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

HMY01K06
Επιστημονικός Προγραμματισμός με Python

6+1 ερωτήσεις επανάληψης για τη Διάλεξη 03

Τελεστές, Είσοδος / Έξοδος

Δομές Επιλογής και Επανάληψης

Φθινόπωρο 2025

Ερώτηση 1/7

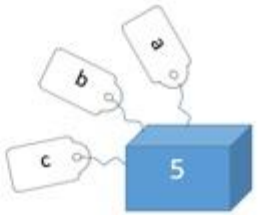
Ποιοι είναι δυο *εναλλακτικοί τρόποι χρήσης*
του τελεστή εκχώρησης;

- Εκχώρηση του *ίδιου αντικειμένου* σε *περισσότερους από έναν προσδιοριστές*

  (στην ουσία μαρκάρισμα του ίδιου αντικειμένου με περισσότερους από έναν προσδιοριστές)

```
a = b = c = 5
```

οι προσδιοριστές a, b και c παίρνουν όλοι την τιμή 5



- Ταυτόχρονη εκχώρηση *διαφορετικών τιμών* σε *διαφορετικούς προσδιοριστές*

```
a, b, c = 5, 12.4, 'Hello'
```

Αποτελεί συντόμευση του `a = 5` `b = 12.4` `c = 'Hello'`

Παρεμπιπτόντως: `x < y < z` σημαίνει `(x < y) and (y < z)` και επαληθεύεται από αριστερά προς τα δεξιά

Ερώτηση 2/7

Τι ακριβώς είναι η *υπερφόρτωση τελεστών*;

Υπερφόρτωση Τελεστών *(operator overloading)* ονομάζεται η ιδιότητα ορισμένων τελεστών να αποκτούν *διαφορετική λειτουργικότητα* ανάλογα με το *περιβάλλον χρήσης* τους *(δηλαδή ανάλογα με τους τελεστέους πάνω στους οποίους επιδρούν)*

Παράδειγμα 1: ο τελεστής της πρόσθεσης (+)

- Σε "κανονική" χρήση, προσθέτει δυο αριθμούς 3 + 5 ⇨ 8
- Μπορούμε όμως να τον χρησιμοποιήσουμε και για να συνενώσουμε συμβολοσειρές

  'ΚΑΛΗ' + 'ΜΕΡΑ' ⇨ 'ΚΑΛΗΜΕΡΑ'

- Ή ακόμα για να συνενώσουμε δυο λίστες

  [1, 4, 2, 9] + [5, 2.7, 4] ⇨ [1, 4, 2, 9, 5, 2.7 ,4]

Η Python μας δίνει τη δυνατότητα να υπερφορτώνουμε και οι ίδιοι τους τελεστές της ώστε να εξυπηρετούν τις δικές μας ανάγκες

Αυτό γίνεται ορίζοντας κατάλληλα ειδικές κλάσεις για αυτή τη δουλειά.

Παράδειγμα 2: ο τελεστής του πολλαπλασιασμού (*)

- Σε "κανονική" χρήση, πολλαπλασιάζει δυο αριθμούς 3 * 5 ⇨ 15
- Μπορούμε όμως να τον χρησιμοποιήσουμε και στις συμβολοσειρές ως εξής

  'χα ' * 5 ⇨ 'χα χα χα χα χα '

Ερώτηση 3/7

Τι είναι οι παράμετροι `sep` και `end` στη συνάρτηση `print();`

Είναι *ορίσματα-κλειδιά* (*keyword arguments*) που μπαίνουν προαιρετικά στο τέλος της λίστας των ορισμάτων της **`print`** και καθορίζουν την συμπεριφορά της

Η παράμετρος **`sep`** καθορίζει τον/τους *διαχωριστικούς χαρακτήρες* που θέλουμε να εμφανίζονται *ανάμεσα* στα διαδοχικά ορίσματα που εκτυπώνονται. Αν δεν υπάρχει, νοείται ως ένας κενός χαρακτήρας `' '`

```
>>> print('one', 'two', 'three', sep='/') Τα αποτελέσματα διαχωρίζονται με /
one/two/three

>>> print('one', 'two', 'three', sep='...') διαχωρίζονται με ...
one...two...three
```

Η παράμετρος **`end`** καθορίζει έναν ή περισσότερους χαρακτήρες που θέλουμε να εκτυπωθούν *στο τέλος της γραμμής*, αφού έχουν πια εκτυπωθεί όλες οι τιμές. Αν δεν υπάρχει, νοείται ως ο χαρακτήρας `'\n'` *NEWLINE - κατέβασμα στην επόμενη γραμμή*

```
print('Πρωί', end ='')
print('Βράδι')
ΠρωίΒράδι
```

Ερώτηση 4/7

Τι ακριβώς είναι ενα μπλοκ εντολών στην Python;

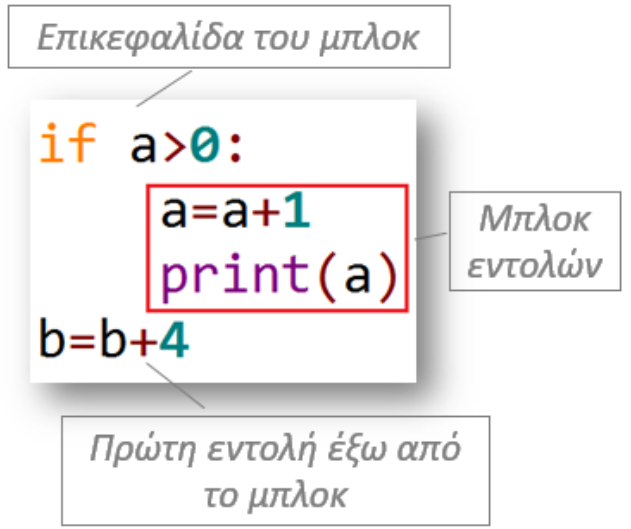
Ενα *μπλοκ εντολών* στην Python είναι μια *ακολουθία εντολών* που αποτελεί μια *αυτόνομη ενιαία οντότητα.*

Παραδείγματα:

— ο *κώδικας* μιας συνάρτησης

— οι *εντολές* μέσα σε μια επανάληψη (for, while κλπ.)

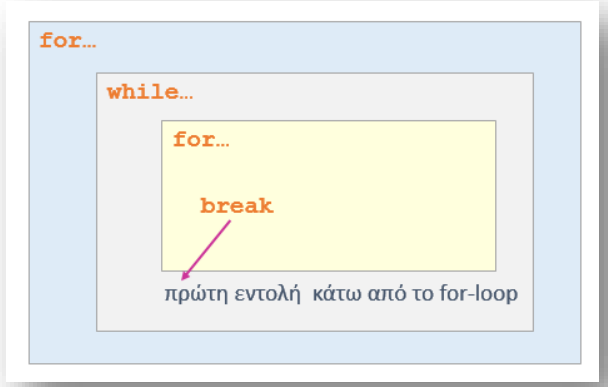
— οι *εντολές* που ακολουθούν ένα if ή ένα else



- Ένα μπλοκ εντολών ορίζεται από εσοχές (indentations)
- Συνήθως η εσοχή γίνεται με *4 κενά* ή με το *πλήκτρο tab*, *αλλά γενικά προτείνεται η χρήση 4 κενών για συνέπεια*
- Οι εντολές που ανήκουν στο ίδιο μπλοκ πρέπει να έχουν το ίδιο επίπεδο εσοχής

Αντίθετα με άλλες γλώσσες προγραμματισμού που χρησιμοποιούν σύμβολα όπως {} ή begin/end για τον καθορισμό των μπλοκ, *η Python βασίζεται αποκλειστικά στις εσοχές*

Ερώτηση 5/7



Τι ακριβώς κάνει η εντολή `break`
και τι η εντολή `continue` σε μια επανάληψη;

- Η εντολή `break` *σπάει εντελώς την επανάληψη* και η εκτέλεση του προγράμματος μεταφέρεται στην πρώτη εντολή *έξω* από το μπλοκ της επανάληψης

  Αν υπάρχει ξεχωριστό μπλοκ `else` στο τέλος της επανάληψης *(κάτι που γίνεται στην Python)* αυτό δεν εκτελείται όταν η επανάληψη σπάσει με `break`

  Η εντολή `break` μιας *εσωτερικής επανάληψης* ισχύει μόνο ***τοπικά*** δηλαδή μόνο στο δικό της επίπεδο βάθους - οι *εξωτερικές* από αυτήν επαναληπτικές δομές *δεν επηρεάζονται*

- Η εντολή `continue` *διακόπτει την εκτέλεση του τρέχοντος κύκλου της επανάληψης* και ξεκινάει τον επόμενο κύκλο της επανάληψης

Ερώτηση 6/7

Τι γνωρίζετε για την συνάρτηση range()

Η συνάρτηση **range()** είναι μια *γεννητορική συνάρτηση* της Python *(περισσότερα σε επόμενες διαλέξεις)* η οποία χρησιμοποιείται για την παραγωγή ακολουθιών ακεραίων και ορίζεται ως εξής:

```
range([start], stop, [step])
```

- Τα ***start***, ***stop*** και ***step*** είναι ακέραιες εκφράσεις
- Τα ***start*** και ***step*** είναι προαιρετικά
- Παράγει διαδοχικούς ακέραιους μέχρι το ***stop*** *χωρίς αυτό να συμπεριλαμβάνεται (δηλ. μέχρι το stop–1)*
- Αν δεν υπάρχει το προαιρετικό ***start*** η συνάρτηση ξεκινάει *από το 0*
- Αν δεν υπάρχει το προαιρετικό ***step***, τότε ως βήμα παραγωγής θεωρείται το 1

```
range(stop)
range(start, stop)
range(start, stop, step)
```

`range(4)`	Παράγει την ακολουθία αριθμών `0,1,2,3` *ισοδύναμο με* ***`range(0,4)`***
`range(1,7)`	Παράγει την ακολουθία αριθμών `1,2,3,4,5,6`
`range(1,7,2)`	Παράγει την ακολουθία αριθμών `1,3,5`
`range(8,5,-1)`	Παράγει την ακολουθία αριθμών `8,7,6`
`range(8,5)`	Δεν παράγει τίποτε *(γιατί;)*

Bonus Ερώτηση 7/7

Τι είναι ο τριμερής τελεστής `if-else` στην Python;

ternary operator

Είναι ενας τρόπος γραφής του `if-else` σε μια γραμμή

```
<εντολή 1> if <συνθήκη> else <εντολή 2>
```

Αν η συνθήκη είναι *αληθής* εκτελείται η *εντολή 1* αλλιώς η *εντολή 2*

```
grade = 7.3
if grade >= 5:
    result = 'pass'
else:
    result = 'fail'
print(result)
```

Εναλλακτικά:

```
grade = 7.3
result = 'pass' if grade >= 5 else result = 'fail'
print(result)
```

Θυμόμαστε ότι στην Python *δεν υπάρχει το keyword then*