ΠΕΡΙΕΧΟΜΕΝΑ:

Νίκος Θ.
Ασημένιος Χορηγός Μαθήματος

Αριστείδης Φ.
Χάλκινος Χορηγός Μαθήματος

ΜΑΘΗΜΑ 14: Modules και Εμβέλεια

1. Modules

python 3 psounis

- Τώρα που τα προγράμματα μας "ξεφεύγουν" σε μέγεθος:
  - Οργανώνουμε τον κώδικα σε διαφορετικά αρχεία, τα οποία τα λέμε modules.
  - **Module είναι ένα αρχείο κώδικα το οποίο το ενσωματώσουμε (import) στο πρόγραμμα μας** ώστε:
    - Να αποκρύπτουμε λεπτομέρειες της υλοποίησης
    - Να μπορούμε να το ξαναχρησιμοποιούμε σε άλλα προγράμματα
    - Να τα διαμοιραζόμαστε με άλλους προγραμματιστές.
- Υπάρχουν τρεις τρόποι για να κάνουμε την ενσωμάτωση:

1ος τρόπος: Ενσωματώνουμε όλο το module
- Σύνταξη: **import module_name**
- Χρήση περιεχομένων: **module_name.func_name** (με την τελεία)

Παράδειγμα 1: module.py & import1.py

```
# module.py
def func():
   print("Hello!")
```

```
# import1.py
import module

module.func()
```

Παρατήρησεις:
- Τα ίδια ισχύουν και για την ενσωμάτωση απλών μεταβλητών
- Αλλά και για την ενσωμάτωση κλάσεων (επόμενο μάθημα)

2ος τρόπος: Ενσωματώνουμε μόνο κάποια συνάρτηση:
- Σύνταξη: **from module_name import func_name**
- Χρήση συνάρτησης: **func_name** (χωρίς την τελεία)

Παράδειγμα 2: import2.py

```
from module import func

func()
```

3ος τρόπος: Ενσωματώνουμε με συνώνυμο:
- Σύνταξη: **from module_name import func_name as synonym**
- Χρήση συνάρτησης: **synonym**

Παράδειγμα 3: import3.py

```
from module import func as f

f()
```

- Το module πρέπει να βρίσκεται στον ίδιο φάκελο με το κυρίως πρόγραμμα.
- Μπορούμε να οργανώσουμε πολλά modules σε έναν φάκελο φτιάχνοντας ένα πακέτο (package) [σε επόμενα μαθήματα..]

- Οι **εσωτερικές συναρτήσεις (inner functions):**
  - Είναι **συναρτήσεις που ορίζονται μέσα σε συναρτήσεις**
  - Και μπορουν να κληθούν μόνο από τη συνάρτηση μέσα στην οποία έχουν οριστεί.

Παράδειγμα 4: inner.function.py

```
def f():
    def g():
        print("Hello!")
    g()
    g()

f()
```

Η g() είναι εσωτερική συνάρτηση στην f()
Σημειώστε ότι δεν μπορούμε να καλέσουμε την g() π.χ. από το κυρίως πρόγραμμα.

Οι εσωτερικές συναρτήσεις είναι χρήσιμες όταν:

- Για να προφυλάξουμε τις εσωτερικές συναρτήσεις από ότι συμβαίνει εκτός της συνάρτησης.
- Για να κρατήσουμε τον κώδικα καθαρό, χωρίς περιττούς ορισμούς συναρτήσεων
- Σαν περίβλημα σε πιο περίπλοκες συναρτήσεις (βλ. παράδειγμα 5 και άσκηση 1) και στα εργοστάσια συναρτήσεων (επομ.διαφ.)

Παράδειγμα 5: fibonacci.py

```
def fibonacci(n):
    def fib_rec(n):
        if n == 0:
            return 0
        elif n == 1:
            return 1
        else:
            return fib_rec(n - 1) + fib_rec(n - 2)

    if n < 0:
        return None
    else:
        return fib_rec(n)


for i in range(11):
    print(f"fibonacci({i})={fibonacci(i)}")
```

Άσκηση 1:
Τροποποιήστε τον ορισμό της συνάρτησης της δυαδικής αναζήτησης (μάθημα 12, άσκηση 8), ώστε το πρωτότυπό της να μην έχει παραμέτρους ορίων του πίνακα (Χρησιμοποιήστε εσωτερική συνάρτηση)

- Τα **εργοστάσια συναρτήσεων (function factories):**
  - Είναι **συναρτήσεις που κατασκευάζουν συναρτήσεις.**
  - Συγκεκριμένα, **επιστρέφουν μια εσωτερική συνάρτηση, στην οποία έχουν πάρει συγκεκριμένες τιμές κάποιες μεταβλητές.**

Παράδειγμα 5: function.factory.py

```
def factory_power(power):
    def nth_power(number):
        return number ** power

    return nth_power


square = factory_power(2)
print(square(4))

cube = factory_power(3)
print(cube(4))
```

Η factory_power:

- Πρώτα ορίζει μια συνάρτηση(nth_power) ορίζει μια συνάρτηση, στην οποία η μεταβλητή power έχει αρχικοποιηθεί με την τιμή του ορίσματος.
- Έπειτα την επιστρέφει.

Παρατηρήσεις:

- Η πλήρης κατανόηση των εργοστασίων συναρτήσεων απαιτεί γνώσεις αντικειμενοστρεφούς προγραμματισμού σε Python (μαθήματα 16-18)
- Ωστόσο μπορούμε να παρατηρήσουμε με το ακόλουθο παράδειγμα:

```
def f():
    print("Hello!")

print(type(f))
```

- ότι και οι συναρτήσεις είναι αντικείμενα τύπου “function” και έτσι έχουν χρήσεις που είναι αντίστοιχες των συνήθων μεταβλητών.
- (Περισσότερα στα επόμενα μαθήματα)

Άσκηση 2:

Ορίστε το εργοστάσιο δευτεροβάθμιων πολυωνύμων, δηλαδή συναρτήσεων της μορφής $f(x) = ax^2 + bx + c$

Παράδειγμα κλήσης:

```
pol = factory(1,1,1)
print(pol(1))
```

που θα τυπώνει 3 (αφού $\text{pol}(x) = x^2 + x + 1$)

- Με τον όρο **εμβέλεια (scope)** ενός ονόματος (π.χ. μεταβλητής, συνάρτησης, αντικειμένου) εννοούμε:
  - Την περιοχή του προγράμματος στην οποία έχουμε πρόσβαση σε αυτό το όνομα.
- Υπάρχουν οι ακόλουθες εμβέλειες:
  - **Τοπική Εμβέλεια σε Συνάρτηση (local scope):**
    - Ονόματα που έχουν οριστεί στη συνάρτηση
    - Είναι ορατά μόνο στον κώδικα της συνάρτησης
    - Δημιουργούνται όταν γίνεται κλήση της συνάρτησης
  - **Εξωτερική Εμβέλεια (enclosing scope):** Η εμβέλεια σε μία συνάρτηση που περιέχει εσωτερικές συναρτήσεις.
    - Ονόματα που έχουν οριστεί στην συνάρτηση που περιέχει εσωτερικές συναρτήσεις
    - Ορατά από τη συνάρτηση και τις εσωτερικές της συναρτήσεις
  - **Καθολική Εμβέλεια (global scope):**
    - Ονόματα που ορίζονται στο κυρίως πρόγραμμα (ή γίνονται import σε αυτό)
    - Ορατά παντού στον κώδικα.
  - **Εμβέλεια ενσωματωμένων στοιχείων (built-in scope)**
    - Λέξεις-Κλειδιά, ενσωματωμένες συναρτήσεις κ.α.
    - Επίσης ορατά παντού στον κώδικα.
- **Προσοχή!** Σε περίπτωση συγκρούσεων ονομάτων, επικρατεί το όνομα που βρίσκεται πιο πάνω στην παραπάνω ιεραρχία.

Παράδειγμα 6: scope.py

```
def outer():
    x = 3
    def inner():
        x=4
        print("inner x= " + str(x))
    inner()
    print("outer x= " + str(x))

x = 2
print("global x= " + str(x))
outer()
```

(Συμβουλευθείτε και το βίντεο για το παράδειγμα αυτό.

Παράδειγμα 7: built.in.py

```
def max(x,y):
    return x>y

print(max(1,2,3))
```

Υπενθύμιση:

- Τα ονόματα δημιουργούνται στις αναθέσεις, όταν κάνουμε import, όταν ορίζουμε μια συνάρτηση, στα ορίσματα συναρτήσεων και στον ορισμό των κλάσεων (Μαθ.16)

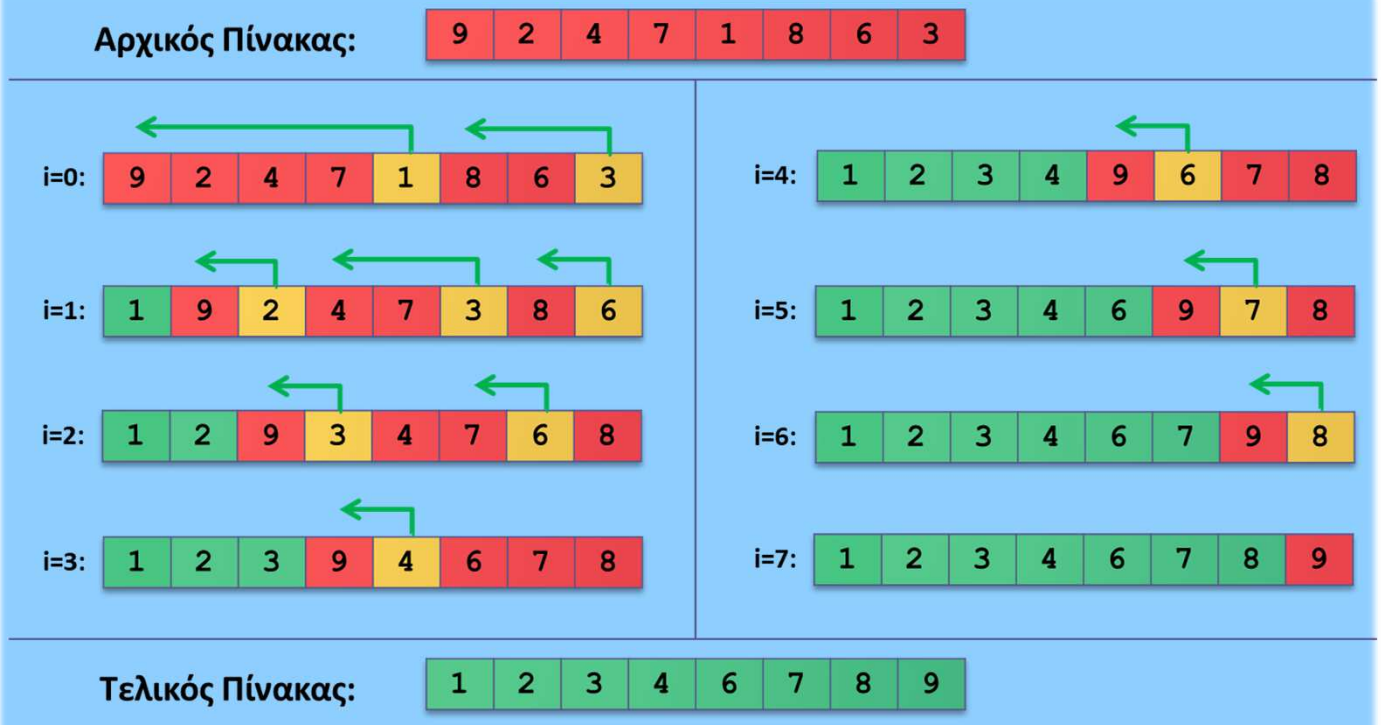
ΜΑΘΗΜΑ 14: Modules και Εμβέλεια Algorithm: Bubble Sort (Ταξινόμηση Φυσαλίδας) python 3 psounis

Συνεχίζουμε με το πρόβλημα της ταξινόμησης, με έναν ακόμη αλγόριθμο που το επιλύει.

Το **σκεπτικό της ταξινόμησης φυσαλίδας (bubble sort)** είναι:

- Έχοντας ταξινομήσει ήδη τις θέσεις 0..i-1
- Από το τέλος του πίνακα (θέση N-1) έως τη θέση i το "πιο ελαφρύ" στοιχείο (το μικρότερο) ανεβαίνει προς τα πάνω (το στοιχείο ανταλάσσει τη θέση του με το στοιχείο στα αριστερά του, αν έχει μικρότερη τιμή από αυτό.)

Παράδειγμα Εκτέλεσης:



Άσκηση 3:

Υλοποιήστε τον αλγόριθμο Bubble Sort (σε μία συνάρτηση)

Άσκηση 4:

- Κατασκευάστε ένα 2Δ 5x5 πίνακα.
- Κατασκευάστε μία συνάρτηση που να ταξινομεί τα στοιχεία κάθε γραμμής σε αύξουσα σειρά
- Κατασκευάστε μία συνάρτηση που να ταξινομεί τα στοιχεία κάθε στήλης σε φθίνουσα σειρα.

Άσκηση 5.1: Module φοιτητών

Μεταφέρετε τη λίστα και τις συναρτήσεις των μαθητών σε ένα module με όνομα mod_pupils.

Άσκηση 5.2: Καθηγητές

Ενσωματώνουμε πληροφορία στο σύστημα για τους καθηγητές του σχολείου. Ένας καθηγητής ορίζεται από τα εξής στοιχεία:

- teacher_id: Κωδικός, μοναδικός για κάθε καθηγητή
- name: Όνομα
- surname: Επώνυμο

Κατασκευάστε module με όνομα mod_teachers το οποίο θα περιέχει τις τυπικές CRUD ενέργειες:

- create_teacher: Παίρνει ως παράμετρους τα στοιχεία (όνομα-επώνυμο) και προσθέτει τον καθηγητή. Ελέγχει αν ο καθηγητής υπάρχει ήδη.
- read_teacher: Δέχεται το id του καθηγητή και επιστρέφει το λεξικό που τον αντιπροσωπεύει.
- update teacher: Δέχεται το id του καθηγητή, το κλειδί του πεδίου και την νέα τιμή του πεδίου και κάνει την ενημέρωση.
- delete_teacher: Δέχεται το id του καθηγητή και τον διαγράφει από τη λίστα καθηγητών.

Ένας καθηγητής θα αποθηκεύεται ως λεξικό. Το σύνολο των καθηγητών θα αποθηκεύεται σε μία λίστα με όνομα teachers.

Άσκηση 5.3: Μενού

Επεκτείνετε το μενού με 4 νέες επιλογές:

- Εισαγωγή καθηγητή: Διαβάζει από την είσοδο και καλεί τη create_teacher
- Διάβασμα καθηγητή: Διαβάζει από την είσοδο το id και τυπώνει τα στοιχεία του καθηγητή
- Ενημέρωση καθηγητή: Διαβάζει από την είσοδο το id και ενημερώνει διαδοχικά τα στοιχεία του.
- Διαγραφή καθηγητή: Διαβάζει από την είσοδο το id και διαγράφει τον καθηγητή.

"If the implementation is easy to explain, it may be a good idea."

Zen of Python #18