IT SCHOOL
from zero to hero

# Python Session 5

## Dictionaries

- Dictionarele  sunt folosite sa salveze multiple  date in aceeasi variablia.
- Dictionarele sunt una dintre cele 4 modalitati de a tine si de a folosi o colectie de date in Python.
- Un dictionar se creaza folosind {} si are format de cheie-valoare ex:  my_dict = {“my_key”: 12}
- Pentru a crea un dictionar putem sa folosim si functia constructor dict() ex: dict(((1, 2), (2, 3)))

## Dictionaries

- Dictionarele pot tine tipuri de date diferite

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

- Dictionarele nu ne permit chei duplicate

## Accesare elemente

- Accesarea elementelor se face folosind cheia. my_dict[cheie]
- Accesarea se poate face folosind si metoda get(), la aceasta metoda putem sa oferim si un default daca cheia nu este gasita. my_dict.get(cheie,valoare_default)
- Folosind operatorul ‘in’ putem verifica daca ce cautam noi se afla in cheile dictionarului

## Dictionary view objects

- Folosind metoda keys(), extragem cheiile folosite in dictionar
- Folosind metoda values(), extragem valorile din dictionar
- Putem sa folosim metoda items() sa returnam dictionarul in forma de ‘lista’ de tuple
- Tot ce este returnat de aceste 3 metode sunt niste dictionary views si sunt dinamice

## Adaugare elemente

- Putem adauga un nou element folosind o noua cheie si sa ii asignam o valoare

```
thisdict["color"] = "red"
```

- Folosind metoda update(), prin aceasta metoda dictionarul o sa fie updatat cu elementele noi primite in metoda.

* Datele transmise metodei update trebuie sa fie dictionar sau un obiect iterabil cu formatul cheie -valoare

## Update elemente

- Putem schimba un element prin a folosi cheia acestuia.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
```

- Folosind metoda update(), prin aceasta metoda dictionarul o sa fie updatat cu informatiile primite in metoda.

* Datele transmise metodei update trebuie sa fie dictionar sau un obiect iterabil cu formatul cheie -valoare

## Eliminare elemente

- Folosind metoda pop(‘key’, default(optional))

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

- Folosind metoda popitem(), elimina ultimul element inserat (LIFO)
  - Inainte de python 3.7 un element random era eliminat

## Nested dictionaries

- Un dictionar poate sa contina alte dictionare, in acest caz vorbim despre nested dictionaries

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```

## Dictionaries Methods

| | |
|---|---|
| clear() | Removes all the elements from the dictionary |
| copy() | Returns a copy of the dictionary |
| get() | Returns the value of the specified key |
| items() | Returns a ‘list’ containing a tuple for each key value pair |
| keys() | Returns a ‘list’ containing the dictionary's keys |
| pop() | Removes the element with the specified key |
| popitem() | Removes the last inserted key-value pair |
| setdefault() | Returns the value of the specified key. If the key does not exist: insert the key, with the specified value |
| update() | Updates the dictionary with the specified key-value pairs |
| values() | Returns a list of all the values in the dictionary |

IT SCHOOL from zero to hero

## Exercitii

- Get the value for the model of the car and the color

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

- Add a color to the car dict
- Add a list of possible colors to the car
- Retrieve the owner_name from the car dict , if not present set it.
- Clear the dictionary

## IF Statements

- In python daca dorim sa executam instructiuni doar in anumite cazuri putem sa folosim **if** statements

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

## IF Statements

- Cuvantul **elif** il putem folosi sa verificam o noua conditie daca cea precedenta nu a fost adevarata

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

## IF Statements

- Cuvantul **else**, este folosit sa folosit sa ‘prindem’ toate cazurile in care conditiile precedente nu au fost adevarate

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

## IF Statements

- Cand avem de executat o singura linie de cod daca conditia este adevarata o putem scrie  pe aceeasi linie cu if-ul (aka Short Hand if)

```
if a > b: print("a is greater than b")
```

- Acest lucru este valabil si pentru else(aka Short Hand if else)

```
print("A") if a > b else print("B")
```

- Daca nu avem cod de executat cand conditia este valida, nu putem pur si simplu sa lasam gol. Folosim **pass**

## Nested IFs

- Putem sa avem if -uri in cadrul altor if-uri. In acest caz se numesc nested ifs

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

## While Loops

- While loops.
  - Acest loop executa instructuiun cat timp conditia este adevarata

```
i = 1
while i < 6:
  print(i)
  i += 1
```

- Daca conditia ramane mereu adevarata si nu se schimba o sa ramanem intr-un while loop permanent

## While Loops

- Daca dorim sa iesim din loop inainte de finalizarea acestuia putem sa folosim statementul **break**

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

## While Loops

- Daca dormi sa oprim iteratia curenta a loopului si sa trecem la urmatoarea folosim **continue**

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

- ! Mare atentie sa nu ramanem intr-un infinite loop

# Nested While Loops

- Putem sa folosim while loops inauntrul altor while loops => nested while loops

```
i = 1
j = 1

while i <= 3:
    print(i, "Outer loop is executed")
    while j <= 3:
        print(j, "Inner loop is executed")
        j += 1
    i += 1
```

## Exercitii

- Print “Salutare” 5 times using while loops
- Print every letter from the string “Hello” on a new line except for “e” and “o” using while loops
- Print the letters from the string “hello” on a new line until we find the letter “l” then stop printing
- Retrieve from the user using the input() method 5 numbers and print the average
- Pop all elements from the list fruitsList = ["Mango","Apple","Orange","Guava"] using a while loop
- Printing the items in a tuple using while loop

## Homework

- Write a program that will tell if a dictionary is empty or not
- Write a program that will compute the factorial of a number imputed by the user (using while loops)
- The user will input 5 numbers (one at a time) print the min and the max values
- Finding the sum of numbers in a list using while loop

## Homework

- Number guessing game, we want to create a game where the user needs to guess the number the computer is thinking of.

Steps:

- The computer chooses a number between 1 and 10
  - This is done using the random.randint() function .
  - Use the code snippet below exactly and you will have a random number assigned to variable chosen_number

```
import random
start_no = 1
end_no = 10
chosen_number = random.randint(start_no , end_no)
```

- The computer tells the user what range the number is in.
- User inputs guess
- The computer tells the user if it's higher or lower or if the user succeeded in guessing the number

Optional :

- Add guess limit

## For Loops

- Un for loop este folosit pentru a (itera peste/ parcurge) o seventa precum string,list,tuples,dict,set

- Cu ajutorul for-ului putem sa executam cod pe pentru fiecare element din lista/set/tuple etc

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

## For Loops

- In python in cadrul for ului nu avem nevoie de un index setat precum in alte limbaje

  Java:  for (int i = 0; i < 10; i++)

- Daca dorim sa parcurgem de la un anumit index la altul si sa putem folosii pasul dorit in python o sa folosim functia range()

  Python:  for x in range(0, 10, 1):

IT SCHOOL from zero to hero

## For Loops

- Functia range()
  - Poate sa primeasca 3 argumente
    - start_index
    - end_index
    - Step
- Avem **continue** si **break** si **pass** care functioneaza exact priecum in while loop

## Nested for loops

- Si la for loops putem avea nested for loops (for loop in for loop)

```
adj = ["red", "yellow", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

## Exercitii

- Write a program that prints all keys of a dictionary using for loops
- Write a program that prints the keys and values of a dictionary
- Write a program that prints only the even numbers from a list
- Write a program that finds and prints the largest number from a list of lists

## Homework

- Write a program that will compute the factorial of a number imputed by the user (using for loops)
- Finding the sum of numbers in a list using for loop
- Get all values from the dictionary and add them to a list but don’t add duplicates (use for loops and do it without for loops)

  data = {'jan': 47, 'feb': 52, 'march': 47, 'April': 44, 'May': 52, 'June': 53, 'july': 54, 'Aug': 44, 'Sept': 54}