Introduction:

Python is a general use programing language which supports object-oriented programing, functional programing among other programing paradigms. But you don’t know what the actual flask is that, and you shouldn’t. First things first, before digging into this let’s make this clear. This is not difficult, is not supposed to be, and you are not supposed to know things that are not yet in your head so let’s take it slow.

Now being serious what is python?

Yes, is indeed a programing language, or how I like to call it, magic itself. Just think for a second how many things can be achieved by touching some keys in your keyboard, even from very far away. Python is in fact the programming language used in google, YouTube and Instagram’s websites. This is also the programing language which runs inside the robots, yes AI ( Artificial Intelligence) runs on python. Among the programing languages, this is the most powerful, it can be used to create ANYTHING or to destroy EVERYTHING. Yes, is one of the main languages used in hacking, and also to protect us from it.

I’m sure ill get your attention as soon as we dig into it so let’s do that right now.

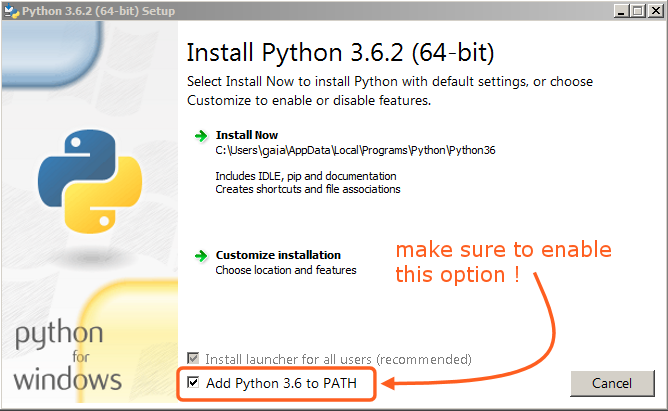
Installation

Inside the main folder there is a folder with all the complementary files needed for Installation and also, there is an embedded version in case you don’t want to install it.

I have included two main versions since windows 7 seems to be the biggest breach between users, and also between python versions. It it not possible to use the 3.9 version in windows 7 anymore. As a replacement, for those who have windows 7, the version 3.8.10 is included.

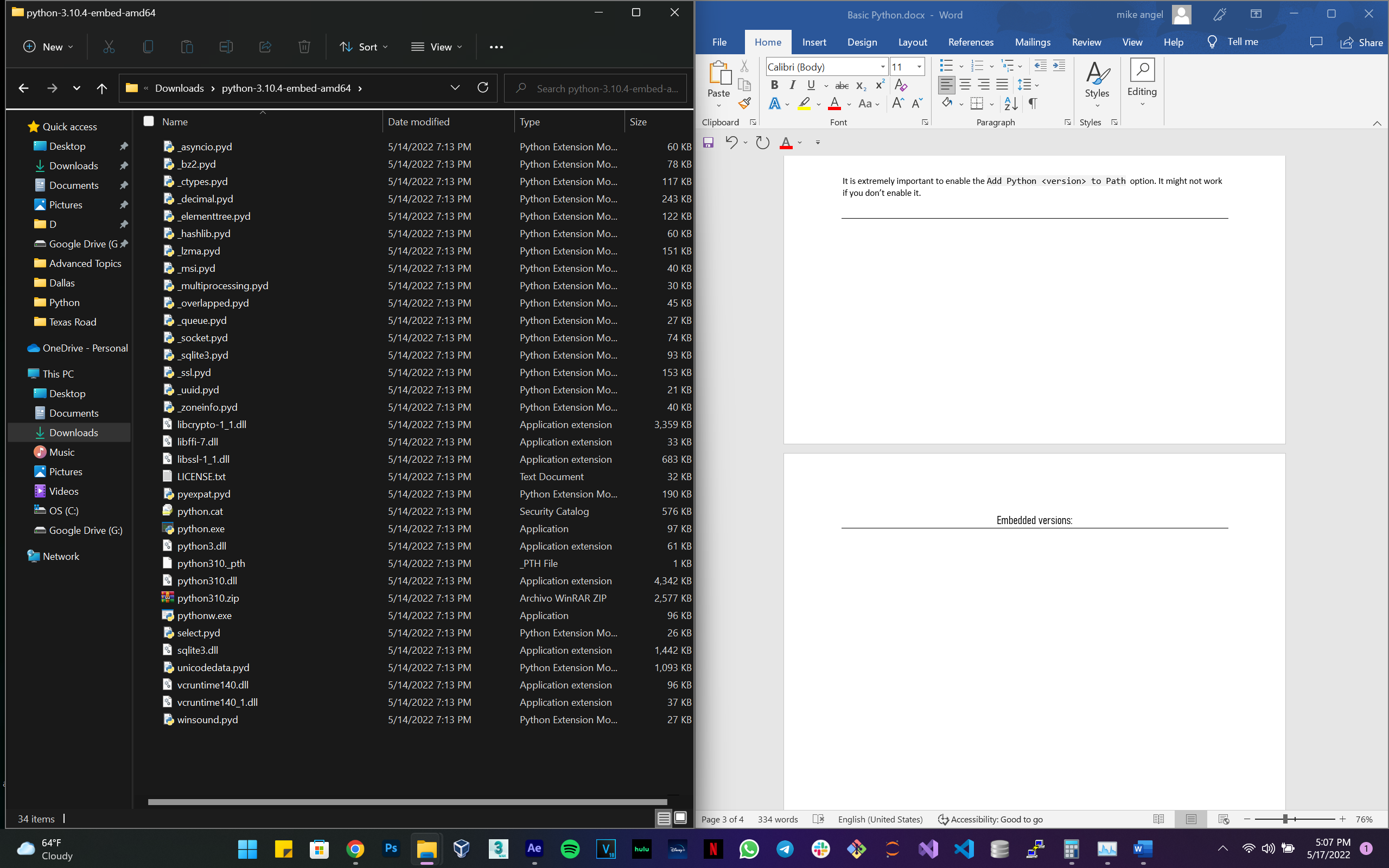
If they don’t work you can just go to python.org, which is the official website for python and get it there,

Installation for all versions:



It is extremely important to enable the Add Python <version> to Path option. It might not work if you don’t enable it.

Embedded versions:



The name of the file that you should open is <python.exe>. Just in case, open it as administrator.

Let’s Go

Once installed or opened the embedded version, you should be able to start programing. What? Yes sr. We are going to start programing right now. Let’s break the ice. For the embedded version you only have to open it, and for the installed version, you must open a session in the terminal. How?

Three steps:

* [Windows] + [R]
* [Enter]
* Type the command python (marked down in blue)

Once you do that this is what you get:

C:\Users\Mike Ehl>python

Python 3.9.5 (default, May 18, 2021, 14:42:02) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:

This Python interpreter is in a conda environment, but the environment has

not been activated. Libraries may fail to load. To activate this environment

please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.

>>>

When you see those three little side arrows, it means you’re in the right place. In the case of the embedded version, you will get this just by opening it

IMPORTANT

It is better to work with the installed version, please consider to use the embedded version just as last resource in case you’re not able to install it on your device, you might lose some features.

The ritual

There is an old ritual which has been passed generation to generation of programmers. You of course are not the exception. By going through this ritual, you break the ice, the fear of the ‘terminal’ and become a programmer. My typing the sacred words on the terminal you will be granted the power of changing everything with the movement of your hands, yes, the world will bend before you!!!!!!.

Ok that was too much I think, anyways:

Type this in your terminal and let the magic begin:

>>> print("Hello, World")

By typing print("Hello, World") you have written your first line of code, you made the computer reed on the screen the words Hello, World. This is usually the first line of code that every one of the great programmers that exist today, printed on the screen. So, this is the first time that you talk to your computer!!

Output:

>>> print("Hello World")

Hello World

>>>

IMPORTANT

Yes, as long as you change the text between the quotes, you can make your computer talk, and say what you want it to say. This line could’ve been anything else, but Hello, World is the classic one.

This is the first line, so this looks like a good moment to make it clear. Exerimentation is the key here, if you did already, GREAT! But if not, don’t be afraid, your system is not going to break (I think) so don’t take anything as an absolute, try modifying the lines and making your own.

The base of programing:

Before we jump on to the next commands, there is some very basic low-level info that I think is important to know. Programing is about giving an input to the computer, save it or modify it, and then return it to someone, either in the same way, or modified. That is the base of the base. This is why the first concepts we are going to see now are how to:

* Print
* Save
* Modify
* Read

The words print and read can be tricky so be careful. All those actions are being taken by the device, no by you. The device prints, saves, modifies, and read **to you** the information or data you give to it. This is not me, is just ca convention, something we all agree on because that is how it is since all this began.

Saving the data:

You already know how to make your computer print something on the screen. Now let’s see how can we save those letters contained in quotes we saw before, which by the way, those are called strings.

What we are going to see now is something known as variables. No they are not the same variables as in school LOL. Then what are they?

Variables 1.01:

A variable is a piece of RAM which is dedicated to hold the piece of data which a program tells it to.

What?

Yes ladies and gentlemen, it is the RAM the place where our data live. More specifically, In a chunk of the RAM, when we create a variable, or we just (type something without a variable) that goes somewhere in the RAM. The difference is that when we do it without a name (variable) it is stored somewhere, but we don’t know where. On the other hand, when we use a variable and assign a name to that chunk of ram where our data lives, we can read it later because we know where it is.

Let’s create one and then we can discuss the syntaxis:

>>> name = "Alice"

>>>

Here we have three elements:

* name : Variable
* = : Assignation Symbol(NOT EQUAL)
* “Alice” : Data

Here, name would be the name that the chunk of the same receives, and which will remember so when we call name later, it will remember that the data is the <string> “Alice”

The = symbol doesn’t mean equal; the equal symbol is represented by two equal symbols (==).

Why?

Very funny, when the creators where creating, (LOL) they thought: the chunk of RAM called name is equal to Alice right, so let’s do equal symbol and that’s it. They didn’t have more complex stuff at the time. Long story short, Assignation was born before Equality comparisons, so Assignation stole equality Symbol for herself. She is evil, I know, but she is the bigger sister.

If we then call the variable name, we’ll get its value in the RAM:

>>> name

'Alice'

>>>

There is more to variable, and we will cover it later, for now, this is the basic, enough to write your first script

Modifying the data:

Now let’s change Alice’s name. We could do that very easy by just rewriting the variable name with another value. Wat we aim to do here is extend Alice’s name, so the text string is longer, more concretely, Alice’s name and last name.

We could do that like this:

>>> name = 'Alice Synthesis'

>>> name

'Alice Synthesis'

>>>

But we are not “modifying” anything right? We are just overwriting on it. Let’s try something else. The idea is making two different strings, one will hold Alice’s first name, which is alice, and the other one will hold Alice’s last name.

>>> first\_name = 'Alice'

>>> last\_name = 'Synthesis'

>>>

Now, we can do something that we know from mathematics, a basic addition of two items that are the same type. Remember if you add one apple and one apple you can say “I have two apples”. If you add one apple and one tomato, they might look alike, but you have to still say “I have one apple, AND one tomato”. This means that you can’t add two different data types.

Now let’s build Alice’s name and call it:

>>> name = first\_name + last\_name

>>> name

'AliceSynthesis'

>>>

Ok but why is it all together? Well this is something that happens very frequently and I wanted to show you early so you don’t suffer later. Check Strings 1.01 for more info.

Strings 1.01:

First of all they are called strings, so you can deduce that they are not just a single item, they are a string of items, this is your first sequence and it has some awesome features and characteristics we’ll see later, but for now, you have to know this:

The string of characters known as “Alice” is actually the combination of the individual characters “A”+”l”+”i”+”c”+”e” . You can check that the string “Alice” , the result of the addition “A”+”l”+”i”+”c”+”e”, and the variable first\_name, are the same object . We can check that because everything that exists, whether is has a variable attached to it or not has an ID which we can find by using the id function. Functions are what we use to print, modify, read, and sometimes save, like the one that we saw at first print(“Hello, World”).

>>> id("Alice")

1889456280304

>>> id("A"+"l"+"i"+"c"+"e")

1889456280304

>>> id(first\_name)

1889456280304

>>>

Now that we saw this we can figure what the problem with the addition of the variables first\_name and last\_name is, we didn’t add a space at the endo of “Alice” or the beginning of “Synthesis”.

Now we have seen how to concatenate (add) strings we have seen how to modify a piece of data we have saved and also we’ve seen how to print it on the screen.

Now let’s see how ca we get a string from the user, and also save it to a variable.

Reading (getting ) the data from the user:

There are many ways we can get input from the user. And when I say input, I mean electricity because that is how computers work, we can do that with the mouse, touch pad or the keyboard. Back in the days the creators had saw that a keyboard is enough, and it was good. The function we are about to see is the basic way of getting input from a user on the terminal, the input() function. I know, what a plot twist.

Let’s just go and type that and see what happens:

>>> input()

''

>>>

When we call the function input() we are asked to type something, and press enter to confirm our string. Since I didn’t enter anything, an empty string was returned.

Now if we type something that will go somewhere in memory, but it wont be saved, and we don’t what that, so let’s save some input to some variables:

>>> name\_2 = input(“What is your name”)

Eugeo

>>>

This time the string is not returned, instead we can just continue as if nothing happened. But something did happen. Let’s open the variable name\_2 and combine it with a print().

>>> name\_2 = input("What is your name")

What is your nameEugeo

>>> print("Hello, my name is", name\_2 )

Hello, my name is Eugeo

>>>

Two things are to be noted here, we just printed something that we got form the user, AND, for some reason there is a space now between the first string and our name\_2 string . This is because of the way that the print() function works. Print takes as many arguments as you cant, string type arguments of course, all separated by a coma, and then it adds a space between each one. This is different than concatenate strings which means literally put a character after the other.

EX:

>>>print(“string 1 ”, “string 1 ”,“string 1 ”,)

string 1 string 1 string 1

**Practice:**

Create a variable with your name by asking you to input the name

Greet you from the terminal using concatenation or by separating values by coma

The numbers:

What if we need to work with some numbers? You will say, well, then we just pass in some numbers. Well, that is correct, but it might be a little bit different, you will see why right now and you will deduce what it’s going on super-fast:

>>> a = '5'

>>> a

'5'

>>> a\*2

'55'

>>>

So, since when 2 times 5 is 55? LOL. What you just saw is an operator, we can use those also to modify data like numbers (which we won’t be calling numbers anymore) or strings. The numbers are called differently depending on their structure, you already know that from mathematics right? When we are working with a whole number, we call that an integer they are also known as the <int> type, (by the way strings are known as the <str> type)

So the deal is, that what we have here in the variable <a> is actually a string, which is not the same as an integer. I will not try to explain anything, the difference can be spotted right away by the syntaxis, but also, there is an elephant in the room which is that the mathematical operations, must be done by using integers, because text, is text, and two times a text, well, is the same text repeated, right?

IMPORTANT:

Notice that the syntax difference is that integers are the only data type that don’t need any kind of syntactic specification, that is because they are first-borns in the programing world, I mean they are computers, they only talk in zeros and ones, so it makes sense that the second thing they learned was 2,3,4… Ok bad joke, but that’s how it was

Same example as before, but using integers:

>>> b = 5

>>> b

5

>>> b-2

3

>>> b\*2

10

>>>

As we can see, now we are able to make mathematical operations with the data stored in the variables. Yes, all normal operations work, addition, subtraction, multiplication, division, and exponential operations. Here is a little summary of them, and also the type of data they return:

>>> 2+5

7

>>> type(2+5)

<class 'int'>

>>> 2-5

-3

>>> type(2-5)

<class 'int'>

>>> 2\*5

10

>>> type(2\*5)

<class 'int'>

>>> 2\*\*5

32

>>> type(2\*\*5)

<class 'int'>

>>> 5//2

2

>>> type(5//2)

<class 'int'>

>>> type(5/2)

<class 'float'>

>>> 5/2

2.5

>>>

Several things are to be noticed from the last example. First thing is that we have a couple of operations to do with numbers ill name them in order()