Σημειωματάριο Δευτέρας 9 Οκτ. 2017

Η δομή ελέγχου `if` ... `else` ... `elif`

Βλέπουμε τώρα πώς μπορούμε να γράψουμε προγράμματα που η εκτέλεσή τους ακολουθεί διαφορετική πορεία ανάλογα με τα δεδομένα. Χωρίς τέτοιες δομές ελέγχουν μόνο πολύ απλά και όχι πολύ χρήσιμα προγράμματα μπορεί να γράψει κανείς, σε οποιαδήποτε γλώσσα προγραμματισμού και αν δουλεύει.

Η βασική δομή ελέγχουν είναι το `if` με τις παραλλαγές του:

```
if (συνθήκη):
    εντολή 1
    εντολή 2
    ...
    εντολή N
```

Ο παραπάνω κώδικάς (εντολές 1 έως N) θα εκτελεστεί μόνο αν ισχύει η συνθήκη που εμφανίζεται δίπλα στο `if` ισχύει όταν η ροή του προγράμματος φτάσει εκεί. Προσέξτε ότι οι εντολές που υπάγονται στο `if` είναι γραμμένες πιο δεξιά από το ίδιο το `if` και είναι μεταξύ τους στοιχισμένες. Αυτό είναι υποχρεωτικό στην python και δεν το κάνουμε για αισθητικούς λόγους (όπως γίνεται σε πολλές άλλες γλώσσες, ανάλογα πάντα με τα γούστα του προγραμματιστή). Είναι η τρόπος που έχει η γλώσσα python να δηλώνει ότι κάποιες εντολές υπάγονται σε μια άλλη.

Για να δούμε καλύτερα πώς δουλεύει το `if` ας λύσουμε το εξής πρόβλημα. Έχουμε δύο διαστήματα [a,b] και [c,d] στην πραγματική ευθεία (με b < c) και θέλουμε δοθέντος ενός αριθμού x στη μεταβλητή `result` να μπαίνει η τιμή 0, 1 ή 2 ανάλογα με το αν το x αντίστοιχα δεν ανήκει σε κανένα από τα δύο διαστήματα, ανήκει στο αριστερό διάστημα [a,b] ή ανήκει στο δεξί διάστημα [c,d].

Για να το πετύχουμε αυτό χρησιμοποιούμε το `if`. Προσέξτε ότι σε αυτό το πρόγραμμα ακριβώς ένα από τα `if` θα δουλέψει (ισχύει δηλ. η συνθήκη του και άρα θα εκτελεστούν οι εντολές του).

In [1]:

```
a = 1 # Εδώ δίνουμε τιμές στα άκρα των διαστημάτων
b = 2.1
c = 3.5
d = 8

x = 2.5 # Εδώ δίνουμε τιμή στο x

if x < a: # Αν το x είναι αριστερά του a τότε result γίνεται 0
    result = 0

if x > d: # Ομοίως αν το x είναι δεξιά του d
    result = 0

if b < x and x < c: # Ομοίως αν το x είναι ανάμεσα στο b και στο c
    result = 0

if a <= x and x <=b: # Αν το x είναι ανάμεσα στο a και στο b τότε result γίνεται
 1
    result = 1

if c <= x and x <= d: # Αν το x είναι ανάμεσα στο c και στο d τότε result γίνετα
ι 2
    result = 2

print("Για x={} το αποτέλεσμα είναι {}".format(x, result)) # Τυπώνουμε το αποτέλ
εσμα
```

```
Για x=2.5 το αποτέλεσμα είναι 0
```

Εδώ βλέπουμε τη δομή `if` ... `else`.

```
if (συνθήκη):
    εντολή 1
    ...
    εντολή N
else:
    εντολή Α
    ...
    εντολή Ω
```

Αν η συνθήκη του `if` ισχύει τότε εκτελούνται οι εντολές που υπάγονται στο `if`. Αν όχι τότε εκτελούνται οι εντολές που υπάγονται στο `else`.

Έτσι ξαναγράφουμε το προηγούμενο πρόγραμμα ώστε να είναι πιο σύντομο.

In [2]:

```
a = 1
b = 2.1
c = 3.5
d = 8

x = 2.5 # Μέχρι εδώ, όπως πριν

if x < a or x > d or (b < x and x < c): # εδώ ελέγχουμε την ακριβή συνθήκη που π
ρέπει να ισχύει για result=0
    result = 0
else: # στο else μέσα μπαίνει το πρόγραμμα μόνο αν το x δεν είναι εκτός των δύο
 διαστημάτων
    if a <= x and x <=b: # αν το x είναι στο αριστερό διάστημα τότε result γίνετ
αι 1
        result = 1
    else: # αλλιώς το result γίνεται 2
        result = 2

print("Για x={} το αποτέλεσμα είναι {}".format(x, result)) # τυπώνουμε όπως πριν
```

```
Για x=2.5 το αποτέλεσμα είναι 0
```

Στο επόμενο βλέπουμε το ίδιο πρόγραμμα αλλά τώρα αντί να δίνουμε τιμή στο x μέσα στο πρόγραμμα ζητάμε από το χρήστη να μας δώσει το x με την εντολή

```
x = float(input("Δώστε ένα πραγματικό αριθμό x: "))
```

Η συνάρτηση `input(s)` τυπώνει το string `s` και μας επιστρέφει το string το οποίο έδωσε ο χρήστης στο πληκτρολόγιο. Επειδή στην περίπτωση αυτή ζητάμε από το χρήστη ένα πραγματικό αριθμό (floating point number, και ο αντίστοιχος τύπος δεδομένω στην python είναι ο `float`) το αποτέλεσμα της `input` το δίνουμε στη συνάρτηση `float(w)` η οποία μετατρέπει το string `w` σε πραγματικό αριθμό (αν αυτό γίνεται, αλλιώς προκύπτει σφάλμα).

Θα μπορούσαμε να είχαμε το ίδιο αποτέλεσμα αν αντί για την παραπάνω εντολή δίναμε

```
s = input("Δώστε ένα πραγματικό αριθμό x: ")
x = float(s)
```

αλλά με το να δώσουμε κατ' ευθείαν το output της `input` στη συνάρτηση `float` κάνουμε το πρόγραμμά μας και συντομότερο και πιο κατανοητό.

Αν θέλαμε να διαβάσουμε από το χρήστη ένα άλλο τύπο δεδομένων τότε θα δίναμε άλλη συνάρτηση μετατροπής. Αν θέλαμε π.χ. να διαβάσουμε ένα ακέραιο τότε θα δίναμε `int(input("μπλα μπλα"))` ενώ αν θέλαμε να διαβάσουμε απλά ένα string τότε δε θα κάναμε καμιά μετατροπή και θα δίναμε μόνο τη συνάρτηση `input`.

In [12]:

```
a = 1
b = 2.1
c = 3.5
d = 8

x = float(input("Δώστε ένα πραγματικό αριθμό x: "))

if x < a or x > d or (b < x and x < c):
    result = 0
else:
    if a <= x and x <=b:
        result = 1
    else:
        result = 2

print("Για x={} το αποτέλεσμα είναι {}".format(x, result))
```

```
Δώστε ένα πραγματικό αριθμό x: 3.4
Για x=3.4 το αποτέλεσμα είναι 0
```

Αξιοσημείωτη είναι επίσης και η συνάρτηση `eval(...)` την οποία θα μπορούσαμε να είχαμε χρησιμοποιήσει στη θέση της `float` αλλά και της `int` στην περίπτωση που διαβάζαμε ακέραιο. Η `eval` παίρνει ένα όρισμα τύπου string και το υπολογίζει σαν να είναι μια έκφραση python.

Δείτε τα παρακάτω παραδείγματα.

In [3]:

```
x = eval(input("Δώστε ένα αριθμό: "))
print(x)
```

```
Δώστε ένα αριθμό: 1.1
1.1
```

In [4]:

```
x = eval(input("Δώστε ένα αριθμό: "))
print(x)
```

```
Δώστε ένα αριθμό: -3
-3
```

In [5]:

```
x = eval(input("Δώστε ένα αριθμό: "))
print(x)
```

```
Δώστε ένα αριθμό: 1+2.4
3.4
```

In [6]:

```
x = eval(input("Δώστε ένα αριθμό: "))
print(x)
```

```
Δώστε ένα αριθμό: 3/4
0.75
```

In [7]:

```
x = eval(input("Δώστε ένα αριθμό: "))
print(x)
```

```
Δώστε ένα αριθμό: 5%3
2
```

In [8]:

```
import math
x = eval(input("Δώστε ένα αριθμό: "))
print(x)
```

```
Δώστε ένα αριθμό: math.pi
3.141592653589793
```

In [9]:

```
y = 3
x = eval(input("Δώστε ένα αριθμό: "))
print(x)
```

```
Δώστε ένα αριθμό: y+4.2
7.2
```

Παρακάτω χρησιμοποιούμε το `if` ... `else` και γράφουμε ένα πρόγραμμα για την επίλυση της εξίσωσης

$$ax^2 + bx + c = 0.$$

Το πρώτο πράγμα που κάνουμε είναι να υπολογίσουμε τη διακρίνουσα $D = b^2 - 4ac$ και να διακρίνουμε περιπτώσεις ανάλογα με το αν η διακρίνουσα είναι αρνητική, θετική ή 0.

In [10]:

```
# Πρόγραμμα για επίλυση του τριωνύμου ax^2+bx+c=0
import math

print("Πρόγραμμα για επίλυση του τριωνύμου ax^2+bx+c=0")
print("-----------------------------------------------")
a = float(input("Δώστε το συντελεστή a: ")) # Διαβάζουμε τους συντελεστές του τριωνύμου
b = float(input("Δώστε το συντελεστή b: "))
c = float(input("Δώστε το συντελεστή c: "))

D = b*b-4*a*c # Διακρίνουσα

if D<0:
    print("Αδύνατη εξίσωση.")
else:
    if D>0: # Αν η διακρίνουσα είναι θετική τότε
        sqD = math.sqrt(D) # υπολογίζουμε την τετραγωνική της ρίζα στη μεταβλητή sqD
        x1 = (-b-sqD)/(2*a) # και τις δύο ρίζες. x1 είναι η μικρότερη από τις δύο
        x2 = (-b+sqD)/(2*a)
        print("Οι ρίζες είναι η {} και η {}".format(x1, x2)) # τυπώνουμε το αποτέλεσμα
    else: # αν φτάσουμε εδώ τότε η διακρίνουσα είναι 0, άρα το τριώνυμο έχει μια "διπλή" ρίζα
        x = -b/(2*a)
        print("Διπλή ρίζα {}".format(x))
```

```
Πρόγραμμα για επίλυση του τριωνύμου ax^2+bx+c=0
-----------------------------------------------
Δώστε το συντελεστή a: 1
Δώστε το συντελεστή b: -5
Δώστε το συντελεστή c: 6
Οι ρίζες είναι η 2.0 και η 3.0
```

Τώρα βλέπουμε την παραλλαγή του `if .. else` με το `elif` που είναι κατά κάποιο τρόπο συντομογραφία του `else if`.

```
if συνθήκη 1:
    εντολές
elif συνθήκη 2:
    εντολές'
elif συνθήκη 3:
    εντολές''
....
else:
    εντολές '''
```

Αν ισχύσει η συνθήκη 1 τότε εκτελούνται οι εντολές που υπάγονται στο `if`, αλλιώς αν ισχύει η συνθήκη 2 εκτελούνται οι εντολές που υπάγονται στο πρώτο `elif` (εντολές'), αλλιώς (αν δηλ. δεν ισχύει ούτε η συνθήκη 1 ούτε η συνθήκη 2) αν ισχύει η συνθήκη 3 τότε εκτελούνται οι εντολές που υπάγονται στο δεύτερο `elif` (εντολές''), κλπ. Αν δεν ισχύσει καμία συνθήκη έως να φτάσουμε στο τελικό `else` (το οποίο μπορεί και να μην υπάρχει) τότε εκτελούνται οι εντολές που υπάγονται στο `else` (εντολές''').

Το `elif` είναι πολύ χρήσιμο αν θέλουμε να ελέγξουμε σε ποια από πολλές αλληλοαποκλειόμενες συνθήκης βρισκόμαστε.

Παρακάτω ξαναγράφουμε το πρώτο πρόγραμμα της ημέρας όπου αποφασίζουμε αν το `result` θα γίνει 0, 1 ή 2 ανάλογα με το πού είναι το x.

In [18]:

```
a = float(input("Δώσε το a: "))
b = float(input("Δώσε το b: "))
c = float(input("Δώσε το c: "))
d = float(input("Δώσε το d: "))

x = float(input("Δώσε και το x: "))

# Έλεγχος του τι έδωσε ο χρήστης. Αν ο χρήστης δεν τήρησε την αύξουσα διάταξη γι
α τα a, b, c, d τυπώνουμε
# μήνυμα σφάλματος και δεν κάνουμε τίποτε άλλο. Προσέξτε ότι η συνθήκη στο `if`
 είναι η άρνηση αυτού που ΘΕΛΟΥΜΕ
# να συμβαίνει. Αν δηλ. δε συμβαίνει το επιθυμητό τότε έχουμε σφάλμα εισόδου.
if not(a<b and b<c and c<d):
    print("Έκανες σφάλμα. Γεια.")
else: # από δω και κάτω ελέγχουμε πού είναι το x για να υπολογίσουμε το result.
      # ξεκιναμε από αριστερά και προχωράμε προς τα δεξιά μέχρι να βρούμε που πε
ριέχεται το x
    if x<a:
        result = 0
    elif x<=b:
        result = 1
    elif x<c:
        result = 0
    elif x<=d:
        result = 2
    else:
        result = 0
print("Για x={} το αποτέλεσμα είναι {}.".format(x, result))
```

```
Δώσε το a: 1
Δώσε το b: 2.1
Δώσε το c: 3.5
Δώσε το d: 8
Δώσε και το x: 1.5
Για x=1.5 το αποτέλεσμα είναι 1.
```

Για το παραπάνω πρόγραμμα, αν θέλαμε να γράψουμε το `if` ... `elif` ... με `if` ... `else` μόνο τότε θα το γράφαμε ως εξής:

```
if x<a:
    result = 0
else:
    if x<=b:
        result = 1
    else:
        if x<c:
            result = 0
        else:
            if x<=d:
                result = 2
            else:
                result = 0
```

Το κομμάτι αυτό κώδικα είναι λειτουργικά το ίδιο με το κομμάτι που αντικατέστησε, αλλά είναι και πιο δύσκολο να το γράψει κανείς σωστά και πολύ πιο δύσκολο να το τροποποιήσει, λόγω του "φωλιάσματος" (nesting) του ενός `if`...`else` μέσα στο προηγούμενο.

Το επόμενο πρόβλημα που λύνουμε είναι το εξής. Μας δίνεται ένα ορθογώνιο παραλληλόγραμμο στο επίπεδο

$$R = [a,b] \times [c,d] = \{(x,y) : a \leq x \leq b,\ c \leq y \leq d\}.$$

Αυτό είναι το ορθογώνιο που είναι παράλληλο με τους άξονες και προβάλλεται στο διάστημα $[a,b]$ στον x-άξονα και στο διάστημα $[c,d]$ στον y-άξονα. Για να ανήκει ένα σημείο

$$P = (p_x, p_y)$$

στο R πρέπει να ισχύει $a \leq p_x \leq b$ και $c \leq p_y \leq d$.

Στο πρόγραμμα ο χρήστης δίνει τα a, b, c, d, p_x, p_y και το πρόγραμμα μας λέει αν το σημείο (p_x, p_y) ανήκει στο ορθογώνιο R ή όχι.

In [20]:

```
a = float(input("Δώσε το a: "))
b = float(input("Δώσε το b: "))
c = float(input("Δώσε το c: "))
d = float(input("Δώσε το d: "))

px = float(input("Δώσε το px: "))
py = float(input("Δώσε και το py: ")) # Εδώ τελειώνει η εισαγωγή δεδομένων από το χρήστη

if not( a<b and c<d): # Ελέγχουμε αν υπάρχει σφάλμα στα δεδομένα. Αυτό που ΠΡΕΠΕΙ να ισχύει είναι αυτό
                      # είναι μέσα στο not. Αν υπάρχει σφάλμα τυπώνουμε μήνυμα και δεν κάνουμε τίποτε άλλο.
    print("Λάθος δεδομένα. Γεια.")
else: # από δω και κάτω ελέγχουμε αν το σημείο είναι μέσα στο ορθογώνιο
    is_in = (a <= px and px <= b) and (c <= py and py <= d) # Η λογική μεταβλητή is_in γίνεται True αν ισχύει
        # η σωστή συνθήκη, αλλιώς είναι False.
    if is_in:
        print("Το σημείο ({},{}) είναι μέσα.".format(px, py))
    else: # Αν το σημείο δεν είναι μέσα
        print("Το σημείο ({},{}) δεν είναι μέσα.".format(px, py))
```

```
Δώσε το a: 1
Δώσε το b: 2
Δώσε το c: 3
Δώσε το d: 5
Δώσε το px: 0.5
Δώσε και το py: 4
Το σημείο (0.5,4.0) δεν είναι μέσα.
```

Στο επόμενο βλέπουμε μια παραλλαγή του προηγούμενου προγράμματος όπου χρησιμοποιούμε μόνο ένα `print` και τροποποιούμε το μήνυμα ανάλογα με το αν το σημείο είναι μέσα ή όχι.

In [22]:

```
a = float(input("Δώσε το a: "))
b = float(input("Δώσε το b: "))
c = float(input("Δώσε το c: "))
d = float(input("Δώσε το d: "))

px = float(input("Δώσε το px: "))
py = float(input("Δώσε και το py: "))

if not( a<b and c<d):
    print("Λάθος δεδομένα. Γεια.")
else:
    is_in = (a <= px and px <= b) and (c <= py and py <= d)
    word='' # αυτό θα παραμείνει κενό αν το σημείο είναι μέσα
    if not is_in:
        word = 'δεν ' # αλλιώς θα γίνει το string `δεν `
    print("Το σημείο ({},{}) {} είναι μέσα.".format(px, py, word))
```

```
Δώσε το a: 1
Δώσε το b: 2
Δώσε το c: 3
Δώσε το d: 5
Δώσε το px: 0.5
Δώσε και το py: 4
Το σημείο (0.5,4.0) δεν  είναι μέσα.
```

Στο τελευταίο κομμάτι του μαθήματος κάναμε μια μικρή εισαγωγή στις λίστες, που είναι ένας πολύ βασικός τύπος δεδομένων στην Python. Μοιάζουν πολύ με τα strings μόνο που αντί για γράμματα μπορούν στην κάθε θέση να έχουν οτιδήποτε, ακόμη και άλλες λίστες. Τα στοιχεία τους μπαίνουν μέσα σε αγκύλες διαχωρισμένα από κόμματα.