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

HMY01K06
Επιστημονικός Προγραμματισμός με Python

6+1 ερωτήσεις επανάληψης για τη Διάλεξη 04
Συμβολοσειρές

Φθινόπωρο 2025

Ερώτηση 1/7

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)*

Ερώτηση 2/7

Τι είναι οι συμβολοσειρές με *τριπλά* εισαγωγικά (docstrings);

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

print(s)

>>>This
   is
   a
   test
```

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.)*

Ερώτηση 3/7

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
```

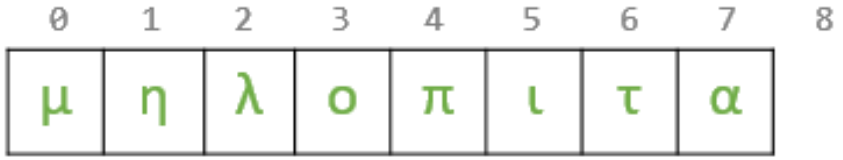
Από τη στιγμή που ορίσουμε μια συμβολοσειρά,
δεν μπορούμε πια να την να αλλάξουμε

Θυμόμαστε ότι:
Οι συμβολοσειρές στην Python είναι *αμετάβλητα αντικείμενα* *immutables*

Ερώτηση 4/7

Πως παίρνουμε ένα *τμήμα* μιας συμβολοσειράς (slicing);

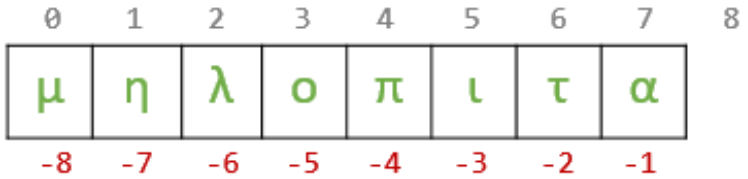
```
>>> s='μηλοπιτα'
>>> len(s)
8
>>> s[0]
'μ'
>>> s[7]
'α'
>>> s[0:4]
'μηλο'
>>> s[4:7]
'πιτ'
>>> s[4:8]
'πιτα'
>>> s[0:8]
'μηλοπιτα'
```



- 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

Ερώτηση 5/7

Πως μπορούμε να προσπελάσουμε τα στοιχεία μιας συμβολοσειράς χρησιμοποιώντας *αρνητικούς δείκτες*;



- 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

Ερώτηση 6/7

Τι είναι και τι υποδηλώνει το *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*

Bonus Ερώτηση 7/7

Πως μπορούμε να δούμε *όλες τις μεθόδους* που υποστηρίζει μια κλάση (εν προκειμένω η κλάση των συμβολοσειρών `str`);

```
>>> for x in dir(str):
...     if x[0:2] != '__': print (x)
...
...
capitalize
casefold
center
count
encode
endswith
expandtabs
find
format
format_map
index
isalnum
isalpha
isascii
isdecimal
isdigit
isidentifier
islower
isnumeric
isprintable
isspace
istitle
isupper
join
ljust
lower
lstrip
maketrans
partition
removeprefix
removesuffix
replace
rfind
rindex
rjust
rpartition
rsplit
rstrip
split
splitlines
startswith
strip
swapcase
title
translate
upper
zfill
```

Αν δεν θυμόμαστε το ακριβές όνομα μιας κλάσης ώστε να το βάλουμε ως όρισμα στη συνάρτηση `dir()` μπορούμε εύκολα να το βρούμε μέσω της συνάρτησης `type()`

```
>>> type('aaa')
<class 'str'>
>>> type(4)
<class 'int'>
>>> type([1,2,3])
<class 'list'>
```