Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών
www.csd.uoc.gr

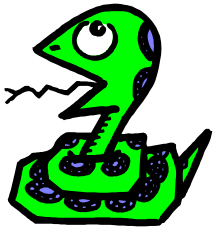
Εργαστήριο Υπηρεσιών Μετασχηματισμού
www.tsl.gr

ΗΥ-100 Εισαγωγή στην Επιστήμη Υπολογιστών

http://efront.tsl.gr

Όγδοη Διάλεξη

Python: επανάληψη (εντολή while)



Διδάσκων: Νικολάου Χρήστος

Χειμερινό εξάμηνο 2009-2010
Τετάρτη, 21/10/2009

Πολλαπλές εκχωρήσεις -1

- Όπως ίσως θα έχετε διαπιστώσει, είναι νόμιμο να γίνουν παραπάνω από μια εκχωρήσεις σε μια μεταβλητή. Μια νέα εκχώρηση κάνει μια μεταβλητή να αναφέρεται σε μια νέα τιμή και να σταματήσει να αναφέρεται στην παλιά τιμή.

```
amount = 4
print(amount)
amount = 9
print(amount)
```

Run:

```
>>>
4
9
>>>
```

Πολλαπλές εκχωρήσεις -2

- Στην Python πρέπει να θυμόμαστε να διακρίνουμε μεταξύ μιας εντολής εκχώρησης και μιας δήλωσης ισότητας!
  - άλλο x=3 και άλλο x==3
- Παρατηρείστε επίσης ότι ενώ στα μαθηματικά a=7, 7=a, είναι νόμιμα, στην Python a=7 είναι νόμιμο, ενώ 7=a δεν είναι!
- Υπάρχει και η περίπτωση της ανανέωσης της τιμής μιας μεταβλητής, όπου η νέα τιμή εξαρτάται από την παλιά, π.χ.

  x = x+1

Πολλαπλές εκχωρήσεις -3

- Σε μερικές γλώσσες προγραμματισμού χρησιμοποιείται διαφορετικό σύμβολο για την εκχώρηση, π.χ. <- ή := για την αποφυγή σύγχυσης.
- Χρησιμοποιείτε τις πολλαπλές εκχωρήσεις στην ίδια μεταβλητή με προσοχή – κάνουν το πρόγραμμα πιο δύσκολο να διαβαστεί και να ελεγχθεί.

Η εντολή while -1

- Οι υπολογιστές συχνά χρησιμοποιούνται για να κάνουν επαναλαμβανόμενες (ανιαρές) εργασίες. Η επανάληψη ίδιων ή παραπλήσιων εργασιών χωρίς λάθη είναι κάτι που οι υπολογιστές κάνουν καλά και όχι οι άνθρωποι.
- Είδαμε ήδη τη συνάρτηση countdown, που χρησιμοποιεί αναδρομή για να κάνει μια επαναληπτική εργασία. Επειδή η επανάληψη είναι κάτι πολύ συνηθισμένο, οι γλώσσες προγραμματισμού έχουν συνήθως διάφορες εντολές για το σκοπό αυτό.

Η εντολή while -2

- Η πρώτη εντολή που θα δούμε είναι η εντολή while.
  Παράδειγμα: countdown με while:

```
def countdown(n):
    while n > 0:
        print(n)
        n = n-1
    print("Blastoff!")

countdown(4)
```

Run:

```
>>>
4
3
2
1
Blastoff!
>>>
```

- Λίγο πολύ μπορούμε να διαβάσουμε το πρόγραμμα σαν να ήταν γραμμένο σε ανθρώπινη γλώσσα. Λέει: «Όσο το n είναι μεγαλύτερο από το 0, συνέχισε να εκτυπώνεις την τιμή του n και να τη μειώνεις κατά μια μονάδα. Όταν φτάσεις στο 0, δείξε στην οθόνη τη λέξη “Blastoff”».

Η εντολή while -3

- Αυστηρά, σε μια εντολή while υπάρχει η ακόλουθη ροή προγράμματος:

  1. Υπολόγισε τη συνθήκη, είτε σε αληθή (TRUE, 1) είτε σε ψευδή (FALSE, 0).
  2. Αν η συνθήκη είναι ψευδής, βγες από την εντολή while και συνέχισε την εκτέλεση της επόμενης μετά το while εντολής.
  3. Αν η συνθήκη είναι αληθής, τότε εκτέλεσε όλες τις εντολές μέσα στο «σώμα» εντολών που ακολουθεί και πήγαινε πίσω στο βήμα 1.

- Αυτός ο τύπος ροής ελέγχου ονομάζεται **βρόγχος,** επειδή από το τρίτο βήμα πάμε πίσω στο πρώτο.
- Παρατηρήστε ότι, αν η συνθήκη είναι ψευδής από την πρώτη φορά που ελέγχεται, το «σώμα» δεν εκτελείται ποτέ.

Η εντολή while -4

- Το «σώμα» του βρόγχου πρέπει να αλλάζει τις τιμές μιας ή περισσότερων μεταβλητών, τις οποίες ελέγχει η συνθήκη στην επικεφαλίδα του βρόγχου while, ώστε η συνθήκη να γίνεται τελικά ψευδής και ο βρόγχος να τελειώνει.
- Διαφορετικά ο βρόγχος θα επαναλαμβάνεται επ' άπειρον, γι' αυτό και στην περίπτωση αυτή ονομάζεται **ατέρμων βρόγχος (infinite loop).**
- Στην περίπτωση της συνάρτησης countdown μπορούμε να αποδείξουμε ότι ο βρόγχος τελειώνει, επειδή ξέρουμε ότι η τιμή του n είναι θετική και πεπερασμένη, βλέπουμε ότι η τιμή του n μειώνεται κάθε φορά που εκτελείται ο βρόγχος, άρα τελικά θα γίνει 0.

Η εντολή while -5

- Σε άλλες περιπτώσεις δεν είναι εύκολο να το αποδείξουμε:

```
def sequence(n):
    while n != 1:
        print(n)
        if n%2 == 0:   # n άρτιος
            n = n//2
        else:          # n περιττός
            n = n*3+1

sequence(3)
```

Run:

```
>>>
3
10
5
16
8
4
2
>>>
```

- Η συνθήκη του βρόγχου είναι n != 1, άρα ο βρόγχος συνεχίζει μέχρι το n να γίνει 1, πράγμα που θα κάνει τη συνθήκη ψευδή.
- Κάθε φορά που εκτελεί το σώμα του βρόγχου, το πρόγραμμα δείχνει την τιμή του n και μετά ελέγχει αν ο n είναι άρτιος ή περιττός. Αν είναι άρτιος, ο n διαιρείται με το 2, αν είναι περιττός, αντικαθίσταται με το n*3+1.

Η εντολή while -6

- Βλέπουμε ότι το n μια μεγαλώνει και μια μικραίνει, άρα δεν υπάρχει προφανής απόδειξη ότι θα γίνει ποτέ 1, και επομένως θα τελειώσει το πρόγραμμα.
- Για συγκεκριμένες τιμές του n μπορούμε να αποδείξουμε τερματισμό.
- Π.χ. αν το n είναι δύναμη του 2, τότε το n θα είναι πάντα άρτιο μέσα στο βρόγχο, μέχρι να φτάσει τη μονάδα (π.χ. με n=16 το πρόγραμμα τερματίζει).
- Αλλά μπορούμε να έχουμε μια γενική απόδειξη ότι το πρόγραμμα τερματίζει για κάθε τιμή του n; Ακόμη την ψάχνουμε!!!

Εκτύπωση πινάκων -1

- Ένα από τα πράγματα που μπορούμε να κάνουμε καλά με βρόγχους είναι η εκτύπωση πινάκων δεδομένων.
- Στην προ των υπολογιστών εποχή, έπρεπε να υπολογίζουμε λογαρίθμους, ημίτονα και συνημίτονα, και άλλες μαθηματικές συναρτήσεις με το χέρι. Για ευκολία υπήρχαν βιβλία με τεράστιους πίνακες τιμών αυτών των συναρτήσεων. Η δημιουργία των πινάκων αυτών ήταν αργή και ανιαρή, και γεμάτη λάθη.
- Με την έλευση των υπολογιστών, αυτοί οι πίνακες εξαφανίστηκαν...
- Εκτός από ορισμένες περιπτώσεις που οι ίδιοι οι υπολογιστές χρησιμοποιούν πίνακες τιμών για υπολογισμό τιμών κάποιων συναρτήσεων προσεγγιστικά, και στη συνέχεια βελτιώνουν την προσέγγιση με επιπλέον υπολογισμούς, π.χ. σε πράξεις floating point.
- Ας δούμε λοιπόν πως μπορούμε να εκτυπώσουμε έναν πίνακα λογαρίθμων χρησιμοποιώντας βρόγχους.

Εκτύπωση πινάκων -2

- Το επόμενο πρόγραμμα τυπώνει μια σειρά τιμών στην αριστερή κολώνα και τους λογαρίθμους τους στη δεξιά:

```
import math

x = 1.0
while x <= 5.0:
    print(x, '\t', math.log(x))
    x = x + 1.0
```

Run:

```
>>>
1.0		0.0
2.0		0.69314718056
3.0		1.09861228867
4.0		1.38629436112
5.0		1.60943791243
>>>
```

- Η συμβολοσειρά '\t' παριστάνει το χαρακτήρα «tab» (από τη λέξη tabulation, που σημαίνει εκτύπωση πίνακα δεδομένων).
- Καθώς χαρακτήρες και ομάδες χαρακτήρων τυπώνονται στην οθόνη, ο δρομέας οθόνης (κέρσορας, cursor) δείχνει πάντα που θα τυπωθεί ο επόμενος χαρακτήρας. Μετά από μια εντολή print, ο δρομέας συνήθως δείχνει στην αρχή της επόμενης γραμμής. Ο χαρακτήρας tab (στηλογνώμονας στη γραφομηχανή) μετατοπίζει το στηλογνώμονα μέχρι το επόμενο tab stop.

Εκτύπωση πινάκων -3

- Αυτοί οι λογάριθμοι είναι με βάση το e. Επειδή οι δυνάμεις του 2 είναι σημαντικές στην επιστήμη των υπολογιστών, συχνά χρειάζεται να υπολογίσουμε λογαρίθμους με βάση το 2. Για το σκοπό αυτό μπορούμε να χρησιμοποιήσουμε τον εξής τύπο:

$$\log_2 x = \frac{\log_e x}{\log_e 2}$$

- Στην πράξη:
  - Αλλάξτε το πρόγραμμα της προηγούμενης διαφάνειας ώστε να τυπώνει λογαρίθμους με βάση το 2.

Εκτύπωση πινάκων -4

```
import math

x = 1.0
while x <= 10.0:
    print(x, '\t', math.log(x)/math.log(2.0))
    x = x + 1.0
```

Run:

```
>>>
1.0     0.0
2.0     1.0
3.0     1.58496250072
4.0     2.0
5.0     2.32192809489
6.0     2.58496250072
7.0     2.80735492206
8.0     3.0
9.0     3.16992500144
10.0    3.32192809489
```

- Διακρίνουμε τις δυνάμεις του 2: 2, 4, 8, ...

Εκτύπωση δισδιάστατων πινάκων

- Ένας δισδιάστατος πίνακας είναι ένας πίνακας όπου διαβάζουμε τιμές στη διασταύρωση σειρών και στηλών. Καλό παράδειγμα είναι ένας πίνακας πολλαπλασιασμού, π.χ. των τιμών από 1 έως 6.
- Ας αρχίσουμε τυπώνοντας τα πολλαπλάσια του 2 σε μια γραμμή:

```
i = 1
while i <= 6:
    print(2*i, '\t', end=' ')
    i = i + 1
print()
```

Run:

```
>>>
2       4       6       8       10      12
>>>
```

- Το όρισμα end=' ' στην print χρησιμοποιείται ώστε σε κάθε επανάληψη να διατηρείται η εκτύπωση στην ίδια γραμμή.

Ενθυλάκωση και Γενίκευση -1

- Ενθυλάκωση είναι το να «περιτυλίξουμε» ένα κομμάτι κώδικα μέσα σε μια συνάρτηση, για να εκμεταλλευτούμε όλα όσα προσφέρει μια συνάρτηση, θυμηθείτε την isDivisible.
- Γενίκευση είναι η διαδικασία του να πάρουμε κάτι συγκεκριμένο, όπως η εκτύπωση των πολλαπλασίων του 2, και να το κάνουμε πιο γενικό, όπως π.χ. η εκτύπωση των πολλαπλασίων κάθε ακεραίου.
- Η παρακάτω συνάρτηση ενθυλακώνει τον προηγούμενο βρόγχο και τυπώνει πολλαπλάσια του n:

```
def printMultiples(n):
    i = 1
    while i <= 6:
        print(n*i, '\t', end=' ')
        i = i + 1
    print()
```

Ενθυλάκωση και Γενίκευση -2

- Εάν καλέσουμε τη συνάρτηση αυτή με όρισμα 2 παίρνουμε ακριβώς το ίδιο αποτέλεσμα όπως πριν.
- Με όρισμα 3 η έξοδος είναι:

```
>>>
3       6       9       12      15      18
>>>
```

- Με όρισμα 4 η έξοδος είναι:

```
>>>
4       8       12      16      20      24
>>>
```

Ενθυλάκωση και Γενίκευση -3

- Στην πράξη:
  - Προχωρήστε και φτιάξτε έναν πίνακα πολλαπλασιασμού, καλώντας την printMultiples κατ' επανάληψη και με διαφορετικά ορίσματα, π.χ. μέσα από ένα καινούργιο βρόγχο.

Ενθυλάκωση και Γενίκευση -4

```
def printMultiples(n):
    i = 1
    while i <= 6:
        print(n*i, '\t', end=' ')
        i = i + 1
    print()

i = 1
while i <= 6:
    printMultiples(i)
    i = i + 1
```

Run:

```
>>>
1       2       3       4       5       6
2       4       6       8       10      12
3       6       9       12      15      18
4       8       12      16      20      24
5       10      15      20      25      30
6       12      18      24      30      36
>>>
```

- Προσέξτε πόσο παρόμοιος είναι αυτός ο βρόγχος με το βρόγχο μέσα στην printMultiples. Το μόνο που κάναμε είναι να αντικαταστήσουμε την εντολή print με μια κλήση συνάρτησης.

Ενθυλάκωση και Γενίκευση -5

- Μπορούμε βέβαια να πάρουμε πάλι τον προηγούμενο κώδικα και να τον περιτυλίξουμε μέσα σε μια νέα συνάρτηση:

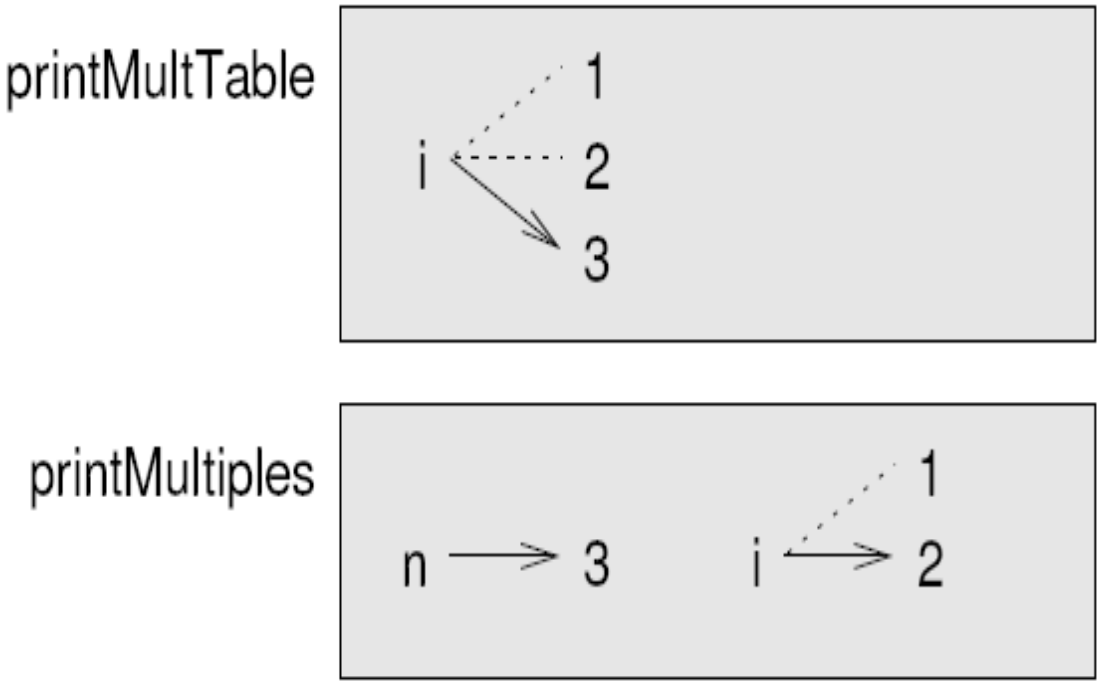
```
def printMultTable():
    i = 1
    while i <= 6:
        printMultiples(i)
        i = i + 1
```

- Αυτή η διαδικασία χρησιμοποιείται συχνά για την ανάπτυξη κώδικα. Αναπτύσσουμε πρώτα κώδικα έξω από συνάρτηση, π.χ. δακτυλογραφώντας στο διερμηνευτή. Όταν σιγουρευτούμε ότι ο κώδικας δουλεύει, τον περιτυλίσσουμε με μια συνάρτηση. Μια τέτοια μέθοδος είναι ιδιαίτερα χρήσιμη όταν δεν γνωρίζουμε από την αρχή πως να χωρίσουμε ένα μεγάλο πρόγραμμα σε συναρτήσεις.

Τοπικές Μεταβλητές

- Μπορεί να διερωτάστε: πως είναι δυνατόν να χρησιμοποιούμε το ίδιο όνομα μεταβλητής i στις printMultiples και printMultTable;
- Δεν δημιουργείται πρόβλημα όταν η μια συνάρτηση αλλάζει την τιμή αυτής της μεταβλητής;
- Η απάντηση είναι όχι, γιατί το i στην printMultiples δεν είναι το ίδιο με το i στην printMultTable.
- Μεταβλητές που δημιουργούνται μέσα σε μια συνάρτηση είναι **τοπικές, και δεν μπορεί να τις προσπελάσει κανείς έξω από τη συνάρτηση που είναι το «σπίτι» της!**
- Αυτό σημαίνει ότι είστε ελεύθεροι να χρησιμοποιείτε μεταβλητές με το ίδιο όνομα, εφόσον δεν χρησιμοποιούνται μέσα στην ίδια συνάρτηση!

Τοπικές μεταβλητές - διάγραμμα στοίβας



Η εντολή break

- Με την εντολή break τερματίζονται άμεσα οι επαναλήψεις μιας εντολής while.
- Παράδειγμα: γράφουμε ένα πρόγραμμα το οποίο δέχεται συνεχώς είσοδο από το πληκτρολόγιο, τυπώνει στην οθόνη την είσοδο που διάβασε και τερματίζει μόνο όταν διαβάσει την λέξη end:

```
while True:
    s = input('Type something:')
    print(s)
    if s == 'end':
        break
```

Τέλος