Εισαγωγή στην Python

Η εκτύπωση μηνυμάτων γίνεται με απλό τρόπο χρησιμοποιώντας την εντολή `print`

In [ ]:
```
print("Hello World!")
```

Σε αντίθεση με άλλες γλώσσες δεν ορίζουμε ποιές μεταβλητές θα χρησιμοποιήσουμε ούτε τον τύπο των μεταβλητών. Εκτός από τις ακολουθίες χαρακτήρων `string` υπάρχουν διάφοροι αριθμητικοί τύποι.

- ακέραιοι (integer)
- κινητής υποδιαστολής (float)
- μιγαδικοί (complex)

In [ ]:
```
x=7 #ανάθεση ακεραίου
print(x)
x=7.0 #ανάθεση float
print(x)
x=float(7) #ανάθεση float
print(x)
```

Η ανάθεση `string` γίνεται με τον ακόλουθο τρόπο

In [ ]:
```
x="Hello" # Τα strings δηλώνονται με "διπλές" αποστρόφους
print(x)
x='Hello' # Τα strings δηλώνονται και με "μονές" αποστρόφους
print(x)
```

Άσκηση 1:

Φτιάξτε πρόγραμμα python όπου θα δημιουργείται ένα `string` , έναν `integer` και έναν `float` . Το `string` θα καλείται `mystring` και η τιμή του θα είναι `"Hello"` , ο `integer` θα καλείται `myint` και η τιμή του θα είναι 10 και ο `float` θα καλείται `myfloat` και η τιμή του θα είναι 20.5

Λίστες

Οι λίστες (`list`) είναι ακολουθίες που περιέχουν αριθμούς, `strings` , ή άλλα αντικείμενα ακόμα και άλλες λίστες. Δηλώνονται με τετράγωνη αγκύλη [ ].

In [ ]:
```
L=[] #κενή λίστα
L.append(1) # 0 αριθμός 1 τοποθετείτε στη L
L.append(2) # 0 αριθμός 2 τοποθετείτε στη L (στο τέλος)
L.append(3) # 0 αριθμός 3 τοποθετείτε στη L (στο τέλος)
print(L[0]) #τυπώνει το πρώτο στοιχείο (το 1)
```

Άσκηση 2:

Φτιάξτε πρόγραμμα python όπου θα δημιουργείτε μια κενή λίστα `L` . Στη συνέχεια θα προσθέστε τους αριθμούς 10, 20, 30 χρησιμοποιώντας τη εντολή `append` .

Φορμαρισμένη εκτύπωση

Μπορούμε να υποδείξουμε στην python τον τρόπο που θέλουμε να τυπωθεί μια μεταβλητή.

- %s # φορμάρισμα για `string`
- %d # φορμάρισμα για `int`
- %f # φορμάρισμα για `float`
- %e # φορμάρισμα για `float` (επιστημονική μορφή)

Αν θέλουμε να εκτυπώσουμε το `string` `"John"` μετά τη φράση `"Hello, my name is"`

In [ ]:
```
x="John"
print("Hello, my name is %s"%x)
```

Σε μια φορμαρισμένη εκτύπωση μπορούμε να χρησιμοποιήσουμε 2 ή περισσότερες μεταβλητές.

In [ ]:
```
x="John"
a=23
print("Hello, my name is %s. I am %d years old"%(x,a))
```

Έναν `float` μπορώ να τον τυπώσω με διαφορετικό πλήθος δεκαδικών

In [ ]:
```
x=1234.56789
print("The number x is %f"%x) #Τον τυπώνει ως έχει με 6 δεκαδικά
print("The number x is %.3f"%x) #Τον τυπώνει με 3 δεκαδικά ψηφία
```

Λογικές Συνθήκες

Λογικές μεταβλητές (`True` – `False`)

In [ ]:
```
x=2 # ανάθεση τιμής
print(x==2) #Έλεγχος αν το x είναι ίσο με 2. Τυπώνει True
print(x==3) #Τυπώνει False
print(x!=2) #Έλεγχος αν το x δεν είναι ίσο με 2. Τυπώνει False
print(x<3) #Τυπώνει True
```

Λογικός έλεγχος (if ... (elif) ... else)

In [ ]:
```
x=2
if x==2: # έλεγχος αν x είναι ίσο με 2
  print ("x equals 2")
elif x<2: # έλεγχος αν x είναι μικρότερο από 2
  print ("x less than 2")
else: #έλεγχος αν x είναι μεγαλύτερο από 2
  print ("x greater than 2")
```

Λογικοί τελεστές (and, or, not)

In [ ]:
```
x=2
y=10 # ανάθεση τιμών
if x==2 and y==10: # έλεγχος αν δύο συνθήκες ισχύουν ταυτόχρονα
    z=x+y
if x!=2 or y==10: # έλεγχος αν δύο συνθήκες ισχύει είτε η μια είτε η άλ
λη
    z=y-x
```

Άσκηση 3:

Φτάξτε έναν έλεγχο (εντολή if) που όταν ένας αριθμός x είναι άρτιος να τυπώνει το μήνυμα “x is even” και όταν περιττός το μήνυμα “x is even”

Λογικός τελεστής `in`

Ελέγχουμε αν ένα αντικείμενο περιέχεται σε μια λίστα.

In [ ]:
```
L=[1,2,3] # Δημιουργία μιας λίστας
2 in L # Έλεγχος αν το 2 ανήκει στην L (επιστρέφει True)
```

Άσκηση 4:
Φτάξτε έναν έλεγχο (εντολή `if`) που να ελέγχει αν ένας αριθμός `x` υπάρχει σε μια λίστα `L`. Αν υπάρχει τότε να τυπώνει το μήνυμα `"x is in L"` και όταν δεν υπάρχει το μήνυμα `"x is not in L"`

Επαναλήψεις (for)

In [ ]:
```
for x in [10,20,30]: #το x παίρνει διαδοχικά τις τιμές της λιστας
    print(x) # για κάθε υλοποίηση εκτελείται η εντολή
```

Η λίστα σύμφωνα με την οποία γίνεται η επανάληψη μπορεί να περιέχει τιμές διαφορετικές από αριθμούς

In [ ]:
```
for x in ["Apple",20,True]:
    print(x)
```

Με την εντολή `range` μπορούμε να δημιουργήσουμε μια λίστα από διαδοχικούς αριθμούς και να τους χρησιμοποιήσουμε ως δείκτες σε μια επανάληψη.

In [ ]:
```
L=range(4) # δημιουργεί τη λίστα [0,1,2,3]
L=range(2,5) # δημιουργεί τη λίστα [2,3,4]
```

Άσκηση 5:
Φτιάξτε μια επανάληψη τύπου `for` που να γεμίζει μια κενή λίστα με τους αριθμούς 10,20,30,40,50

Επαναλήψεις (while)

In [ ]:
```
s=1
while s<10: # λογική συνθήκη η οποία αν επαληθεύεται
            # εκτελείται το block εντολών που ακολουθεί
 print(s)
 s=s+1
```

Άσκηση 6:
Φτιάξτε μια επανάληψη τύπου while που να γεμίζει μια κενή λίστα με τους αριθμούς 10,20,30,40,50

Επαναλήψεις `break` και `continue`

In [ ]:
```
for s in range(10):
    print(s)
    if s>=5:
      break # σταματαει την εκτέλεση της επαναληψης
```

In [ ]:
```
for i in range(20):
    if i%5==0:
       continue # δεν εκτελείται η επαναληψη και ο δείκτης παει
                #στο επόμενο στοιχειο της λιστας των αριθμών
    print(i)
```

Συναρτήσεις

Ο ορισμός μιας συνάρτησης στη `python` είναι ο ακόλουθος.

```
def myfunction(): # Συνάρτηση (όχι απαραίτητο να επιστρέφει τιμή)
   εντολές
   εντολές
   .....
```

Αν θελήσουμε να ορίσουμε μια συνάστηση η οποία τυπώνει π.χ. τη φράση 'Hello World! My name is John', μπορεί να γίνει ως εξής:

In [ ]:
```
def myfunction():
    print('Hello World!')
    print('My name is John')
```

Για να εκτελέσουμε μια συνάρτηση απλώς την καλούμε.

In [ ]:
```
myfunction()
```

Για να επιστρέφει η συνάρτηση το αποτέλεσμα μιας διεργασίας χρειάζεται να το δηλώσουμε με τη εντολή `return` . Έτσι ο ορισμός μιας συνάρτησης τροποποιείται ως εξής:

```
def myfunction():
    εντολές
    εντολές
    .....
    return <αυτό που θέλω να επιστρέφει>
```

Η ακόλουθη συνάρτηση επιστρέφει την τιμή της μεταβλητής `s` , που είναι πάντα το 23.

In [ ]:
```
def myfunction(): # Συνάρτηση
  s=23
  return s

#κλήση της συνάρτησης και ανάθεση στη μεταβητή a
a=myfunction()
print(a)
```

Άσκηση 7:

Φτιάξτε μια συνάρτηση που να δέχεται ως όρισμα 2 αριθμούς x, y και να επιστρέφει το άθροισμα τους