python notes - ACHRAF KHABAR

Made by : ashraf khabar .

generale

every thing in python is an object , and every variable is an attribute , and every function is a methode .

Type of objects

the function `type` return `<class 'type'>` .

```
    print(type(5)) # return <class 'int'>
    print(type(5.2)) # return <class 'float'>
    print(type("ashraf")) # return <class 'str'>
    print(type(print)) # return <class 'builtin_function_or_method'>
```

Operations :

Division :

```
    print(5/5) # return 1.0 ( a float )
                # if we want to return a intiger we need to do '//' instead of
'/'
    print(5//5)
```

power :

```
    print( 5**5 ) # the operator '**'
```

variable declaration :

```
    age = 21
    print(age) # output gonna be : 21
```

Dara structures :

string :

```
name = 'ashraf khabar'
print(name) # output gonna be : ashraf khabar
name2 = "he's so good"
print(name2) # output gonna be : he's so good
```

element of string :

```
name = 'ashraf khabar'
print(name[0]) # output gonna be : a
print(name[1]) # output gonna be : s
print(name[2]) # output gonna be : h
print(name[3]) # output gonna be : r
print(name[4]) # output gonna be : a
print(name[5]) # output gonna be : f
print(name[-1]) # the last element 'r'
print(name[-2]) # before the last element 'a'
```

slice of string :

```
name = 'ashraf khabar'
print(name[0:6]) #output : ashraf
```

All these things doesn't change the first variable `name` , if we want to change it we need to do what we called `affectation` :

```
name = 'ashraf khabar'
print(name) # print ashraf khabar

name = name[0:6]
print(name) # print just ashraf
```

concatenation :

```
name = 'ashraf '
name2 = 'khabar'
print(name+name2) # print ashraf khabar
```

times string :

Times string means repeating a string `n` time by the operator `*` :

```
name = 'ashraf '
print(name*4) # print ashraf ashraf ashraf ashraf
```

string formatting :

What about other useful way to print a string concatenation :

```
name1 = 'ashraf'
name2 = 'khabar'
print('the fist name is {0} and the last name is {1}'.format(name2,name2))
```

And we got exactly the same thing as `print(the first name is,name1,the last name is ,name2)` . And in this kind of `format` we can define the number of digits after coma we want to show using the syntax `.:n` :

```
number1 = 3.252
number2 = 1.252
print('the fist number is {0.:2} , and the second is
{1.:3}'.format(number1,number2))
# the number of digits after the coma we want to show .
```

Also,we can do it in other way as :

```
number1 = 3.252
number2 = 1.252
print(f'the first number is {number2:.3} and the second {number1:.2}')
```

string methods :

As we said all items in python are objects and classes , so `string` is a class and has a methods :

- Uper case and Lower case :

  ```
  name = 'ashraf '
  print(name.upper())
  str = 'ASHRAF'
  print(str.lower())
  ```

- Split :

```
name = 'ashraf,khabar,21'
print(name.split(',')) # output : ['ashraf', 'khabar', '21']
```

- length od a string :

```
name = 'ashraf'
print(len(name)) # output : 6
```

lists :

There is no difference between `string` and `list` except that `string` a specific case of `list` :

```
lst = [1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9]
print(lst) # outpust : [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

For manipulation of the `list` is same as what we did in the `string` , using the `[n]` to have access to the element number `n+1` of the `list` .

I can use some methods like `epend(n)` to add an element in the las the sequence or the method `pop()` to remove an element :

```
lst = [1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9]
print(lst) # outpust : [1, 2, 3, 4, 5, 6, 7, 8, 9]
lst.append(10)
print(lst) # outpust : [1, 2, 3, 4, 5, 6, 7, 8, 9 ,10]
lst.pop()
print(lst) # outpust : [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

And if i want to remove a peticular element from the list is simply using the method `remove(n)` :

```
lst = [1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9]
lst.remove(6)
print(lst) # outpust : [1, 2, 3, 4, 5, 7, 8, 9]
```

We have also the method `del( name_of_list[n] )` , in order to delete an element using the index .

We can make a `list` of `string` like :

```
brench = ['ginf1' , 'ginf2' , 'ginf3']
# and I can add a digit or a number if I want iven if it doesn't make any sens
brench.append(10)
```

Standard input :

We use the method `input` :

```
input('what is your name :')
# the output is : what is your name , and you can add what ever you want
```

And if we want to have the return value of the input we just need the affect these value to a variable :

```
name = input('waht is your name')
print('your name is',name) # output : the name what we entered
```

if statement :

if :

The syntax is a little different from other languages like `C++`,`java` or `php` .

```
age = int(input('enter your age :'))
if age > 18:
     # instructions
     print('You are major . ')
```

❗ : `int()` is for casting the return value to int and not something else .

else :

The opposite of `if` :

```
age = int(input('enter your age :'))
if age > 18:
     # instructions
     print('You are major . ')
else:
     print('You are not major')
```

elif :

The synonymy of `else if` in other languages :

```
age = int(input('enter your age :'))
if age > 18:
     # instructions
     print('You are major . ')
elif age > 100:
     print('Seriously ??')
else:
     print('You are not major')
```

For loop:

We do the same thing as other languages :

```
branches = ['ginf1','ginf2','ginf3']
for branch in branches:
  print(branch)
```

and

```
branches = ['ginf1','ginf2','ginf3']
for branch in branches[0:1]:
  print(branch)
```

A simple code:

A simple code for `if statement` :

```
first_name = input('enter your first name : ')
second_name = input('enter your second name : ')
age = int( input('enter your age : ') )
filiere = input('enter your branch : ')

if age < 18:
    print("You don't belong to here")
    if filiere != 'ginf1' or filiere != 'ginf2' or filiere != 'ginf3':
        print("You don't belong to here or You enter a wrong branch")
else:
    if filiere != 'ginf1' or filiere != 'ginf2' or filiere != 'ginf3':
        print("You don't belong to here or You enter a wrong branch")
    else:
        print(f'your name is {first_name} {second_name}')
        print(f'age : {age}')
        print(f'filiere : {filiere}')
        print(f'Welcome {first_name}')
```

While loops :

```
age = 25
num = 0

while num < age :
  # the code will be here
  num = num + 1
  print(num)
```

Ranges :

```
for n in range(10) : # generate numbers from 0 to 9
  # do somthing
```

instead of :

```
numbers = [0,1,2,3,4,5,6,7,8,9]
for number in numbers :
  # do somthing
```

- we can do also :

```
for n in range(3,10) : # generate numbers from 3 to 9
  # do somthing
```

- we can do also :

```
for n in range(3,10,2) : # generate numbers from 3 to 9 by the step 2
  # do somthing
```

- we can do also :

```
burgers = ['poulet' , 'VH' , 'Nuggets','fish']
for n in range(len(burgers)) :
  print(n , burgers[n])
```

Functions :

- ```
  def greeting():
    print("hello world")
  greeting()
  ```

- ```
  def factorial(Number) :
      fac = 1
      i = 1
      while i <= int(Number) :
          fac = fac * i
          i = i+1
      return fac
  number1 = input("Give a number : ")
  Factorial = factorial(number1)
  print(f'The factoriel of {number1} is {number1}! = {Factorial}')
  ```

Dictionaries :

- ```
  person = {"name" : 'ashraf khabar' , "age" : 22 , "branch" : 'GINF2'}
  print(person)
  ```

- ```
  person = {"name" : 'ashraf khabar' , "age" : 22 , "branch" : 'GINF2'}
  print(person['name'])
  print(person['age'])
  print(person['branch'])
  ```

- how to get the keys (values):

  ```
  person = {"name" : 'ashraf khabar' , "age" : 22 , "branch" : 'GINF2'}
  print(person.keys()) # person.values
  # dict_keys(['name', 'age', 'branch'])
  print( list( person.keys() ) )
  # ['name', 'age', 'branch']
  ```

- ```
  person = {"name" : 'ashraf khabar' , "age" : 22 , "branch" : 'GINF2'}
  vals = list(person.values())
  print(vals)
  # ['ashraf khabar', 22, 'GINF2']
  print(vals.count('ashraf khabar'))
  # 1
  print(vals.count('age'))
  ```
  ```

```
# 1
print(vals.count('A'))
# 0
```

- other way to define a dectionary :

```
from builtins import dict
person = dict( name ="ashraf khabar" , age = 22 , branch = "ginf2")
print(person)
```

- In genrale :

```
def present(dictionary):
    for key,val in dictionary.items():
        print(f'The student {key} in the branch {val}')

person = {}

while True:
    name = input("name : ")
    branch = input("branch : ")
    person[name] = branch

    other = input("add something ? [y/n] : ")
    if other == 'y' :
        continue
    else:
        break
present(person)
# output :
name : ashraf khabar
branch : ginf2
add something ? [y/n] : y
name : sami aouad
branch : ginf3
add something ? [y/n] : n
The student ashraf khabar in the branch ginf2
The student sami aouad in the branch ginf3
```

sorting :

```
nums = [1,2,6,4,3,6,9,1,0,3,6,4,1]
print(nums)
print("AFTER SORTING")
print(sorted(nums))

#
```

```
[1, 2, 6, 4, 3, 6, 9, 1, 0, 3, 6, 4, 1]
AFTER SORTING
[0, 1, 1, 1, 2, 3, 3, 4, 4, 6, 6, 6, 9]
```

sets :

```
nums = [1,2,6,4,3,6,9,1,0,3,6,4,1]
print(nums)
print("AFTER SETS")
print(set(nums))

#
[1, 2, 6, 4, 3, 6, 9, 1, 0, 3, 6, 4, 1]
AFTER SETS
{0, 1, 2, 3, 4, 6, 9}
```

Classes :

```
class Person:
def __init__(self):
    self.name = "ashraf"
    self.surname = "khabar"
    self.age = 20
P1 = Person()
print(f'name is {P1.name} , surname is {P1.surname} , age is {P1.age}')
```

The init function :

```
class Person:
  def __init__(self, name, surname, age):
      self.name = name
      self.surname = surname
      self.age = age

  def display(self):
      print(f'name : {self.name} , surname : {self.surname} , age : {self.age}')
  P1 = Person("ashraf", "khabar", 20)
  P2 = Person("sami", "aouad", 20)

  P1.display()
  P2.display()
```

Methods & Attributes :

- we have instance methods and we have class level mothods , which means that instance method are specified for every instance meanwhile class level methods are for the hole class .

```
class Student:

  school = "ENSAT" # means tha all Students are in ENSAT

  def __init__(self):
    # .......

  def display(self):
    # .......

  # we will define a cummon method between all instances of this class
  @classmethod
  def displayScool(C):
  print(f'name of school is : {C.school}')

  @staticmethod
  def graduate(montion = 'very good'):
  print(f'the student has graduated by the montion of: {montion}')
```

Modules & Packages :

- file Student

```
    class Student:
    school = 'ENSAT'

    def __init__(self , name , prenom , cne , age ):
        self.name = name
        self.prenom = prenom
        self.cne = cne
        self.age = age

    def display(self):
        print(f'name : {self.name} | prenom : {self.prenom} | cne : {self.cne}
  | age : {self.age}')

    @classmethod
    def displaySchool(STA):
        print(f'name of school : {STA.school}')
```

- file main

```
    from Student import Student as S

    P1 = S("ashraf","khabar","DM12535",22)
```

```
P2 = S("sami","aouad","DM14235",22)
P3 = S("hassan","tihihit","DM13635",22)

classe = [P1.name , P2.name , P3.name]
print(classe)
```

- But inside : classes / Student

```
from classes.Student import Student as S

P1 = S("ashraf","khabar","DM12535",22)
P2 = S("sami","aouad","DM14235",22)
P3 = S("hassan","tihihit","DM13635",22)

classe = [P1.name , P2.name , P3.name]
print(classe)
```