


Introduction to python

Author	Initially designed by Guido Van Rossum and developed by python software foundation.
Implementation started in	1989
First version 0.9.0 released in	1991
Version 1.0 released in	1994
Version 2.0 released in	2000
Version 3.0 released in	2008
Version 3.8 released in	2019
Programming approach	Procedural and object oriented both
Platform	Cross platform
Inspired from	ABC language, Modula-3
Available IDEs	Anaconda, Thonny, Pycharm, Netbeans or Eclipse
Development Areas	Web development (server-side), software development, Mathematics, System scripting
Used by	Many large companies use the Python programming language include NASA, Google, YouTube, BitTorrent, etc.
Python logo	

Some Interesting points

1. The language is named after the BBC show **“Monty Python’s Flying Circus”** and has nothing to do with reptiles.
2. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
3. Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Installation of python

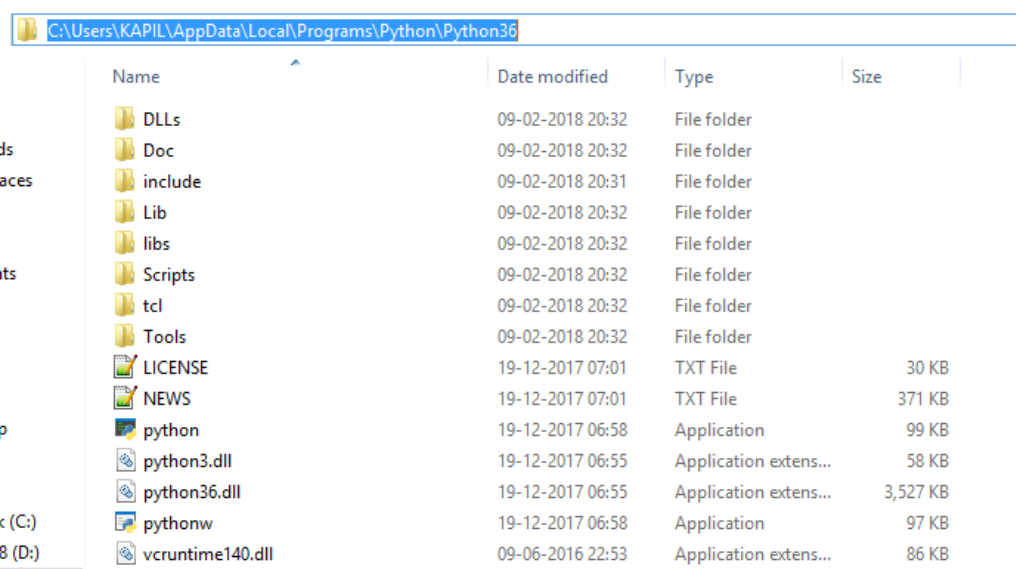
Download latest python set up from

<https://www.python.org/downloads/>

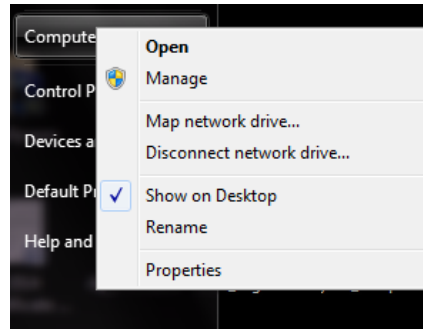
Simply install it by double clicking on the exe file downloaded and now you need to set the path of python as follow.

Setting path of python

Go to the python installation directory. For my PC it is C:\Users\KAPIL\AppData\Local\Programs\Python\Python36 folder. Take a look at screenshot here



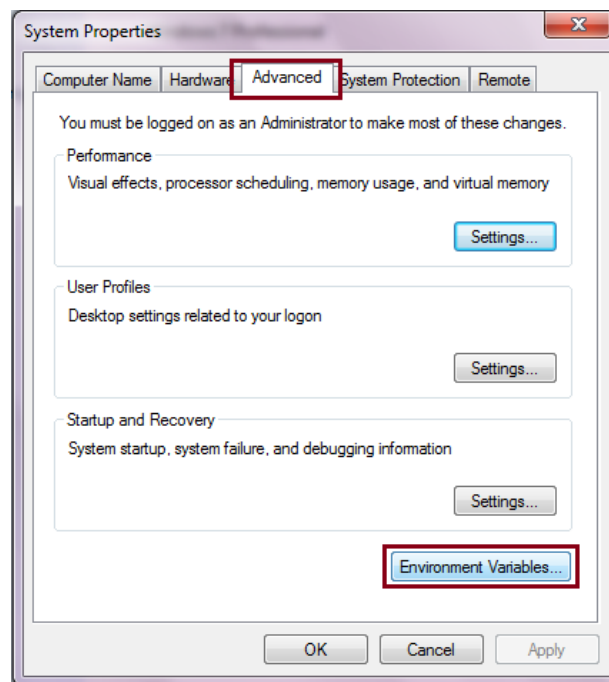
Now you need to add this installation path to the PATH environment variable. To do so right click on “My Computer” then click on properties.



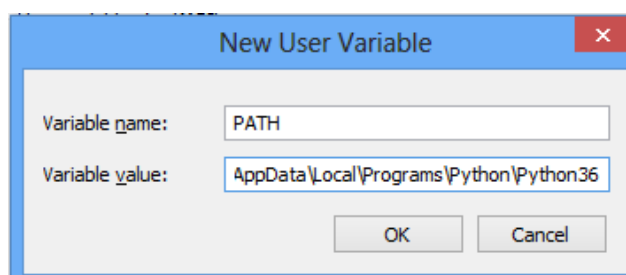
After that click on Advanced System settings



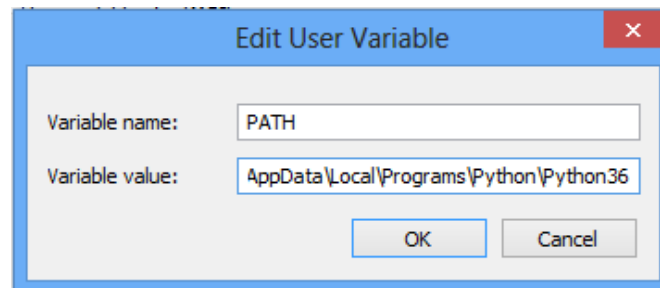
Now a new dialog box will be appeared, in that dialog box click on the environment variable button it will open another dialog box.



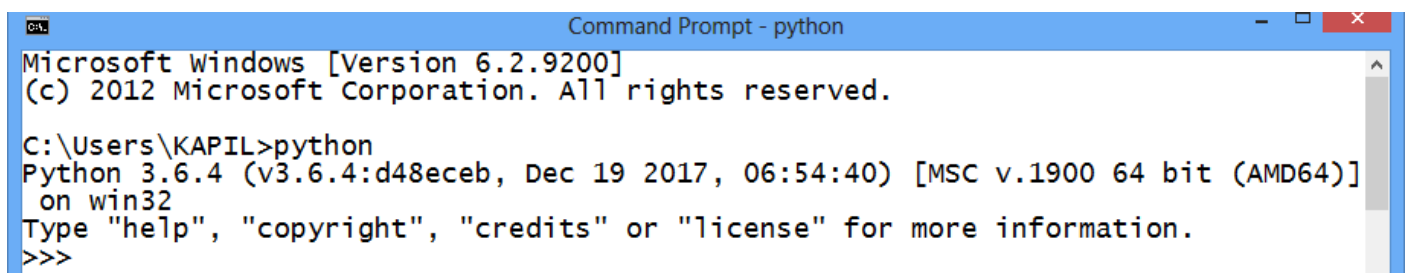
Now check if environment variable path exists or not. If it doesn't exist then click on new button. Type PATH for variable name and C:\Users\KAPIL\AppData\Local\Programs\Python\Python36 for value.



If PATH variable already exists then click on that variable and then click on edit button and append C:\Users\KAPIL\AppData\Local\Programs\Python\Python36 to existing value of PATH. Make sure to place semi-colon to separate two paths value.



Now to check if python is successfully installed or not open console and then type python followed with enter. If you are getting screen shot like that then path is successfully set. Now you are ready to start work with python.



To return on the console we have to write quit() or we need to press ctrl + Z.

Apart from that you can also use python idle. This software allows to run python code interactive mode as well as using scripting mode.

Make sure to add “C:\Users\KAPIL\AppData\Local\Programs\Python\Python36\Scripts” to PATH variable also as it is required for installation for some modules.

Installing jupyter notebook

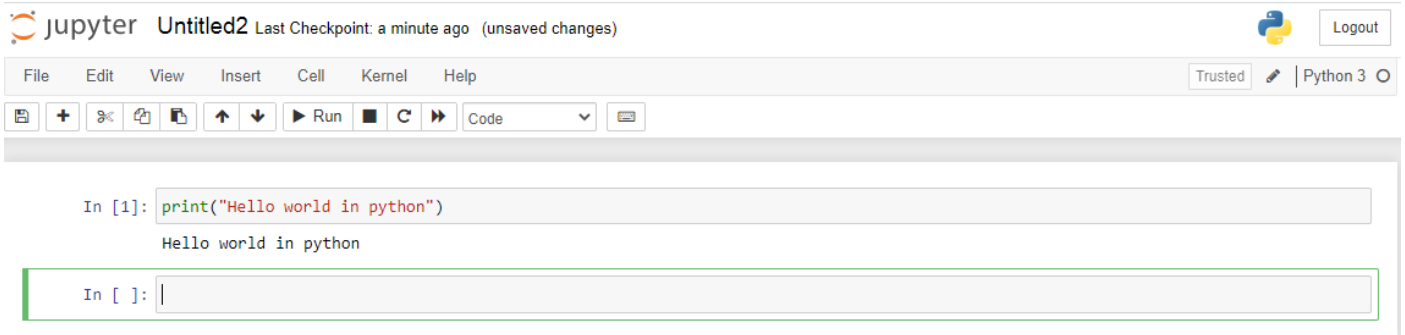
To install the jupyter notebook on your computer, you need to open command prompt in administrative mode (Search command prompt and then right click on that option for ‘Run as administrator’ will be there) and then type following command-

```
pip install jupyterlab
```

This command will take approximate (7-8 minutes, subjected to speed of internet), once the installation is completed then you need to Open the anaconda prompt and then type command `jupyter notebook` there, then it will be opened in web browser and the interface will be as follow-



Now click on the New button on the right top corner and then click on the python3. New tab will be opened with following interface-



You can type the code in the cell, to run the code, click on the run button and you will find the output. You can save the notebook for future reference also. To close the jupyter notebook, you need to click on the quit button as closing the tab is not sufficient.

The print() function

To print output on the console we have to use print() function that function automatically add a new line just after printing output and it uses whitespace as a separator while printing values. Multiple values can be passed to the print() function and these values are comma separated. The print() function returns None that actually stands for absence of value. The general format of print() function is as follow

```
print(parameter, sep = ' ', end = '\n')
```

An Example

```
print("Hello world")
print("Hi Heaven")
```

The above code will produce following output

```
Hello world
Hi Heaven
```

To suppress new line and adding space just after the printing statement we have to use 'end =' with print() statement.

```
print("Hello world",end = " ")
print("Hi Heaven")
```

On the other hand this code will produce following output

```
Hello world Hi Heaven
```

Comments

In python comments are started with # sign and they are of single line.

Literals in Python

Python supports integer literals (decimal, octal and hexadecimal), long integer, floating point literals (fractional and exponential), complex literals, String literals (single, double and triple quoted), Boolean literals (True and False) and None.

We can create collection of literals using list, tuple and dictionary. We will discuss about the same later.

Variables

A Python identifier is a name used to identify a variable, function, class, module or other object. Here we are mentioning some rules that one has to remember while choosing the name of identifier.

1. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
2. Python does not allow punctuation characters such as @, \$, and % within identifiers.
3. Python is a case sensitive programming language. `Manpower` and `manpower` are two different identifiers in Python.
4. An identifier cannot be a reserved keyword.

In C/C++ programming, Variables are different in python as compare to C/C++. In C/C++ programming languages, variable refers to some physical memory location such that that location stores some values that can be changed. Whenever value is changes than memory location remains same but value on that location changes.

In Python, variables are references to objects, but the actual data is contained in the objects. We are going to have a detailed discussion about that definition later in the same chapter.

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
count = 50 #it is an integer assignment
pi = 3.14 #it is a floating point assignment
name = "hello" #it is a string assignment
print(count)
print(pi)
print(name)
```

It will produce following output

```
50
3.14
hello
```

We can perform multiple assignments also like

```
a = b = 10
```

```
print(a)
print(b)
```

It will produce following output

```
10
10
```

```
count, pi, name = 50, 3.14, "hello";
print(count)
print(pi)
print(name)
```

It will produce following output

```
50
3.14
hello
```

Standard Data Types

Python has following standard data types

Numbers

Number data types store numeric values. Number objects are created when you assign a value to them. A Number data type can store value of three types that are integer values (e.g. 10, 100, -789), floating point real values (e.g. 3.14, 98.96). Apart from that it also supports complex number. The integer values are also of two types that are int and long int.

To represent a long number we have to use a lowercase L or uppercase L with long, but it is recommended that you use only an uppercase L to avoid confusion with the number 1. Python displays long integers with an uppercase L. Use L when you are providing input from console. Don't use it directly when providing value directly.

Number data types are *immutable data types*, means that changing the value of a number data type results in a newly allocated object.

Note: All inputs taken from console are considered as String to convert them to specific data type we have following functions

int(x)	To convert x to int value
long(x)	To convert x to long integer
float(x)	To convert x to float

String

Any contiguous set of characters represented in the single or double quotation marks is considered as String.

```
s1 = 'Python King' #it is a string
s2 = "Python king" #it is a string too
```

The first character is stored at location 0 and the final character is stored at location n-1 where n is the number of characters in a String.

Python's triple quotes comes to the rescue by allowing strings to span multiple lines, including verbatim NEWLINES, TABs, and any other special characters. The triple quotes can be three consecutive single quotes or double quotes like

```
s1 = """This is a String
and this string will preserve
new line"""
print(s1);
```

The above example will produce following output

```
This is a String
and this string will preserve
new line
```

Note: The triple quoted string is often used for multi line commenting

A single/double quoted string can be split into multiple lines as follow

```
str1 = 'Hello\
Jaipur'

str2 = "Hello\
Jaipur"
```

We can use the same way for triple quoted string.

```
str3 = """Hello\
Jaipur"""
```

Strings str1, str2 and str3 have length 11 characters i.e. splitting a string to multiple line(s) using \ does not increase length of string but we have to keep one thing remember in case of multi line string, splitting string into multiple line(s) without \ increases length.

```
str4 = """Hello
Jaipur"""
```

The above String str4 has length 12.

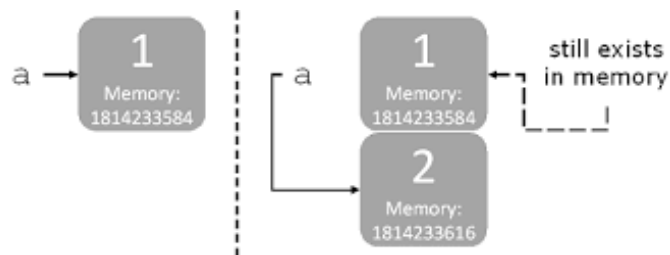
Immutable and mutable data types

Number & String, both are considered as *immutable data types*. Apart from that we have two more immutable data types that are tuple and frozenset (discussed later). Before discussing immutability let us recall definition of variables “*Variables are references to objects, but the actual data is contained in the objects*” i.e. a variable contains address of an object but the value is stored at memory location referred by variable. Immutability says that value of the object can't be changed (mind these words) but variable can change the address i.e. it can point to another object, look at the following code

```
a = 1
```

```
a = 2
```

Look, what exactly happened in memory



The variable `a` points to the object containing value 1, when we assign value 2 to variable `a` then it stops pointing to object containing value 1. It starts pointing to some other memory location that actually has value 2. The whole scenario happened due to immutability. It says that whenever value of a variable is changed then it actually starts pointing to new memory location, it does not change value at previous memory location.

Apart from the immutable data types we have mutable data types (examples: list, dictionary and set, will discuss in subsequent chapters), In case of mutable data type whenever any changes in value of variable takes place then variable continues to point old memory location and value at the same memory location gets changed.

To verify the concept of mutability, we have `id()` function which actually returns address of the object being pointed by variable. Take a look at example

```
a = 10
print(a, id(a))
```

```
a = 15
print(a, id(a))
```

```
a = 10
print(a, id(a))
```

Output of the code given above is

```
10 1987800832
15 1987800992
10 1987800832
```

Garbage Collection and del keyword

The object's reference count decreases when it's deleted with `del`, its reference is reassigned, or its reference goes out of scope. Python deletes unneeded objects automatically to free the memory space when an object's reference count reaches zero, Python collects it automatically. The process of reclaiming blocks of memory that are no longer in use is termed Garbage Collection. The garbage collector runs during program execution and is triggered when an object's reference count reaches zero.

An Example

```
a = 10
print(a)
del a
print(a)
```

The output of above code is

```
10
Traceback (most recent call last):
  File "Seventh.py", line 4, in <module>
    print(a)
NameError: name 'a' is not defined
```

An Example

```
a = 10
b = 10

print("a = ",a,"id(a) = ",id(a))
print("b = ",b,"id(b) = ",id(b))
#Observe output: a and b have same id i.e. the same memory has two referenced
del a
#Now memory has only one reference

print("\nb = ",b,"id(b) = ",id(b))
#print("a = ",a,"id(a) = ",id(a))      Error

del b
#Now memory is eligible for garbage collection because reference count is zero.
```

The output of above code is

```
a = 10 id(a) = 1987800832
b = 10 id(b) = 1987800832

b = 10 id(b) = 1987800832
```

Type Conversion

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion. Python has two types of type conversion, that are Implicit Type Conversion & Explicit Type Conversion

In **Implicit type conversion**, Python automatically converts one data type to another data type. This process doesn't need any user involvement. This type of conversion takes place in expressions when multiple types are involved like int and float.

```
a = 10
b = 10.5
c = a + b
print(c)
```

In the above example, in the expression `a + b`, value of variable `a` is promoted to float during calculation, so data type of variable `c` is float

For **explicit type conversion**, we have to use `int()`, `float()` and `chr()` function. We have already discussed the same.

Operators

An **operator** is a symbol that represents an operation or action. The values on which action is applied are known as **operands**. A combination of operator and operand is known as **expression**.

The operators that work on a single operand are called **unary operators** (e.g. `+`, `-`, `~`) on the other hand the operators that work on two operands are called **binary operators** (e.g. `*`, `/`, `<`, `>`, `+`, `-`, etc). The operator that works on three operands is termed as **ternary operator**, python has only one ternary operator.

Python language has a variety of operators that are divided into following categories like Arithmetic operators, Relational operators, Logical operators, Bitwise operators, Membership operator and Identity operators etc.

Arithmetic Operators

These operators are used to perform arithmetic operations. Some of the operators also work on string, list and other data types. Take a look at the list of operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Floor Division
%	Modulus
**	Exponential

*Note: ** enjoys highest priority and having associativity right to left. /, //, *, % have same priority (lower than **) but their associativity is left to right. +, - has same priority (lower than /) their associativity is also left to right.*

An Example:

```
x = 5
```

```
y = 3
```

```
print(x + y)      #will result 8
print(x - y)      #will result 2
print(x * y)      #will result 15
print(x / y)      #will result 1.6666666666666667
print(x // y)     #will result 1
print(x % y)      #will result 2
print(x ** y)     #will result 125
```

The + and * operators also work for string & list (discussed later) data type. The + operator works for concatenation in String data type and * operator works for replication. For + operator both values must be of string type and for * operator one operand must be of int data type

An Example

```
s1 = 'ABC'
print(s1 + 'DEF')    #will result ABCDEF
#print(s1 + 45)      TypeError: must be str, not int
print(s1 + str(45))  #will result ABC45
print(s1 * 3)        #will result in ABCABCABC
#print(s1 * 'D')     can't multiply sequence by non-int of type 'str'
```

Note: the str() function is used to convert int/float value to string.