Ελληνικό Μεσογειακό Πανεπιστήμιο
Σχολή Μηχανικών / Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

HMY01K06
Επιστημονικός Προγραμματισμός με Python

Διάλεξη Τέταρτη
Συμβολοσειρές

Φθινόπωρο 2025

Ορισμός

Συμβολοσειρά: Ακολουθία χαρακτήρων που περικλείεται από *μονά* ή *διπλά* εισαγωγικά

(ο ορισμός αυτός ισχύει εφόσον εκτείνεται σε μια μόνο γραμμή)

```
>>> city = 'Ηράκλειο'
>>> city
'Ηράκλειο'
>>> city[0]
'Η'
```

```
>>> city = "Ηράκλειο"
>>> city
'Ηράκλειο'
>>> city[0]
'Η'
```

Η Python δεν κάνει διάκριση ανάμεσα στα *μονά* ή τα *διπλά* εισαγωγικά. Αν το κείμενό μας περιέχει το ένα είδος εισαγωγικών, τότε το περικλείουμε χρησιμοποιώντας το άλλο είδος

```
a = 'Μας είπε "Καλημέρα" και χαμογέλασε'
b = "Το σημείο Α' είναι συμμετρικό του Α"
```

Εναλλακτικά μπορούμε να χρησιμοποιούμε το ίδιο είδος εισαγωγικών και μέσα στο κείμενο βάζοντας ενα *backslash* (\) ακριβώς πριν

```
a = "Μας είπε \"Καλημέρα\" και χαμογέλασε"
b = 'Το σημείο Α\' είναι συμμετρικό του Α'
```

Σε μια συμβολοσειρά, το *backslash* πριν από έναν χαρακτήρα σημαίνει γενικά ότι αυτός ο χαρακτήρας έχει ξεχωριστή σημασία

Συμβολοσειρές με τριπλά εισαγωγικά *(docstrings)*

Με *τριπλά* εισαγωγικά - τρία *μονά* (' ' ') ή τρία *διπλά* (" " ") - μπορούμε να ορίσουμε συμβολοσειρές που εκτείνονται σε *περισσότερες από μια γραμμές*

```
s = """This
is
a
test"""

print(s)
```

Τρία διπλά εισαγωγικά

```
>>>This
   is
   a
   test
```

```
s = '''This
      is
      a
      test'''

print(s)
```

Τρία μονά εισαγωγικά

```
>>>This
   is
   a
   test
```

Συμβολοσειρές με τριπλά εισαγωγικά *(docstrings)*

Οι συμβολοσειρές με τριπλά εισαγωγικά όταν εμφανίζονται ως *πρώτη πρόταση* κατά τον ορισμό

ενός *πακέτου* / μιας *συνάρτησης* / μιας *κλάσης* / ή μιας *μεθόδου*

τότε ονομάζονται *docstrings (συμβολοσειρές τεκμηρίωσης)* και περιέχουν συνήθως ένα *σύντομο* κείμενο *(ορισμένες φορές ίσως και λίγο πιο αναλυτικό)* περιγραφής της λειτουργίας του συγκεκριμένου στοιχείου

```
def max(a, b):
    """Returns the maximum of a and b.
    a and b may be integer or real"""
   <κώδικας συνάρτησης>
```

```
>>> help(max)
Returns the maximum of a and b.
a and b may be integer or real
```

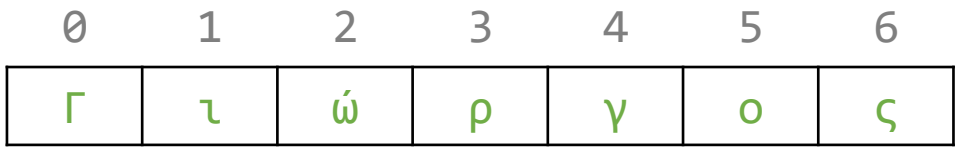
Το κείμενο *αυτό εκχωρείται αυτόματα σε έναν ειδικό προσδιοριστή* *__doc__*

και μπορούμε να το εμφανίσουμε από την κονσόλα δίνοντας την εντολή *help()* και σε παρένθεση το όνομα αυτού του αντικειμένου *(πακέτου, συνάρτησης κ.λπ.)*

Θα δούμε σχετικά παραδείγματα σε επόμενες διαλέξεις

Δημιουργία συμβολοσειρών

```
>>> name = "Γιώργος"
>>> name
'Γιώργος'
>>> name[0]
'Γ'
>>> name[3]
'ρ'
>>> len(name)
7
```



- Σκεφτόμαστε τις συμβολοσειρές σαν *μονοδιάστατους πίνακες* που σε κάθε κελί τους περιέχουν *εναν ακριβώς χαρακτήρα*
- Ο δείκτης του πίνακα *ξεκινάει να μετράει* ***πάντα από το μηδέν***
- Η συνάρτηση `len(s)` επιστρέφει το πλήθος χαρακτήρων μιας συμβολοσειράς `s`

Προσέξτε τη διαφορά στα παρακάτω:

```
>>> S1 = " "   Συμβολοσειρά που περιέχει ένα κενό. Το len(S1) επιστρέφει 1
>>> S2 = ""    Κενή συμβολοσειρά. Το len(S2) επιστρέφει 0
```

Μπορούμε να αλλάξουμε ένα χαρακτήρα σε μια συμβολοσειρά;

```
>>> a='μηλοπιτα'
>>> a[4]
'π'
>>> a[4]="ρ"
Traceback (most recent call last):
  File "<pyshell#64>", line 1, in <module>
    a[4]="ρ"
TypeError: 'str' object does not support item assignment
```

Από τη στιγμή που ορίσουμε μια συμβολοσειρά,

δεν μπορούμε πια να την να αλλάξουμε

Θυμόμαστε ότι:

Οι συμβολοσειρές στην Python είναι *αμετάβλητα αντικείμενα* ***immutables***

Χειρισμός συμβολοσειρών

Συνένωση και *"πολλαπλασιασμός"* συμβολοσειρών

```
>>> a = "Monty"
>>> b = "Python"
>>> a+b
'MontyPython'

>>> a + ' ' + b
'Monty Python'

>>> a*2
'MontyMonty'

>>> a + ' ' * 2
'Monty  '
```

(και όχι 'Monty Monty' – γιατί;)

Αν `S1` και `S2` είναι συμβολοσειρές και το `n` είναι ακέραιος, τότε:

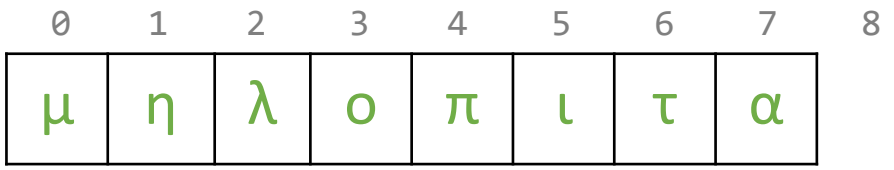
- Η πράξη `S1 + S2` επιστρέφει μια νέα συμβολοσειρά η οποία προκύπτει από την συνένωση των αρχικών
- Η πράξη `S1 * n` επιστρέφει μια νέα συμβολοσειρά η οποία προκύπτει από την επανάληψη της αρχικής `n` φορές
  Αν το n είναι μηδέν ή αρνητικός τότε επιστρέφεται μια κενή συμβολοσειρά

```
>>> a+2
Traceback (most recent call last):
  File "<pyshell#58>", line 1, in <module>
    a+2
TypeError: can only concatenate str
(not "int") to str
```

Χειρισμός συμβολοσειρών

Slicing - Αποκοπή και χρήση τμήματος συμβολοσειράς 1/2

```
>>> s='μηλοπιτα'
>>> len(s)
8
>>> s[0]
'μ'
>>> s[7]
'α'
>>> s[0:4]
'μηλο'
>>> s[4:7]
'πιτ'
>>> s[4:8]
'πιτα'
>>> s[0:8]
'μηλοπιτα'
```

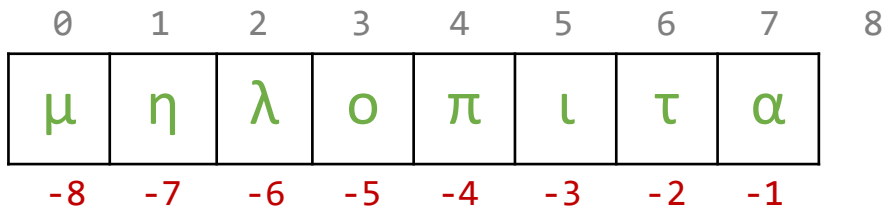


- Το πρώτο στοιχείο μιας συμβολοσειράς έχει πάντα δείκτη `0` και το τελευταίο έχει δείκτη `n-1` όπου `n` το μήκος (αριθμός χαρακτήρων) της συμβολοσειράς
- Η έκφραση `s[i:j]`επιστρέφει το τμήμα της συμβολοσειράς από τον χαρακτήρα `s[i]` μέχρι και τον χαρακτήρα `s[j-1]`
- Μπορούμε να παραλείψουμε τον έναν ή και τους δυο δείκτες
  - `s[:j]` σημαίνει από την αρχή μέχρι τη θέση `j-1`
  - `s[i:]` σημαίνει από τη θέση `i` μέχρι το τέλος

Χειρισμός συμβολοσειρών

Slicing - Αποκοπή και χρήση τμήματος συμβολοσειράς 2/2

```
>>> s='μηλοπιτα'
>>> s[:3]
'μηλ'
>>> s[2:]
'λοπιτα'
>>> s[:]
'μηλοπιτα'
>>> s[-1]
'α'
>>> s[:-1]
'μηλοπιτ'
>>> s[-3: -1]
'ιτ'
>>> s[-8]
'μ'
>>> s[8]
Traceback (most recent call last):
s[8] IndexError: string index out
of range
```



- Μπορούμε να προσπελάσουμε τα στοιχεία μιας συμβολοσειράς και με ***αρνητικούς*** δείκτες. Ο αρνητικός δείκτης `-j` είναι ίδιος με τον θετικό δείκτη `n-j` όπου `n` το μήκος της συμβολοσειράς

  Πχ αν `s='μηλοπιτα'` *(μήκος 8 χαρακτήρες)* τότε το `s[-4:-2]` ισοδυναμεί με το `s[8-4:8-2]` δηλαδή το `s[4:6]` που είναι `'πι'`

Γενικότερα για οποιαδήποτε συμβολοσειρά `s` ισχύει:

- `s[0]` πρώτο στοιχείο
- `s[-1]` τελευταίο στοιχείο
- `s[:]` ή `s[::]` ολόκληρη η συμβολοσειρά
- `s[::-1]` *η αντίστροφη* συμβολοσειρά

Συναρτήσεις και μέθοδοι χειρισμού συμβολοσειρών

Σε τι διαφέρει μια *συνάρτηση* χειρισμού ενός αντικειμένου από μια *μέθοδο* χειρισμού ενός αντικειμένου;

Η *συνάρτηση* χειρισμού ενός αντικειμένου *δέχεται σαν όρισμα το αντικείμενο (πιθανόν να δέχεται και άλλα επιπλέον ορίσματα)* και επιστρέφει μια τιμή, που συνήθως εκχωρείται σε έναν προσδιοριστή

πχ. `a = len("abc")` *το a θα πάρει την τιμή 3*

Η *μέθοδος* χειρισμού ενός αντικειμένου είναι μια *ειδική συνάρτηση* που *εφαρμόζεται πάνω σε ένα αντικείμενο* και δηλώνεται ως εξής:

`αντικείμενο.μέθοδος()`

πχ. `s = "abc".upper()` *το s θα πάρει την τιμή "ABC"*

Μια μέθοδος μπορεί να έχει δικά της ορίσματα *(δηλώνονται μέσα στην παρένθεση της μεθόδου)* ή να μην έχει καθόλου *(κενή παρένθεση)*

Συναρτήσεις χειρισμού συμβολοσειρών

`len(s)` Επιστρέφει το πλήθος χαρακτήρων μιας συμβολοσειράς `s`, συμπεριλαμβανομένων και όσων είναι μη εκτυπώσιμοι π.χ. `\n (newline), \t (tab)`

`len('Hello')` επιστρέφει 5

`len('Line1\nLine2')` επιστρέφει 11 *(το \n είναι και αυτό **ένας** χαρακτήρας)*

`str(obj)` Μετατρέπει το αντικείμενο `obj` σε συμβολοσειρά

`str(12)` επιστρέφει `'12'`

`str(12/7)` επιστρέφει `'1.7142857142857142'`

`str(2+3j)` επιστρέφει `'(2+3j)'`

`str([1, 7, "Hi"])` επιστρέφει `'[1, 7, "Hi"]'`

Η Python βάζει εδώ την παρένθεση για να υποδηλώσει ότι πρόκειται για μιγαδικό αριθμό που έγινε συμβολοσειρά

`sorted(s)` Επιστρέφει μια *λίστα* με όλους τους χαρακτήρες της **s** ταξινομημένους σύμφωνα με την θέση τους στον χρησιμοποιούμενο πίνακα κωδικοποίησης (πχ ASCII)

`sorted('test')` επιστρέφει `['e', 's', 't', 't']`

Μέθοδοι χειρισμού συμβολοσειρών

Μέθοδοι ελέγχου περιεχομένου – επιστρέφουν `True` ή `False`

`s.isalnum()`	`True` αν η s αποτελείται ***μόνο*** από γράμματα ή/και αριθμητικά(*) ψηφία πχ. `'abc1²23'`
`s.isalpha()`	`True` αν η s αποτελείται ***μόνο*** από γράμματα
`s.isdecimal()`	`True` αν η s αποτελείται ***μόνο*** από τα δέκα δεκαδικά ψηφία (0…9)
`s.isdigit()`	`True` αν η s αποτελείται ***μόνο*** από αριθμητικά(*) ψηφία πχ.`'27'` `'2²'`
`s.isnumeric()`	`True` αν η s είναι μια ***γενικότερα έγκυρη αριθμητική έκφραση*** πχ. `'5⅔'`
`s.startswith(s1)`	`True` αν η s αρχίζει από s1 `'Hello'.startswith('He')` επιστρέφει `True`
`s.endswith(s1)`	`True` αν η s τελειώνει σε s1 `'Hello'.endswith('elo')` επιστρέφει `False`
`s.isidentifier()`	`True` αν η s είναι έγκυρος προσδιοριστής `'4a'.isidentifier` επιστρέφει `False`
`s.isupper()`	`True` αν όλα τα γράμματα που περιέχει η s είναι κεφαλαία
`s.islower()`	`True` αν όλα τα γράμματα που περιέχει η s είναι πεζά
`s.istitle()`	`True` αν η s έχει Κεφαλαίο το πρώτο γράμμα κάθε λέξης που περιέχει
`s.isprintable()`	`True` αν όλοι οι χαρακτήρες της s είναι εκτυπώσιμοι *(δεν περιέχει `'\n'`,`'\t'` κλπ.)*

(*) *ως αριθμητικά ψηφία νοούνται τα δεκαδικά ψηφία (0…9) είτε στη συνηθισμένη τους μορφή, είτε σε μορφή εκθέτη.*

Μέθοδοι χειρισμού συμβολοσειρών

Διαφορές ανάμεσα στις μεθόδους `isdecimal`, `isdigit` και `isnumeric`

Είδος συμβολοσειράς	Παράδειγμα	`isdecimal`	`isdigit`	`isnumeric`
Ψηφία 0…9	`'0123'`	`True`	`True`	`True`
Εκθέτες και δείκτες	`'2²'`	`False`	`True`	`True`
Κλάσματα Ρωμαϊκοί αριθμοί…	`'5⅗'`, `'Ð'`	`False`	`False`	`True`

Είναι True μόνο για θετικούς ακέραιους

Πως ελέγχουμε αν μια συμβολοσειρά είναι δεκαδικός αριθμός;

Δεν υπάρχει αντίστοιχη μέθοδος *όπως πχ. .isfloat()* οπότε το κάνουμε με διάφορους τρόπους

Πχ αν `s = '12.3'`

- Αντικαθιστούμε την τελεία `"."` με `""` *(θα δούμε πως)* και ελέγχουμε με την μέθοδο `isdecimal`
- Δοκιμάζουμε να την μετατρέψομε σε float με το `float(s)` και αν "χτυπήσει" το χειριζόμαστε κατάλληλα *(χρησιμοποιούμε την δομή `try-except` που θα δούμε σε επόμενη διάλεξη)*

Μέθοδοι χειρισμού συμβολοσειρών

Μέθοδοι διαχωρισμού / συνένωσης (1/2)

`s.split(sep, maxsplit)`	Σπάει μια συμβολοσειρά σε μικρότερα κομμάτια που διαχωρίζονται από το `sep` τα οποία επιστρέφει σε μια λίστα. Ο προαιρετικός ακέραιος `maxsplit` καθορίζει το μέγιστο αριθμό κομματιών που διαχωρίζονται. Αν παραληφθεί το `sep` , νοείται το κενό
	πχ αν `s = '12t!bv!34 5!!1z7'` τότε
	`s.split('!')` επιστρέφει `['12t', 'bv', '34 5', '', '1z7']`
	`s.split('!',2)` επιστρέφει `['12t', 'bv', '34 5!!1z7']`
	`s.split()` επιστρέφει `['12t!bv!34', '5!!1z7']`

`s.rsplit(sep, maxsplit)`	Ίδιο όπως το `split` αν δεν υπάρχει το `maxsplit` Αν υπάρχει, τα `maxsplit` κομμάτια διαχωρίζονται από το τέλος *(δεξιά προς τα αριστερά)* `s.rsplit('!',2)` επιστρέφει `['12t!bv!345', '', '1z7']`
`s.splitlines()`	Σπάει την `s` σε γραμμές. Ουσιαστικά είναι ισοδύναμη με την `s.split('\n')` πχ αν `s = 'hello\nworld\nagain'` `s.splitlines()` επιστρέφει `['hello', 'world', 'again']`

Μέθοδοι χειρισμού συμβολοσειρών

Μέθοδοι διαχωρισμού / συνένωσης (2/2)

`s.join(iter)`	Επιστρέφει μια νέα συμβολοσειρά όπου η `s` παρεμβάλλεται ανάμεσα στα στοιχεία του επαναληπτικού τύπου(*) `iter` πχ. `'...'.join('123')` επιστρέφει `'1...2...3'`

`s.partition(sep)`	Σπάει την συμβολοσειρά στην ***πρώτη*** εμφάνιση του `sep` και επιστρέφει μια *πλειάδα* τριών στοιχείων: (κομμάτι πριν, `sep` , κομμάτι μετά)
	πχ αν `s = '12t!bv!345!!1z7'` τότε
	`s.partition('!')` επιστρέφει `('12t', '!', 'bv!345!!1z7')`
	`s.partition('A')` επιστρέφει `('12t!bv!345!!1z7','', '')`

`s.rpartition(sep)`	Σπάει την συμβολοσειρά στην ***τελευταία*** εμφάνιση του `sep` και επιστρέφει μια *πλειάδα* τριών στοιχείων: (κομμάτι πριν, `sep` , κομμάτι μετά)
	πχ αν `s = '12t!bv!345!!1z7'` τότε
	`s.rpartition('!')` επιστρέφει `('12t!bv!345!', '!', '1z7')`
	`s.rpartition('A')` επιστρέφει `('', '','12t!bv!345!!1z7')`

(*) *Επαναληπτικοί τύποι της Python είναι μεταξύ άλλων οι λίστες, οι πλειάδες, οι συμβολοσειρές, τα λεξικά και τα σύνολα*

Μέθοδοι χειρισμού συμβολοσειρών

Μέθοδοι εύρεσης / αντικατάστασης (1/2)

`s.find(s1)`
`s.find(s1,`*`start,end`*`)`
	Επιστρέφει τον *δείκτη* της θέσης που συναντάται για ***πρώτη*** φορά η `s1` μέσα στην `s`
	Αν έχουν οριστεί τα `start` και `end` τότε η αναζήτηση περιορίζεται στο κομμάτι της συμβολοσειράς που ξεκινάει από τη θέση `start` και τελειώνει στην θέση `end-1`
	πχ αν **`s`** `= '12t!bv!345!!1z7'` τότε
	`s.find('!')` επιστρέφει `3` *(θυμηθείτε, ξεκινάμε από το 0)*
	`s.find('!',4,7)` επιστρέφει `6`
	`s.find('2t')` επιστρέφει `1`
	`s.find('@')` επιστρέφει `-1` *(-1 σημαίνει δεν βρέθηκε)*

`s.rfind(s2)`
r: from the right
	Επιστρέφει τον *δείκτη* της θέσης που συναντάται για ***τελευταία*** φορά η `s1` μέσα στην `s`
	πχ αν `s = '12t!bv!345!!1z7'`τότε
	`s.rfind('!')` επιστρέφει `11`
	`s.rfind('2t')` επιστρέφει `1`
	`s.rfind('@')` επιστρέφει `-1`

`s.index(s1)`
`s.index(s1,`*`start,end`*`)`
`s.rindex(s1)`
	Ακριβώς αντίστοιχες με τις `find` και `rfind` με μόνη διαφορά ότι αν η `s1` δεν βρεθεί, τότε η Python βγάζει το μήνυμα λάθους `ValueError` *(αντί να επιστρέψει -1)*

Μέθοδοι χειρισμού συμβολοσειρών

Μέθοδοι εύρεσης / αντικατάστασης (2/2)

`s.replace(s1, s2)`	Επιστρέφει μια νέα συμβολοσειρά που προκύπτει από την αντικατάσταση της `s1` με `s2` παντού μέσα στην `s`
	πχ αν `s = 'this is it!'` τότε
	`s.replace('i','a')` επιστρέφει `'thas as at'`
	`s.replace('is','abc')` επιστρέφει `'thabc abc it'`

`s.replace(s1, s2, n)`	Επιστρέφει μια νέα συμβολοσειρά που προκύπτει από την αντικατάσταση της `s1` με `s2` μόνο <u>τις πρώτες n φορές</u> που την εντοπίζει μέσα στην `s` πχ αν `s = 'this is it!'` τότε `s.replace('i','a', 2)` επιστρέφει `'thas as it'`

Απορία: Μα αφού είπαμε ότι οι συμβολοσειρές είναι <u>αμετάβλητα</u> αντικείμενα, πως γίνεται να κάνουμε εύρεση / αντικατάσταση;

Απάντηση: Στην ουσία <u>*δεν αλλάζει η αρχική μας συμβολοσειρά*</u>, καθώς <u>*δημιουργείται και επιστρέφεται μια νέα*</u> στην οποία έχει εφαρμοστεί η εύρεση/αντικατάσταση

Μέθοδοι χειρισμού συμβολοσειρών

Μέθοδοι αλλαγής πεζών/κεφαλαίων

`s.capitalize()`	Μετατρέπει τον *πρώτο* χαρακτήρα κεφαλαίο και τους υπόλοιπους πεζά `'heLLo'.capitalize()` επιστρέφει `'Hello'`
`s.lower()`	Μετατρέπει *όλους* τους χαρακτήρες σε πεζά `'HeLLO'.lower()` επιστρέφει `'hello'`
`s.upper()`	Μετατρέπει *όλους* τους χαρακτήρες σε κεφαλαία `'helLo'.upper()` επιστρέφει `'HELLO'`
`s.swapcase()`	Αντιστρέφει κεφαλαία ⇔ πεζά `'HeLLo'.swapcase()` επιστρέφει `'hEllO'`
`s.title()`	Μετατρέπει τον πρώτο χαρακτήρα *κάθε λέξης* κεφαλαίο και όλους τους υπόλοιπους πεζά `'heLLo wORLd!'.title()` επιστρέφει `'Hello World!'`

Μέθοδοι χειρισμού συμβολοσειρών

Μέθοδοι χειρισμού ελεύθερου χώρου (κενών) 1/2

`s.center(width,`*`pad`*`)`	Κεντράρει την `s` σε πεδίο πλάτους `width` με κενά δεξιά και αριστερά.
	Εναλλακτικά αντί κενού μπορεί να χρησιμοποιηθεί ο χαρακτήρας **`pad`**
	`'Hi'.center(10)` επιστρέφει `'  Hi  '`
	`'Hi'.center(10,'-')` επιστρέφει `'----Hi----'`
	
	Αν το `width` είναι μικρότερο από το μήκος της `s` τότε την αφήνει ως έχει
	`'Hi'.center(1,'-')` επιστρέφει `'Hi'`

`s.ljust(width,`*`pad`*`)`
`s.rjust(width,`*`pad`*`)`
	Τοποθετεί την `s` στην αριστερή (αντίστοιχα δεξιά) άκρη ενός πεδίου με κενά πλάτους `width`.
	Εναλλακτικά αντί κενού μπορεί να χρησιμοποιηθεί ό χαρακτήρας pad
	`'Hi'.ljust(10)` επιστρέφει `'Hi        '`
	`'Hi'.rjust(10,'-')` επιστρέφει `'--------Hi'`
	
	Αν το `width` είναι μικρότερο από το μήκος της `s` τότε την αφήνει ως έχει
	`'Hi'.rjust(1,'-')` επιστρέφει `'Hi'`

Μέθοδοι χειρισμού συμβολοσειρών

Μέθοδοι χειρισμού ελεύθερου χώρου (κενών) 2/2

`s.lstrip()`
`s.rstrip()`
	Καθαρίζει όλα κενά από την *αριστερή (αντίστοιχα την δεξιά)* άκρη της την `s`
	`'  Hi  '.lstrip()` επιστρέφει `'Hi  '`
	`'  Hi  '.rstrip()` επιστρέφει `'  Hi'`

`s.strip()`	Καθαρίζει όλα τα κενά *αριστερά και δεξιά* της την `s` *Ισοδυναμεί με την ταυτόχρονη εφαρμογή των lstrip και rstrip πάνω στην s* `'  Hi there  '.strip()` επιστρέφει `'Hi there'`
`s.expandtabs(n)`	Αντικαθιστά κάθε μη εκτυπώσιμο χαρακτήρα `tab` `(\t)` που υπάρχει στην `s` με `n` κενά. Αν παραλείψουμε να δώσουμε το `n`, τότε κάθε tab αντικαθίσταται από 8 κενά *(που είναι το προκαθορισμένο μήκος tab)* `'hello\tworld'.expandtabs(4)` επιστρέφει `'hello    world'` (4 κενά) `'hello\tworld'.expandtabs()` επιστρέφει `'hello        world'` (8 κενά)

Μέθοδοι χειρισμού συμβολοσειρών

Ειδική περίπτωση: Αντιστροφή συμβολοσειράς

Για τις συμβολοσειρές δεν υπάρχει κάποια ιδιαίτερη μέθοδος αντιστροφής (πχ reverse)

Ο απλούστερος *(και συνάμα ταχύτερος)* τρόπος να γίνει είναι με *slicing*

```
s[::-1]
```

Παράδειγμα:

Αν `s = 'Η Python είναι μια πολύ δυνατή και εύχρηστη γλώσσα'`

τότε η `s[::-1]` επιστρέφει

`'ασσώλγ ητσηρχύε ιακ ήτανυδ ύλοπ αιμ ιανίε nohtyP Η'`

Επ' ευκαιρία ξαναθυμόμαστε ότι για οποιαδήποτε συμβολοσειρά `s`:

- `s[0]` επιστρέφει το *πρώτο* στοιχείο της συμβολοσειράς
- `s[-1]` επιστρέφει το *τελευταίο* στοιχείο της συμβολοσειράς
- `s[:]` ή `s[::]` επιστρέφει *ολόκληρη* την συμβολοσειρά

Παραδείγματα εμπέδωσης (1/2)

```
>>> a="Hello Everybody"
>>> len(a)
15
>>> str(2+3j)  #complex number
'(2+3j)'
```

```
>>> a= "Book"
>>> a.lower()
'book'
>>> a.upper()
'BOOK'
```

```
>>> a= "    Book  "
>>> a.strip()
'Book'
```

```
>>> "Book  123"[::-1]
'321 kooB'
```

```
>>> a="University"
>>> a.find("ver")
3
>>> a.find("zzz")
-1
```

```
>>> 'banana'.replace('na','ha')
'bahaha'
```

```
>>> 'No. Okay. Why?'.split('.')
['No', ' Okay', ' Why?']
```

```
>>> "*".join(['red','green','blue'])
'red*green*blue'
```

```
>>> "".join(['red','green','blue'])
'redgreenblue'
```

Παραδείγματα εμπέδωσης (2/2)

```
>>> a='μηλοπιτα'
>>> b='φασκο'
>>> b + a[0:4]
'φασκομηλο'

>>> b[3:] + a[6:]
'κοτα'

>>> a[6:] + a[-4:-2]
'ταπι'
```

```
>>> b[?:?] + a[?:?]
'φασκοπιτα'

b[:] + a[4:]

b + a[-4:]

b[:5] + a[-4:8]
```

Μορφοποίηση συμβολοσειρών για εκτύπωση 1/3

1ος τρόπος: Χρήση του τελεστή μορφοποίησης %

```
>>> Name, age  = "John", 23
>>> print("%s is %d years old" % (Name, age))
John is 23 years old
```

```
>>> a, b = 20, 3
>>> c=a/b
>>> print("%d / %d is %.4f" % (a,b,c))
20 / 3 is 6.6667
```

```
>>> k=127
>>> print ("%d in hex is %X" % (k,k))
127 in hex is 7F
```

`%s` - συμβολοσειρά (ή ο,τιδήποτε μπορεί να αναπαρασταθεί σαν συμβολοσειρά – πχ αριθμοί)

`%d` - Ακέραιοι

`%f` - Δεκαδικοί

`%.<αρ. ψηφίων>f` - Δεκαδικοί με συγκεκριμένο πλήθος ψηφίων

`%x/%X` – Ακέραιοι σε δεκαεξαδική μορφή (πεζά abcdef /κεφαλαία ABCDEF)

Μορφοποίηση συμβολοσειρών για εκτύπωση 2/3

2ος τρόπος: Χρήση της μεθόδου format

```
>>> print("{} is {} years old".format(Name,age))
John is 23 years old

>>> print("{1} is {0} years old".format(age, Name))
John is 23 years old

>>> print("{Name:10} is {age:6.2f} years old".format(Name="John",age=23))
John       is  23.00 years old
```

Όταν υπάρχει *άνω-κάτω τελεία (:)* μετά το όνομα της μεταβλητής, τότε η τιμής της μορφοποιείται σύμφωνα με τις οδηγίες που ακολουθούν. Συνήθως *ο πρώτος αριθμός* που ακολουθεί υποδηλώνει *το πλήθος των θέσεων* που εκχωρούνται για την εκτύπωση της μεταβλητής. Ειδικά για αριθμητικές τιμές μπορεί να ακολουθεί *μια τελεία και ο αριθμός των δεκαδικών ψηφίων* που θέλουμε να τυπωθούν, καθώς και ο χαρακτήρας `d` ή `f` που υποδηλώνει αν η αριθμητική μας τιμή θέλουμε να εκτυπώσουμε είναι αντίστοιχα *ακέραιος* ή *δεκαδικός*.

Πχ στο παραπάνω παράδειγμα το `{Name:10}` υποδηλώνει ότι η τιμή του `Name` θα εκτυπωθεί σε ένα πεδίο 10 θέσεων ενώ το `{age:6.2f}` υποδηλώνει ότι η τιμή του `age` θα εκτυπωθεί σαν δεκαδικός σε ένα πεδίο 6 θέσεων με 2 δεκαδικά ψηφία

Μορφοποίηση συμβολοσειρών για εκτύπωση 3/3

3ος τρόπος: Χρήση f-strings (formatted string literals)

(μορφοποιημένα κυριολεκτήματα συμβολοσειρών)

- *Νεότερος* και *πιο αποδοτικός* τρόπος μορφοποίησης συμβολοσειρών (παρουσιάστηκε στην *Python 3.6*), αποτελεί εξέλιξη της μεθόδου .format
- Δημιουργούνται τοποθετώντας ένα `f` αμέσως *πριν το αρχικό εισαγωγικό* μιας απλής συμβολοσειράς π.χ. `f'Hello'`
- Οι f-strings μέσα στο κείμενο μπορεί να περιέχουν *πεδία* που περικλείονται σε άγκιστρα `{}`
- Το περιεχόμενο κάθε *πεδίου* μπορεί να είναι οποιαδήποτε έκφραση της Python *(πχ απλή μεταβλητή, αριθμητική παράσταση, αποτέλεσμα συνάρτησης…)* όπως πχ. `{12}` ή `{age+1}` ή `{Name}` ή `{Name.upper()}` κλπ.
- Κατά την *εκτέλεση του προγράμματος*, τα *πεδία* των f-strings *επαληθεύονται* και στη θέση τους εμφανίζεται η τιμή τους
- Μπορούμε να μορφοποιήσουμε τα πεδία *(πχ αριθμός δεκαδικών, συνολικό πλάτος πεδίου κλπ.)* χρησιμοποιώντας τους ίδιους τελεστές μορφοποίησης που είδαμε νωρίτερα

Παραδείγματα εκτυπώσεων με f-strings

```
>>> Name, age = 'John', 23
>>> print(f'{Name} is {age} years old')
John is 23 years old
>>> print(f"{Name.upper()} is {age:6.2f} years old")
JOHN is  23.00 years old

>>> s = 'μηλοπιτα'
>>> print(f'Φάγαμε {s[4:8]}κια με {s[:4]}')
Φάγαμε πιτακια με μηλο

>>> print(f'Φάγαμε {len(s)-3} {s[4:8]}κια \nμε {s[:4]}')
Φάγαμε 5 πιτακια
με μηλο
```

Χαρακτήρες Ελέγχου

- Ακολουθία δυο *(ενίοτε και περισσότερων)* χαρακτήρων όπου ο πρώτος είναι το backslash `"\"`
- Έχουν ειδική σημασία η οποία υποδηλώνεται από τον χαρακτήρα που ακολουθεί το `"\"`
- Μας δίνουν την δυνατότητα να προσθέσουμε στο κείμενό μας χαρακτήρες ελέγχου της ροής του όπως tabs, newlines κλπ.
- Οι δυο πιο συνηθισμένοι είναι οι `"\n"` και `"\t"` αλλά υπάρχουν και αρκετοί άλλοι

Χαρακτήρας	Περιγραφή
\a	Bell or alert
\b	Backspace
\e	Escape
\f	Formfeed
\n	Newline
\nnn	Octal notation, where n is in the range 0.7

Χαρακτήρας	Περιγραφή
\r	Carriage return
\s	Space
\t	*Tab*
\v	Vertical tab
\x	Character x
\xnn	Hexadecimal notation, where n is in the range 0.9, a.f, or A.F

Πρόθεμα συμβολοσειράς

String prefix

Χαρακτήρας(*) που μπαίνει *ακριβώς μπροστά* από μια συμβολοσειρά (*χωρίς κενό ενδιάμεσα*) για να δηλώσει ειδικό τρόπο χειρισμού του περιεχομένου της. Τα ευρύτερα χρησιμοποιούμενα string prefixes είναι:

`f"...."` : f-string ή formatted-string: μπορεί να περιέχει *πεδία μορφοποίησης* που περικλείονται σε άγκιστρα `{ }` *(βλ. προηγούμενες διαφάνειες)*

`r"...."` : raw string: οι *χαρακτήρες ελέγχου* (*control characters*) αγνοούνται

πχ. `print(r"Hello\n")` θα τυπώσει `Hello\n`

`u"...."` : Unicode string: Το περιεχόμενό της είναι κωδικοποιημένο σύμφωνα με την κωδικοποίηση *Unicode* και όχι την παλαιότερη ASCII.

Περιττό στην Python 3.x καθώς όλες οι συμβολοσειρές ακολουθούν την κωδικοποίηση Unicode

`b"...."` : bytes string: Το περιεχόμενό της ερμηνεύεται ως σειρά από διαδοχικά bytes και όχι ως Unicode *(θα δούμε παράδειγμα αργότερα)*

(*) *χωρίς διάκριση πεζών-κεφαλαίων*

Τέλος Διάλεξης

Ερωτήσεις;

Τμήματα αυτής της διάλεξης περιέχουν πληροφορίες από πηγές που είναι ελεύθερες στο διαδίκτυο όπως η Βικιπαίδεια και ανοιχτές σημειώσεις παρεμφερών διαλέξεων.