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

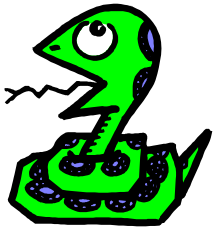
Εργαστήριο Υπηρεσιών Μετασχηματισμού
www.tsl.gr

ΗΥ-100 Εισαγωγή στην Επιστήμη Υπολογιστών

http://efront.tsl.gr

Έκτη Διάλεξη

Python: εκτέλεση υπό συνθήκη, αναδρομή, είσοδος από το πληκτρολόγιο



Διδάσκων: Νικολάου Χρήστος

Χειμερινό εξάμηνο 2009-2010
Τετάρτη, 14/10/2009

Ο τελεστής modulus -1

- Ο τελεστής modulus εφαρμόζεται σε ακεραίους (και εκφράσεις ακεραίων) και έχει ως αποτέλεσμα το υπόλοιπο της διαίρεσης του πρώτου ακέραιου από το δεύτερο. Στην Python το σύμβολό του είναι το %.
- Παράδειγμα:

```
>>> quotient = 7 // 3
>>> print(quotient)
2
>>> remainder = 7 % 3
>>> print(remainder)
1
```

Ο τελεστής modulus -2

- Ο modulus είναι εξαιρετικά χρήσιμος. Παραδείγματος χάριν, για να ελέγξουμε κατά πόσο ο αριθμός x είναι διαιρετός από τον αριθμό y, εκτελούμε x % y και αν το αποτέλεσμα είναι μηδέν, τότε ο y διαιρεί τον x.
- Επίσης μπορούμε να εξαγάγουμε το πιο δεξιό ψηφίο ενός ακεραίου με x % 10, ή τα δύο τελευταία ψηφία ενός ακεραίου με x % 100.

```
>>> print(37%10)
7
>>> print(375%100)
75
```

Λογικές εκφράσεις (boolean expressions) -1

- Μια λογική έκφραση είναι είτε αληθής (true) είτε ψευδής (false). Ένας τρόπος να γράψουμε λογική έκφραση είναι με χρήση του τελεστή == που συγκρίνει δύο τιμές και παράγει μια λογική (boolean) τιμή:

```
>>> 5 == 5
True
>>> 5 == 6
False
```

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

- Στην πρώτη εντολή οι δύο αριθμοί είναι ίσοι, άρα η τιμή της λογικής έκφρασης είναι True, ενώ στη δεύτερη εντολή οι δύο αριθμοί είναι άνισοι, άρα παίρνουμε False.
- Οι True και False είναι ειδικές τιμές που είναι μέρος της Python (λέξεις κλειδιά) και ανήκουν στον τύπο bool.

Λογικές εκφράσεις (boolean expressions) -2

- Ο τελεστής == είναι ένας από τους τελεστές σύγκρισης, οι υπόλοιποι είναι οι παρακάτω:

```
x != y      # x δεν είναι ίσος με y
x > y       # x είναι μεγαλύτερος από y
x < y       # x είναι μικρότερος από y
x >= y      # x είναι μεγαλύτερος ή ίσος με y
x <= y      # x είναι μικρότερος ή ίσος με y
```

- Προσοχή να μην γίνεται σύγχυση μεταξύ των συμβόλων της Python και ίδιων ή παρόμοιων μαθηματικών συμβόλων.
- Συνηθισμένο λάθος: χρήση του απλού μαθηματικού συμβόλου ισότητας = αντί του συμβόλου ισότητας στην Python που είναι ==.
- = σημαίνει εκχώρηση στη Python, ενώ == είναι τελεστής ισότητας.
- Επίσης δεν υπάρχουν τα =< και =>

Λογικοί τελεστές

- Υπάρχουν τρεις λογικοί τελεστές: and, or και not. Η σημαντική αυτών των τελεστών είναι πολύ παρόμοια με την καθημερινή σημασία τους.
- Π.χ. x > 0 and x < 10 είναι αληθής μόνο και μόνον όταν το x είναι μεγαλύτερο του μηδενός **και** μικρότερο του 10.
- n%2 == 0 or n%3 == 0 είναι αληθής όταν **τουλάχιστον** μια από τις συνθήκες είναι αληθής, δηλαδή αν ο αριθμός n διαιρείται από το 2 ή το 3.
- Τέλος ο τελεστής not **αντιστρέφει** την τιμή μιας λογικής έκφρασης, π.χ. not(x > y) είναι αληθής αν (x > y) είναι ψευδής, δηλαδή αν το x είναι μικρότερο ή ίσο του y.
- Αυστηρά μιλώντας, οι λογικοί τελεστές πρέπει να εφαρμόζονται πάνω σε λογικές εκφράσεις. Όμως η Python δεν είναι αυστηρή και ερμηνεύει (σε λογικές εκφράσεις) κάθε μη μηδενικό αριθμό ως True.

```
>>> x = 1
>>> x and 2
2
>>> y = 0
>>> y and 2
0
```

Υπό συνθήκη εκτέλεση προγράμματος -1

- Για να μπορούμε να δημιουργούμε χρήσιμα προγράμματα, σχεδόν πάντα χρειαζόμαστε τη δυνατότητα να ελέγχουμε την ισχύ συνθηκών και να αλλάζουμε τη συμπεριφορά του προγράμματος ανάλογα με το αποτέλεσμα του ελέγχου. Οι εντολές συνθήκης μας δίνουν αυτή τη δυνατότητα. Στην απλούστερή τους μορφή έχουμε την εντολή if:

```
>>> x = 5
>>> if x > 0:
        print("x is positive")


x is positive
>>>
```

- Η λογική έκφραση μετά το if ονομάζεται συνθήκη. Αν είναι αληθής τότε εκτελείται η εντολή (οι εντολές) που είναι μετατοπισμένες προς τα δεξιά. Αν είναι ψευδής δεν εκτελείται τίποτε. Η εντολή if είναι γενικά της μορφής μιας σύνθετης εντολής (compound statement):

```
HEADER:
  FIRST STATEMENT
  ...
  LAST STATEMENT
```

Υπό συνθήκη εκτέλεση προγράμματος -2

- Η **εντολή επικεφαλίδα (header)** ξεκινά πάντοτε σε νέα γραμμή και τελειώνει με άνω και κάτω τελεία (:).
- Οι προς τα δεξιά μετατοπισμένες εντολές που ακολουθούν αποτελούν ένα μπλοκ εντολών. Η πρώτη εντολή που ευθυγραμμίζεται με το αριστερό περιθώριο είναι η πρώτη εντολή έξω από το μπλοκ.
- Ένα μπλοκ εντολών μέσα σε μια σύνθετη εντολή ονομάζεται το **σώμα** (body) της εντολής.
- Δεν υπάρχει όριο στο πλήθος των εντολών που μπορούν να εμφανισθούν στο σώμα μιας εντολής if, αλλά πρέπει να υπάρχει τουλάχιστον μια.
- Αν παρόλα αυτά χρειαζόμαστε ένα σώμα χωρίς εντολές (π.χ. για να θυμηθούμε αργότερα να πάμε και να συμπληρώσουμε εντολές), τότε μπορεί να χρησιμοποιήσουμε την εντολή pass που δεν κάνει τίποτε.

Εναλλακτική εκτέλεση

- Μια δεύτερη μορφή της εντολής if είναι η εναλλακτική εξέταση, κατά την οποία υπάρχουν δύο δυνατότητες (ή κλάδοι εκτέλεσης) και η συνθήκη καθορίζει ποια εκτελείται. Η σύνταξη είναι η εξής:

```
x=4
if x%2 == 0:
    print(x, "is even")
else:
    print(x, "is odd")
```

Run:

```
>>>
4 is even
>>>
```

```
x=5
if x%2 == 0:
    print(x, "is even")
else:
    print(x, "is odd")
```

Run:

```
>>>
5 is odd
>>>
```

Αλυσιδωτές συνθήκες -1

- Παράδειγμα:

```
x=7
y=4

if x < y:
    print(x, "is less than", y)
elif x > y:
    print(x, "is greater than", y)
else:
    print(x, "and", y, "are equal")
```

→ Run:

```
>>>
7 is greater than 4
>>>
```

- elif είναι συντόμευση του else if.
- Ένας μόνο κλάδος εκτελείται.
- Δεν υπάρχει όριο στο πλήθος των εντολών elif.

Αλυσιδωτές συνθήκες -2

```
if choice == 'A':
    functionA()
elif choice == 'B':
    functionB()
elif choice == 'C':
    functionC()
else:
    print('Invalid choice.')
```

- Κάθε συνθήκη εξετάζεται με τη σειρά, αν η πρώτη είναι ψευδής πάμε στη δεύτερη, κ.ο.κ.
- Αν κάποια συνθήκη είναι αληθής, τότε ο αντίστοιχος κλάδος εκτελείται και η εντολή τελειώνει, ακόμη και αν άλλες συνθήκες (παρακάτω) είναι αληθείς.

Εμφωλευμένες συνθήκες (Nested conditionals) -1

- Παράδειγμα:

```
x=7
y=4

if x == y:
    print(x, "and", y, "are equal")
else:
    if x < y:
        print(x, "is less than", y)
    else:
        print(x, "is greater than", y)
```

→ Run:

```
>>>
7 is greater than 4
>>>
```

- Οι εμφωλευμένες συνθήκες κάνουν το πρόγραμμα δυσανάγνωστο και γι' **αυτό πρέπει να αποφεύγονται.**

Εμφωλευμένες συνθήκες (Nested conditionals) -2

- Οι λογικοί τελεστές βοηθούν στην απλοποίηση των εμφωλευμένων συνθηκών. Παράδειγμα:

```
if  0 < x:
    if x < 10:
        print("x is a positive single digit.")
```

- Η εντολή εκτύπωσης εκτελείται μόνο αν περάσουμε επιτυχώς και τους δυο ελέγχους, άρα μπορούμε να χρησιμοποιήσουμε τον λογικό τελεστή and:

```
if 0 < x and x < 10:
    print("x is a positive single digit.")
```

- Η Python δέχεται επίσης και το:

```
if 0 < x < 10:
    print("x is a positive single digit.")
```

Η εντολή return

- Μας επιτρέπει τον τερματισμό της εκτέλεσης μιας συνάρτησης πριν φθάσουμε το τέλος του κώδικα της συνάρτησης αυτής, π.χ. διότι ανακαλύψαμε λάθος:

```
import math

def printLogarithm(x):
    if x <= 0:
        print("Positive numbers only, please.")
        return
    result = math.log(x)
    print("The log of x is", result)

printLogarithm(-2)
printLogarithm(2)
```

→Run:

```
>>>
Positive numbers only, please.
The log of x is 0.69314718056
>>>
```

Τι είναι η αναδρομή; -1

- Έχουμε πει ήδη ότι είναι νόμιμο μια συνάρτηση να καλεί μια άλλη. Αλλά είναι επίσης νόμιμο μια συνάρτηση να καλεί τον εαυτό της. Μπορεί να μην είναι προφανές γιατί αυτό μπορεί να είναι κάτι καλό, αλλά σίγουρα είναι ένα από τα πιο κομψά πράγματα που μπορούμε να κάνουμε με μια γλώσσα προγραμματισμού.
- Η διαδικασία όπου μια συνάρτηση καλεί τον εαυτό της λέγεται **αναδρομή** και οι συναρτήσεις αυτές λέγονται **αναδρομικές**.

Τι είναι η αναδρομή; -2

- Παράδειγμα:

```
def countdown(n):
    if n == 0:
        print("Blastoff!")
    else:
        print(n)
        countdown(n-1)
```

- Η countdown έχει μια παράμετρο, την n, που είναι θετικός ακέραιος. Αν το n είναι 0 τότε τυπώνει “Blastoff!”. Αλλιώς, τυπώνει n και μετά καλεί τη συνάρτηση countdown (τον εαυτό της δηλαδή) περνώντας το n-1 ως όρισμα.
- Τι συμβαίνει εάν την τρέξουμε με όρισμα τον αριθμό 3;

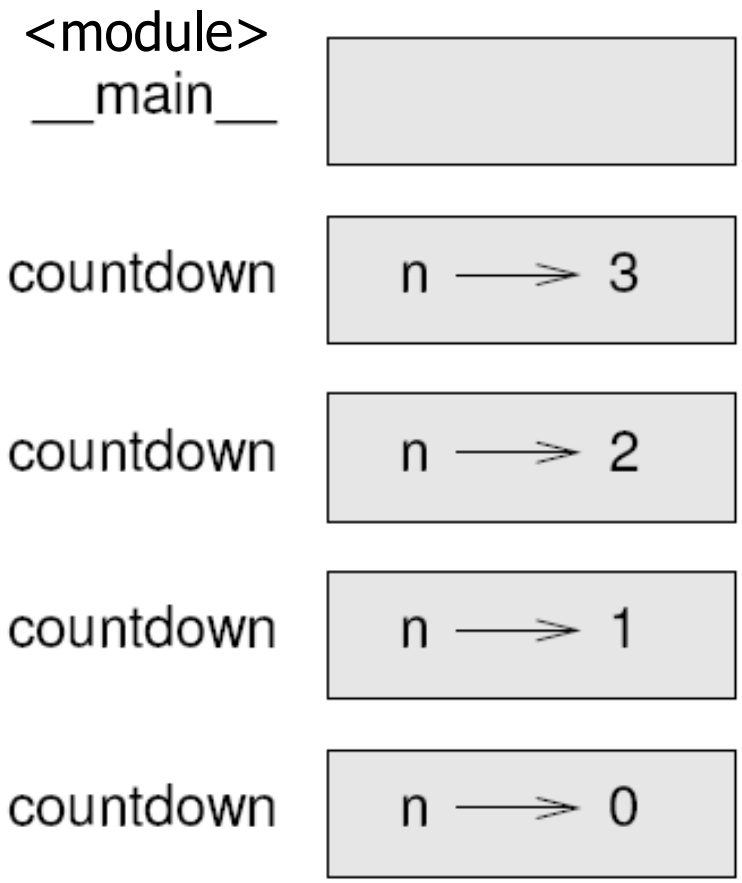
  countdown(3)

Τι είναι η αναδρομή; -3

- Η εκτέλεση της countdown ξεκινά με n=3, και επειδή το n δεν είναι 0, τυπώνει την τιμή 3 και μετά καλεί τον εαυτό της...
- Η εκτέλεση της countdown ξεκινά με n=2, και επειδή το n δεν είναι 0, τυπώνει την τιμή 2 και μετά καλεί τον εαυτό της...
- Η εκτέλεση της countdown ξεκινά με n=1, και επειδή το n δεν είναι 0, τυπώνει την τιμή 1 και μετά καλεί τον εαυτό της...
- Η εκτέλεση της countdown ξεκινά με n=0, και επειδή το n είναι 0, τυπώνει την λέξη “blastoff!” και μετά επιστρέφει.
- Η countdown με n=1 επιστρέφει.
- Η countdown με n=2 επιστρέφει.
- Η countdown με n=3 επιστρέφει.
- Και τώρα είμαστε πίσω στο main (κυρίως πρόγραμμα). Άρα συνολικά η έξοδος φαίνεται ως εξής:

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

Διάγραμμα στοίβας για αναδρομικές συναρτήσεις



Άπειρη Αναδρομή

- Εάν μια αναδρομή δεν φτάνει ποτέ τη λεγόμενη «περίπτωση βάσης», π.χ. n==0 στο παράδειγμα μας, συνεχίζει να κάνει αναδρομικές κλήσεις επ' άπειρον και το πρόγραμμα δεν τελειώνει ποτέ. Αυτό λέγεται άπειρη αναδρομή και εν γένει δεν είναι καθόλου καλή ιδέα! Παράδειγμα άπειρης αναδρομής:

```
def recurse():
    recurse()

recurse()
```

- Στις περισσότερες γλώσσες προγραμματισμού, η άπειρη αναδρομή δεν επιτρέπεται γιατί απλούστατα δεν μπορεί να επιτραπεί άπειρα μεγάλη στοίβα (κάποτε θα τελειώσει η μνήμη του υπολογιστή). Στην Python:

```
  File "C:\Users\admin\Desktop\testing2.py", line 4, in recurse
    recurse()
RuntimeError: maximum recursion depth exceeded
>>>
```

Είσοδος από το πληκτρολόγιο -1

- Τα προγράμματα που έχουμε γράψει μέχρι τώρα είναι αγενή γιατί δεν δέχονται είσοδο από τον χρήστη! Άρα κάνουν το ίδιο πράγμα κάθε φορά, κάτι που είναι μάλλον περιορισμένο και πληκτικό.
- Η Python διαθέτει τη συνάρτηση input για αυτό.
- Όταν καλείται η συνάρτηση αυτή το πρόγραμμα σταματάει και περιμένει ο χρήστης να πληκτρολογήσει κάτι. Όταν ο χρήστης πατήσει τα πλήκτρα Return ή Enter, το πρόγραμμα συνεχίζει και η input επιστρέφει αυτό που πληκτρολόγησε ο χρήστης σαν συμβολοσειρά.

```
>>> message = input()
Hello
>>> message
'Hello'
>>> n = input()
5
>>> n
'5'
>>>
```

Είσοδος από το πληκτρολόγιο -2

- Πριν καλέσετε την input, είναι καλή ιδέα να τυπώσετε ένα μήνυμα που θα πληροφορεί το χρήστη τι είσοδο περιμένετε. Το μήνυμα αυτό λέγεται «prompt», και μπορούμε να το δώσουμε ως όρισμα στην input:

```
>>> n = input("How old are you?")
How old are you?20
>>>
```

Είσοδος από το πληκτρολόγιο -3

- Επειδή η input επιστέφει μία συμβολοσειρά, μπορείτε να χρησιμοποιήσετε μία από τις συναρτήσεις μετατροπής τύπων εάν η είσοδος σας είναι ακέραιος ή πραγματικός αριθμός.
- Παράδειγμα:

```
>>> x = input("Give the first number:")
Give the first number:5
>>> y = input("Give the second number:")
Give the second number:7
>>> print(x,"+",y,"=",int(x)+int(y))
5 + 7 = 12
>>>
```

Τέλος