ΠΕΡΙΕΧΟΜΕΝΑ:

1. Frozen Sets
2. Μνήμη
    1. Κλάσεις και Αντικείμενα
    2. Αναπαράσταση Απλών Τύπων
    3. Αναπαράσταση Σύνθετων Τύπων
3. Debugging Part 2: Googling..
4. Game Project: Κρεμάλα
5. Data Project: CRUD - Create

Κώστας Μ.

Σμαραγδένιος Χορηγός Μαθήματος

Νίκος Θ.

Ασημένιος Χορηγός Μαθήματος

- **Το Frozen Set** είναι ένα **immutable σύνολο**
  - Έχει ακριβώς ίδια χρήση με το σύνολο
  - Με τη διαφορά ότι είναι immutable (δεν μπορεί να τροποποιηθεί)
  - (Συνεπώς δεν υποστηρίζει μεθόδους για τροποποίηση (π.χ. προσθήκη και διαγραφή)

- **Δήλωση Frozen Set**
  - Ορίζουμε ένα frozen set χρησιμοποιώντας κάποιο όνομα μεταβλητής και ορίζοντας τη μετατροπή κάποιου collection σε frozen set.
  - (Δεν υπάρχει δηλαδή κάποιος ειδικός συμβολισμός για frozen sets)

Παράδειγμα 1: string.definition.py
Στο παράδειγμα βλέπουμε τον ορισμό τρίων frozen sets:

```
empty = frozenset()
frozen_set_int = frozenset({1, 2, 3})
fs_in_set = {frozenset(), frozenset({1,2})}
print(empty)
print(fs_in_set)
print(type(frozen_set_int))
```

- **Συμπεριφορά:**
  - Ως immutable μπορεί να είναι μέρος σε σύνολα ή κλειδί σε λεξικά.
  - Υποστηρίζει πράξεις όπως η ένωση και η τομή (βλ. σύνολα)

Άσκηση 1:
Κατασκευάστε ένα σύνολο με όλα τα υποσύνολα με 2 στοιχεία του {1,2,…,N}

Άσκηση 2:
Κατασκευάστε ένα σύνολο με όλα τα υποσύνολα του {1,2,…,N}

[Υπόδειξη: Αυξημένης Δυσκολίας Άσκηση. Συμβουλευθείτε το βίντεο]

Κλάσεις

Οι κλάσεις είναι το βασικό εργαλείο για να ομαδοποιήσουμε:

- Δεδομένα
- Συμπεριφορά επί των δεδομένων (μεθόδους)

Ήδη κάναμε το εξής στα προηγούμενα μαθήματα:

- Ορίσαμε έναν τύπο δεδομένων (π.χ. τη λίστα) που ουσιαστικά είναι ένας τρόπος αναπαράστασης δεδομένων
  - και έπειτα είδαμε τις μεθόδους της
- Ουσιαστικά η λίστα είναι μία **κλάση (δεδομένα + μέθοδοι)**
- και ομοίως κλάση είναι και όλοι οι τύποι δεδομένων που είδαμε (σύνολο, λεξικό κ.λπ.)
- και παρατηρούσαμε ότι κάθε μεταβλητή ήταν κάποιας κλάσης (με τη κλήση της συνάρτησης type(μεταβλητή))

Όταν πηγαίνουμε και ορίζουμε μία συγκεκριμένη "μεταβλητή" του τύπου δεδομένων:

- Θα λέμε ότι πλέον ορίζουμε ένα **αντικείμενο** της κλάσης
- Στο πρόγραμμά μας μπορούμε να έχουμε πολλά αντικείμενα της κλάσης.

Στην πραγματικότητα, όλοι οι τύποι δεδομένων στην Python είναι κλάσεις!

- Ακόμη και οι ακέραιοι ή οι πραγματικοί
- και μάλιστα έχουν αρκετές μεθόδους.

Παράδειγμα 2: data.types.py

```
print(type(1))
print(type(1.2))
print(type(1+2j))
print(type(True))
print(type("Hello!"))
print(type({1,2}))
print(type((1,2)))
print(type({1:2}))
print(type(range(3)))
print(type(frozenset({1,2})))
```

Άσκηση 3:

Ορίστε 2 μιγαδικούς αριθμούς, προσθέστε τους και τυπώστε το αποτέλεσμα.
Έπειτα πολλαπλασιάστε τους και τυπώστε το αποτέλεσμα.
Επίσης παρατηρήστε τη μέθοδο conjugate() που επιστρέφει το συζυγή του μιγαδικού αριθμού.

Παρατήρηση:

Σε επόμενα μαθήματα θα μάθουμε να ορίζουμε και δικές μας κλάσεις.

Αναπαράσταση Απλών Τύπων

- (Απλούς τύπους θεωρούμε τους int, float, bool και string)
- Τα 5, 3.14, True, “Hello” είναι αντικείμενα των αντίστοιχων κλάσεων.

Εναλλακτικά με την ορολογία που χρησιμοποιούμε ως τώρα:

- **ΜΕΤΑΒΛΗΤΗ = ΤΙΜΗ(ΤΥΠΟΣ)**

Θα χρησιμοποιούμε την ορολογία:

- **ΟΝΟΜΑ = ΑΝΤΙΚΕΙΜΕΝΟ(ΚΛΑΣΗ)**

Προσοχή όμως!

- Στην πραγματικότητα η Python λειτουργεί με ιδιαίτερο τρόπο.
- Έχει δύο “χώρους” στη μνήμη
  - Το χώρο των ονομάτων (που είναι τα ονόματα των μεταβλητών)
  - Το χώρο των αντικειμένων
    - Βλέπουμε τη διεύθυνση μνήμης που πιάνει το αντικείμενο με τη συνάρτηση **id()**
- Και όταν γράφουμε την εντολή καταχώρησης
  - Το όνομα, σχετίζεται με το συγκεκριμένο αντικείμενο.
  - Αλλά το αντικείμενο υπάρχει μόνο μία φορά στη μνήμη!
  - και λέμε ότι κάνουμε **bind** το όνομα στο αντικείμενο.
- **Λέμε ότι τα ονόματα των μεταβλητών είναι αναφορές (references)** στα αντικείμενα, με την έννοια ότι δεν έχουν κάποιο χώρο μνήμης αλλά αναφέρονται στα αντικείμενα (που έχουν χώρο μνήμης.

Παράδειγμα 3: id.py

```
x = 5
print(f"x: {id(x)}, 5: {id(5)}")
y = 5
print(f"y: {id(y)}")
x = 6
print(f"x: {id(x)}, 6: {id(6)}")
y = x
print(f"x: {id(x)}, y: {id(y)}")
```

Όλοι οι απλοί τύποι είναι immutable

- Τι σημαίνει όμως πραγματικά immutable;
- Immutable σημαίνει ότι ένα αντικείμενο δεν μπορεί να αλλάξει, χωρίς να αλλάξει η ταυτότητα τού (id).
- Δηλαδή π.χ. το αντικείμενο “6” δεν μπορεί να τροποποιηθεί και να γίνει “7”

Παράδειγμα 4: id2.py

```
x = "Hello!"
print(f"x: {id(x)}")
x = "Hello World!"
print(f"x: {id(x)}")
```

Προσοχή, ότι το αντικείμενο αλλάζει, ενώ το όνομα παραμένει το ίδιο.

Αναπαράσταση Σύνθετων Τύπων

- (Σύνθετους θεωρούμε τους υπόλοιπους τύπους πλην των απλών)

Μελετάμε π.χ. ξανά τη λίστα

- Μία λίστα είναι μια διατεταγμένη συλλογή από ονόματα (που έχουν δεθεί σε αντικείμενα)
- **Η λίστα είναι mutable, άρα μπορεί να αλλάξει χωρίς να πρέπει να αλλάξει η ταυτότητά της.**

Παράδειγμα 5: id3.py

```
x = [1, 2]
print(f"x:{id(x)}, x[0]:{id(x[0])}, x[1]:{id(x[1])}")
y = x
print(f"y:{id(y)}, y[0]:{id(y[0])}, y[1]:{id(y[1])}")
y = x.copy() # y = x[:]
print(f"y:{id(y)}, y[0]:{id(y[0])}, y[1]:{id(y[1])}")
y[1] = 3
print(f"x:{x} x:{id(x)}, x[0]:{id(x[0])}, x[1]:{id(x[1])}")
print(f"y:{y} y:{id(y)}, y[0]:{id(y[0])}, y[1]:{id(y[1])}")
```

Παρατηρούμε ότι:

- Η copy δημιουργεί μία νέα λίστα, αλλά οι αναφορές στα αντικείμενα που περιέχονται σε αυτήν είναι οι ίδιες.
- Αυτό μπορεί να οδηγήσει σε ένα πρόβλημα..

Παράδειγμα 6: id4.py

```
x = [[1,2], [3,4]]
y = x.copy()
print(f"x:{x} x:{id(x)}, x[0]:{id(x[0])}, x[1]:{id(x[1])}")
print(f"y:{y} y:{id(y)}, y[0]:{id(y[0])}, y[1]:{id(y[1])}")
x[0].append(0)
print(f"x:{x} y:{y}")
```

Ναι μεν η λίστα y είναι άλλο αντικείμενο, αλλά περιέχει αναφορές στις ίδιες υπολίστες που περιέχει η x.

Shallow vs Deep Copy

- Η μέθοδος copy δημιουργεί ένα shallow copy με το πρόβλημα του π.χ. 5
- Αν έχουμε collection που περιέχει collection και θέλουμε αντίγραφό του, θα πρέπει να κατασκευάσουμε ένα deep copy, όπως φαίνεται στο ακόλουθο παράδειγμα.

Παράδειγμα 7: id4.py και id5.py

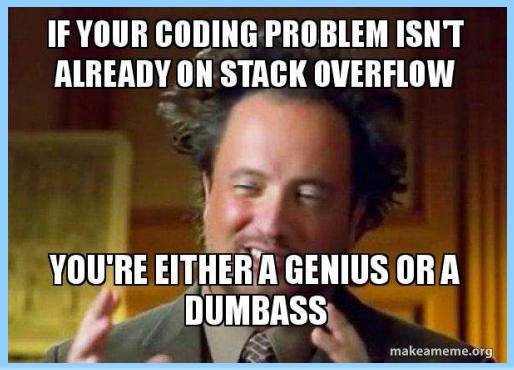
```
import copy

x = [[1,2], [3,4]]
y = copy.deepcopy(x)
print(f"x:{x} x:{id(x)}, x[0]:{id(x[0])}, x[1]:{id(x[1])}")
print(f"y:{y} y:{id(y)}, y[0]:{id(y[0])}, y[1]:{id(y[1])}")
x[0].append(0)
print(f"x:{x} y:{y}")
```

Κατά τη διάρκεια του debugging το google μπορεί να βοηθήσει…

- Περιέχει άπειρα tutorials, παρουσιάσεις θεμάτων, walkarounds, projects κ.λπ.
- αλλά κυρίως, **ένα πρόβλημα που αντιμετωπίζεις, μάλλον το έχει ήδη αντιμετωπίσει κάποιος άλλος**

Που να μπάινω;

- **Stack Overflow** (ερωτήσεις και απαντήσεις προγραμματιστών)
  - Για να ψάξεις ένα error που δεν καταλαβαίνεις (copy-paste στο google το error => stack overflow)
  - Για να αναζητήσεις κάτι που δεν καταλαβαίνεις καλά (αναζήτηση με λέξεις κλειδιά)
- **Συντακτικό – tutorials - άρθρα**
  - www.w3schools.com
  - www.medium.com
  - www.programiz.com
  - www.geeksforgeeks.org
  - www.realpython.com
  - docs.python.org/3/ [επίσημο documentation της Python]



Άσκηση 4: Ο προγραμματιστής προσπαθεί να ορίσει ένα σύνολο που περιέχει τα υποσύνολα του {1,2}. Γράφει λοιπόν τον εξής κώδικα:

```
x = {{}, {1}, {2}, {1,2}}
```

Η Python διαμαρτύρεται δίνοντας το λάθος:

```
Traceback (most recent call last):
  File "C:/python/lesson10/temp.py", line 14, in <module>
    x = {{}, {1}, {2}, {1,2}}
TypeError: unhashable type: 'dict'
```

Αναζητήσε το λάθος (google it) για να βρείτε τι πάει στραβά με τον κώδικα.

Άσκηση 5: Αναζητήστε άρθρα για τη διαφορά των mutable από τους immutable τύπους δεδομένων. Έχουμε καλύψει στα μαθήματα όλους τους τύπους δεδομένων της Python;

Άσκηση 6: Αναζητήστε τη λειτουργία της μεθόδου intersection (των συνόλων) στο επίσημο documentation της Python.

Άσκηση 7.1: Κρεμάλα

Στην κρεμάλα υπάρχει μία κρυφή λέξη και προσπαθούμε να τη βρούμε, μαντεύοντας τα γράμματα που αυτή περιέχει. Ξεκινήστε την κατασκευή:

- Ορίζοντας μία λίστα με καμια 15αριά λέξεις από τις οποίες το πρόγραμμα θα διαλέγει κάποια με τυχαίο τρόπο και θα την αποθηκεύει σε μία συμβολοσειρά με όνομα hidden_word

Άσκηση 7.2: Επανάληψη

Κατασκευάστε τον επαναληπτικό βρόχο:

- Ο παίκτης πληκτρολογεί ένα γράμμα.
- Το γράμμα προστίθεται σε μια λίστα με όνομα guessed_letters
- Εκτυπώνεται ένα μήνυμα που λέει στον παίκτη πόσες φορές υπάρχει το γράμμα στην κρυμμένη λέξη
- Εκτυπώνεται η κρυμμένη λέξη ως εξής: Αν το γράμμα ανήκει στην guessed_letters τότε εκτυπώνεται το γράμμα, αλλιώς να εκτυπώνεται ένα underscore

Άσκηση 7.3: Τερματισμός με επιτυχία

Τροποποιήστε το πρόγραμμα ώστε να εμφανίζει ένα κατάλληλο μήνυμα, αν ο χρήστης βρήκε όλα τα γράμματα της κρυμμένης λέξης.

Άσκηση 7.4: Τερματισμός με αποτυχία

Το παιχνίδι θα επιτρέπει στο χρήστη να κάνει 10 επιλογές γράμματος. Αν βρει τη λέξη κερδίζει, αλλιώς χάνει. Ορίστε το πλήθος των επιλογών σε μια μεταβλητή με όνομα max_rounds και ενσωματώστε εκτυπώσεις, ώστε να εμφανίζεται ο τρέχων γύρος καθώς και μήνυμα αποτυχίας σε περίπτωση που ο χρήστης φτάσει στο μέγιστο πλήθος γύρων και δεν έχει βρει σωστά τη λέξη.

Άσκηση 7.5: Εκλεπτύνσεις

Πειράξτε το πρόγραμμα ώστε:

- Το πρόγραμμα να λειτουργεί, είτε ο χρήστης πληκρολογήσει μικρό, είτε κεφαλαίο χαρακτήρα
- Να βγαίνει μήνυμα λάθους αν ο χρήστης εισάγει κάτι μη έγκυρο (π.χ. αριθμό, πολλά γράμματα κ.λπ.)
- Να βγαίνει μήνυμα λάθους αν ο χρήστης εισάγει χαρακτήρα, τον οποίο έχει ήδη μαντέψει σε προηγούμενο γύρο.

Άσκηση 8.1: Περιγραφή Δεδομένων

Η γραμματεία ενός δημοτικού σχολείου διατηρεί τα εξής στοιχεία για κάθε μαθητή:

- **Όνομα**
- **Επώνυμο**
- **Πατρώνυμο**
- **Ηλικία**
- **Τάξη** με τιμές (1, 2, 3, 4, 5,6)
- **Αρ. Ταυτότητας (αν υπάρχει)**

Για να ταυτοποιεί εύκολα έναν μαθητή, διατηρεί επίσης ένα κωδικό ταυτοποίησης (identity number – id) που είναι ένας απλός αύξων αριθμός, μοναδικός για κάθε μαθητή [Η αρίθμηση ξεκινά από το 1000].

Κατασκευάστε πρόγραμμα που να αρχικοποιεί σε μία λίστα τρεις μαθητές.

Άσκηση 8.2: Μενού Επιλογών

Κατασκευάστε ένα μενού το οποίο να εμφανίζεται επαναληπτικά στο χρήστη και να έχει τις εξής επιλογές:

1. Δημιουργία Εγγραφής
2. Εκτύπωση
3. Ενημέρωση Εγγραφής
4. Διαγραφή Εγγραφής
5. Έξοδος

Άσκηση 8.3: Δημιουργία Εγγραφής

Υλοποιήστε τη δημιουργία εγγραφής. Ο χρήστης θα εισάγει τα δεδομένα του μαθητή και ο νέος μαθητής θα προστίθεται στη λίστα.

Προσοχή, θα πρέπει να ελέγχεται ότι δεν υπάρχει ήδη ο μαθητής στη λίστα. Συγκεκριμένα:

- Όταν ο χρήστης εισάγει το Όνομα, το Επώνυμο και το Πατρώνυμο, θα ελέγχεται αν ο μαθητής υπάρχει ήδη.
- Αν υπάρχει, θα ενημερώνει το χρήστη ότι υπάρχει ήδη ένας μαθητής και θα τυπώνει τα στοιχεία του. Στη συνέχεια θα ρωτάει το χρήστη αν θέλει να προχωρήσει με την εισαγωγή των υπόλοιπων στοιχείων. Ο χρήστης θα επιλέγει αν θέλει να εισάγει την εγγραφή ή όχι (ανάλογα με το αν είναι απλή συνωνυμία ή όχι)
- Αν δεν υπάρχει, θα προχωρά κανονικά με την εισαγωγή των υπολοίπων στοιχείων.

Στις περιπτώσεις που όντως έγινε δημιουργία εγγραφής, να τυπώνεται και ολοκληρωμένη η εγγραφή με ένα μήνυμα επιτυχίας

Παρατηρήσεις:

Το ακρωνύμιο CRUD, προέρχεται από τα Create – Read – Update – Delete που είναι οι καθιερωμένες ενέργειες σε προγράμματα επεξεργασίας δεδομένων.

(Σημείωση: Η άσκηση θα συνεχιστεί στο επόμενο μάθημα)