



ΜΥΥ105: Εισαγωγή στον Προγραμματισμό

Αλφαριθμητικά

Χειμερινό Εξάμηνο 2014

[Λειτουργίες σε αλφαριθμητικά]

- Μπορούμε να εφαρμόσουμε όλες τις λειτουργίες που έχουμε δει για πλειάδες και λίστες (π.χ. slicing) σε αλφαριθμητικά
- **Προσοχή!**
 - Τα αλφαριθμητικά **δεν τροποποιούνται**

```
>>> website = 'http://www.python.org'
```

```
>>> website[-3:] = 'com'
```

```
Traceback (most recent call last):
```

```
File "<pyshell#4>", line 1, in <module>
```

```
    website[-3:] = 'com'
```

```
TypeError: 'str' object does not support item assignment
```

[Slicing αλφαριθμητικών]

```
>>> s='Hello World'
```

```
>>> s[4]
```

```
'o'
```

```
>>> s[3:7]
```

```
'lo W'
```

```
>>> s[-4:-1]
```

```
'orl'
```

```
>>> s[4:2]
```

```
"
```

```
>>> s[7:2:-2]
```

```
'o l'
```

*μη έγκυρο
διάστημα*

[Άλλες λειτουργίες]

```
>>> s='Hello World'
>>> min(s)
''
>>> max(s)
'r'
>>> 'Hello' in s
True
>>> len(s)
11
>>> 'Hello'*2+'World'
'HelloHelloWorld'
```

Μέθοδοι Αλφαριθμητικών

```
>>> s='Hello Hello World'
>>> s.count('Hello')
2
>>> s.replace('Hello','Bye')
'Bye Bye World'
>>> s.split()
['Hello', 'Hello', 'World']
>>> s = 'hello'
>>> s.capitalize()
'Hello'
>>> s.upper()
'HELLO'
```

*επιστρέφει το αλφαριθμητικό
μετά την αντικατάσταση αλλά
δεν αλλάζει το αρχικό s*

*τα αλφαριθμητικά δεν
τροποποιούνται!*

Μέθοδοι Αλφαριθμητικών

```
>>> s='Hello Hello World'
```

*επιστρέφει τη θέση της
πρώτης εμφάνισης του
'Hello' στο s*

```
>>> s.find('Hello')
```

```
0
```

```
>>> s.find('Hello',1)
```

*άρχισε να ψάχνεις από τη
θέση 1 και μετά*

```
6
```

```
>>> s.find('Help')
```

*δηλώνει ότι το 'Help' δεν
υπάρχει στο s*

```
-1
```

```
>>> seq = ['1', '2', '3', '4', '5']
```

```
>>> sep = '+'
```

```
>>> sep.join(seq)
```

*συνένωση των στοιχείων της
λίστας seq σε ένα
αλφαριθμητικό*

```
'1+2+3+4+5'
```

[Που είναι το λάθος;]

```
>>> seq = ['1', '2', '3', '4', 5]
>>> sep = '+'
>>> sep.join(seq)
```

*ΟΛΑ τα στοιχεία της
λίστας πρέπει να είναι
αλφαριθμητικά!*

Traceback (most recent call last):

File "<pyshell#106>", line 1, in <module>

sep.join(seq)

TypeError: sequence item 4: expected str instance, int found

[Παράδειγμα της join]

```
>>> dirs = "", 'usr', 'bin', 'env'
```

```
>>> '/'.join(dirs)
```

```
'/usr/bin/env'
```

```
>>> print('C:' + '\\'.join(dirs))
```

```
C:\usr\bin\env
```

ο χαρακτήρας '\'

[string formatting]

- Χρησιμοποιείται στον ορισμό η τύπωση (στοιχισμένων) αλφαριθμητικών με διαφόρους τύπους δεδομένων

δηλώνει μετατροπή σε αλφαριθμητικό

```
>>> '%s plus %s equals %s' % (1, 1, 2)
```

```
'1 plus 1 equals 2'
```

```
>>> from math import pi
```

```
>>> 'Pi: %f..' % pi δηλώνει μετατροπή σε float
```

```
'Pi: 3.141593...'
```

```
>>> 'Very inexact estimate of pi: %i' % pi δηλώνει μετατροπή σε int
```

```
'Very inexact estimate of pi: 3'
```

Πλάτος και ακρίβεια

```
>>> '%10f' % pi # Field width 10
' 3.141593' η έξοδος είναι αλφαριθμητικό πλάτους 10
>>> '%10.2f' % pi # Field width 10,
precision 2
' 3.14'
>>> '%.2f' % pi # Precision 2
'3.14'
>>> '%.5s' % 'Guido van Rossum'
'Guido' παραμετροποιημένο πλάτος
>>> '%.*s' % (5, 'Guido van Rossum')
'Guido'
>>> '%010.2f' % pi '0' αντί για κενά στις θέσεις που δεν χρησιμοποιούνται
'0000003.14'
```

[Στοίχιση]

```
>>> '%-10.2f' % pi
3.14
>>> print('% 5d' % 10) + '\n' + ('% 5d' % -10)
 10
-10
>>> print('%+5d' % 10) + '\n' + ('%+5d' % -10)
+10
-10
```

στοίχιση στα αριστερά

θετικός: κενό, αρνητικός: -

θετικός: +, αρνητικός: -

[Παραδείγματα]

```
>>> x=1234.5678
>>> print(x)
1234.5678
>>> print('%d' % x)
1234
>>> print('%10d' % x)
      1234
>>> print('%s' % x)
1234.5678
>>> print('%10s' % x)
1234.5678
```

[Παραδείγματα]

```
>>> x=1234.5678
>>> print('%9.4f' % x)
1234.5678
>>> print('%10.4f' % x)
 1234.5678
>>> print('%10.3f' % x)
 1234.568
>>> print('%10.5f' % x)
1234.56780
```

Παράδειγμα: τύπωμα τιμοκαταλόγου

```
# Print a formatted price list with a given width
width = int(input('Please enter width: '))
price_width = 10
item_width = width - price_width
header_format = '%-*s%*s'
fformat = '%-*s%*.2f'
print('=' * width)
print(header_format % (item_width, 'Item', price_width, 'Price'))
print('-' * width)
print(fformat % (item_width, 'Apples', price_width, 0.4))
print(fformat % (item_width, 'Pears', price_width, 0.5))
print(fformat % (item_width, 'Cantaloupes', price_width, 1.92))
print(fformat % (item_width, 'Dried Apricots (16 oz.)', price_width, 8))
print(fformat % (item_width, 'Prunes (4 lbs.)', price_width, 12))
print('=' * width)
```

Παράδειγμα: τύπωμα τιμοκαταλόγου

Please enter width: 40

```
=====
```

Item	Price
Apples	0.40
Pears	0.50
Cantaloupes	1.92
Dried Apricots (16 oz.)	8.00
Prunes (4 lbs.)	12.00

```
=====
```

[Αλφαριθμητικές σταθερές]

```
>>> import string
>>> string.digits
'0123456789'
>>> string.printable
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHI
IJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[
\]^_`{|}~ \t\n\r\x0b\x0c'
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
>>> '5' in string.digits
True
```

χρήση του string module

[Αλφαριθμητικές σταθερές]

- `string.digits` A string containing the digits 0–9
- `string.letters` A string containing all letters (upper- and lowercase)
- `string.lowercase` A string containing all lowercase letters
- `string.printable` A string containing all printable characters
- `string.punctuation` A string containing all punctuation characters
- `string.uppercase` A string containing all uppercase letters