Σημειώσεις

Προηγούμενο Επόμενο

Λογικές εκφράσεις και μεταβλητές

Μπορούμε να γράψουμε μία λογική έκφραση η οποία θα είναι είτε αληθής είτε ψευδής. Μία συνηθισμένη λογική έκφραση είναι αυτή που ελέγχει αν μία ισότητα ή μία ανισότητα ισχύει ή όχι.

Παραδείγματα. (Λογικές εκφράσεις.)

```
>>> x = 5     # assign a value to the variable x
>>> y = 7     # assign a value to the variable y
>>> x == y    # check if x is equal to y
False
>>> x < y     # check if x is smaller than y, etc
True
>>> x <= y
True
>>> x > y
False
>>> x >= y
False
>>> x != y    # check if x is not equal to y
True
```

Παράδειγμα. (Λογικές μεταβλητές.) Η μεταβλητή `cond` είναι μία λογική μεταβλητή (`bool`) και περιέχει το αποτέλεσμα μίας λογικής έκφρασης. Η `cond` μπορεί να πάρει τις τιμές `True` ή `False`.

```
>>> cond = (x == y)
>>> cond
False
>>> type(cond)
<class 'bool'>
```

Παράδειγμα. Είναι το φετινό έτος δίσεκτο;

```
>>> year = 2017
>>> isleap = year%4 == 0
>>> print(isleap)
```

Σημείωση: η παραπάνω συνθήκη χρειάζεται βελτίωση για να είναι πλήρως σωστή (θα το δούμε αργότερα).

Εντολές ελέγχου

Εκτέλεση υπό συνθήκη (εντολή `if`)

Η εντολή **`if`** ακολουθείται από μία λογική έκφραση και το σύμβολο **`:`** (άνω-κάτω τελεία). Εάν η λογική έκφραση είναι `True` τότε εκτελούνται όλες οι εντολές οι οποίες ακολουθούν την `if` και βρίσκονται σε μία εσοχή ως προς την `if`.

Παράδειγμα.

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

Παρατήρηση: Προσέξτε τη σημασία της εσοχής στην `if` (και γενικότερα στην python). Όλες οι εντολές που ακολουθούν την `if` και εξαρτώνται από αυτήν ξεκινούν από την ίδια ακριβώς στήλη (είναι ευθυγραμμισμένες στα αριστερά).

`if - else`: Εάν η λογική έκφραση η οποία ακολουθεί την `if` έχει τιμή `False` τότε δεν εκτελούνται οι εντολές που ακολουθούν (εντός της εσοχής) την `if`. Έχουμε όμως τη δυνατότητα να εκτελέσουμε κάποιες εντολές, εκείνες που ακολουθούν (εντός εσοχής) την νέα εντολή `else`.

Παράδειγμα.

```
>>> x = 5
```

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

Για τις εντολές `if - else` η γενική μορφή της σύνταξης είναι

```
if expression:
    statements_true
else:
    statements_false
```

Εδώ, *expression* δηλώνει οποιαδήποτε έκφραση η οποία η τιμή της οποίας είναι `True` ή `False`, ενώ *statements_true* και *statements_false* δηλώνουν μία ή περισσότερες εντολές Python. Παρατηρήστε την άνω και κάτω τελεία που ακολουθεί τόσο την έκφραση *expression* όσο και τη λέξη else. Προσέξτε και πάλι ότι οι εντολές *statements_true* και *statements_fals*e έχουν μετακινηθεί (κατά μια εσοχή) προς τα δεξιά.

`if - elif - else`: Μπορεί να συμβαίνει να θέλουμε να ελέγξουμε διαφορετικές περιπτώσεις, δηλαδή, το αποτέλεσμα διαφορετικών λογικών εκφράσεων. Είναι δυνατόν να ισχύουν ταυτοχρόνως περισσότερες από μία λογικές εκφράσεις (δηλαδή να έχουν την τιμή `True`), να ισχύει μόνο η μία ή να μην ισχύει καμμία.

Παράδειγμα.

```
>>> x = 5
>>> y = 7
>>> if x>y:
        print(x,'is greater than',y)
    elif x<y:
        print(x,'is less than',y)
    else:
        print(x,'and',y,'are equal')
```

Παράδειγμα. Είναι το φετινό έτος δίσεκτο;

```
>>> year = 2017
>>> isleap = year%4 == 0
>>> if isleap:
        print("Year",year,"is leap year")
    else:
        print("Year",year,"is not leap year")
```

Η συνθήκη που χρησιμοποιήσαμε στο παραπάνω παράδειγμα χρειάζεται βελτίωση:

```
>>> year = 2017
>>> isleap = (year%4 == 0) and (year%100 != 0)
>>> print(isleap)
```

Έχουμε χρησιμοποιείσει δύο λογικές εκφράσεις και τις έχουμε ενώσει με την λέξη `and`. Η λογική έκφραση που προκύπτει είναι αληθής μόνο στην περίπτωση που και οι δύο επιμέρους λογικές εκφράσεις είναι αληθείς.

Η συνθήκη που χρησιμοποιήσαμε στο παραπάνω παράδειγμα χρειάζεται επιπλέον βελτίωση:

```
>>> year = 2017
>>> isleap = (year%4 == 0) and (year%100 != 0) or (year%400 == 0)
>>> print(isleap)
```

Τελεστές για λογικές μεταβλητές *b, c* (κατά σειρά χαμηλότερης προτεραιότητας)

- *b* `or` c (είναι αληθής εάν η *a* ή η *b* είναι αληθής)
- *b* `and` c (είναι αληθής εάν η *a* και η *b* είναι αληθής)
- `not` b (είναι αληθής εάν η *a* είναι ψευδής)

Παράδειγμα. Η παραπάνω εντολή `if` για τα δίσεκτα έτη είναι ισοδύναμη με την

```
>>> isleap = (year%4 == 0 and year%100 != 0) or (year%400 == 0)
```

Άσκηση. Υπάρχει κάποιο λάθος στις παρακάτω εντολές; Τι νομίζετε ότι προσπαθούσε να γράψει ο προγραμματιστής;

```
a == 1
if a = 1:
   print('a is one')
```

Φωλιασμένες εντολές if

Εντολές ελέγχου `if-elif-else` μπορούν να είναι μέρος εντολών οι οποίες εκτελούνται υπό μία συνθήκη.

Παράδειγμα. Οι παρακάτω εντολές επιλέγουν τον μέγιστο μεταξύ των τριών αριθμών *x, y, z* :

```
if x >= y:
    if x >= z:
        print('x is largest')
    else:
        print('z is largest')
else:
    if y >= z:
        print('y is largest')
    else:
        print('z is largest')
```

Μια κομψότερη, ίσως, λύση (η οποία μπορεί εύκολα να γενικευθεί σε μεγαλύτερο πλήθος αριθμών) είναι η ακόλουθη:

```
maxnum = x
if y > maxnum:
    maxnum = y
if z > maxnum:
    maxnum = z
print('largest number is', maxnum)
```

Παράδειγμα. Εισάγετε τρεις πραγματικούς αριθμούς *x, y, z* και τυπώστε τους κατά αύξουσα σειρά.

```
x = float(input("Give a number: "))
y = float(input("Give a number: "))
z = float(input("Give a number: "))

if y > z:
   a = z; z = y; y = a
if x > z:
   a = z; z = x; x = a
if x > y:
   a = y; y = x; x = a

print("{} {} {}".format(x,y,z))
```

Παρατηρήστε ότι μπορούμε να γράψουμε σε μία γραμμή περισσότερες από μία εντολές, τις οποίες χωρίζουμε με το semicolon (;).

Παρατηρήστε ότι για να εναλλάξουμε τις τιμές δύο μεταβλητών χρησιμοποιήσαμε μία βοηθητική μεταβλητή (*a*).

Παράδειγμα. Θα βρούμε τις ρίζες του τριωνύμου $ax^2 + bx + c = 0$

```
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("Δεν υπάρχει λύση.")
else:
    if D>0:                                    # Αν η διακρίνουσα είναι
θετική
        sqD = math.sqrt(D)
        x1 = (-b-sqD)/(2*a)                    # έχουμε δύο ρίζες: x1,x2
        x2 = (-b+sqD)/(2*a)
        print("Οι ρίζες είναι η {} και η {}".format(x1, x2)) # τυπώνουμε
```

```
το αποτέλεσμα
    else:                                   # Η διακρίνουσα είναι 0, άρα
το τριώνυμο έχει μια διπλή ρίζα
        x = -b/(2*a)
        print("Διπλή ρίζα {}".format(x))
```

Αλγόριθμοι

Παράδειγμα. (Εύρεση κυβικής ρίζας τέλειου κύβου.) Η κυβική ρίζα κάποιων αριθμών είναι ακέραιος. Μπορούμε να βρούμε την κυβική ρίζα του $a = 216$ (και άλλων τελείων κύβων) χρησιμοποιώντας το παρακάτω πρόγραμμα. Θα πρέπει να το τρέξουμε πολλές φορές δοκιμάζοντας διαδοχικούς ακεραίους. Η μέθοδος ονομάζεται *εξαντλητική απαρίθμηση*.

```
perfect_cube = 216   # this is a perfect cube

print("Try to guess the cubic root of",perfect_cube)
x = input("Give an integer: ")
x = int(x)

if x**3 == perfect_cube:
    print("Yes, indeed!", x,"is the cubic root of",perfect_cube)
elif x**3 < perfect_cube:
     print("You need a larger number")
elif x**3 > perfect_cube:
     print("You need a smaller number")
```

Παράδειγμα. (Εύρεση ρίζας θετικού αριθμού.) Ο Ήρων ο Αλεξανδρεύς πρότεινε την εύρεση ρίζας του x ως εξής

- Θεωρούμε μία εκτίμηση της ρίζας, έστω, g.
- Αν το $g \cdot g$ είναι αρκετά κοντά στο x, τότε έχουμε τη ρίζα.
- Αλλιώς, κατασκευάζουμε το μέσο όρο των g και x/g, δηλ., $\frac{1}{2}(g + x/g)$ και αυτό είναι μία καλύτερη προσέγγιση της ρίζας.
- Ονομάζουμε το αποτέλεσμα πάλι g και επαναλαμβάνουμε τη διαδικασία έως ότου το $g \cdot g$ είναι κοντά στο x.

Ο αλγόριθμος του Ήρωνα υλοποιείται με τον κώδικα:

```
x = 2.0
prompt = "Give a number close to the root of "+str(x)+": "
g = input(prompt)
g = float(g)

print("The square of",g,"is:",g*g)
g = (g+x/g)/2.0
print("g =",g,"is a better approximation: g*g =",g*g)
```

Παράδειγμα. (Εύρεση ρίζας εξίσωσης με διχοτόμηση διαστήματος.) Θεωρούμε ένα κυβικό πολυώνυμο $a_0 + a_1x + a_2x^2 + a_3x^3$. Θα πρέπει αρχικά να γνωρίζουμε έαν διάστημα εντοπισμού της ρίζας. Ακολούθως, να τρέξουμε το πρόγραμμα πολλές φορές ελέγχοντας στο κέντρο του διαστήματος και διχοτομόντας το διάστημα. Η μέθοδος ονομάζεται *μέθοδος διχοτόμησης*.

```
# coefficients of cubic polynomial
a0 = 1.0; a1 = 2.0; a2 = 0.0; a3 = 3.0

# input x and find f(x), at two points x=x1, x=x2
x1 = float(input("Give x1: "))
fx1 = a0 + a1*x1 + a2*x1**2 + a3*x1**3

x2 = float(input("Give x2: "))
fx2 = a0 + a1*x2 + a2*x2**2 + a3*x2**3

print("f(x1) =",fx1,"\nf(x2) =",fx2)

# Check if a root of the polynomial is in the interval [x1,x2].
# Try to go closer to the root.
if fx1*fx2 <= 0:
    print("A root is between",x1,"and",x2)
    x = (x1+x2)/2.0
    fx = a0 + a1*x + a2*x**2 + a3*x**3
    print("It may be close to",x,"where f =",fx)
else:
    print("I cannot locate a root of the polynomial")
```

Άσκηση. Εφαρμόστε την παραπάνω μέθοδο (διχοτόμησης) για τη συνάρτηση $a\ \sin(x) + b\ \cos(x)$ (όπου a, b είναι σταθερές).

Πληροφορία.

- Μπορεί να μας δωθεί υπό τη μορφή μίας πρότασης. Π.χ., η τετραγωνική ρίζα αριθμού x είναι ένας y τέτοιος ώστε y*y=x.
- Μπορεί να προκύπτει στο τέλος μίας διαδικασίας. Δηλαδή, μπορεί να μας δωθεί μία ακολουθία οδηγιών στο τέλος των οποίων προκύπτει η πληροφορία. Ένα παράδειγμα είναι η μέθοδος εύρεσης τετραγωνικής ρίζας αριθμού του Ήρωνα.

Ένας **αλγόριθμος** αποτελείται από βήματα υπολογισμών τα οποία εφαρμόζονται σε κάποια αρχικά δεδομένα και, αφού εξελιχθούν μέσω διαφόρων καταστάσεων, δίνουν κάποια τελικά δεδομένα (δηλαδή, ένα αποτέλεσμα).

Για να εφαρμόσουμε έναν αλγόριθμο χρειαζόμαστε μία **γλώσσα προγραμματισμού**. Μία γλώσσα προγραμματισμού δίνει εντολές στον υπολογιστή ώστε να πραγματοποιηθούν οι πράξεις, οι οποίες αποτελούν τα βήματα οποιουδήποτε αλγορίθμου.

Μελέτη

Βιβλιογραφία

1. Δ. Καρολίδης, *Μαθαίνετε εύκολα python*.
2. Κ. Μαγκούτης, Χ. Νικολάου, *Εισαγωγή στον αντικειμενοστραφή προγραμματισμό με Python*, (Αποθετήριο "Κάλλιπος", 2016) - Κεφάλαιο 4. Συναρτήσεις και εκτέλεση υπό συνθήκη.
3. J.V. Guttag, *Υπολογισμοί και προγραμματισμός με την python* (Παράγραφοι 2.2, 3.1).

Προηγούμενο Επόμενο