ΠΕΡΙΕΧΟΜΕΝΑ:

Ιωάννης Τ.

Χρυσός Χορηγός Μαθήματος

Έφη

Ασημένιος Χορηγός Μαθήματος

- **Λεξικό (dictionary)** είναι μία **mutable συλλογή (collection) από ζευγάρια κλειδιού – τιμής (key-value pairs)**.
  - Δεν έχει σημασία η σειρά των ζευγαριών key-value.
  - Τα κλειδιά είναι immutable (άρα δεν μπορούν να είναι σύνολα και λίστες)
  - Οι τιμές είναι mutable.

- **Δήλωση λεξικού**
  - Ορίζουμε ένα λεξικό χρησιμοποιώντας κάποιο όνομα μεταβλητής και ενθέτοντας τα ζευγάρια σε άγκιστρα.
  - Τα ζευγάρια χωρίζονται με κόμμα
  - Το κλειδί χωρίζεται από την τιμή με άνω κάτω τελεία.

Παράδειγμα 1: dict.definition.py

Στο παράδειγμα βλέπουμε τον ορισμό δύο λεξικών:

```
empty = {}
person = {
   "grade": 13,
   (1,2): 29,
   "name": "Jim",
}
print(person)
print(type(person))
```

Παρατηρήσεις:

- Κάθε κλειδί πρέπει να υπάρχει ακριβώς μία φορά.
- Το τελευταίο κλειδί-τιμή μπορεί να ακολουθείται από κόμμα.

Χρησιμοποιούμε το λεξικό:

Όταν τα δεδομένα μιας οντότητας μπορούν να απεικονιστούν ως ζεύγη κλειδιού – τιμής (όπως ένα λεξικό)

- **Συμπεριφορά:**
  - Indexes και ranges δεν υπάρχουν (δεν έχουν νόημα)
  - Έχουμε πρόσβαση σε μία τιμή μέσω του κλειδιού.
  - Προσθέτουμε/Αφαιρούμε ζεύγη κλειδιού-τιμής
  - Διατρέχουμε όλα τα ζευγάρια μέσω κλειδιών ή τιμών.
- Και γενικά είναι ένα πανίσχυρο μέσο αποθήκευσης πληροφοριάς (αποθήκευση χαρακτηριστικών μιας οντότητας)

Παράδειγμα 2: dict.example.py

```
person = {
   "name": "Tony Stark",
   "alias": "Iron Man“
   "power": 20
}
print(person)
```

1. **Πρόσβαση σε κάποιο στοιχείο:**
   - **dict_name[key]** ή
   - dict_name.**get**(key)
2. **Προσθήκη στοιχείου σε λεξικό**
   - **dict_name[key] =** value
     - προσθέτει το κλειδί με την τιμή
     - Αν υπάρχει το κλειδί, τροποποιεί την τιμή.

Παράδειγμα 3: dict.add.py

```
hero = {"name": "Bruce Banner", "alias": "Iron Man"}
print(hero)
hero["name"] = "Tony Stark"
hero["equipment"] = "suite"
print(hero)
```

3. **Αντιγραφή λεξικού**
   - Το = και πάλι δείχνει στο ίδιο λεξικό (αναφορά)
   - Για να κάνουμε καταχώρηση και αντιγραφή σε νέο λεξικό, χρησιμοποιούμε τη μέθοδο **copy**()

Παράδειγμα 4: dict.copy.py

```
hero1 = {"name": "Tony Stark", "alias": "Iron Man"}
hero2 = hero1.copy()
hero2["equipment"] = "suite"
print(hero1)
print(hero2)
```

4. **Μέθοδοι αφαίρεσης ζεύγους από λεξικό**
   - dict_name.**pop**(key)
     - Αφαιρεί το ζεύγος με κλειδί key από το λεξικό.
   - dict_name.**popitem**()
     - Αφαιρεί το τελευταίο ζεύγος με βάση τη σειρά προσθήκης των ζευγών στο λεξικό.
   - dict_name.**clear**()
     - Αφαιρεί όλα τα ζεύγη από το λεξικό

5. **Μετατροπές από άλλους τύπους συλλογών**
   - dict_name = **dict**(collection): Μετατρέπει το collection (list, tuple ή set) σε λεξικό (μόνο εφόσον τα στοιχεία του collection περιέχει tuples με δύο στοιχεία).

Παράδειγμα 5: dict.convert.delete.py

```
a_list = [("name", "Natasha Romanoff"), ("alias", "Black Widow")]
hero = dict(a_list)
hero["ability"] = "hand-to-hand combat"
print(hero)
hero.pop("name")
print(hero)
hero.clear()
print(hero)
```

Άσκηση 1: Λεξικό

Κατασκευάστε πρόγραμμα το οποίο

- Θα κατασκευάζει ένα πραγματικό λεξικό.
- Να περιέχει αρχικά τα λήμματα:
  - ιταμός: Προκλητικός, αυθάδης, αναιδής
  - όνειδος: ντροπή, καταισχύνη
  - πομφόλυγες: αερολογίες, ανοησίες
- Να τυπώνει το λεξικό
- Να προσθέτει έπειτα το λήμμα:
  - φληναφήματα: ανοησίες, σαχλαμάρες
- Να ζητάει από το χρήστη ένα νέο ζεύγος κλειδιού - τιμής
- Να προσθέτει το ζεύγος στο λεξικό και να τυπώνει το λεξικό.

Παρατήρηση:

- Οι συμβολοσειρές της Python μπορούν να δεχτούν και ελληνικούς χαρακτήρες

Άσκηση 2: Οντότητα με χαρακτηριστικά

Στην Βάση Δεδομένων της Γ' Υποδιεύθυνσης Α/Β/23/8 του Υπουργείου Γραφειοκρατίας, διατηρείται μητρώο των εργαζομένων. Για κάθε εργαζόμενο διατηρούνται τα στοιχεία:

- Όνομα
- Επώνυμο
- Πατρώνυμο
- Ημερομηνία Γέννησης
- Διεύθυνση
- Τηλέφωνο

Κατασκευάστε ένα πρόγραμμα Python το οποίο θα αρχικοποιεί (με στοιχεία της επιλογής σας) έναν εργαζόμενο με τα παραπάνω στοιχεία.

Έπειτα να τυπώνει τα στοιχεία του εργαζομένου, μορφοποιημένα ως εξής:

```
Ονοματεπώνυμο: Κωνσταντίνος Γεώργιος Γεωργίου,
Ημ/νία Γέννησης: 11/10/1957
Διεύθυνση       : Πανεπιστημίου 24
Τηλέφωνο        : 2101234567
```

Παρατήρηση:

Στις ασκήσεις αυτές είδαμε (α) τη χρήση λεξικού με την αναμενόμενη χρήση του (κλειδιά που προσδιορίζουν τιμές) και (β) ως το μέσο για να ομαδοποιήσουμε τα χαρακτηριστικά (attributes) μιας οντότητας.

Έλεγχος (για χρήση στην if):

- key **in** dict: Ελέγχει αν το κλειδί key υπάρχει στο λεξικό
- key **not in** dict: Ελέγχει αν το κλειδί key δεν υπάρχει στο λεξικό

Παράδειγμα 6: dict.if.py

```
hero = {"name": "Bruce Banner", "alias": "Hulk"}
if "equipment" not in hero:
    print(hero["alias"] + " has no equipment")
```

Επανάληψη: Μπορούμε να διατρέξουμε τα ζεύγη του λεξικού με διαφορετικούς τρόπους:

- for **key** in dict()
  - Διατρέχει τα κλειδιά
- for **key, value** in dict.**items**()
  - Διατρέχει ταυτόχρονα κλειδιά και τιμές
- for **key** in dict.**keys**()
  - Διατρέχει τα κλειδιά (χωρίς ταξινόμηση)
- for **value** in dict.**values**()
  - Διατρέχει τις τιμές (με επαναλήψεις)

Χρήσιμες ιδέες για τις Διαπεράσεις:

- for key in sorted(dict.keys())
  - Διατρέχει τα κλειδιά (με αύξουσα ταξινόμηση)
- for key in set(dict.values())
  - Διατρέχει τις τιμές (χωρίς επαναλήψεις)

Παράδειγμα 7: dict.loop.py

```
heroes_weapons = {
    "Black Panther": "Anti-Metal Claws",
    "Wolverine": "Claws",
    "Ultron": "Plasma Weapons",
    "Spider-Man": "Web-shooters",
    "Beast": "Claws",
    "Venom": "Web-shooters"
}

print("Key-value loop: ")
for key, value in heroes_weapons.items():
    print(key + " has " + value)

print("\nOrdered Key loop: ")
for key in sorted(heroes_weapons.keys()):
    print(key + " has " + heroes_weapons[key])

print("\nWeapons Gallery: ")
for value in set(heroes_weapons.values()):
    print(value, end=", ")
```

Παρατηρήσεις για τη συνάρτηση sorted της Python:

- Η sorted παίρνει ως πρώτο όρισμα ακολουθία, σύνολο ή λεξικό.
- Επιστρέφει μία λίστα με τα στοιχεία ταξινομημένα.
- Αν βάλουμε 2ο όρισμα reverse=True ταξινομεί τα στοιχεία σε φθίνουσα σειρά

Άσκηση 3: Να παίξουμε μπαρμπούτι στον υπολογιστή;

Κατασκευάστε πρόγραμμα το οποίο θα ελέγχει αν οι τυχαίοι αριθμοί είναι αρκετά τυχαίοι.. ώστε να μπορούμε να τους χρησιμοποιήσουμε για τυχερά παίγνια, όπως το μπαρμπούτι.

- Κατασκευάστε ένα λεξικό με την εξής δομή
  - κλειδί: οι αριθμοί (ζαριές) από το 1-6
  - τιμή: Ένας μετρητης για κάθε κλειδί. Αρχικά θα είναι 0 για όλα τα κλειδιά.
- Έπειτα για 1.000.000 φορές:
  - Θα ρίχνει ένα ζάρι (τυχαίος αριθμός από το 1-6)
  - Θα αυξάνει τον μετρητή του αντίστοιχου ζαριού
- Θα εμφανίζει τα ποσοστά εμφάνισης κάθε ζαριάς (διαιρώντας τον αντίστοιχο μετρητή με το 1.000.000)

Επαναλάβετε για 1000 ρίψεις ζαριού και 10 ρίψεις ζαριού.

Παρατήρηση:

- Η παραπάνω άσκηση θα μπορούσε να γίνει και με λίστα αντί για λεξικό.
- Ωστόσο πιο περίπλοκα κλειδιά απαιτούν τη χρήση λεξικού.

Άσκηση 4: Στατιστική Ανάλυση Κειμένου

Κάντε copy paste ένα κείμενο στα αγγλικά από κάποια σελίδα και αποθηκεύστε το σε μία συμβολοσειρά (ας είναι μια παράγραφος 5-6 γραμμών). Έπειτα:

- Μετατρέψτε το string σε λίστα
- Τυπώστε τη λίστα και παρατηρήστε τα περιεχόμενα της.
- Διατρέξτε τη λίστα ώστε να κατασκευάστε ένα λεξικό πλήθους εμφανίσεων κάθε χαρακτήρα (για κάθε χαρακτήρα, να αποθηκεύεται πόσες φορές εμφανίζεται ο χαρακτήρας στο κείμενο)
- Τυπώστε τον χαρακτήρα (ή τους χαρακτήρες) με το μέγιστο πλήθος εμφανίσεων.

Παρατήρηση:

Οι ασκήσεις αυτές δείχνουν την 3η χρήση του λεξικού ως "συσσωρευτές" πληροφορίας.

- **Τα περιγραφικά λεξικά:**
  - Σε αντιστοιχία με τις λίστες και τα σύνολα
  - Ορίζουν με ένα πιο γρήγορο τρόπο ένα λεξικό.

Ορισμός περιγραφικού λεξικού:

- Υπάρχουν ακριβώς οι ίδιες δυνατότητες με αυτές που είδαμε στις λίστες και τα σύνολα.
- Μόνη διαφορά ότι το στοιχείο που μπαίνει τελικά στο λεξικό έχει τη μορφή key: value

Παράδειγμα 8: dict.comprehensions.py

```
dict1 = {v:v**2 for v in range(10)}
print(dict1)

dict2 = {v:v**2 for v in range(10) if v%2==0}
print(dict2)

dict3 = {(i, j): 0 for i in range(1,7)
                   for j in range(1,7)}
print(dict3)
```

- **Ενσωματωμένες συναρτήσεις:**
  - Η **len** επιστρέφει το πλήθος των ζευγών κλειδιού-τιμής.
  - Η **max** επιστρέφει το μέγιστο κλειδί
  - Η **min** επιστρέφει το ελάχιστο κλειδί

Άσκηση 5: Μαθαίνοντας εμπόριο

Ένας μικρο-έμπορος χωρίς επαγγελματική συνείδηση κάνει τα εξής:

- Αποθηκεύει σε ένα λεξικό με όνομα (merchandise) τις τιμές αγοράς των προϊόντων του
- Συγκεκριμένα εμπορεύεται (κατασκευάστε το λεξικό):
  - Αντίτυπα των “50 αποχρώσεων του γκρι” με τιμή αγοράς 10.18 ευρώ
  - Ματσάκια μαϊντανού με τιμή αγοράς 0.22 ευρώ
  - Επισκευαστικό τσιμέντο με τιμή αγοράς 5.17 ευρώ
  - πειρατικά CD της Billie Eilish με τιμή αγοράς 0.05 ευρώ.
- Έπειτα με το έμπειρο μάτι του βαθμολογεί τον πελάτη που εισέρχεται με έναν θετικό πραγματικό αριθμό (rate) από το 0 έως το 5 (όσο πιο μικρός ο αριθμός, τόσο πιο μικρή και η απόπειρα να κερδίσει από το συγκεκριμένο πελάτη)
- Στη συνέχεια κατασκευάζει (μέσω dictonary comprehension) ένα νέο λεξικό με όνομα new_values όπου κάθε τιμή έχει πολλαπλασιαστεί με το (1+rate)
- Τυπώνει το νέο λεξικό ώστε να δείξει το νέο τιμοκατάλογο στον πελάτη του.

Άσκηση 6: Λεξικό από λίστες

Το αρχείο κώδικα exercise06.initial περιέχει ένα λεξικό με τα 40 υψηλότερα κτίρια στον κόσμο όπου. Ως κλειδί χρησιμοποιείται το όνομα του κτιρίου και ως τιμή η χώρα στην οποία βρίσκεται το κτίριο.

- Κατασκευάστε ένα νέο λεξικό με όνομα country_buildings το οποίο θα περιέχει:
  - κλειδί: οι χώρες που εμφανίζονται στο λεξικό
  - τιμή: μία λίστα με τα υψηλότερα κτίρια που βρίσκονται στη συγκεκριμένη χώρα.

(Σημείωση: Προφανώς η μετατροπή πρέπει να γίνει προγραμματιστικά, δηλαδή διατρέχοντας τα στοιχεία του ενός λεξικού πρέπει να κατασκευάζουμε το 2ο λέξικο)

Σχόλιο:

Για την αποθήκευση της πληροφορίας της άσκησης, "πιο σωστός" τρόπος αποθήκευσης είναι ο δεύτερος, διότι κάθε πληροφορία (χώρα ή κτίριο) εμφανίζεται μία φορά στο λεξικό.

Άσκηση 7: Οικογενειακό Δένδρο των Simpsons

Μελετήστε με προσοχή την οργάνωση των πληροφοριών μιας οικογένειας σε ένα σύνθετο λεξικό, στο αρχείο exercise07.initial.py
Έπειτα κατασκευάστε ένα πρόγραμμα το οποίο θα τυπώνει:

- Το όνομα κάθε μέλους της οικογένειας συνοδευόμενο από μία τυχαία ατάκα του.

Σημείωση: Η άσκηση έχει αυξημένη δυσκολία. Συμβουλευθείτε το βίντεο.

Παρατήρηση:

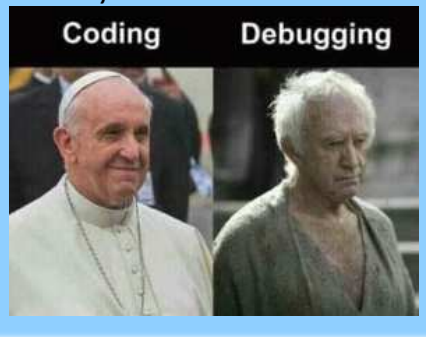
Τέτοιου τύπου οργάνωση πληροφορίας είναι συχνή σε πραγματικές εφαρμογές (4η χρήση των λεξικών)

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ = Επανάλαβε τα παρακάτω 2 βήματα εώσότου το πρόγραμμα είναι ολοκληρωμένο:

1. Γράψε κώδικα
2. Βρες τα λάθη στον κώδικά που έγραψες (διώξε τα ζουζούνια, κάνε απεντόμωση – **debugging** (bug=έντομο))

Έχουμε ήδη κάνει:

- **Συντακτικά Λάθη (syntax errors)**, π.χ. να ξεχάσουμε ένα κόμμα
  - Το Pycharm τα υπογραμμίζει ή τα κοκκινίζει
  - Η Python πετάει error
- **Λογικά Λάθη (logic errors)**: ο προγραμματιστής κάνει ένα μη συντακτικό λάθος που οδηγεί σε μη αναμενόμενη συμπεριφορά:
  - Δεν υπάρχει τρόπος προ-εντοπισμού τους.
  - Είναι πολύ συχνά.
  - Πρέπει να κάνουμε συστηματικό έλεγχο ορθότητας του προγράμματος μας (testing)
  - **Συχνά προσπαθούμε να εντοπίσουμε τα λογικά λάθη εκτυπώνοντας ενδιάμεσες τιμές** (κυρίως σε επαναλήψεις) ώστε με παρατήρηση, να εντοπίσουμε που υπάρχει το λάθος.



Άσκηση 8: Ένας προγραμματιστής προσλαμβάνεται για να υπολογίσει το άθροισμα: 1 + 1/2 + 1/3 + 1/4 + … + 1/100
Κατόπιν ωρίμου σκέψεως παραδίδει τον εξής κώδικα:

```
s = 0
for i in range(1, 100):
   s += 1//i
print(s)
```

Ο κώδικας – ατυχώς – βγάζει αποτέλεσμα 1, το οποίο προφανώς δεν είναι σωστό.
Ο προγραμματιστής εισάγει μια γραμμή εκτύπωσης, ώστε να τυπώσει τα ενδιάμεσα αποτελέσματα και να τα παρατηρήσει:

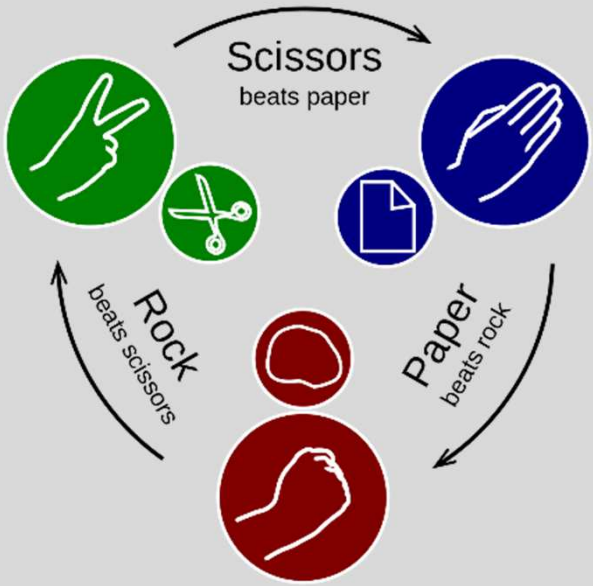
```
s = 0
for i in range(1, 100):
   s += 1//i
   print("i=" + str(i) +" 1/i=" + str(1//i) + " s=" + str(s))
print(s)
```

Πλέον ο προγραμματιστής μπορεί να βρει το (ή τα) λογικά λάθη του προγράμματος του. Εμείς μπορούμε; (exercise08.initial.py)

Σχετικό περίφημο ρητό του Donald Knuth:
"Beware of bugs in the above code. I have only proved it correct, not tried it"

Άσκηση 9.1: Πέτρα – Ψαλίδι – Χαρτί

Θα παίξουμε το παιχνίδι με τον υπολογιστή. Υπενθύμιση:



Κατασκευάστε πρόγραμμα έτσι ώστε να παίζεται μία φορά το παιχνίδι μεταξύ υπολογιστή και χρήστη:

- Ο χρήστης να επιλέγει από την είσοδο.
- Ο υπολογιστής να επιλέγει τυχαία.
- Να υπολογίζεται και να εκτυπώνεται ποιος νίκησε.

[Hint: Για την ώρα σκεφθείτε απλά και αποθηκεύστε τις επιλογές σε συμβολοσειρές “scissors” ή “rock” ή “paper”]

Άσκηση 9.2: Βγαίνουμε στις 3 νίκες

Τροποποιήστε το πρόγραμμα ώστε να επαναλαμβάνεται το παιχνίδι, μετρώντας τις νίκες κάθε παίκτη. Σε κάθε γύρο να εμφανίζεται το σκορ. Το πρόγραμμα να τερματίζει όταν ο ένας από τους δύο παίκτες φτάσει τις 3 νίκες.

Άσκηση 9.3: Ιστορικό

Τροποποιήστε το πρόγραμμα ώστε να αποθηκεύονται με κατάλληλο τρόπο οι πληροφορίες κάθε γύρου. Στο τέλος του παιχνιδιού να εκτυπώνονται εκτός του νικητή και του σκορ, και το ιστορικό του πως εξελίχθηκε το παιχνίδι (π.χ. στον 1ο γύρο “Round 1: Player: Rock, Computers: Scissors, Score: 1-0”)

Άσκηση 9.4: Τροποποίηση ελέγχου νικητή

‘Ενας φίλος μας – πύραυλος προγραμματιστής – μας προτείνει το εξής:

- Η επιλογή κάθε παίκτη να κωδικοποιείται με ακέραιο 0 ή 1 ή 2 (αντίστοιχα για κάθε επιλογή – πέτρα, ψαλίδι ή χαρτί)
- Ο έλεγχος, λέει, μπορεί να γίνει κάνοντας μία απλή αφαίρεση.

Πως μπορούμε να ενσωματώσουμε την ιδέα του στο πρόγραμμά μάς;