

Python:
Data structures, control flow,
OO Programming, Regular
Expressions, System Programming

AAA-Python Edition



Plan

- 1- if / else , For, While
- 2- Lists, Tuples, List comprehensions
- 3- Dictionaries
- 4- Sets
- 5- Object Oriented Programming
- 6- Regular Expression
- 7- System Programming



If / else

 These statements are used to control which block of code to execute:

```
[1]
       2 if a>8):
          print("a is greater than 8") -
        else:
          print("a is not greater than 8")
    a is not greater than 8
[4]
      1 a=3
          print("a is greater than 8")
        elif a==3:
          print("actually, a=3")
      6 else:
          print("a is not greater than 8")
    actually, a=3 🗻
```

If the "condition" is true
(the corresponding expression
Is Evaluated to True), then
the if "clause" is executed
(the if block)

The condition was false, so The "else" clause was executed

The "elif" clause is executed, If its condition is true



While

 This statement is used to control how many times a block of code has to be executed:

```
[21] 1    i=j=1
    while(i>0):
        print("**** execution number "+str(j)+"***" )
        i=float(input("Give a float value for i: "))
        print("Last given i =",i)
        j=j+1
```

```
Give a float value for i: 7.3

Last given i = 7.3

**** execution number 2***

Give a float value for 100

Last given i = 0.0
```

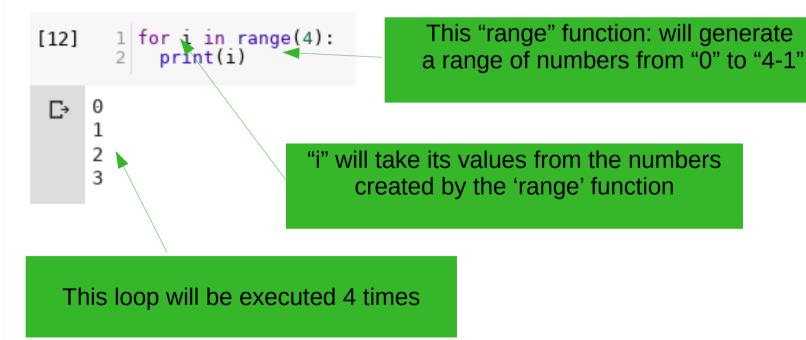
While the condition is true, the block code will be executed.

In this loop, the block has been executed 2 times



For

 This statement is used to execute a block of code a certain number of times

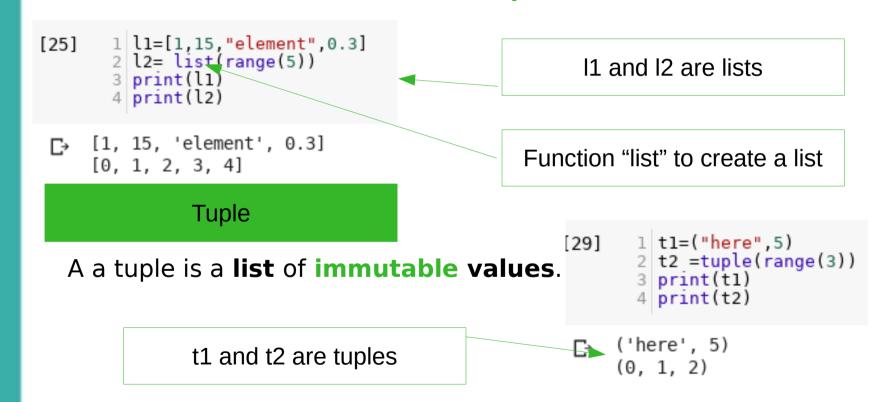






List

A list is a value that contains multiple values.





Lists and tuples (suite)

Modifying the value of the element of index 0 (first element)

```
1 l1=[2,5,9]
2 l1[0]="first"
3 print("l1=",l1)
4 print("t1[0]=",t1[0])
5 t1[0]=5
```

Access to the first element

Trying to modify the value of an element of a tuple



Some operations with lists

```
[60]
       1 l1=list(range(-5,2))
       2 print("l1=","l1)
                                                   a slice: values
        l2=list(range(7,20,3))
                                             From index 2 to index 4
        print("l2= ",l2)
         l3=l1[2:4]
                                                         Concatenating two
       8 print("l3= ",l3)
                                                                 lists
      10 del(l3[0]);print ("l3=",l3)
      12 | 14=13+[10,11]; print("14=",14)
      13
      14 l5=7*[2];print("l5=",l5); print("l5 has",str(len(l5))+" elements")
      15
      16
    ll= [-5, -4, -3, -2, -1, 0, 1]
                                                     Number of elements
    l2= [7, 10, 13, 16, 19]
                                                            of a list
     13= [-3, -2]
    l3= [-2]
    14= [-2, 10, 11]
     15= [2, 2, 2, 2, 2, 2, 2]
    15 has 7 elements
```



Some operations with lists (suite)

Iterate through list

```
1 ll= list("ABC")
   [70]
              2 for i in l1:
                     print(i)
                                                                              "not" with "in"
                 for i in range(len(l1)):
    print(str(i)+"- "+l1[i])
                                                                    Functions "min" and "max"
                 if 'G' not in ll:
                     print ("G is not in l1")
             10
                                                 1 l1=list(range(2,10,5))
2 l2=list(range(5,25,9))
3 print(l1);print(l2)
4 print("The greatest value in l1=",max(l1))
5 print("The smallest value in l2",min(l2))
                                        [20]
     Г≯
           В
                                                    x,y=l1
                                                    print(x,y)
           2 - C
           G is not in l1
                                              [2, 7]
                                         ₽
                                              [5, 14, 23]
  Affecting list values
                                              The greatest value in l1= 7
  To multiple variables
                                              The smallest value in 12 5
                                              2 7
[By Amina Delali]
```



List comprehensions

1 l1=[x for x in range(4) if x!= 2]

Filtering elements

[22]

2 print (l1)



List methods

Finding an element in a list

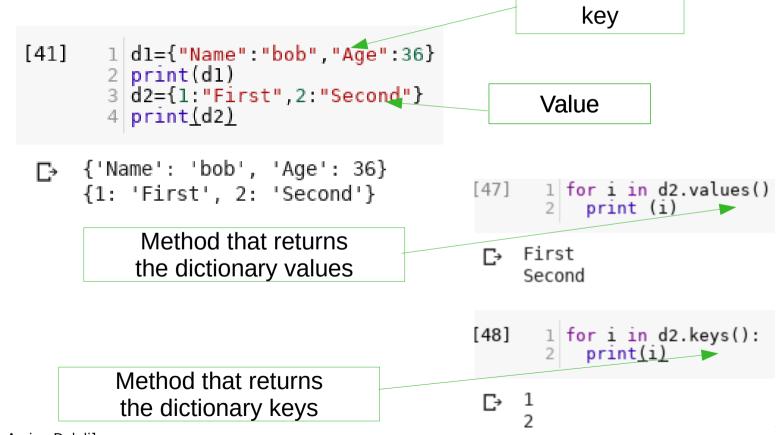
A list has some methods. We will talk about methods later.

```
Add an element to the
             1 ll=list("LETTERS")
     [33]
                                                        end of a list
             2 print(ll.index("R"))
                                     1 ll.append("G")
      ₽
                                     2 print(l1)
                                   ['L', 'E', 'T', 'T', 'E', 'R', 'S', 'G']
  [35]
         1 ll.insert(5, "H")
         2 print(l1)
                                       Insert an element at a certain position
       ['L', 'E', 'T', 'T', 'E', 'H', 'R', 'S', 'G']
                              [36]
                                     1 | l1.remove("T")
 Remove an element
                                     2 print(l1)
       from a list
                                   ['L', 'E', 'T', 'E', 'H', 'R', 'S', 'G']
[By Amina Delali]
```



Dictionaries

A dictionary is a list of values with corresponding keys



[By Amina Delali]



Dictionaries (suite)

```
for i in d2.items():
    print(i)
    k, l=i
    print(k,l)
```

```
C→ (1, 'First')<sup>A</sup>
1 First
(2, 'Second')
2 Second
```

The key doesn't exist so a default value is given

A key is created with a default value

The key already exists, So no other key is created

[By Amina Delali]

Method that returns the dictionary items: pairs of key,value

```
print(dl.get("Name"))
print(dl.get("name", "Smith"))
[56]
              The key exists, its value
     bob
   Smith
                      Is returned
        d2.setdefault(3,"third")
[63]
        2 print(d2)
    {1: 'First', 2: 'Second', 3: 'third'}
[64]
       1 d2.setdefault(3,"other element")
       2 print(d2)
    {1: 'First', 2: 'Second', 3: 'third'}
```



Sets

A set is a list of distinct values.

```
1 print(s1 & s2)
[82]
```

{1}

Intersection between s1 and s2

Elements in S1 and not in **S2**

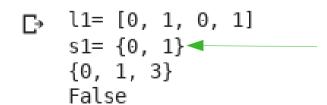
```
[84]
        1 print(s1-s2)
       2 print(s1<=s2)
     \{0, 3\}
     False
```

Is s1 a subset of

S2

```
1 l1=list(range(2))+list(range(2))
2 s1=set(l1)
3 print("l1=",l1)
4 print("sl=",sl)
5 sl.add(3);print(sl)
6 s2=set(list(range(1,3)))
7 print(s1.isdisjoint(s2))
```

The duplicates are eliminated







Classes

- In Python, we can define "classes": a defined prototype that encapsulates data and the functions to operate on them.
- An instance of a class is called an "object". We already used objects when we used lists, sets and dictionaries.

```
Name of the class
       1 #definition of class MyTable
                             Called when creating
                                                          To indicate that elements of
         class MyTable:
           ll=0 an object of that class

def init (self, name, length=0):
    self.length=length
                                                          range are not the elements
                                                                     of the list
              self.name=name
              self.myList=[None for fin range(length)]
 Data
attribute
            # to be sure that the lenght represents the actual list length
           def validL(self):
              self.length=len(self.myList)
                                                       A method (a function
                                                              attribute)
          # print the type of the list
           def myType(self):
      15
              print("I am a TABLE 1 ")
[By Amina Delali]
```



5- Object Oriented Programming

Classes

```
18
     # insert doesn't accept negative values or values greater than length
     def insert (self,ind,val):
19
    self.validL()
20
       if ind >= self.length :
21
                                                                  A comment
         print("The given index: "+str(ind)+
22
               " exceeds the table length: "+str(self.length))
23
24
       elif ind < 0:
         print("The given index: "+str(ind)+" is negative")
25
26
       else:
         self.myList.insert(ind,val)
27
         print("The value has been inserted at the index"+str(ind) )
28
29
30
                                                             Each time we use
31
     # print myList and the length attribute
     def printme(self):
32
                                                             myList, we ensure
     self.validL()
33
                                                               that length==
       print(self.name+" ("+str(self.length)+"): ",end=",")
34
35
       for i in self.myList:
                                                                 len(myList)
         print (i,end=" ")
36
       print(" ")
37
38
39
      # append a new element at the end of the
                                                      Optional attribute for
     def add (self, val=None):
40
                                                          function print
     self.validL()
41
       self.myList=self.myList +[val]
42
       self.length=self.length+1
43
```



5- Object Oriented Programming

Classes

```
# MyTable2 inherit MyTable1 functions
   class MyTable2(MyTable):
     def myType(self):
   print("I am a TABLE 2 ")
   class MyTable3(MyTable):
     def myType(self):
       print("I_am a TABLE 3 ")
   print("#####")
  a= MyTable("table1",3)
a.insert(22,"B")
   a.insert(2, "A")
   a.printme()
62 a.add(2)
                    Object creation(
   a.printme()
                    Call of init
64 a.add()
65 a.printme()_
66 a.length=8
   print("a.length=",a.length)
   a.printme()
69 print("#####")
```

A subclass of class MyTable
Inherits all its methods
and data attributes:
We can use them without
redefining them.

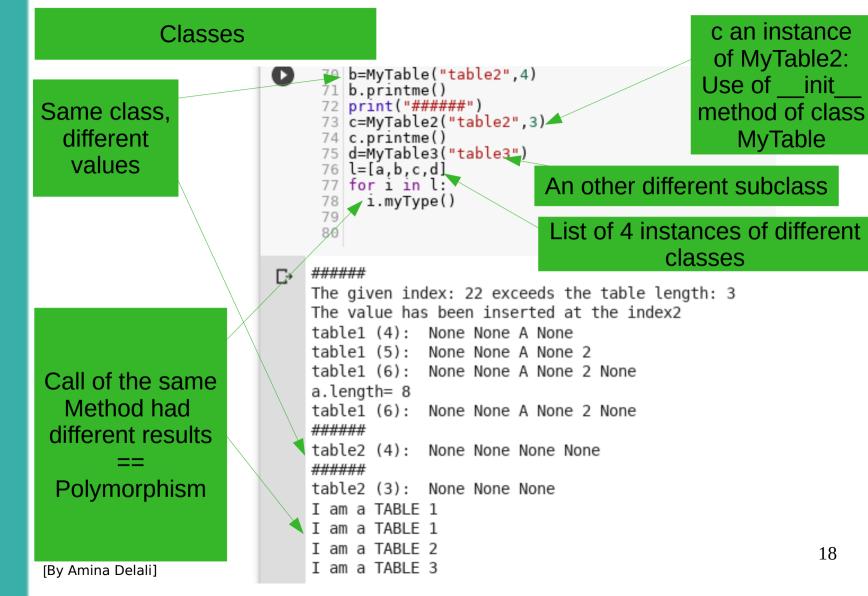
Redefine "myType"
(already defined in MyTable)
It's overriding myType
parent class method

Call of a method

Access of a data attribute



5- Object Oriented Programming





Regular expressions

A year from 1970 to 2999

A regular expression is a description of pattern of text

```
Need of module re
[190]
        1 import re
        2 #creating a regex pattern object
         myReg=re.compile(r"([0-2][1-9]|30|31)-(0[1-9]|1[0-2])-(19[7-9][0-9]|2[0-9]{3})")
         myReg2=re.compile(r"[a]+")
        6 print(myReg.findall("It \starts from 11-02-2018 and ends at 25-09-2029."))
          res=myReq.search("It starts from 11-02-2018* and ends at 25-09-2029.")
         print("*"+res.group()+"*")
                                                                                     3 digits
      9 print(myReg.match("It starts from 11-02-2018 and ends at 25-09-2029."))
10 print(myReg2.findall("a string aa and aaaa"))
       11 print(myReg2.findall("my st\ring"))
     [('11', '02', '2018'), ('25', '09\', '2029')]
                                                          A month: composed of:
     *11-02-2018*
                                                     0 and a digit from 1 to 9 (0[1-9])
                                       One or
     None
           `aa', 'a', 'aaaa']
                                      more a(+)
                                                     1 and a digit from 0 to 2 (1[0-2])
 Search for the first
         date
```

19



7- System programming

System Programming

- We will focus on system programming in Colab.
- Some Python functions can be simply done on Colab.
- For example the bash commands: they can be used as they are by prefixing them by "!" or "%": !ls, !mkdir, !git, !pip, %cd ... etc

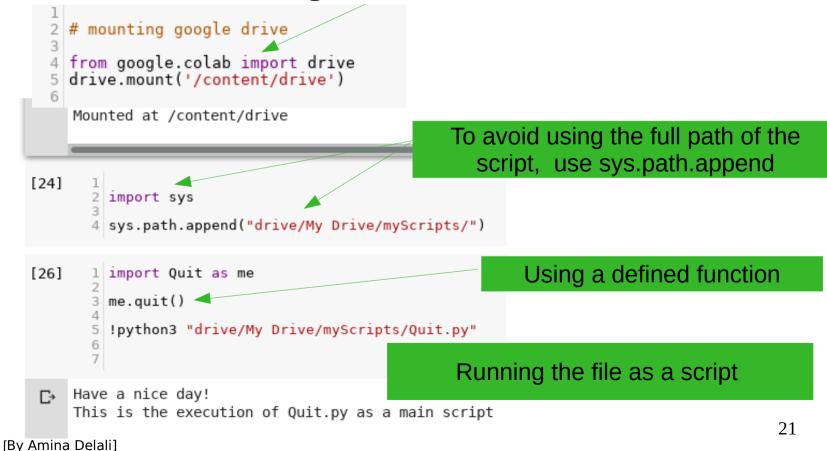
```
Running 'ls'
  1 import subprocess
                                                           using subprocess
    p = subprocess.run(["ls", "-l"])
   from google.colab import files
  5 myFile= files.upload()
                                                                Import a local
                                                                      file
             Hello.py
                                  Cancel upload
   Browse...
  1 import Hello as h
                                                         After selecting the
  2 h.sayHello()
                                                         Script file, import it
Hello.
 Welcome to School Of AI!
```



7- System programming

System Programming

- Second way of using a user defined script:
- We have to mount Google Drive





System Programming

7- System programming

of the file #printing the script content using cat command 2 print("-3 !cat "drive/My Drive/myScripts/Quit.py" 5 #printing the script content using path.join an open functions 6 import os print("\n-----9 myFile=os.path.join("drive", "My Drive", "myScripts", "Quit.py") 10 f=open(myFile,'r') lines=f.readlines() Creating the file path 12 f.close() 13 for l in lines: print(l,end="") Open and read the file content into a list Ľ⇒ def quit(): print ("Have a nice day!") if name == " main ": Print the list print("This is the execution of Quit.py as a main script\n") def quit(): Use if name ==" main " for the code print ("Have a nice day!") To be executed if the module is not imported And run as a script if name == " main ": / print("This is the execution of Quit.py as a main script\n") 22 [By Amina Delali]

Using '!cat' to print the content



References

- Duchesnay Edouard and Löfstedt Tommy. Statistics and machine learning in python release 0.2. On-line at ftp://ftp.cea.fr/pub/unati/people/educhesnay/pystatml/M1_IMSV/StatisticsMachineL earningPythonDraft.pdf. Accessed on 23-09-2018.
- Python Software Foundation. The python standard library. On-line at https://docs.python.org/3.6/library/index.html. Accessed on 23-09-2018.
- Joel Grus. Data science from scratch: first principles with python. O'Reilly Media, Inc, 2015.
- Chandat Sunny. Learn Python in 24 Hours. CreateSpace Independent Publishing Platform, 2016.
- Al Sweigart. Automate the boring stuff with Python: practical programming for total beginners. No Starch Press, 2015.



Thank you!

FOR ALL YOUR TIME