ΠΕΡΙΕΧΟΜΕΝΑ:

Αγγελική Γ.
Σταυριανή Γ.
Χάλκινος Χορηγός Μαθήματος

Νίκος Θ.
Χάλκινος Χορηγός Μαθήματος

- **Σύνολο (set)** είναι μία **συλλογή (collection) στοιχείων χωρίς συγκεκριμένη σειρά**.
  - Προσοχή, ότι όπως στα μαθηματικά σύνολα **δεν υπάρχει διάταξη** και **δεν επιτρέπεται η επανάληψη κάποιου στοιχείου**.
  - Το σύνολο μπορεί να αλλάξει (mutable) π.χ. προσθέτωντας ή αφαιρώντας στοιχεία σε αυτό
    - Αλλά ένα στοιχείο του συνόλου δεν μπορεί να αλλάξει (τα στοιχεία είναι immutable)

- **Δήλωση συνόλου**
  - Ορίζουμε ένα σύνολο χρησιμοποιώντας κάποιο όνομα μεταβλητής και ενθέτοντας τις τιμές σε άγκιστρα.

Χρησιμοποιούμε το σύνολο:

- Όταν μας ενδιαφέρει μόνο το ποια στοιχεία ανήκουν στη συλλογή

Παράδειγμα 1: tuples.py

Στο παράδειγμα βλέπουμε τον ορισμό 4 συνολων:

```
int_set = {1, 2, 3}
collection = {"hi", 3.14, True}
empty_set = set()
duplicates = {1, 2, 3, 1, 1, 2}

print(duplicates)
print(type(int_set))
```

Παρατηρήσεις:

- Κάθε στοιχείο πρέπει να υπάρχει ακριβώς μία φορά στο σύνολο, άρα τυχόν διπλότυπα αγνοούνται
- Υπάρχει ιδιαιτερότητα με το κενό σύνολο. Δεν δουλευεί το προφανές {} (μιας και αυτό έχει δεσμευτεί για τον ορισμό του άδειου "λεξικού" όπως θα δούμε σε επόμενο μάθημα)
- **Αναμένουμε ότι η σειρά των στοιχείων θα είναι τυχαία.**

- **Συμπεριφορά:**
  - Indexes και ranges δεν υπάρχουν (δεν έχουν νόημα)
  - Βασικα κάνουμε:
    - έλεγχους: π.χ. if element in set:
    - loops: π.χ. for element in set:
  - Έχει μεθόδους που τροποποιούν το σύνολο (προσθήκη, διαγραφή στοιχείων) και πράξεις (τομή, ένωση, κ.λπ.)
  - Δεν μπορέι να περιέχει π.χ. λίστες (mutable)

Παράδειγμα 2: no.lists.in.set.py

```
my_set = {1, 2, (1, 2)}
print(my_set)
x = 3
my_set = {x, x**3, (x, 3*x)}
print(my_set)
my_set = {x, [x, 2]}
```

1. Μέθοδοι προσθήκης στοιχείου/ων σε σύνολο

- set_name.**add**(element)
  - Προσθέτει το element στο σύνολο set_name.
- set_name.**update**(collection)
  - Προσθέτει τα στοιχεία της collection (list, tuple, set, string, range) στο σύνολο set_name.

Παράδειγμα 3: set.add.elements.py

```
my_set = {1,2,3}
my_set.add(5)
my_set.add(3)
print(my_set)
my_set.update((1,4,5))
print(my_set)
```

2. Μέθοδοι αφαίρεσης στοιχείου/ων από σύνολο

- set_name.**remove**(element)
  - Αφαιρεί το element από το σύνολο set_name.
  - Προκαλεί σφάλμα αν το στοιχείο δεν υπάρχει.
- set_name.**discard**(element)
  - Αφαιρεί το element από το σύνολο set_name.
  - Δεν προκαλεί σφάλμα αν το στοιχείο δεν υπάρχει.
- set_name.**pop**()
  - Αφαιρεί κάποιο στοιχείο από το σύνολο.
- set_name.**clear**()
  - Αφαιρεί όλα τα στοιχεία από το σύνολο

3. Αντιγραφή συνόλου

- Το = και πάλι δείχνει στο ίδιο σύνολο (αναφορά)
- Για να κάνουμε καταχώρηση χρησιμοποιούμε τη μέθοδο copy()

Παράδειγμα 4: set.copy.py

```
set1 = {1,2,3}
set2 = set1.copy()
set2.add(4)
print(set1)
print(set2)
```

4. Μετατροπές από άλλους τύπους συλλογών

- set_name = **set**(collection): Μετατρέπει το collection σε σύνολο.

Παράδειγμα 5: set.convert.delete.py

```
my_list = [number for number in range(10) if number % 2 == 0]
print(my_list)
my_set = set(my_list)
print(my_set)
my_set.discard(2)
print(my_set)
my_set.remove(2)
print(my_set)
```

Άσκηση 1: Κατασκευή Υποσυνόλων

Κατασκευάστε πρόγραμμα το οποίο

- Θα αρχικοποιεί μία μεταβλητή N με την τιμή 5.
- Θα κατασκευάζει το σύνολο A={1,2,…,N}
- Έπειτα για κάθε στοιχείο του συνόλου, θα κατασκευάζει ένα tuple με τον αριθμό και το τετράγωνό του
- Όλα αυτά τα στοιχεία θα πρέπει να ανήκουν σε ένα σύνολο.

Η απάντηση θα πρέπει να είναι (τα tuples όχι απαραίτητα με αυτή τη σειρά):
{(1,1), (2,4), (3,9), (4,16), (5,25)}

Άσκηση 2: Κατασκευή εξάδων στο ΛΟΤΤΟ

Κατασκευάστε πρόγραμμα που θα επιλέγει τυχαίες 6-άδες του ΛΟΤΤΟ με τις ιδιότητες:

- Δύο από τους αριθμούς θα είναι στο εύρος από 10-19
- Δύο από τους αριθμούς θα είναι στο εύρος από 20-39
- Ένας αριθμός θα είναι ζυγός στο εύρος 1-9
- Ένας αριθμός θα είναι μονός στο εύρος 40-49

Το πρόγραμμα να κατασκευάζει 10 τέτοιες τυχαίες 6-άδες και να τις τυπώνει.

Κατασκευή τυχαίων αριθμών:

- Συμβουλευθείτε το βίντεο για την κατασκευή τυχαίων αριθμών
- Χρησιμοποιούμε το πρόγραμμα :

```
from random import seed
from random import randrange
from datetime import datetime  # all 3 at the beginning

seed(datetime.now())  # once, before randint call

x = randrange(10, 20)  # from 10 to 19
print(x)
```

(βλ. randrange.py)

Πράξεις επί συνόλων: Έχουν υλοποιηθεί όλες οι βασικές πράξεις συνόλων που ξέρουμε από τα μαθηματικά:

Πράξη	Εξήγηση
R = A | B R = A.union(B)	Επιστρέφει την ένωση $A \cup B$ (στοιχεία του A ή του B)
R = A & B R = A.intersection(B)	Επιστρέφει την τομή $A \cap B$ (στοιχεία του A και του B)
R = A - B R = A.difference(B)	Επιστρέφει την διαφορά $A - B$ (στοιχεία του A που δεν ανήκουν στο B)
R = A ^ B R = A.symmetric_difference(B)	Επιστρέφει την συμμετρική διαφορά $(A - B) \cap (B - A)$ (μη κοινά στοιχεία των A και B)
A.issubset(B)	T/F ανάλογα με το αν $A \subseteq B$
A.issuperset(B)	T/F ανάλογα με το αν $A \supseteq B$
A.isdisjoint(B)	T/F ανάλογα με το αν A, B ξένα μεταξύ τους

A.update(B) # union
A.intersection_update(B)
A.difference_update(B)
A.symmetric_difference_update(B)
	Παραλλαγές των πράξεων, που δεν επιστρέφουν νέο σύνολο, αλλά επενεργούν πάνω στο σύνολο που καλεί τη μέθοδο. Π.χ. η A.update(B) είναι ισοδύναμο με το $A = A \cup B$

Συναρτήσεις (built-in):

- **len**(A): πληθάριθμος συνόλου A
- **max**(A): μέγιστο στοιχείο του A
- **min**(A): ελάχιστο στοιχείο του A
- **sum**(A): άθροισμα στοιχείων του A

Παράδειγμα 6: set.operations.py

```
A = {1,2,3,4}
B = {4,5}
print("union: " + str(A | B))
print("intersect: " + str(A & B))
print("diff A-B: " + str(A - B))
print("diff B-A: " + str(B - A))
print("symm.dif.: " + str(A ^ B))
print("A subset B: " +
      str(A.issubset(B)))
print("A, B disjoint: " +
      str(A.isdisjoint(B)))

A.update(B)
print("A= A | B: " + str(A))
A.intersection_update(B)
print("A= A & B: " + str(A))
```

Άσκηση 3: Σύνολα Αριθμών

Κατασκευάστε τα εξής σύνολα φυσικών:

1) Το σύνολο των αρτίων από το 0 έως το 100
2) Το σύνολο των περιττών από το 0 έως το 100
3) Τα πολλαπλάσια του 3 από το 0 έως το 100
4) Το σύνολο των πρώτων από το 0 έως το 100 (βλ. και προηγούμενο μάθημα, άσκηση 4)

Τυπώστε τα περιεχόμενά τους, καθώς και το πλήθος των στοιχείων κάθε συνόλου.

Έπειτα μέσω αυτών κατασκευάστε και τυπώστε τα σύνολα φυσικών από το 0 έως το 100 που:

1) Είναι άρτιοι ή πολλαπλάσια του 3
2) Είναι περιττοί πρώτοι
3) Είναι πρώτοι αλλά δεν είναι περιττοί
4) Είναι περιττοί ή πρώτοι, αλλά όχι ταυτόχρονα και περιττοί και πρώτοι.

Άσκηση 4: Χωρισμός σε Ομάδες

Η δασκάλα μιας τάξης θέλει να κατασκευάσει ένα πρόγραμμα το οποίο:

- Δεδομένων των N (έστω N άρτιος) μαθητών της
- Θα τους χωρίζει σε N/2 ομάδες των 2 μαθητών, με τυχαίο τρόπο, ώστε να τους αναθέσει μια ομαδική εργασία.

Βοηθήστε τη δασκάλα ώστε να κάνει δύο χωρισμούς σε **τυχαίες** ομάδες των 2, έναν για το μάθημα των μαθηματικών και έναν για το μάθημα της γεωγραφίας.

- **Τα περιγραφικά σύνολα (set comprehensions):**
  - Παρέχουν ένα "πιο γρήγορο" συντακτικό για να κατασκευάσουμε επαναληπτικά ένα σύνολο
  - Χρησιμοποιείται εντελώς αντίστοιχο συντακτικό με τις περιγραφικές λίστες

- **Περίπτωση 1: Μια απλή for..in**

```
my_set = {number for number in range(3)}
```

  - Επιστρέφει το σύνολο: {0,1,2}

- **Περίπτωση 2: Μια απλή for..in με if**

```
my_set = {number for number in range(10) if number%2 == 0}
```

  - Επιστρέφει το σύνολο: {0,2,4,6,8}

- **Περίπτωση 3: Μια απλή for..in με if..else**

```
my_set = {number if number%2 == 0 else number/2
                    for number in range(10)}
```

  - Επιστρέφει το σύνολο: {0, 0.5, 2, 1.5, 4, 2.5, 6, 3.5, 8, 4.5}

βλ. και set.comprehensions1.py

Παρατήρηση:
- Τα παραπάνω (1-3) είναι ακριβώς ίδια με αυτά που είδαμε στις λίστες. Χρησιμοποιούν απλά {, } αντί για [,]
- Ενώ το 4 μπορεί επίσης να χρησιμοποιηθούν για λίστες.

Περίπτωση 4: Nested Loops
- Αντί να γράψουμε

```
my_set = set()
for i in range(2):
    for j in range(3):
        my_set.add((i,j))
```

- μπορούμε να γράψουμε (σε μία γραμμή) το παρακάτω:

```
my_set = {(i, j) for i in range(2)
                 for j in range(3)}
```

Περίπτωση 5: Nested Loops με συνθήκη
- Αντί να γράψουμε

```
my_set = set()
for i in range(6):
    if i%2==0:
        for j in range(6,10):
            if j%2==1:
                my_set.add((i,j))
```

- μπορούμε να γράψουμε (σε μία γραμμή) το παρακάτω:

```
my_set = {(i, j) for i in range(6) if i%2==0
                 for j in range(6,10) if j%2==1}
```

(βλ. και set.comprehensions2.py, set.comprehensions3.py)

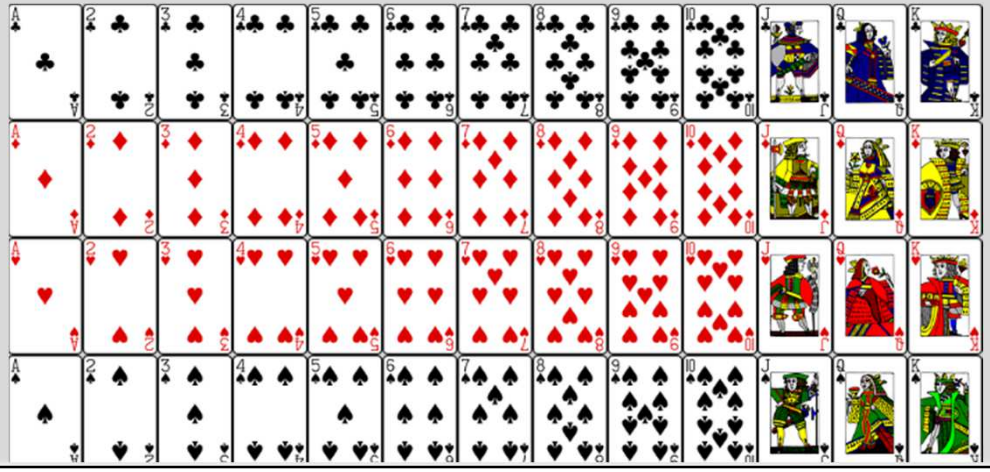
Άσκηση 5.1: Πόκερ

Κατασκευάστε πρόγραμμα το οποίο:

1) Θα κατασκευάζει ένα σύνολο με τα 52 φύλλα της τράπουλας. Κωδικοποιήστε ένα φύλλο ως ένα tuple με δύο χαρακτηριστικά:

- Το πρώτο (είδος) είναι ένα από τα heart, diamond, spade, club
- Το δεύτερο (αριθμός) είναι ένα από τα ace,2,3,…,9,10, jack, queen, king

2) Θα ορίζει δύο παίκτες. Κάθε παίκτης θα συμβολίζεται με ένα (αρχικά κενό) σύνολο με τα χαρτιά που κρατάει στο χέρι του.
3) Επαναληπτικά θα μοιράζονται τυχαία 5 φύλλα σε κάθε παίκτη (το μοίρασμα να γίνεται εναλλάξ στους δύο παίκτες)



Άσκηση 5.2: Καρέ του άσσου

Συνεχίζοντας το προηγούμενο πρόγραμμα:

- Ελέγξτε αν κάποιος παίκτης έχει καρέ του άσσου (δηλαδή τέσσερις άσσους και ακόμη ένα φύλλο)

Άσκηση 5.3: Κέντα

Συνεχίζοντας το προηγούμενο πρόγραμμα:

- Ελέγξτε αν κάποιος παίκτης έχει κέντα (δηλαδή και τα 5 φύλλα να μπαίνουν σε σειρά στους αριθμούς π.χ. 2,3,4,5,6 – τα είδη τους δεν μας ενδιαφέρουν)

Άσκηση 6.1: Τρίλιζα

Αν χρειάζεστε υπενθύμιση για την τρίλιζα :

- google: “Play tic tac toe”
- Και οι δύο παίκτες θα ελέγχονται από τον χρήστη

Θα αποθηκεύσουμε το πλάισιο σε έναν πίνακα 3x3 και τα βασικά βήματα σε μία επανάληψη είναι:

- Εκτύπωση του πλαισίου
- Επιλογή του επόμενου παίκτη
- Ο παίκτης πληκτρολογεί το τετράγωνο που παίζει
- Έλεγχος αν νίκησε ο παίκτης. Αν ναι, γίνεται εκτύπωση του πλαισίου, το πρόγραμμα τερματίζει και εμφανίζεται κατάλληλο μήνυμα.

Αν γεμίσουν όλες οι θέσεις χωρίς να νίκησε κάποιος παίκτης, τότε γίνεται εκτύπωση του πλαισίου, το πρόγραμμα τερματίζει και εμφανίζεται κατάλληλο μήνυμα.

Άσκηση 6.2: Εκτύπωση πλαισίου

Ξεκινήστε με την εκτύπωση του πλαισίου (να είναι κομψή π.χ.:)

```
+---+---+---+
| O | X |   |
+---+---+---+
|   |   | X |
+---+---+---+
| O |   |   |
+---+---+---+
```

Άσκηση 6.3: Επόμενος παίκτης

Κατασκευάστε το μηχανισμό για την επιλογή του επόμενου παίκτη, καθώς και το μηχανισμό και την εκτύπωση αν έχουμε ισοπαλία (έχουμε φτάσει στον 9ο γύρο).

Άσκηση 6.4: Είσοδος Χρήστη

Κατασκευάστε το μέρος για την είσοδο του χρήστη.

- Να γίνεται έλεγχος ότι η είσοδος του χρήστη είναι έγκυρη (στα όρια του πίνακα) και ότι το τετράγωνο είναι κενό.
- Να γεμίζει το τετράγωνο με το σύμβολο του παίκτη.

Άσκηση 6.5: Έλεγχος Νικητή

Κατασκευάστε το μέρος για τον έλεγχο αν ο παίκτης νίκησε

- Πρέπει να γίνεται έλεγχος σε κάθε γραμμή, στήλη και τις δύο διαγωνίους.
- Αν έχουμε νικητή το πρόγραμμα να εκτυπώνει το πλαίσιο, να τυπώνει ποιος νίκησε και έπειτα να τερματίζει