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

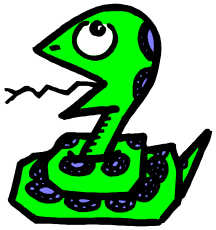
Εργαστήριο Υπηρεσιών Μετασχηματισμού
www.tsl.gr

HY-100 Εισαγωγή στην Επιστήμη Υπολογιστών

http://efront.tsl.gr

Τέταρτη Διάλεξη

Python: εκφράσεις, τελεστές, σχόλια



Διδάσκων: Νικολάου Χρήστος

Χειμερινό εξάμηνο 2009-2010
Τετάρτη, 08/10/2009

Εκφράσεις (Expressions) -1

- Μια έκφραση (expression) είναι συνδυασμός τιμών, μεταβλητών και τελεστών.
- Οι τελεστές (operators) είναι λειτουργίες που κάνουν κάτι και μπορούν να αναπαρασταθούν με σύμβολα όπως το + ή με λέξεις κλειδιά όπως το and.
- Οι τιμές και οι μεταβλητές, πάνω στις οποίες εφαρμόζονται οι τελεστές, ονομάζονται τελεστέοι (operands).
- Εάν εισάγετε μια έκφραση στον διερμηνευτή, αυτός την υπολογίζει και δείχνει το αποτέλεσμα:

```
>>> 2+1
3
>>> x=4
>>> x+2
6
```

- Δεν χρειάζεται μια έκφραση να περιέχει ταυτόχρονα και τιμές και μεταβλητές και τελεστές. Μια τιμή, όπως και μια μεταβλητή, από μόνες τους είναι επίσης εκφράσεις:

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

Προσοχή:
στη μεταβλητή x θα πρέπει να έχει ανατεθεί νωρίτερα κάποια τιμή

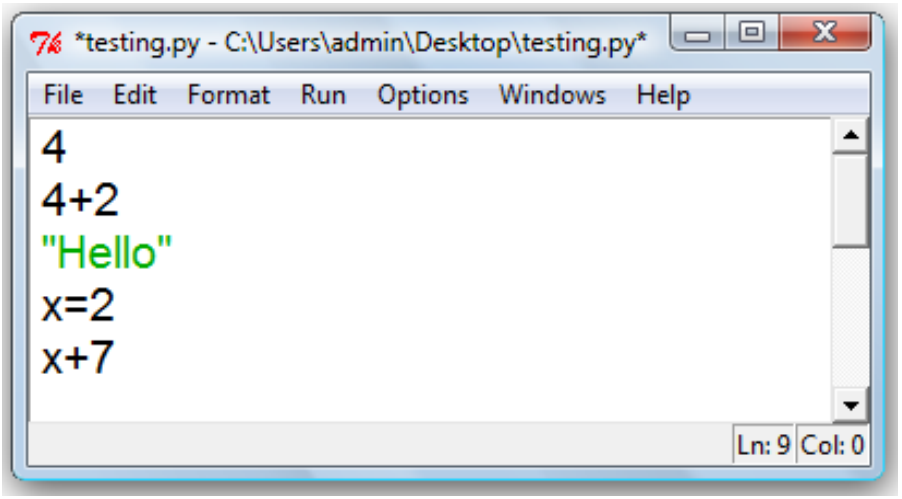
Εκφράσεις (Expressions) -2

- Παρατηρείστε τη διαφορά μεταξύ του υπολογισμού μιας έκφρασης και της εκτύπωσής της:

```
>>> message = "What's up, Doc?"
>>> message
"What's up, Doc?"
>>> print(message)
What's up, Doc?
```

Εκφράσεις (Expressions) -3

- Σε ένα σενάριο (script), μια έκφραση είναι από μόνη της μια νόμιμη εντολή, αλλά δεν παράγει έξοδο:



Run

```
>>> ================================ RESTART ===
>>>
>>>
```

- Πως θα παραχθεί έξοδος;
  - χρησιμοποιούμε την print

Τελεστές -1

+	Πρόσθεση αριθμών ή αλληλουχία συμβολοσειρών.
-	Αφαίρεση ενός αριθμού από έναν άλλο.
*	Γινόμενο δύο αριθμών ή επανάληψη μιας συμβολοσειράς τόσες φορές.
**	Ύψωση αριθμού σε δύναμη.
/	Διαίρεση δύο αριθμών.
//	Διαίρεση δύο αριθμών στρογγυλοποιημένη προς τα κάτω (floor division).
%	Υπόλοιπο διαίρεσης δύο αριθμών.

- Υπάρχουν και άλλοι τελεστές, θα τους δούμε σε επόμενες διαλέξεις.

Τελεστές -2

- Παραδείγματα πράξεων με αριθμούς:

```
>>> 2+3
5
>>> 7-4
3
>>> 2*5
10
>>> 5**2
25
>>> 10/2
5.0
>>> 10//2
5
>>> 10/3
3.3333333333333335
>>> 10//3
3
>>> 1/2
0.5
>>> 1//2
0
>>> 1.0//2
0.0
>>> 7%3
1
```

```
>>> 5.5+0.5
6.0
>>> 3+2.0
5.0
>>> 7.0-2
5.0
>>> 2*5.0
10.0
>>> 2.0**2
4.0
>>> 4**2.0
16.0
```

Παραδείγματα εκφράσεων

- 12+24*hour-6
- hour*60+minute
- minute/60
- 6**3
- (6+9)*(31-7)

- Όταν μια μεταβλητή εμφανίζεται σε έκφραση, αντικαθίσταται από την τιμή της πριν υπολογιστεί η έκφραση:

```
>>> x=5
>>> x+2
7
```

Προτεραιότητα τελεστών -1

- Όταν περισσότερα από ένα σύμβολα τελεστών εμφανίζονται σε μια έκφραση, η σειρά υπολογισμού εξαρτάται από τους κανόνες προτεραιότητας. Το ακρωνύμιο **PEMDAS** βοηθάει να θυμόμαστε αυτούς τους κανόνες.

- Οι **παρενθέσεις (P**arenthesis) έχουν τη μεγαλύτερη προτεραιότητα και χρησιμοποιούνται για να αναγκάσουν την Python να υπολογίσει μια έκφραση σύμφωνα με τη σειρά που θέλουμε. Εκφράσεις σε παρενθέσεις υπολογίζονται πρώτες. Παρενθέσεις χρησιμοποιούνται επίσης για να κάνουν τις εκφράσεις πιο αναγνώσιμες, χωρίς να αλλάζουν το αποτέλεσμα.

```
>>> 2 * (3-1)
4
>>> (2+1)**(5-2)
27
```

```
>>> 2*3-1
5
>>> 2+1**5-2
1
```

```
>>> 5*2+3
13
>>> (5*2)+3
13
```

- Η **ύψωση σε δύναμη (E**xponentiation) έχει την επόμενη μεγαλύτερη προτεραιότητα.

```
>>> 2**1+1
3
>>> 3*1**3
3
```

Προτεραιότητα τελεστών -2

- **Πολλαπλασιασμός (M**ultiplication) και **διαίρεση (D**ivision) έχουν την ίδια προτεραιότητα, που είναι μεγαλύτερη από την **πρόσθεση (A**ddition) και την **αφαίρεση (S**ubtraction) που έχουν επίσης την ίδια προτεραιότητα.

```
>>> 3*4+2
14
>>> 15/3-2
3.0
```

- Τελεστές με την ίδια προτεραιότητα υπολογίζονται από τα αριστερά προς τα δεξιά.

Πράξεις στις συμβολοσειρές -1

- Γενικά δεν μπορούμε να κάνουμε μαθηματικές πράξεις με συμβολοσειρές (strings). Τα παρακάτω δεν είναι σωστά (υποθέτοντας ότι η μεταβλητή message είναι τύπου string):

  message-1 "Hello"/2 "Hello"/"H" message*"Hello" "15"+2

- Ειδικά ο τελεστής + έχει νόημα για συμβολοσειρές:
  - Δημιουργεί μια αλληλουχία (concatenation) συμβολοσειρών, δηλαδή συνδέει τους δύο «προσθετέους» φτιάχνοντας νέα συμβολοσειρά, όπου η αρχή της δεύτερης βρίσκεται μετά το τέλος της πρώτης.
    - Παράδειγμα:

```
>>> first_name="James"
>>> last_name=" Bond"
>>> my_name=first_name + last_name
>>> print(my_name)
James Bond
```

Πράξεις στις συμβολοσειρές -2

- Επιπλέον ο τελεστής * έχει νόημα στις συμβολοσειρές, εκτελεί επανάληψη:

```
>>> 3*"Hello"
'HelloHelloHello'
>>> print(3*"Hello")
HelloHelloHello
```

- Ο ένας από τους πολλαπλασιαστέους πρέπει να είναι συμβολοσειρά και ο άλλος ακέραιος.
- Υπάρχουν ομοιότητες και διαφορές με τις αντίστοιχες πράξεις της αριθμητικής.
  - Όπως το 3*4 είναι ισοδύναμο με το 4+4+4, έτσι και το 3*"Hello" είναι ισοδύναμο με το "Hello"+"Hello"+"Hello".
  - Μπορείτε να σκεφτείτε μία ιδιότητα που έχει η πρόσθεση (addition) αριθμών, ενώ η αλληλουχία (concatenation) συμβολοσειρών δεν την έχει;

Σύνθεση

- Στην Python υπάρχει η δυνατότητα σύνθεσης μεταβλητών, εκφράσεων και εντολών, για παράδειγμα:

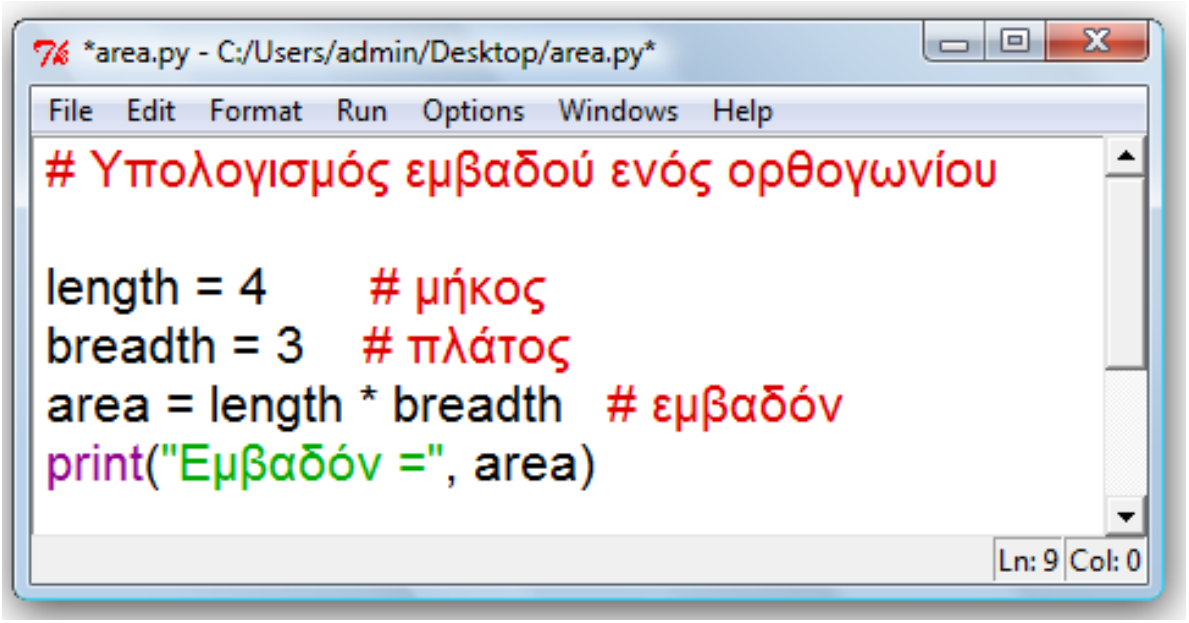
```
>>> hours=2
>>> minutes=50
>>> print("Number of minutes since midnight:", hours*60+minutes)
Number of minutes since midnight: 170
```

- Υπάρχουν βέβαια όρια σε αυτό, π.χ. η εκχώρηση: minute+1 = hour δεν είναι σωστή.

Σχόλια (Comments)

- Καθώς γράφουμε όλο και μεγαλύτερα προγράμματα, μεγαλώνει και η δυσκολία να καταλαβαίνουμε αυτά που γράφουμε.
- Οι γλώσσες προγραμματισμού συμπυκνώνουν νοήματα, γι' αυτό και είναι δύσκολο να τις διαβάσει κανείς, να καταλάβει τι θέλει να κάνει ένα κομμάτι κώδικα και γιατί. Γι' αυτό και χρειάζεται να προσθέτουμε σχόλια στα προγράμματα που γράφουμε.
- Τα σχόλια στην Python αρχίζουν πάντα με το χαρακτήρα #
  - Οτιδήποτε ακολουθεί μετά το # αγνοείται από την Python μέχρι το τέλος της γραμμής.

Σχόλια - παράδειγμα



Run

```
>>> ================================ RESTART ===
>>>
Εμβαδόν = 12
>>>
```

Τέλος