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

HMY01K06
Επιστημονικός Προγραμματισμός με Python

6+1 ερωτήσεις επανάληψης για τη Διάλεξη 07

Συναρτήσεις , Δομοστοιχεία, Πακέτα

Φθινόπωρο 2025

Ερώτηση 1/7

Υπάρχουν δυο περιπτώσεις *εξόδου (τερματισμού)* μιας συνάρτησης
Ποιες είναι αυτές;

- όταν εκτελέσει την εντολή `return` κάπου ενδιάμεσα ή στο τέλος του μπλοκ εντολών της

  αν το `return` ακολουθείται από έναν προσδιοριστή ή μια έκφραση (πχ `return x+5`) τότε το αποτέλεσμα αυτό επιστρέφεται στον κώδικα που κάλεσε την συνάρτηση. Διαφορετικά η συνάρτηση επιστρέφει `None`

- όταν η εκτέλεση φτάσει στην *τελευταία γραμμή* του μπλοκ εντολών της οπότε και επιστρέφει `None`

Ερώτηση 2/7

Γιατί άραγε να υπάρχουν συναρτήσεις
που δεν επιστρέφουν τίποτα;

Για δυο τουλάχιστον λόγους:

- Μια συνάρτηση μπορεί να χρησιμοποιείται πχ. για να *εκτυπώσει* κάτι στην οθόνη, ή να *γράψει κάτι σε ένα αρχείο*

- Μια συνάρτηση μπορεί να έχει εσωτερικά δυνατότητα πρόσβασης *(και άρα επεξεργασίας)* σε συγκεκριμένα *αντικείμενα* του μπλοκ του κώδικα που την έχει καλέσει

  Στην περίπτωση αυτή η επεξεργασία αυτών των αντικειμένων *γίνεται εσωτερικά στη συνάρτηση*, οπότε δεν είναι απαραίτητο να επιστραφεί κάτι όταν αυτή ολοκληρωθεί

Ερώτηση 3/7

Οι τύποι των ορισμάτων μιας συνάρτησης - π.χ. `f(a, b, c)`
μπορούν να αλλάζουν από κλήση σε κλήση.
Σωστό ή λάθος;

Σωστό! Να ένα παράδειγμα.

```
def print_a_sequence(any_sequence):
    for x in any_sequence:
        print(x, end=' ')
```

```
# Κλήση της συνάρτησης με όρισμα μια λίστα
fruits = ["apple", "banana", "cherry"]
print_a_sequence(fruits)

Θα εκτυπώσει apple banana cherry
```

```
# Κλήση με όρισμα μια συμβολοσειρά
print_a_sequence("This")
Θα εκτυπώσει T h i s
```

```
# Κλήση με όρισμα μια ακολουθία τιμών
print_a_sequence(range(6,14,2))
   Θα εκτυπώσει 6 8 10 12
```

```
# Όμως: κλήση με όρισμα έναν ακέραιο
print_a_sequence(48)
TypeError: 'int' object is not iterable
```

Ερώτηση 4/7

Ποιοι είναι οι *τέσσερεις* διαφορετικοί τύποι ορισμάτων μιας συνάρτησης;

1. *Ορίσματα θέσης* (*positional arguments*) – υπάρχουν δυο ειδών:
   - *Τυπικά ή απαιτούμενα ή* ορίσματα θέσης (*required positional arguments*)
   - Ορίσματα θέσης *με προκαθορισμένες τιμές* (*default positional arguments*)
2. *Ορίσματα-κλειδιά* (*keyword arguments*)
3. *Ομαδοποιημένα ορίσματα θέσης* μεταβλητού *(απροσδιόριστου)* πλήθους (*args)
4. *Ομαδοποιημένα ορίσματα-κλειδιά* μεταβλητού *(απροσδιόριστου)* πλήθους (**kwargs)

Ερώτηση 5/7

Ποιος είναι ο γενικότερος ορισμός μιας συνάρτησης που περιλαμβάνει τόσο τυπικά όσο και ομαδοποιημένα ορίσματα;

```
def function_name( [arg1], [arg2], . . . [argk], [*args], [**kwargs] ):
```

Όπου

- [arg$_1$], [arg$_2$], . . . [arg$_k$] τα *τυπικά ορίσματα θέσης* (παράμετροι) της συνάρτησης
- [*args] *μια πλειάδα ή λίστα* που περιέχει ομαδοποιημένα *ορίσματα θέσης*
- [**kwargs] ένα *λεξικό* που περιέχει ομαδοποιημένα *ζεύγη από ορίσματα-κλειδιά*

και όπως είπαμε και νωρίτερα:

- Σε κάθε συνάρτηση δεν μπορεί να ορίζονται περισσότερα από ένα **args* και ένα ***kwargs*
- Η δήλωση **args* προηγείται πάντα της δήλωσης ***kwargs*

Ερώτηση 6/7

Τι ακριβώς πληροφορία μας παρέχουν οι συναρτήσεις εξέτασης ονοματοχώρων

`dir()` `locals()` και `globals()`;

`dir()`: Αν κληθεί χωρίς παράμετρο, τότε επιστρέφει μια λίστα με τα ονόματα όλων των προσδιοριστών που είναι διαθέσιμοι στον τοπικό ονοματοχώρο

`dir()`: Αν κληθεί με παράμετρο το όνομα ενός δομοστοιχείου (module) ή μιας βιβλιοθήκης, τότε επιστρέφει τα ονόματα όλων των προσδιοριστών που περιέχονται σε αυτό το στοιχείο *(δηλαδή όλα τα ονόματα στον τοπικό ονοματοχώρο του)*

`locals()` / `globals()`: Επιστρέφουν αντίστοιχα ένα λεξικό με περιεχόμενο τα ονόματα του τοπικού / καθολικού ονοματοχώρου με τις αντίστοιχες τιμές τους.

Τα περιεχόμενα των λεξικών αυτών δεν πρέπει να τροποποιούνται

Οι συναρτήσεις `locals()` και `globals()` δεν δέχονται ορίσματα στις παρενθέσεις τους

Bonus Ερώτηση 7/7

Τι ακριβώς είναι οι "ανώνυμες" συναρτήσεις στην Python
και ποια η χρησιμότητά τους;

Είναι συναρτήσεις *χωρίς όνομα* με *πολύ απλό κώδικα* σε μια γραμμή *(χωρίς if και επαναλήψεις)* οι οποίες ορίζονται χρησιμοποιώντας τη λέξη-κλειδί `lambda`

```
>>> (lambda x: x**2+1) (4)
17
>>>
```

Συνάρτηση

Όρισμα

```
>>> z = lambda x: x**2+1
>>> z(4)
17
```

```
>>> lambda x: x**2+1
<function <lambda> at 0x0000009D36F7B920>
>>> _(4)
17
```

Θυμόμαστε ότι: η κάτω παύλα (_) στο διαδραστικό περιβάλλον της Python περιέχει το αποτέλεσμα της πιο πρόσφατης έκφρασης.

Αν μια συνάρτηση χρησιμοποιείται μόνο μια φορά ή για περιορισμό αριθμό κλήσεων τότε μια ανώνυμη συνάρτηση είναι συντακτικά ελαφρύτερη από μια συνάρτηση με όνομα