Jh Td 2c 3s 8h 3d 7h 3h Qs Ts
4s 4c Ad 6c Ts 5s Th 7h 8s 7d 4d Ah Js
Qh Qc 9s Ad Jd 3c 7h Td Ks 8c 8s Qd 3s
9s 7c 8h 2h 4d As Jc 6s Ac Ts Qh 8s Jd
156-0
```

Όμως κάτι παράξενο γίνεται εδώ. Εμείς όταν ορίσαμε την κλάση pack γράψαμε μόνο την __init__. Δεν προσθέσαμε μέθοδο shuffle(), ούτε καν μέθοδο __str__. Πώς μπορεί και ανακατεύει το πακέττο; Πώς γίνεται και το τυπώνει;

Η απάντηση βρίσκεται στο “class pack(deck):” με το οποίο αρχίσαμε τον ορισμό. Προσέξτε πως όλες οι προηγούμενες κλάσεις που ορίσαμε δεν έγραφαν κάτι μέσα στην παρένθεση. Εδώ όμως γράψαμε το όνομα της κλάσης deck. Με αυτόν τον τρόπο δηλώνουμε στην Python ότι η κλάση pack είναι ειδική περίπτωση της κλάσης deck. Και για αυτό το λόγο **κληρονομεί** από την deck όλες τις μεθόδους και όλα τα δεδομένα που έχουμε παραλείψει να γράψουμε (ακριβώς επειδή είναι ίδια). Αρκεί να

γράψουμε μόνο εκείνα στα οποία διαφέρει (εδώ μόνο την __init__) κι έτσι απλοποιούμε πολύ τον κώδικά μας. Με άλλα λόγια κάθε αντικείμενο τύπου pack θα έχει τα strings values και symbols, τις λίστες content και pile και τις μεθόδους __str__, shuffle(), draw() και collect() ακριβώς όπως κάθε αντικείμενο τύπου deck. Θα διαφέρει μόνο στην __init__ Η οποία θα είναι διαφορετική.

Ας δούμε πόσο: μέσα στην παρένθεση της __init__ μετά από το self, υπάρχει το όρισμα number_of_decks=2 που καθορίζει από πόσες τράπουλες θα αποτελείται το πακέττο. Αυτό το "=2" θα μπορούσαμε να το παραλείψουμε. Έχει την έννοια της **προεπιλεγμένης τιμής** (**default value**). Δηλαδή τώρα που το βάλαμε είτε γράψουμε e=pack(2) είτε σκέτο e=pack() θα μας φτιάξει ένα πακέττο με 2 τράπουλες.

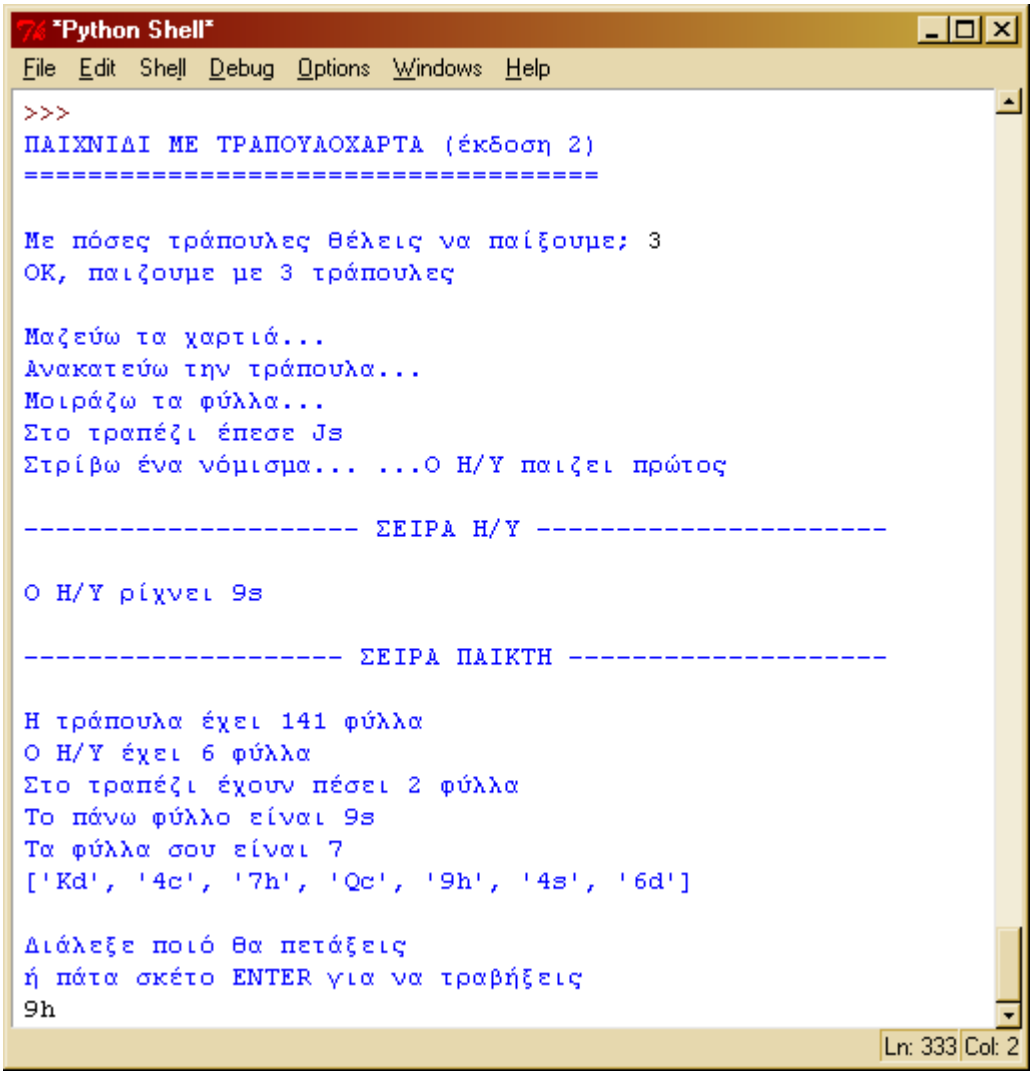
Στη συνέχεια, η __init__ δημιουργεί μια τράπουλα d, ώστε να φτιαχτεί σωστά η λίστα d.content και στη συνέχεια την πολλαπλασιάζει number_of_decks φορές ονομάζοντας το αποτέλεσμα self.content. Δηλαδή η λίστα content του pack θα αποτελείται από πολλά αντίγραφα της λίστας content μιας απλής τράπουλας.

Κι αυτό είναι όλο. Δείτε τώρα τί θα αλλάζαμε στο αρχείο card_game.py:

```
...
# ΚΥΡΙΩΣ ΠΡΟΓΡΑΜΜΑ
print "ΠΑΙΧΝΙΔΙ ΜΕ ΤΡΑΠΟΥΛΟΧΑΡΤΑ (έκδοση 2)"
print "====================================="
print
n=input("Με πόσες τράπουλες θέλεις να παίξουμε; ")
print "ΟΚ, παιζουμε με",n,"τράπουλες"
print

# ΚΑΘΟΛΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ
d=pack(n)
...
```

Ουσιαστικά προσθέσαμε μόνο την input() και αλλάξαμε την καθολική μεταβλητή d από deck() σε pack(n). Όλος ο προηγούμενος κώδικας κι όλος ο επόμενος στο αρχείο μένουν ίδιοι. Βέβαια, αν θέλαμε να ήμασταν πιο συνεπείς, θα έπρεπε να κάνουμε κι έναν έλεγχο μήπως ο παίκτης δώσει ακατάλληλη (πχ. αρνητική) τιμή για το n πριν το δεχτούμε. Όμως αυτό δεν είναι επί του παρόντος. Το σημαντικό ήταν να δείξουμε πόσο χρήσιμη είναι η ιδιότητα της κληρονομικότητας μεταξύ των κλάσεων και πόσο μπορεί να απλοποιήσει τον κώδικά μας.

Δείτε παρακάτω μια οθόνη από το βελτιωμένο πρόγραμμα. Προσέξτε στο σημείο που λέει ότι η τράπουλα έχει 141 φύλλα. 141+1+7+7=156=3x52, σωστά.



9.5 Ασκήσεις

Αφού ετοιμάσετε τα δυο αρχεία playing_cards.py και card_game.py σύμφωνα με τις οδηγίες που προηγήθηκαν, εκπονήστε τις παρακάτω ασκήσεις. Αναρτήστε τα αρχεία σας ως ένα ενιαίο συμπιεσμένο αρχείο.

Άσκηση #1

Κάντε πιο ενδιαφέρον το παιχνίδι προσθέτοντας τους εξής επιπλέον κανόνες:

1. Αν ένας παίκτης έχει Άσσο (A), μπορεί να τον ρίξει ό,τι φύλλο κι αν υπάρχει στο τραπέζι. Φυσικά ο αντίπαλος θα πρέπει να συνεχίσει με το σύμβολο του Άσσου (ή με νέο Άσσο).
2. Όταν ένας παίκτης ρίξει Ρήγα (K) ξαναπαίζει.
3. Όταν ένας παίκτης ρίξει Ντάμα (Q) αναγκάζει τον αντίπαλο να τραβήξει 2 φύλλα από την τράπουλα.

Άσκηση #2

Βελτιώστε τον κώδικα του παιχνιδιού έτσι ώστε ο παίκτης να μπορεί να παίζει με πολλούς αντιπάλους (που θα τους ελέγχει ο υπολογιστής). Στην αρχή ο παίκτης θα επιλέγει το πλήθος τους. (Υπόδειξη: Δημιουργήστε μια κλάση computer_player και προσθέστε της ως μέθοδο την συνάρτηση computer_plays(). Τροποποιήστε την όσο χρειαστεί. Με την ευκαιρία, δοκιμάστε αν μπορείτε να κάνετε τον Η/Υ να παίζει με πιο έξυπνη στρατηγική.)

Εργαστήριο 10: Εξέταση Εργαστηρίου

Γενικά

Στην τελευταία βδομάδα του ακαδημαϊκού εξαμήνου προβλέπεται γραπτή εξέταση του εργαστηριακού έργου. Αυτή θα γίνει στο χώρο του εργαστηρίου θα είναι διάρκειας αντίστοιχης με τις άλλες ασκήσεις. Αυτή η εξέταση είναι αυστηρά ατομική. Οι ερωτήσεις που τίθενται καλύπτουν την ύλη όλων των εργαστηριακών ασκήσεων και οι φοιτητές μπορούν να χρησιμοποιήσουν για να τις απαντήσουν τον υπολογιστή εκτός από εργαλεία συνεργασίας και επικοινωνίας. Επίσης μπορεί να χρησιμοποιηθούν βοηθήματα σε έντυπη μορφή (σημειώσεις, βιβλία κλπ). Οι απαντήσεις στα ερωτήματα θα κατατίθενται σε ηλεκτρονική μορφή, σύμφωνα με τις οδηγίες.

Ο τελικός βαθμός του εργαστηρίου θα προκύπτει ως μέσος όρος από (α) τον βαθμό των εκθέσεων των εργαστηριακών ασκήσεων και της ομαδικής εργασίας και (β) τον βαθμό της εξέτασης στο εργαστήριο. Βλέπε συχνές ερωτήσεις στις σελίδες 13-14. Η επιτυχής ολοκλήρωση του εργαστηρίου είναι απαραίτητη προϋπόθεση για συμμετοχή στην τελική εξέταση του μαθήματος.

3. Ομαδική εργασία

Οδηγίες

Κατά την 6η εβδομάδα του εξαμήνου οι σπουδαστές θα χωριστούν σε ολιγομελείς ομάδες 4 ή 5 φοιτητών η κάθε μια, οι οποίες θα ανακοινωθούν έγκαιρα. Η κάθε ομάδα θα αναλάβει ένα διαφορετικό θέμα που θα πρέπει να εκπονηθεί μέσα σε χρονικά περιθώρια που επίσης θα ανακοινωθούν (περίπου 6 εβδομάδες). Η κάθε ομάδα θα συντάξει μια έκθεση την οποία θα παραδώσει μέσα στο καθορισμένο χρονικό πλαίσιο καθώς και ένα πρόγραμμα σε γλώσσα Python. Οι φοιτητές με δική τους πρωτοβουλία και με συναντήσεις που οι ίδιοι συντονίζουν, θα πρέπει να προγραμματίσουν και να οργανώσουν τις επιμέρους εργασίες και το υλικό που θα αναλάβει κάθε μέλος της ομάδας, ώστε στο τέλος να παραδοθεί έγκαιρα η ομαδική εργασία με ενιαία δομή, εμφάνιση και περιεχόμενο. Είναι ευθύνη των μελών κάθε ομάδας να μοιράσουν ισόποσα τον εργασιακό φόρτο μεταξύ τους ώστε να μην υπάρχει αδικία στην τελική αξιολόγηση της ομαδικής τους εργασίας, μιας και ο βαθμός της ομαδικής εργασίας είναι κοινός για όλα τα μέλη της ομάδας. Επίσης θα γίνει εξέταση των εργασιών αυτών στο τέλος του εξαμήνου. Προτείνεται η οργάνωση της ομάδας να γίνει με χρήση ενός εργαλείου υποστήριξης ομαδικής δραστηριτότητας. Τέτοιο μπορεί να είναι για παράδειγμα το google groups. (http://groups.google.com/). Πρέπει να δοθεί ιδιαίτερη προσοχή ώστε να τηρηθεί αυστηρά η ημερομηνία παράδοσης.

Παρακάτω αναφέρεται η δομή που θα πρέπει να τηρεί η ομαδική εργασία:

- Στο *εξώφυλλο* της έκθεσης που θα παραδοθεί θα πρέπει να αναφέρονται: α) ο κωδικός της ομάδας, β) τα ονοματεπώνυμα και αριθμοί φοιτητικού μητρώου των μελών της ομάδας που εργάστηκαν και γ) ο τίτλος της εργασίας.
- Περίληψη (10 γραμμές).
- Πίνακας Περιεχομένων
- Εισαγωγή (περιγραφή του προβλήματος).
- Μεθοδολογία (πώς εργαστήκατε για να εκτελέσετε την εργασία, συναντήσεις των μελών της ομάδας, καταμερισμός εργασιών, συμφωνίες και διαφωνίες, εργαλεία που χρησιμοποιήσατε κλπ.).
- Οργάνωση και δομή της ομάδας (αναφορά της συγκεκριμένης συνεισφοράς του κάθε μέλους της ομάδας). Μπορείτε να επισυνάψετε αντίγραφο των αλληλεπδράσεων που είχατε με το εργαλείο οργάνωσης της ομάδας σας (π.χ. google groups) από το οποίο να προκύπτει η συμμετοχή του κάθε μέλους της ομάδας στην άσκηση
- Περιγραφή του αλγορίθμου και της οργάνωσης του προγράμματος που κατασκευάσατε.
- Βασική λειτουργία του αλγορίθμου, απόδειξη καλής λειτουργίας.
- Συμπεράσματα.
- Αναφορές (πηγές που χρησιμοποιήσατε).
- Παράρτημα Ο κώδικας που αναπτύξατε με κατάλληλα σχόλια

Μια σημαντική ενημερωμένη πηγή για τις αναζητήσεις υλικού για την εργασία σας είναι το Διαδίκτυο. Αν χρησιμοποιήσετε ιδέες (αλγορίθμους, κώδικα κλπ) θα πρέπει να αναφέρετε υποχρεωτικά τις πηγές σας.

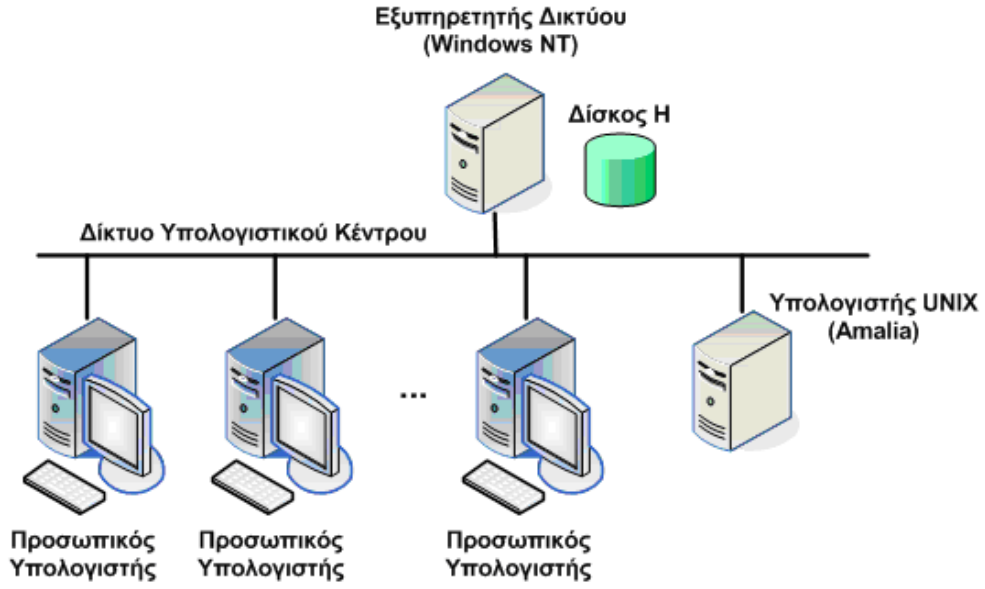
4. Χρήση του Υπολογιστικού Κέντρου

Γενικά

Το Κέντρο Υπολογιστικών και Επικοινωνιακών Συστημάτων (ΚΥΠΕΣ) είναι ένας χώρος ο οποίος περιλαμβάνει 120 θέσεις εργασίας με προσωπικούς υπολογιστές και στον οποίο θα διεξάγονται τα εργαστήρια του μαθήματος «Εισαγωγή στους Υπολογιστές Ι», μεταξύ άλλων. Το ΚΥΠΕΣ διαθέτει δύο αίθουσες: στην μία (αίθουσα 2, 56 θέσεων) διεξάγονται αποκλειστικά τα εργαστήρια, η άλλη (αίθουσα 1, 50 θέσεων) χρησιμοποιείται από τα μέλη του Τμήματος σαν αίθουσα ελεύθερης δράσης (αναζήτηση στο διαδίκτυο, εκπόνηση εργασιών, εξάσκηση στα θέματα που τους θέτονται στις εργαστηριακές ασκήσεις του μαθήματος κλπ). Για σπουδαστές που δεν έχουν προηγούμενη εμπειρία με υπολογιστές προτείνεται να επισκεφθούν το ΚΥΠΕΣ πριν από την πρώτη βδομάδα εργαστηρίου και να πειραματιστούν μόνοι τους με τους υπολογιστές προκειμένου να εξοικειωθούν με την χρήση τους.

Δομή και οργάνωση Υπολογιστικού Κέντρου

Στο ΚΥΠΕΣ υπάρχει ένα σύνολο από υπολογιστές και εξυπηρετητές. Η δομή του φαίνεται στην παρακάτω εικόνα



Δομή ΚΥΠΕΣ

Ο Εξυπηρετητής Δικτύου είναι υπολογιστής που ελέγχει το δίκτυο των προσωπικών υπολογιστών και περιέχει για κάθε σπουδαστή ένα προσωπικό χώρο αποθήκευσης (κάποια MB). Ο χώρος αυτός είναι προσπελάσιμος κατά την σύνδεση σε προσωπικό υπολογιστή ως δίσκος H. Ομοίως ο υπολογιστής UNIX (Amalia) ελέγχει τον λογαριασμό UNIX του κάθε χρήστη. Προσοχή όμως χρειάζεται για τους τοπικούς δίσκους των προσωπικών υπολογιστών οι οποίοι δεν παρέχουν καμία ασφάλεια, αφού σε αυτούς έχουν πρόσβαση όλοι οι χρήστες και κατά συνέπεια δεν θα πρέπει να χρησιμοποιείται σαν αποθηκευτικός χώρος.

Όλοι οι σπουδαστές θα πρέπει να εξοικειωθούν με το υπολογιστικό κέντρο και τις υπηρεσίες του. Αυτό όμως θα πρέπει να γίνει εκτός της ώρας των εργαστηρίων. Και στα δύο περιβάλλοντα (Windows και Unix) θα πρέπει να αποκτήσετε μυστικό συνθηματικό σύνδεσης. Και για τα δύο συστήματα πρέπει να ακολουθήσετε μια διαδικασία σύνδεσης και ορισμού συνθηματικού, η οποία περιγράφεται παρακάτω.

Πληκτρολόγιο

Θα πρέπει αρχικά να εξοικειωθείτε με τη χρήση του πληκτρολογίου (ειδικά αν είναι η πρώτη φορά που το χρησιμοποιείτε). Αν νιώθετε ότι δεν έχετε άνεση με τη χρήση του πληκτρολογίου, συνιστάται να διαθέσετε μια ώρα για πειραματισμό πριν από την έναρξη των εργαστηρίων. Χρησιμοποιήστε όλα τα πλήκτρα και προσέξτε ιδιαίτερα τη χρήση των πλήκτρων Shift, Ctrl, Enter, Alt. Επίσης ζητήστε πληροφορίες για τη διαδικασία ανοίγματος και κλεισίματος της συσκευής, ειδικά στους προσωπικούς υπολογιστές εξοικειωθείτε ακόμη με τη χρήση του ποντικιού.

Αλλαγή password σε περιβάλλον Windows

Το login name για κάθε σπουδαστή ακολουθεί μία συγκεκριμένη μορφή, η οποία είναι eceXXXX, όπου XXXX είναι τα τέσσερα (4) τελευταία ψηφία του αριθμού μητρώου του κάθε σπουδαστή. Το login name είναι μία συμβολοσειρά που μπορεί να γνωρίζουν και άλλοι εκτός από τον πραγματικό κάτοχό του. Όμως το password δεν το ξέρει και θα πρέπει να μην το ξέρει κανείς άλλος εκτός από τον πραγματικό του κάτοχο. Σε κάθε προσωπικό υπολογιστή γίνεται έλεγχος της ταυτότητας του χρήστη. Ο χρήστης θα πρέπει να δώσει το login name και στη συνέχεια password. Την πρώτη φορά που θα συνδεθείτε, πρέπει να δώσετε το συνθηματικό "123456" το οποίο έχει τεθεί αυτόματα μέχρι εσείς να πάτε και να το αλλάξετε με κάποιο της δική σας προτίμησης. Ιδιαίτερη προσοχή χρειάζεται όταν εισάγεται το password αφού οι χαρακτήρες αντικαθίστανται από αστερίσκους για λόγους ασφαλείας.

Επειδή λοιπόν για κάθε λογαριασμό που δημιουργείται το password είναι αυτόματα “123456”, θα πρέπει γρήγορα να αλλαχθεί από το σπουδαστή. Για να αλλάξετε το password, αρκεί μόλις συνδεθείτε να πατήσετε Ctrl+Alt+Del και μετά να επιλέξετε «Change Password». Στην εικόνα που θα εμφανισθεί, θα πρέπει να δώσετε το παλιό password πρώτα (π.χ. 123456) και στη συνέχεια το νέο password δύο (2) φορές. Το password πρέπει έχει μήκος τουλάχιστον 6 χαρακτήρων και να περιέχει υποχρεωτικά τουλάχιστον 2 γράμματα και τουλάχιστον 1 αριθμό ή ειδικό χαρακτήρα (π.χ. -, $, #, &). Το ίδιο συνθηματικό καλό είναι να χρησιμοποιηθεί για τη σύνδεση και στο

σύστημα Unix, ώστε να μην χρειάζεται ο σπουδαστής να θυμάται δύο διαφορετικά passwords. Δεν έχει σημασία βεβαίως σε ποιόν προσωπικό υπολογιστή βρίσκεται ο φοιτητής μπροστά, αφού όλοι είναι συνδεδεμένοι σε δίκτυο και συνδέονται με έναν κεντρικό εξυπηρετητή ο οποίος ελέγχει την ορθότητα του λογαριασμού σας. Ο λογαριασμός αυτός, ο οποίος είναι μονοσήμαντος για κάθε φοιτητή / τρια, θα σας συνοδεύει μέχρι το τέλος των σπουδών σας στο Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών, της Πολυτεχνικής Σχολής, του Πανεπιστημίου Πατρών. Ισχύει για τα πέντε (5) έτη σπουδών και κατ' εξαίρεση για ένα (1) επί πλέον έτος μετά τα πέντε με αιτιολογημένη αίτησή σας. Με την απόκτηση πτυχίου διαγράφεται και ο λογαριασμός σας και από τα δύο συστήματα (Εξυπηρετητής Δικτύου, Amalia). Για περισσότερες πληροφορίες ζητείστε οδηγίες από το προσωπικό του ΚΥΠΕΣ.

Σύνδεση στο σύστημα UNIX μέσω προσωπικού υπολογιστή

Ο σπουδαστής μπορεί να συνδεθεί στο σύστημα UNIX μέσω των προσωπικών υπολογιστών που έχουν λειτουργικό σύστημα Windows. Έτσι μπορεί να αλλάξει το password του λογαριασμού που ισχύει για το σύστημα UNIX. Αυτό επιτυγχάνεται με διπλό κλικ του ποντικιού πάνω στο εικονίδιο "Shortcut to Term.exe" που φαίνεται στο σχήμα 3.1.2. Αν δεν βλέπετε το εικονίδιο στην επιφάνεια εργασίας, μπορείτε να ενεργοποιήσετε την εφαρμογή σύνδεσης με την εντολή: Έναρξη (Start) > Προγράμματα (Program Files) > QvtNet > QVT-Term. Επίσης μπορείτε να χρησιμοποιήσετε τη λειτουργία telnet από τη γραμμή εντολών: Έναρξη (Start) > Εκτέλεση (Run) > "telnet". Στη συνέχεια επιλέγετε από το νέο παράθυρο του προγράμματος αυτού (αν ζητηθεί) Terminal και δίνετε το όνομα του υπολογιστή Unix που θα χρησιμοποιήσετε δηλαδή γράφεται "amalia". Τότε ανοίγει ένα νέο παράθυρο στο οποίο υπάρχει η προτροπή "Login:" του συστήματος Unix. Τώρα είστε έτοιμοι να συνδεθείτε με τον υπολογιστή Unix.

Σχήμα 3.1.2 Πρόγραμμα σύνδεσης στο σύστημα UNIX.

Αλλαγή password σε περιβάλλον UNIX

Όταν συνδέεστε στο σύστημα Unix για πρώτη φορά, θα σας ζητήσει να εισάγετε το νέο σας συνθηματικό κατευθείαν και μάλιστα θα ζητήσει επιβεβαίωση του για δεύτερη φορά για να αποφευχθούν λάθη πληκτρολόγησης. Αυτό γίνεται μόνο την

πρώτη φορά που συνδέεστε. Αν ο υπολογιστής δεν ζητήσει επιβεβαίωση του συνθηματικού, σημαίνει είτε ότι έχετε ήδη συνθηματικό, είτε ότι δεν είστε ακόμη καταγεγραμμένος χρήστης του συστήματος και δεν μπορεί να σας αναγνωρίσει. Όταν επιτυχώς ολοκληρώσετε την εισαγωγή του συνθηματικού "εισέρχεστε" στο σύστημα. Αυτό σας καλωσορίζει με πληροφορίες για τον κατασκευαστή του (Hewlett Packard) και άλλες τρέχουσες πληροφορίες. Χρησιμοποιώντας την εντολή passwd μπορείτε να αλλάξετε τον κωδικό πρόσβασης σας οποιαδήποτε χρονική στιγμή Ο λογαριασμός αυτός, ο οποίος είναι μονοσήμαντος για κάθε φοιτητή / τρια, θα σας συνοδεύει μέχρι το τέλος των σπουδών σας στο Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών, της Πολυτεχνικής Σχολής, του Πανεπιστημίου Πατρών. Ισχύει για τα πέντε (5) έτη σπουδών και κατ' εξαίρεση για ένα (1) επί πλέον έτος μετά τα πέντε με αιτιολογημένη αίτησή σας. Με την απόκτηση πτυχίου διαγράφεται και ο λογαριασμός σας και από τα δύο συστήματα (Εξυπηρετητής Δικτύου, Amalia).

Κανόνες χρήσης ΚΥΠΕΣ

(από **http://kypes.ece.upatras.gr/kypesdocs/UserPolicy.pdf)**

1. Όροι και Κανόνες Λειτουργίας Εγκαταστάσεων

- **Διατήρηση Καθαριότητας:** Όλοι οι χρήστες είναι υποχρεωμένοι να διατηρούν το χώρο του εργαστηρίου καθαρό.
- **Τήρηση Όρων υγιεινής και ασφάλειας:** Απαγορεύεται αυστηρώς το κάπνισμα μέσα στο χώρο του εργαστηρίου. Δεν επιτρέπεται η κατανάλωση φαγητού και ποτού μέσα στο χώρο του ΚΥΠΕΣ.
- **Μετακίνηση Ηλεκτρονικών Υπολογιστών και περιφερειακών συσκευών:** Απαγορεύεται η μετακίνηση και αλλαγή θέσεως των Ηλεκτρονικών Υπολογιστών και των περιφερειακών συσκευών τους.
- **Επέμβαση στα μηχανήματα:** Απαγορεύεται οποιαδήποτε φυσική επέμβαση πάνω στα μηχανήματα που βρίσκονται στο ΚΥΠΕΣ.

2. Όροι και Κανόνες Χρήσης Υπολογιστικών Συστημάτων

- Κάθε χρήστης είναι υπεύθυνος για κάθε είδους δραστηριότητα που αναπτύσσεται στον Η/Υ που χρησιμοποιεί άμεσα (τοπικά) ή έμμεσα (π.χ. μέσω δικτύου).
- Κάθε χρήστης είναι υπεύθυνος για τη διατήρηση της καλής λειτουργίας του εξοπλισμού τον οποίο χειρίζεται. Σε περίπτωση που εμφανιστεί οποιοδήποτε πρόβλημα οφείλει να ενημερώσει αμέσως το προσωπικό του ΚΥΠΕΣ.
- Κάθε χρήστης είναι υπεύθυνος για την επιλογή και διαφύλαξη «ασφαλούς» συνθηματικού (password) για την πρόσβαση στο λογαριασμό του. Τα συνθηματικά απαγορεύεται να δίδονται σε τρίτους.
- Σε περίπτωση απώλειας ή κλοπής του συνθηματικού (password) του, ο χρήστης υποχρεούται να ειδοποιήσει χωρίς καθυστέρηση το προσωπικό του ΚΥΠΕΣ, για απενεργοποίηση του σχετικού κωδικού.
- Κάθε χρήστης αποθηκεύει τα δεδομένα μόνο στον προσωπικό του χώρο στον εξυπηρετητή του ΚΥΠΕΣ.
- Κάθε χρήστης είναι αποκλειστικά υπεύθυνος για την προστασία των δεδομένων και των αρχείων του από τρίτους.
- Κάθε χρήστης υποχρεούται να αναφέρει αμέσως κάθε απόπειρα ή παραβίαση των κανόνων λειτουργίας και ασφαλείας των υπολογιστικών πόρων από τρίτους στο προσωπικό του ΚΥΠΕΣ.

3. Οριοθέτηση Χρήσης Συστημάτων και Δικτυακών Πόρων

- Οι χρήστες οφείλουν να μην μονοπωλούν και καταχρώνται πόρους του δικτύου και των συστημάτων, όπως διαθέσιμη χωρητικότητα συνδέσεων, αποθηκευτικό χώρο σκληρών δίσκων, κύκλους μηχανής επεξεργαστών, άδειες χρήσης λογισμικού, κτλ. Ιδιαίτερη προσοχή θα πρέπει να επιδεικνύεται ώστε να αποφεύγεται άσκοπη χρήση πόρων, όταν αυτό είναι δυνατόν.
- Οι χρήστες οφείλουν να αποδεσμεύουν άδειες χρήσης λογισμικού τις οποίες δεν χρησιμοποιούν.
- Οι χρήστες οφείλουν να αποδεσμεύουν διευθύνσεις δικτύου τις οποίες δεν χρησιμοποιούν.
- Απαγορεύεται ρητά η οποιαδήποτε μορφής εξω-πανεπιστημιακή και εμπορική δραστηριότητα (με ή χωρίς αμοιβή).
- Απαγορεύεται η χρήση και ανάπτυξη ιών λογισμικού, και γενικά προγραμμάτων που προσπαθούν να παραβιάσουν ή να παρακάμψουν τους μηχανισμούς ασφαλείας του δικτύου, να υποκλέψουν συνθηματικά, ή να εξαντλήσουν τους υπολογιστικούς πόρους.
- Απαγορεύεται ρητά η χρήση και διακίνηση μέσω των υπολογιστών και του δικτύου του ΚΥΠΕΣ κλεψίτυπου περιεχομένου και λογισμικού ή προγραμμάτων για παράνομη αντιγραφή και χρήση λογισμικού.

4. Παραδείγματα Κακής Χρήσης Υπολογιστικών και Δικτυακών Πόρων

- Η κακή χρήση των δικτυακών πόρων και του διαθέσιμου εύρους ζώνης που μπορεί να επιφέρει υπερβολικό φορτίο σε δικτυακές συσκευές ή καθυστέρηση και παρεμπόδιση της υπόλοιπης κίνησης (π.χ. επιθέσεις άρνησης υπηρεσιών - Denial of Service attacks και Port scans).
- Η εξάπλωση ιών ή άλλων «βλαβερών» προγραμμάτων (malware).
- Η προσπάθεια παραβίασης υπολογιστικών συστημάτων.
- Η οποιαδήποτε χρήση του δικτύου για εμπορικούς ή κερδοσκοπικούς σκοπούς.
- Η παραβίαση πνευματικών δικαιωμάτων μέσω διακίνησης κατοχυρωμένου υλικού.
- Η διάδοση υβριστικών και γενικότερα ασύμβατων με τα ακαδημαϊκά ήθη πληροφοριών, όπως ρατσιστικό ή πορνογραφικό υλικό.
- Η μαζική αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου (E - Mail spam).

Πρόληψη, Αντιμετώπιση Παραβάσεων, Επιβολή Κυρώσεων

Σε πολλές περιστάσεις μπορεί να προκύψουν παραβιάσεις της παραπάνω πολιτικής, οι οποίες οφείλονται σε άγνοια ή ελλιπή ενημέρωση των χρηστών (π.χ. προφανή συνθηματικά, αθέλητη κατάχρηση υπολογιστικών πόρων κλπ.) Οι χρήστες ενθαρρύνονται να ζητούν, όταν έχουν οποιαδήποτε αμφιβολία, τη βοήθεια των διαχειριστών του ΚΥΠΕΣ. Οι υπεύθυνοι του ΚΥΠΕΣ είναι σε θέση να πάρουν άμεσα μέτρα που τυχόν απαιτούνται για την αντιμετώπιση προβλημάτων και να κατευθύνουν τους χρήστες, ώστε να μην προκύψουν ξανά.

Σε σοβαρότερες περιπτώσεις όπως επαναλαμβανόμενες παραβιάσεις, επαναλαμβανόμενη κατάχρηση των υπολογιστικών και δικτυακών πόρων, κλοπή δεδομένων, παράνομες πράξεις, οι υπεύθυνοι του ΚΥΠΕΣ θα μεταβιβάζουν το πρόβλημα στα αρμόδια όργανα του Τμήματος.

Για οποιαδήποτε περαιτέρω διευκρίνηση ή απορία μπορείτε να επικοινωνείτε με το υπεύθυνους του ΚΥΠΕΣ στην ηλεκτρονική διεύθυνση helpdesk@ee.upatras.gr.

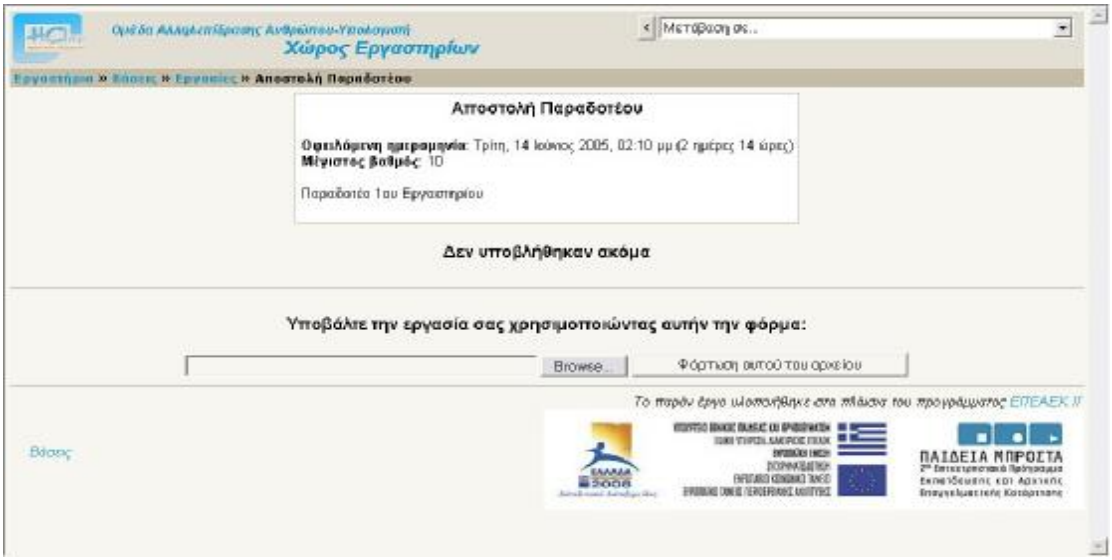
5. Συνήθεις Ερωτήσεις για τον ιστότοπο του εργαστηρίου

Τι πρέπει να γνωρίζω για την υποβολή παραδοτέων?

Απ: Μερικές χρήσιμες πληροφορίες για την **υποβολή των παραδοτέων** σας:

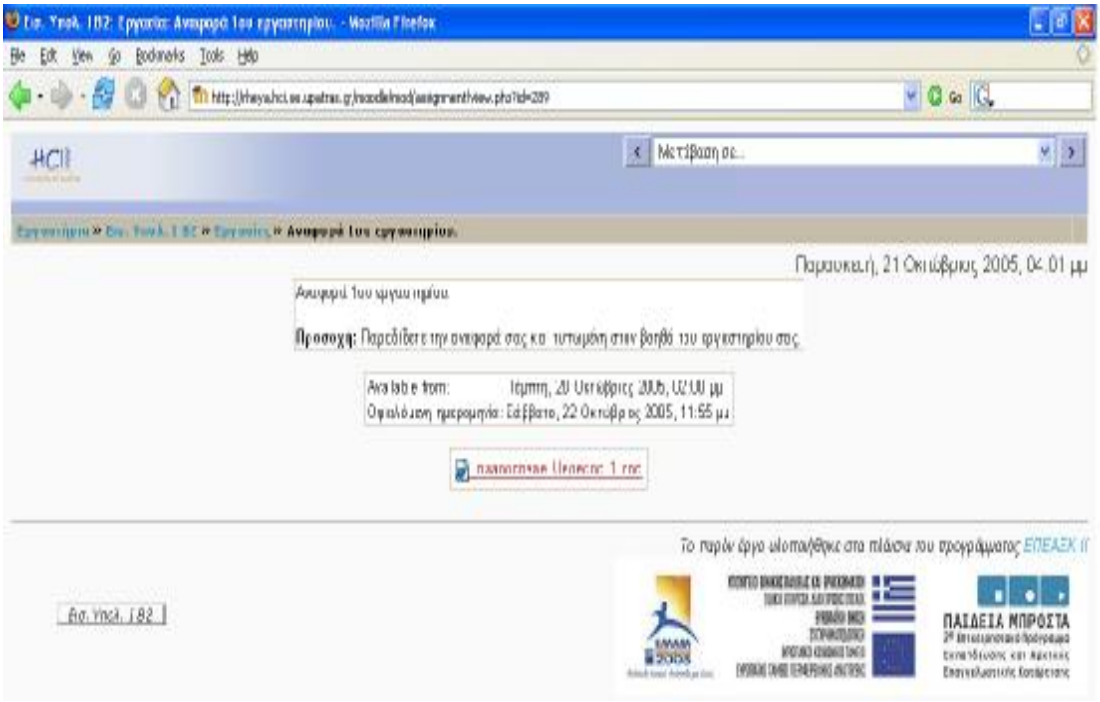
1) Παραδίδετε την αναφορά σας στο link "**Αποστολή Παραδοτέου**" της **αντίστοιχης** εργαστηριακής άσκησης .

2) **Δεν επιτρέπονται** οι εκπρόθεσμες αναφορές. Η επιλογή browse στην αποστολή παραδοτέου θα **κλειδώνει αυτόματα** μετά την λήξη της προθεσμίας υποβολής η οποία θα αναγράφεται για κάθε εργαστηριακή άσκηση στην σελίδα που εμφανίζεται όταν πατήσετε στο link "**Αποστολή Παραδοτέου**".



3) Επιτρέπεται η **επανυποβολή** των παραδοτέων σας, που σημαίνει ότι αν στείλετε την αναφορά σας και θέλετε να την αλλάξετε μπορείτε να ξαναστείλετε την νέα τροποποιημένη αναφορά σας. Στο σύστημα αποθηκεύεται πάντα μόνο το **τελευταίο αρχείο** που έχετε στείλει με την **ημερομηνία** που το στείλατε.

4) Για να διαπιστώστε αν έχετε στείλει ή όχι την αναφορά σας απλά πηγαίνετε ξανά στο link **"Αποστολή Παραδοτέου"** οποιαδήποτε χρονική στιγμή και εφόσον έχετε στείλει κάποιο αρχείο θα σας εμφανιστεί το τελευταίο που στείλατε. Μπορείτε επίσης να κατεβάσετε ή να ανοίξετε το αρχείο εφόσον έχετε στείλει κάτι.



5) **Δεν** στέλνουμε το παραδοτέο με τον λογαριασμό του **φίλου/φίλης μας**. Ο λογαριασμός σας στο σύστημα είναι **προσωπικός** και **δεν πρέπει να τον μοιράζεστε**. Μόνο όταν στέλνετε την αναφορά σας με τον δικό σας λογαριασμό εμφανίζεται με το όνομα σας. Σε περίπτωση που χρησιμοποιήσετε τον λογαριασμό φίλου/φίλης σας για να στείλετε την δική σας αναφορά τότε **αυτόματα σβήνετε** ότι έχει στείλει αυτός και επιπρόσθετα η αναφορά σας **θα φαίνεται με το όνομα του**.
http://rheya.hci.ee.upatras.gr/moodle/mod/forum/post.php?delete=34

Δεν μπορώ να κάνω login στο σύστημα ηλεκτρονικής υποστήριξης εργαστηρίων

Απ: Πιθανές λύσεις για το πρόβλημα σας:

1) Πιθανόν να βάζετε λάθος τον κωδικό πρόσβασης σας. Προσέξτε να μην είναι ενεργοποιημένο το Caps Lock γιατί ο Κωδικός Πρόσβασης είναι Case Sensitive (δηλαδή τα κεφαλαία και τα πεζά γράμματα θεωρούνται ως διαφορετικά σύμβολα)

2) Αν μετά από πολλές προσπάθειες δεν καταφέρατε να μπείτε στο σύστημα πιθανώς να έχετε ξεχάσει τον κωδικό σας. Επικοινωνήστε με κάποιον υπεύθυνο εργαστηρίου είτε με email είτε από κοντά.

Πως σώζω ένα αρχείο στον Υπολογιστή μου

Απ:

Σε περίπτωση που θέλετε να αποθηκεύσετε ένα αρχείο (πχ pdf) από τον ηλεκτρονικό χώρο του εργαστηρίου στον υπολογιστή σας απλά **ανοίγετε το επιθυμητό αρχείο** (με αριστερό κλικ) και χρησιμοποιείτε το **save a copy.**