Προγραμματισμός με τη γλώσσα python

[Alexandros Kanterakis (mailto:kantale@ics.forth.gr)](mailto:kantale@ics.forth.gr) **kantale@ics.forth.gr**

Διάλεξη 2η, Παρασκευή 22 Οκτωβρίου 2019

Τελεστές

Οι τελεστες είναι σύμβολα ή δεσμευμένες λέξεις με τους οποίους εφαρμόζουμε βασικές πράξεις σε διάφορες εκφράσεις. Για πειρσσότερα μπορείτε να διαβάσετε εδώ: https://en.wikipedia.org/wiki/Operator_(computer_programming (https://en.wikipedia.org/wiki/Operator_(computer_programming))

Μερικοί από τους πιο βασικους τελεστές που υποστηρίζει η python είναι:

- +
- -
- /
- //
- *
- %
- **
- <
- >
- <=
- >=
- !=
- ==
- and
- or
- not
- in
- is

Ο τελεστής +

Οι πράξεις που επιστρέπονται με τον τελεστή '+' είναι:

```
In [1]: 3+2
```

```
Out[1]: 5
```

```
In [3]: 3+2.0
```

```
Out[3]: 5.0
```

```
In [5]: 3+0.0
```

```
Out[5]: 3.0
```

In [6]:
```
'ab' + 'cde'
```
Out[6]: 'abcde'

In [7]:
```
True + True + False
```
Out[7]: 2

In [8]:
```
True + 2
```
Out[8]: 3

In [9]:
```
True + 0.0
```
Out[9]: 1.0

In [11]:
```
[1,2,3] + [4,5,6] # Λίστες θα το δούμε αργότερα
```
Out[11]: [1, 2, 3, 4, 5, 6]

Ο τελεστής -

Οι πράξεις που επιτρέπονται με τον τελεστή '-' είναι:

In [13]:
```
3-2
```
Out[13]: 1

In [14]:
```
3-7
```
Out[14]: -4

In [15]:
```
4-6.0
```
Out[15]: -2.0

In [16]:
```
True - True
```
Out[16]: 0

In [17]:
```
True - 6.6
```
Out[17]: -5.6

Ο τελεστής *

Οι πράξεις που επιτρέπονται με τον τελεστή '*' είναι:

In [18]:
```
6*7
```
Out[18]: 42

In [19]:
```
6.6*2
```
Out[19]: 13.2

In [20]:
```
True * 2
```

Out[20]: 2

In [23]:
```
True * False
```

Out[23]: 0

In [24]:
```
True * 2.3
```

Out[24]: 2.3

In [22]:
```
6 * 'hello'
```

Out[22]: 'hellohellohellohellohellohello'

In [26]:
```
[1,2,3] * 5
```

Out[26]: [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

Ο τελεστής '/'

Οι πράξεις που επιτρέπονται με τον τελεστή '/' είναι:

In [27]:
```
4/5
```

Out[27]: 0.8

ΠΡΟΣΟΧΗ!!

In [28]:
```
5/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-28-0106664d39e8> in <module>()
----> 1 5/0

ZeroDivisionError: division by zero
```

In [29]:
```
True/True
```

Out[29]: 1.0

In [30]:
```
6/3
```

Out[30]: 2.0

In [31]:
```
6.5/3
```

Out[31]: 2.1666666666666665

Ο τελεστής '//'

Αυτός ο τελεστής μας δίνει το αποτέλεσμα της ακέραιας διαίρεσης

In [33]:
```
5//2
```

Out[33]: 2

In [34]:
```
11//3
```

Out[34]: 3

In [39]:
```
6.5 // 2.1
```

Out[39]: 3.0

In [40]:
```
True // 2
```

Out[40]: 0

Ο τελεστής %

Αυτός ο τελεστής μας δίνει το υπόλοιπο της ακέραιας διαίρεσης

In [37]:
```
5%2
```

Out[37]: 1

In [38]:
```
5.2 % 2
```

Out[38]: 1.2000000000000002

In [41]:
```
True % 2
```

Out[41]: 1

Ο τελεστής % χρησιμοποιείται (όχι και τόσο συχνά) για να βάλουμε strings μέσα σε strings

In [44]:
```
name = 'mitsos'
pattern = 'my name is %s nice to meet you'
pattern % name
```

Out[44]: 'my name is mitsos nice to meet you'

Μπορείτε να βρείτε εδώ (https://python-reference.readthedocs.io/en/latest/docs/str/formatting.html) για το πως μπορείτε να χρησιμοποιείσετε αυτόν τον τελεστή για strings με περισσότερα παραδείγματα

Ο τελεστής **

Αυτός ο τελεστής μας επιστρέφει το εκθετικό a^b

In [46]:
```
3**2
```

Out[46]: 9

In [47]:
```
3.2**2.3
```

Out[47]: 14.515932837559118

```
In [50]: True ** 2
```

```
Out[50]: 1
```

Οι τελεστές <, >, <=, >=

Αυτοί οι τελεστές επιστρέφουν πάντα True ή False και συγκρίνουν 2 τιμές:

```
In [52]: 2<3
```

```
Out[52]: True
```

```
In [53]: 2<=2
```

```
Out[53]: True
```

```
In [54]: False < True
```

```
Out[54]: True
```

```
In [55]: 2<2
```

```
Out[55]: False
```

Μπορούμε επίσης να τους χρησιμοποιήσουμε παραπάνω από μία φορά:

```
In [57]: 2<3<4
```

```
Out[57]: True
```

```
In [58]: 2<3<3
```

```
Out[58]: False
```

Όταν εφαρμόζονται σε strings, τότε τα συγκρίνουμε αλφαριθμητικά (λεκτικά). Ποιο μικρό θεωρείται αυτό που σε μία ταξινόμηση, παίρνει τη μικρότερη θέση:

```
In [59]: 'ab' < 'fg'
```

```
Out[59]: True
```

```
In [60]: 'ab' < 'b'
```

```
Out[60]: True
```

```
In [61]: 'ab' < 'ac'
```

```
Out[61]: True
```

```
In [62]: 'ab' < 'a'
```

```
Out[62]: False
```

Το άδειο string έχει τη πιο μικρή δυνατή τιμή

In [65]:
```
'' < '0'
```
Out[65]: True

In [112]:
```
"A" < "a"
```
Out[112]: True

In [114]:
```
"05456745674" < "5"
```
Out[114]: True

In [115]:
```
'8' < '09'
```
Out[115]: False

Ο τελεστής !=

Αυτός ο τελεστής ελέγχει αν 2 εκφράσεις **ΔΕΝ** είναι ίδιες

In [66]:
```
2 != 2
```
Out[66]: False

In [67]:
```
2 != 2.0
```
Out[67]: False

In [68]:
```
2 != 'hello'
```
Out[68]: True

In [69]:
```
1 != True
```
Out[69]: False

In [70]:
```
'hello' != ' hello '
```
Out[70]: True

In [71]:
```
'' != ''
```
Out[71]: False

In [72]:
```
'' != ' '
```
Out[72]: True

Ο τελεστής ==

Ελέγχει αν δύο εκφράσεις είναι ίδιες

In [73]:
```
'mitsos' == 'mits' + 'os'
```
Out[73]: True

In [74]:
```
3 == 6/2
```
Out[74]: True

In [75]:
```
True == True or False
```
Out[75]: True

In [76]:
```
3 == True + True + True + False
```
Out[76]: True

In [77]:
```
3 == 'mits' + 'os'
```
Out[77]: False

In [79]:
```
None == None
```
Out[79]: True

In [80]:
```
None == False
```
Out[80]: False

In [81]:
```
[1,2,3] == [1,2,3]
```
Out[81]: True

In [83]:
```
[1,2,3] == [2,1,3]
```
Out[83]: False

Οι τελεστές and και or

τους τελεστές αυτούς τους είδαμε σε προηγούμενο μάθημα

Ο τελεστής not

Ο τελεστής αυτός εφαρμόζεται μόνο σε μία έκφραση. Επιστρέφει True ή False.

In [85]:
```
not True
```
Out[85]: False

In [86]:
```
not False
```
Out[86]: True

In [87]:
```
not ''
```
Out[87]: True

In [88]:
```
not 1
```
Out[88]: False

In [89]:
```
not 0
```
Out[89]: True

In [90]:
```
not []
```
Out[90]: True

In [91]:
```
not [1,2,3]
```
Out[91]: False

In [92]:
```
not 0.0000000001
```
Out[92]: False

In [93]:
```
not None
```
Out[93]: True

In [95]:
```
not 'hello'
```
Out[95]: False

In [96]:
```
not not True
```
Out[96]: True

In [98]:
```
not not not True
```
Out[98]: False

Ο τελεστής in

Αυτός ο τελεστής ελέγχει αν υπάρχει "κάτι" "κάπου"

In [99]:
```
'rak' in 'Heraklion'
```
Out[99]: True

In [100]:
```
'raki' in 'Heraklion'
```
Out[100]: False

In [101]:
```
'h' in 'Heraklion'
```
Out[101]: False

In [102]:
```
'H' in 'Heraklion'
```
Out[102]: True

In [103]:
```
1 in [1,2,3]
```
Out[103]: True

In [104]:
```
[1,2] in [1,2,3]
```
Out[104]: False

```
In [105]: [1,2] in [1, [1,2], 3]
Out[105]: True

In [106]: False in [1, True-True]
Out[106]: True

In [108]: None in [3, None, 4]
Out[108]: True

In [109]: 'ra' in ['Heraklion']
Out[109]: False

In [110]: [] in [1, [], 2]
Out[110]: True
```

Ο τελεστής is

Αυτόν τον τελεστή θα τον δούμε αργότερα

Λίστες

Οι λίστες (https://el.wikipedia.org/wiki/%CE%9B%CE%AF%CF%83%CF%84%CE%B1_%28%CE%B1%CF%86%CE%B7%CF%81%CE%B7%CE%BC%CE%AD%CE%BD%CE%BF%CF%82_%CF%84%CF%8D%CF%80%CE%BF%CF%82_%CE%B4%CE%B5%CE%B4%CE%BF%CE%BC%CE%AD%CE%BD%CF%89%CE%BD%29) είναι ένα μία βασική έννοια της επιστήμης υπολογιστών.

Στην ουσία είναι μια διατεταγμένη σειρά από δεδομένα. Διατεταγμένο = κάθε στοιχείο έχει τη θέση του (1η, 2η, ...)

```
In [1]: a = [1,2,3,4]
        print (a)

        [1, 2, 3, 4]
```

Μία λίστα μπορεί να έχει στοιχεία διαφορετικού τύπου (αριθμοί, δεκαδικά, strings, ...)

```
In [2]: a = [1,2,3,"mitsos", 5, 7.77777778]
        print (a)

        [1, 2, 3, 'mitsos', 5, 7.77777778]
```

Η προσπέλαση των στοιχείων μίας λίστας γίνεται ακριβώς όπως και με τα strings:

```
In [3]: a[0] # Το πρώτο στοιχείο
Out[3]: 1

In [4]: a[0:3] #  Όλα τα στοιχεία από το πρώτο μέχρι το τέταρτο (χωρίς το τέταρτο)
Out[4]: [1, 2, 3]
```

In [5]:
```
a[-1] # Το τελευταίο στοιχείο
```

Out[5]:
```
7.77777778
```

In [6]:
```
a[-2:] # Το προτελευταίο στοιχείο
```

Out[6]:
```
[5, 7.77777778]
```

Ομοίως, μπορούμε να χρησιμοποιήσουμε τα διαστήματα. Έστω:

In [7]:
```
b = [1,2,3,4,5,6]
```

In [8]:
```
b[2:5:2] # Από το 3ο μέχρι το 6ο (χωρίς να πάρουμε ΚΑΙ το 6ο), με βήμα 2
```

Out[8]:
```
[3, 5]
```

In [9]:
```
b[::2] # Από την αρχή μέχρι ΚΑΙ το τέλος με βήμα 2
```

Out[9]:
```
[1, 3, 5]
```

In [10]:
```
b[:3] # Από την αρχή μέχρι το 4ο στοιχείο (χωρίς να πάρουμε ΚΑΙ το 4ο)
```

Out[10]:
```
[1, 2, 3]
```

In [11]:
```
# Όλα τα παρακάτω είναι ισοδύναμα
print (b)
print (b[:])
print (b[::])
print (b[::1])
print (b[0:])
print (b[0::])
print (b[0::1])
print (b[:len(b)])
print (b[:len(b):])
print (b[0:len(b)])
print (b[0:len(b):1])
```

Out[11]:
```
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
```

In [12]:
```
b[-1:0:-1] # Από το τέλος μέχρι την αρχή (χωρίς να πάρουμε ΚΑΙ την αρχή)
```

Out[12]:
```
[6, 5, 4, 3, 2]
```

In [13]:
```
b[-1::-1] # Από το τέλος μέχρι την αρχή (παίρνουμε και την αρχή)
```

Out[13]:
```
[6, 5, 4, 3, 2, 1]
```

In [14]:
```
b[::-1] # Αυτό είναι ισοδύναμε με το παραπάνω
```

Out[14]:
```
[6, 5, 4, 3, 2, 1]
```

Όπως και τα strings έτσι και στις λίστες μπορούμε να εφαρμόσουμε τις `len` , `count` , `index` .

In [15]:
```
a = [1,2,3,"mitsos", 5, 7.77777778]
```

In [16]:
```
len(a) # Το πλήθος όλων των στοιχείων της λίστας
```

Out[16]: 6

In [17]:
```
a.count(1) # Πόσες φορές υπάρχει το 1 μέσα στη λίστα;
```

Out[17]: 1

In [18]:
```
a.count(55) # Πόσες φορές υπάρχει το 55 μέσα στη λίστα;
```

Out[18]: 0

In [19]:
```
a.index("mitsos") # Σε ποια θέση της λίστας εμφανίζεται το "mitsos"?
```

Out[19]: 3

In [20]:
```
a.index(4) # Σε ποια θέση της λίστας εμφανίζεται το 4;
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-20-e644608c73eb> in <module>()
----> 1 a.index(4) # Σε ποια θέση της λίστας εμφανίζεται το 4;

ValueError: 4 is not in list
```

Μία λίστα μπορεί να έχει μέσα άλλες λίστες!

In [21]:
```
a = [1,2, [3,4,5], 6, 7]
```

In [22]:
```
len(a)
```

Out[22]: 5

In [23]:
```
a[2]
```

Out[23]: [3, 4, 5]

In [24]:
```
a[1]
```

Out[24]: 2

In [25]:
```
a[2][1]
```

Out[25]: 4

Υπάρχει επίσης η άδεια λίστα: `[]`

In [26]:
```
a = []
print (len(a))
```

```
0
```

Μπορούμε να γράψουμε τις λίστες με πολλούς τρόπους:

In [27]:
```
# Τα παρακάτω είναι ισοδύναμα:
a = [1,2,3]

a = [
    1,
    2,
    3,
]
```

Προσοχή! Δεν υπάρχει πρόβλημα αν βάλουμε ένα κόμμα στο τέλος μίας λίστας!

In [28]:
```
# Αυτά τα δύο είναι ισοδύναμα:
print ([1,2,3])
print ([1,2,3,])
```

```
[1, 2, 3]
[1, 2, 3]
```

Οπότε αν βάλουμε ένα κόμμα στο τέλος δεν υπάρχει πρόβλημα. Υπάρχει όμως αν ΔΕΝ βάλουμε στη μέση:

In [29]:
```
a = [
    'aaaa',
    'bbbb' # PROSOXH! auto einai 1 string (me to apo katw)
    'cccc'
]
print (len(a))
```

```
2
```

Μία λίστα μπορεί να έχει μία λίστα, που έχει μία λίστα που έχει..

In [30]:
```
a = [[]]
print (len(a))
```

```
1
```

In [31]:
```
a = [[[[[[[[[[[[[[[[[[]]]]]]]]]]]]]]]]]
print (len(a))
```

```
1
```

In [32]:
```
a = [1,2,3,[],4]
```

In [33]:
```
len(a)
```

Out[33]: 5

Μία λίστα μπορεί να έχει μεταβλητές:

In [34]:
```
a=3
b = [a,a,a+1, a/2]
print (b)
```

```
[3, 3, 4, 1.5]
```

Μπορούμε να προσθέσουμε δύο λίστες:

In [35]:
```
[1,2,3] + ["mitsos", "a"]
```

Out[35]:
```
[1, 2, 3, 'mitsos', 'a']
```

Μπορούμε να πολλαπλασιάσουμε μία λίστα με έναν αριθμό:

In [37]:
```
[1,2,3] *4
```

Out[37]:
```
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Δεν μπορούμε να πολλαπλασιάσουμε ή να αφαιρέσουμε δύο λίστες!

In [38]:
```
[1,2,3] * [5,6]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-38-4f81883a1dc4> in <module>()
----> 1 [1,2,3] * [5,6]

TypeError: can't multiply sequence by non-int of type 'list'
```

In [39]:
```
[1,2,3] - ["mitsos", "a"]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-d5954fea0119> in <module>()
----> 1 [1,2,3] - ["mitsos", "a"]

TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

Η Μέθοδος `list` κάνει ότι μπορεί να μετατρέψει κάτι σε λίστα:

In [32]:
```
list("mitsos")
```

Out[32]:
```
['m', 'i', 't', 's', 'o', 's']
```

In [258]:
```
list([1,2,3]) # Δεν κάνει τίποτα
```

Out[258]:
```
[1, 2, 3]
```

Δεν μπορούν να μετατραπούν τα πάντα σε λίστα

In [259]:
```
list(5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-259-0c7f5cd48ec1> in <module>()
----> 1 list(5)

TypeError: 'int' object is not iterable
```

Η εντολή `if`

Η εντολή `if` (εάν) εκτελεί άλλες εντολές ανάλογα με το αν μια έκφραση είναι `True` ή `False`.

In [83]:
```
if True:
    print ("Hello")
```

```
Hello
```

In [269]:
```
if False:
    print ("Hello") # Δεν εκτελείται
```

In [85]:
```
if 1<3:
    print ("Hello")
```

```
Hello
```

In [270]:
```
if [] in [1,2]:
    print ("Hello") # Δεν εκτελείται
```

Προσοχή! όλα τα strings εκτός από το άδειο είναι `True` :

In [87]:
```
if 'mitsos':
    print ("hello")
```

```
hello
```

In [271]:
```
if '':
    print ("hello") # Δεν εκτελείται
```

Όλοι αριθμοί εκτός από το 0 είναι `True`

In [89]:
```
if 3453:
    print ("Hello")
```

```
Hello
```

In [274]:
```
if 0: # Αυτό είναι False
    print ("Hello")
```

In [275]:
```
if 0.000000000001: # Αυτό είναι True!!
    print ("Hello")
```

```
Hello
```

Όλες οι λίστες είναι `True` εκτός από την άδεια:

In [91]:
```
if [1,2,3]:
    print ("Hello")
```

```
Hello
```

In [92]:
```
if []:
    print ("Hello")
```

In [93]:
```
if [0]: # Αυτό είναι True!
    print ("Hello")
```

```
Hello
```

In [276]:
```
if [False]: # Αυτό είναι True!
    print ("Hello")
```

```
Hello
```

Προσοχή στη διαφορά μεταξύ `=` και `==` :

In [117]:
```
a = 3 # Η μεταβλητή a παίρνει την τιμή 3
a == 3 # Ελέγχουμε αν η τιμή της μεταβλητής a είναι 3
```

Out[117]:
```
True
```

Αυτό δεν επιτρέπεται:

In [100]:
```
if a = 3:
    print ("Hello")
```

```
  File "<ipython-input-100-9d66b7bd4d9a>", line 1
    if a = 3:
         ^
SyntaxError: invalid syntax
```

Αυτό επιτρέπεται:

In [118]:
```
if a == 3:
    print ("Hello")
```

```
Hello
```

Μία if μπορεί να έχει "μέσα της" και άλλες if..

In [119]:
```
print ('1')
if True:
    print ('hello')
    if True:
        print ('Kostas')

print ('2')
```

```
1
hello
Kostas
2
```

Αν για κάποιο λόγο δεν θέλουμε να κάνει τίποτα η if, τότε πρέπει να χρησιμοποιήσουμε τη pass

In [120]:
```
print ('1')
if True:
    pass
print ('2')
```

```
1
2
```

Μπορούμε να δηλώσουμε τι θέλουμε να γίνεται όταν η συνθήκη της if ΔΕΝ είναι αληθής με την else:

In [123]:
```
print ('1')
if True:
    print ("hello") # <-- Μπαίνει εδώ
else:
    print ('world') # <-- Δεν μπαίνει εδώ
print ('2')
```

```
1
hello
2
```

In [124]:
```
print ('1')
if False:
    print ("hello") # <-- Δεν μπαίνει εδώ
else:
    print ('world') # <-- Μπαίνει εδώ
print ('2')
```

```
1
world
2
```

Επίσης μπορούμε να δηλώσουμε πολλές συνθήκες με την elif. Η python τις ελέγχει μία-μία και μόλις (και εάν) βρει τη πρώτη αληθή, τότε μπαίνει στο indentation.

In [125]:
```
print ('hello')
a=2
if a==1:
    print ('1')
elif a==2:
    print ('2')
else:
    print ('3')
print ('world')
```

```
hello
2
world
```

Δεν είναι απαραίτητο να υπάρχει η else

In [126]:
```
print ('hello')
a=3
if a==1:
    print ('1')
elif a==2:
    print ('2')
print ('world')
```

```
hello
world
```

Η εντολή `while`

Η εντολή `while` χρησιμοποιείται για επανάληψη. Με την εντολή `while <ΛΟΓΙΚΗ ΣΥΝΘΗΚΗ>:` δηλώνουμε ότι όλες οι εντολές "κάτω" από τη while θα τρέχουν μέχρι η `<ΛΟΓΙΚΗ ΣΥΝΘΗΚΗ>` να γίνει `False` .

In [127]:
```
# Τύπωσε όλους τους αριθμούς από το 0 μέχρι και το 9
a=0
while a<10:
    print (a)
    a += 1
```

```
0
1
2
3
4
5
6
7
8
9
```

Τύπωσε όλους τους **μονούς** αριθμούς από το 1 μέχρι το 100

In [128]:
```
a=1

while a<100:
    if a%2 == 1:
        print (a)
    a+=1
```

```
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
57
59
61
63
65
67
69
71
73
75
77
79
81
83
85
87
89
91
93
95
97
99
```

Τύπωσε τη "προπαίδεια" του 8

In [131]:
```
a = 1
while a<=10:
    print (a*8)
    a+=1
```

```
8
16
24
32
40
48
56
64
72
80
```

Λίγο καλύτερα:

In [132]:
```
a = 1
while a<=10:
    print (a, 'fores to 8 mas kanei', a*8)
    a+=1
```

```
1 fores to 8 mas kanei 8
2 fores to 8 mas kanei 16
3 fores to 8 mas kanei 24
4 fores to 8 mas kanei 32
5 fores to 8 mas kanei 40
6 fores to 8 mas kanei 48
7 fores to 8 mas kanei 56
8 fores to 8 mas kanei 64
9 fores to 8 mas kanei 72
10 fores to 8 mas kanei 80
```

Τύπωσε τη προπαίδεια όλων των αριθμών από το 1 μέχρι το 10

In [133]:
```
a = 1
while a<=10:
    b=1
    while b<=10:
        print (a, 'fores to ', b, ' mas kanei', a*b)
        b+=1
    a+=1
```

```
1 fores to  1  mas kanei 1
1 fores to  2  mas kanei 2
1 fores to  3  mas kanei 3
1 fores to  4  mas kanei 4
1 fores to  5  mas kanei 5
1 fores to  6  mas kanei 6
1 fores to  7  mas kanei 7
1 fores to  8  mas kanei 8
1 fores to  9  mas kanei 9
1 fores to  10  mas kanei 10
2 fores to  1  mas kanei 2
2 fores to  2  mas kanei 4
2 fores to  3  mas kanei 6
2 fores to  4  mas kanei 8
2 fores to  5  mas kanei 10
2 fores to  6  mas kanei 12
2 fores to  7  mas kanei 14
2 fores to  8  mas kanei 16
2 fores to  9  mas kanei 18
2 fores to  10  mas kanei 20
3 fores to  1  mas kanei 3
3 fores to  2  mas kanei 6
3 fores to  3  mas kanei 9
3 fores to  4  mas kanei 12
3 fores to  5  mas kanei 15
3 fores to  6  mas kanei 18
3 fores to  7  mas kanei 21
3 fores to  8  mas kanei 24
3 fores to  9  mas kanei 27
3 fores to  10  mas kanei 30
4 fores to  1  mas kanei 4
4 fores to  2  mas kanei 8
4 fores to  3  mas kanei 12
4 fores to  4  mas kanei 16
4 fores to  5  mas kanei 20
4 fores to  6  mas kanei 24
4 fores to  7  mas kanei 28
4 fores to  8  mas kanei 32
4 fores to  9  mas kanei 36
4 fores to  10  mas kanei 40
5 fores to  1  mas kanei 5
5 fores to  2  mas kanei 10
5 fores to  3  mas kanei 15
5 fores to  4  mas kanei 20
5 fores to  5  mas kanei 25
5 fores to  6  mas kanei 30
5 fores to  7  mas kanei 35
5 fores to  8  mas kanei 40
5 fores to  9  mas kanei 45
5 fores to  10  mas kanei 50
6 fores to  1  mas kanei 6
6 fores to  2  mas kanei 12
6 fores to  3  mas kanei 18
6 fores to  4  mas kanei 24
6 fores to  5  mas kanei 30
6 fores to  6  mas kanei 36
6 fores to  7  mas kanei 42
6 fores to  8  mas kanei 48
6 fores to  9  mas kanei 54
6 fores to  10  mas kanei 60
7 fores to  1  mas kanei 7
7 fores to  2  mas kanei 14
7 fores to  3  mas kanei 21
7 fores to  4  mas kanei 28
7 fores to  5  mas kanei 35
7 fores to  6  mas kanei 42
7 fores to  7  mas kanei 49
7 fores to  8  mas kanei 56
```

Βρες πόσα ψηφία έχει ένας αριθμό:

In [135]:
```
a=51234123
# Πρώτος τρόπος (faste and better)
len(str(a))
```

Out[135]: 8

In [136]:
```
# Δεύτερος τρόπος
# Παρατηρούμε ότι όταν δαιρούμε (ακέραια διαίρεση) έναν αριθμό με το 10, τότε τ
ου αφαιρούμε ένα ψηφίο από το τέλος:
# 51234123 // 10 --> 5123412

a=51234123
c=0
upoloipo = a
while upoloipo != 0:
    upoloipo = upoloipo // 10
    c += 1
print (c)
```

```
8
```

Πόσες φορές υπάρχει το γράμμα a σε ένα string;

In [137]:
```
# Πρώτος τροπος (better / faster)
a = 'zabarakatranemia'
a.count('a')
```

Out[137]: 6

In [138]:
```
# Δεύτερος τρόπος
index = 0
c = 0
while index < len(a):
    if a[index] == 'a':
        c += 1
    index += 1
print (c)
```

```
6
```

Αντιστροφή ενός string.

In [139]:
```
# Πρώτος τρόπος (better / faster)
a = 'zabarakatranemia'
a[::-1]
```

Out[139]: 'aimenartakarabaz'

In [140]:
```
# Δεύτερος τρόπος
index = len(a)-1
anapodo = ''
while index >= 0:
    anapodo = anapodo + a[index]
    index -= 1
print (anapodo)
```

```
aimenartakarabaz
```