Intro

July 22, 2020

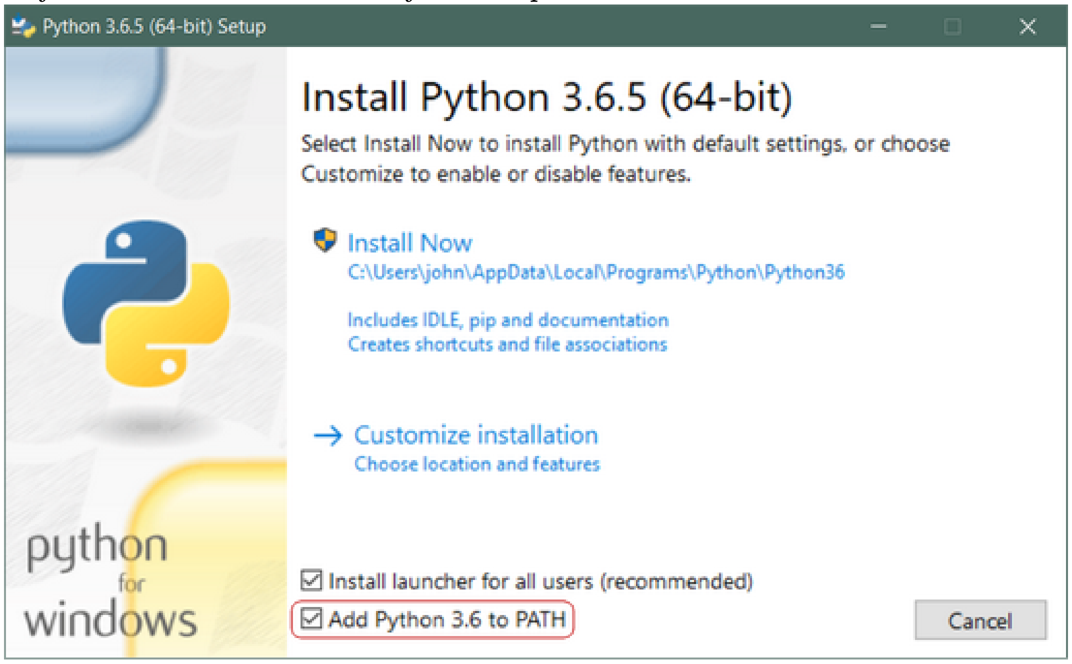
Contents

Chapter 1

How to install Python3

1.1 For Windows

- Download the latest Python release installer for your OS from here
- Launch the installer
- **Pay Attention to select add Python to path**



Python 3.6.5 (64-bit) Setup

Install Python 3.6.5 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\john\AppData\Local\Programs\Python\Python36

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

- Wait until it's done

1.2 For macOS:

You've two alternatives:

1. Proceed as indicated with windows (but obviously download the installer for mac)
2. If you've brew installed just open a terminal and write "*brew install python3*"

1.3 For Linux:

Python3 should be already available on most Linux of distribution, in the rare case you don't have it you can install it by writing on a terminal "*sudo [name of the packet manager] install python3*" (For example in

Ubuntu "*sudo apt-get install python3*")

In any case to check Python wheter has been correctly installed open a terminal and type "*python3*" you should see something like this

```
matteo@matteo-desktop ~> python3
Python 3.6.8 (default, Aug 20 2019, 17:12:48)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Chapter 2

How to install Sublime Text (and set it up for python development)

2.1 Installation

Go on the Sublime website and follow the installation guide for your OS Setup for Python (advanced and optional) — The following part is not mandatory Sublime is a nice editor out of the box but can be a lot more useful with some addons, below it's explained how to install two really useful plugins. You can look online to find even more addons and customize Sublime according to your needs. From here proceed at your own risk!

Open Sublime, click on the botton letf of the window and select console. A small window should appear at the bottom, copy and paste this code:

```
import urllib.request,os,hashlib; h = ‘6f4c264a24d933ce70df5dedcf1dcaee’ +
‘ebe013ee18cced0ef93d5f746d80ef60’; pf = ‘Package Control.sublime-package’; ipp = sub-
lime.installed_packages_path(); urllib.request.install_opener( urllib.request.build_opener( url-
lib.request.ProxyHandler()) ); by = urllib.request.urlopen( ‘http://packagecontrol.io/’ + pf.replace(’
‘,’%20’)).read(); dh = hashlib.sha256(by).hexdigest(); print(‘Error validating download (got %s instead of
%s), please try manual install’ % (dh, h)) if dh != h else open(os.path.join( ipp, pf), ‘wb’ ).write(by)
```

and then press enter

Once the installation is finished go on Preferences and select “Package Control”. From the list select “Package control:Install package”, type “Anaconda” and select the first result

Once the installation is finished go on Preferences and select “Package Control”. From the list select “Package control:Install package”, type “Anaconda” and select the first result

Once the package is installed go on “Preference”>“Package Settings”>“Anaconda”>“Settings-Default” and on line 251 set ***“anaconda_linting”: false***

Now you should've autocompletion enabled and the editor will show you the documentation of the function you're using

Go on Preferences and select “Package Control”. From the list select “Package control:Install package”, type “SideBarEnhancements” and select the first result

Chapter 3

How to launch a Python script

Launching a Python script is quite easy, these are the steps to follow:

1. Open a new terminal (Command line for Windows)
2. Navigate to the folder , "*cd name of the folder*" to open a folder , "*dir*" top list the file in the folder ("*ls*" for Linux)
3. type "*python [name of the file].py*"

For the rest of the course we will always launch our Python scripts in this way

Chapter 4

Exercises

4.1 Exercise 0

Before anything you want to define the "main" as you would do in C. In Python we can do this like it's shown below

```
In [4]: if __name__=="__main__":
            #This is the main
            pass
```

Remember that in Python indentation take places of the bracket, we will see going on with the exercise what this means

4.2 Exercise 1

Print
Printing in python is really easy, you've just to type ~~~ print ("what you want to print") ~~~ An example is shown below

```
In [5]: if __name__=="__main__":
            #This is the main
            print ("Sentence to print")
```

```
Sentence to print
```

Now try to write and launch your Python script and print some words/sentences, you will see that contrarily to C python will print the content of each *print()* in a new line

4.3 Exercise 2

Print with placeholder
In a lot of case you want to print things like

```
the average of the values is: 5.4
```

In general you would like to print a string and the value of a variable or multiple variable. There a lot of way to print in Python, we will see just 2 of them.
First method
The first method consists in using placeholder with "%" like you would do in C, there are different type of plceholder, you will mainly use the following:

- %s to print string
- %d to print integer
- %f to print float
- %e to print in exponential form

So an example could be:

```
print("My name is %s" %myName)
```

In case of floating point number you can specify the number of digits after the comma by writing "***%.nf***" where n is the number of digits so for example if we want to have 3 digits we would write "%.3f". This method will work also with "%s" but in that case the number will specify the number of **characters** that will be used (including the "."). If you want to specify multiple variable you will write

```
print(" %s  %d %f" %(a,b,c))
```

Second method
The second method use the letter "f", in this case the placeholder is the same for every type of variable and is "*{variable}*", so the example we saw before would become:

```
print(f"My name is {myname}")
```

In this case if we eant to print multiple variable we would write:

```
print(f"{a} {b} {c}")
```

Whit this method you can also specify the total number of digits by writing "***{:.n}***" where n is the **total** number of digits so if for example we want to write pi with 3 decimal digits we would write

```
pi=3.15169265
print(f"{pi:.4}").
```

Now try to use both this method to write a Python script that give as output your nam, your age and your birthday in this way:

```
My name is Python and I'm 28 years old, I was born the 20/02/1991
```

4.4 Exercise 3

Math operators
In python we can use a lot of mathematical operators let's check which they are:

- "+" plus
- "-" minus
- "/" slash
- "//" double-slash
- "*" asterisk
- "%" percent
- "<" less-than
- ">" greater-than
- "<=" less-than-equal
- ">=" greater-than-equal
- "==" equals

To try to understand what they do let's write a python script with one operation for each of them, something like the one below:

```
In [10]: if __name__=="__main__":
             print (f"2+3={2+3}")
```

```
2+3=5
```

4.5 Exercise 4

Ask user input
The function to ask inputs from user in Python is simply "*input()*". For example if we want to ask to the user to insert anumber we would write:

```
number=input(" Please write a number")
```

That is quite easy so let's try to write a script that ask the user his name and his age and his height and reply somthing like:

```
So you're Mt. Everest you're 60000 years old and you're tall 8848 m...
```

4.6 Exercise 5

Reading and Writing file Reading and writing file in Python is really easy, the function you need are the following:

open()
this function returns an oject of the type file

```
##Example

#Open file in reading mode
f=open('myFile.txt')

#Open file in writing mode
f=open('myFile.txt','w')

#Open file in append mode
f=open('myFile.txt','a')
```

read()
this function return the whole content of the file as a string

```
##Example

#Open and read the content of the file "myFile.txt"
f=open('myFile.txt')
fileContent=f.read()
```

readline()
this function return one line of the file as a string

```
##Example

#Open the file "myFile.txt" and read one line
f=open('myFile.txt')
fileLine=f.readline()
```

write() this function write on a file

```
#Open the file "myFile.txt" and write one line
f=open('myFile.txt','w')
f.write('line to write')
```

close() this function closes the file

```
#Open the file "myFile.txt", write one line and close the file
f=open('myFile.txt','w')
f.write('line to write')
f.close()
```

As exercise create 2 file: “original.txt” with some random conten and an empty file called “copy.txt”, then write a python script to open the first file, shows its content on the terminal and write the content of “original.txt” in “copy.txt” but add at the beginning “the content of the original file is:”. So the result should look like this.

original.txt

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.”

copyt.txt

“The content of the original file is:
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.”

4.7 Exercise 6

If-elif-else

The easiest way to show how *if-elif-else* works is by making an example. So let’s see a script that asks to the user two number and tells if the first is greater, smaller or equal to the second

```
In [1]: if __name__=="__main__":
            #Receive two number as input
            a=int(input('Write a number\n'))
            b=int(input('Write another number\n'))

            #First case a>b
            if a>b:
                print(f"{a} is greater than {b}")
            #Second case a<b
            elif a<b:
                print(f"{a} is smaller than {b}")
            #All the other case
            else:
                print(f"{a} equals {b}")
```

```
Write a number5
Write another number8
5 is smaller than 8
```

It’s quite self explaying and we can annidate multiple *if-elif-else* just remember to use the correct indentation, most code editors will do it for you but pay attention to it. You can create complex condition with “and” and “or”.
As exercise write a script that takes as input a number and chech whether it’s a multiple of 2 and 3.

4.8 Exercise 7

List, for-loop, while-loop
As you've seen in the slides in python the "array" type is called "list" and there it has some particular function related to it:

- *list[i]*
  Return the i-th element of the list
- *list.append(x)*
  Add an item to the end of the list.
- *list.insert(i, x)*
  Insert an item at a given position. The first argument is the index of the element before which to insert
- *list.remove(x)*
  Remove the first item from the list whose value is x. It is an error if there is no such item.
- *list.pop([i])*
  Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list.
- *list.index(x)*
  Return the index in the list of the first item whose value is x. It is an error if there is no such item.
- *list.count(x)*
  Return the number of times x appears in the list.
- *len(list)*
  Return the number of element in the list

Remember that list in python start at 0
The easiest way to show how the *for-loop* and the *while-loop* works is by making an example. So let's see a script that creates a list with some number and calculates their sum and their product:

```
In [19]: if __name__=="__main__":
             numbers=[1,2,3,4,5]
             list_len=len(numbers)


             ##For-loop for the sum
             #set sum to 0
             sum_of_numbers=0
             for i in range(list_len):
                 #takes the previous value of the sum and add the number at the place i
                 sum_of_numbers=sum_of_numbers+numbers[i]
                 #the more "pythonic" way to do it is
                 #sum_of_numbers+=numbers[i]

             ##While-loop for the product
             #set counter to 0
             i=0
             #Set product to 1
             product_of_numbers=1
             while i<list_len:
                 #Multiply the old value for the number at place i
                 product_of_numbers=product_of_numbers*numbers[i]
```

```
                #Update counter
                i=i+1
                #the more "pythonic" way to do it is
                #product_of_numbers*=numbers[i]
                #i+=1
            print(f"The sum of the numbers in the list is {sum_of_numbers}")
            print(f"The product of the numbers in the list in {product_of_numbers}")
```

```
The sum of the numbers in the list is 15
The product of the numbers in the list in 120
```

There is another method to write the *for-loop* that is a little easier that is

```
for item in list:
```

Using this method for the previous example it becomes

```
In [21]: if __name__=="__main__":
             numbers=[1,2,3,4,5]
             sum_of_numbers=0
             for item in numbers:
                 sum_of_numbers+=item
             print(f"The sum of the numbers in the list is {sum_of_numbers}")
```

```
The sum of the numbers in the list is 15
```

As exercise write a script to calculate the average, the max and the min of a list of number

4.9 Exercise 8

Dictionaries
An really useful datatype of python is “dict”. Dictionaries are collections of couples of key/value, key are of type string and they are unique for each dict, the value of a dict can be of whatever type :int,float,string,list or even other dict. Let’s see an example of dict and what function can be used on a dict.

```
In [3]: if __name__=="__main__":
            building={ "type": "House",
                      "properties": {
                          "id": 1,
                          "RCODE": "RM3",
                          "NAME": "Mezzanone",
                          "TOWN": "Borgo Mezzanone",
                          "website": "https://www.ruritage.eu/RHH/Apulia",
                      },
                      "geometry": {
                          "type": "Point",
                          "coordinates":[ 15.7094462, 41.4205881 ]
                      }
                     }
            #Get the building type
            building_type=building['type']
            #or
            building_type=building.get('type')
```

```
        print(f"The building type is {building_type}")

        #Get the properties of the building
        building_properties=building['properties']
        print(f"building properties:{building_properties}")
        #to print it pretty you can  do
        #print("Building properties")
        #for key,value in building_properties.items():
        #    print(f"\t{key}: {value}")

        #Get the coordinates of the building
        building_coordinates=building['geometry']['coordinates']
        print(f"The coordinates of the building are:\nX:{building_coordinates[0]}\nY:{building_coor
```

```
The building type is House
building properties:{'id': 1, 'RCODE': 'RM3', 'NAME': 'Mezzanone', 'TOWN': 'Borgo Mezzanone', 'website'
The coordinates of the building are:
X:15.7094462
Y:41.4205881
```

As exercise write a script that ask as input the data needed to fill the dict defined below and print the dict as output

```
In [27]: personal_data={"Name":"",
                        "Surname":"",
                        "Birth":{
                            "Place of birth":"",
                            "Birthday":""
                        },
                         "age":""
                        }
```

4.10 Exercise 9

Functions
Functions are usefule to repeat the same operation multiple times without writing the same code multiple times. Let's see an example of a function

```
In [4]: def useless_function(argument):
            argument_v2=argument+argument
            return argument_v2
        if __name__=="__main__":
            x=useless_function(0)
            print(x)
            y=useless_function("IoT")
            print(y)
            z=useless_function([1,2,3])
            print(z)
```

```
0
IoTIoT
[1, 2, 3, 1, 2, 3]
```

As we see in this example we don't need to specify the type of the input of the function, python will understand it by itself (by the way this is a special case due to the fact that int,string and list all support the operator "+" in does **not** work)

As exercise write 4 function to: add, subtract, multiply and divide two given numbers and print the result

```
In [ ]:
```