1.1 Python Basics

1.1.1 Introduction: Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Python is named after a TV Show called 'Monty Python's Flying Circus' and not after Python-the snake. Some of the Features that Makes Python more popular are:

- Python is Simple and Easy to learn and code.
- Python is Free and Open Source. It is freely available at the https://www.python.org/. Python source code is also available to the public, one can download it, use it, and share it.
- Python is High Level Language and supports both Procedure oriented and Object-Oriented Language concepts along with dynamic memory management.
- Python is portable. Python code can be run on any platforms like Linux, Unix, Mac and Windows.
- Python is extensible and integrated. Python code can be extended and integrated with among other languages like C, C++, Java, etc.
- Python is an interpreted language. Python code is executed line by line at a time and there is no need to compile, which makes debugging easier. The
- Python has rich set of libraries for data analytics, machine learning, artificial intelligence, deep learning, mathematical computation, web app development, mobile app development, testing, etc.
- Python is a dynamically typed language. Here the data type for variable is decided at run time. As a result, there is no need to specify the type of variable.

1.1.2 Why One Should Learn Python Program?

Python Programming is a fun, creative and rewarding activity. Python is one of the most widely used programming language across the world for developing software applications. It is named as one of top picked programming languages of most of the universities and industries. Python developer is one of the "10 Most

in Demand Tech Jobs of 2019"[Source : https://www.techrepublic.com/] As of February 23, 2019, the average salary for a Python developer is $123,201 per year in the United States, making it one of the most popular and lucrative careers today [source: https://www.codingdojo.com/].
Python can be used on the following:

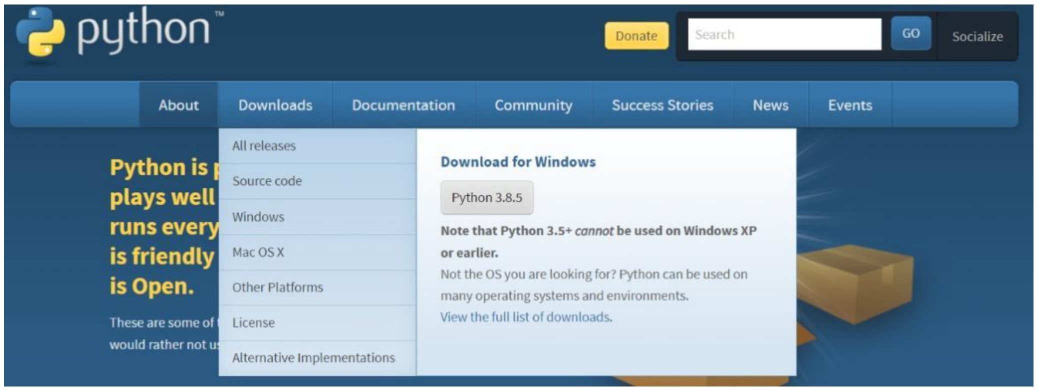
1. Multiple Programming Paradigms
2. Web Testing
3. Data Extraction
4. Artificial Intelligence
5. Machine Learning
6. Data Science
7. Web Application and Internet Development
8. Cybersecurity

Zen of Python [**source:** https://www.python.org/]: Long time Pythoneer Tim Peters succinctly channels the BDFL's guiding principles for Python's design into 20 aphorisms, only 19 of which have been written down:

1. Beautiful is better than ugly.
2. Explicit is better than implicit.
3. Simple is better than complex.
4. Complex is better than complicated.
5. Flat is better than nested.
6. Sparse is better than dense.
7. Readability counts.
8. Special cases are not special enough to break the rules.
9. Although practicality beats purity.
10. Errors should never pass silently.
11. Unless explicitly silenced.
12. In the face of ambiguity, refuse the temptation to guess.
13. There should be one-- and preferably only one --obvious way to do it.
14. Although that way may not be obvious at first unless you're Dutch.
15. Now is better than never.
16. Although never is often better than *right* now.
17. If the implementation is hard to explain, it's a bad idea.
18. If the implementation is easy to explain, it may be a good idea.
19. Namespaces are one honking great idea -- let's do more of those!

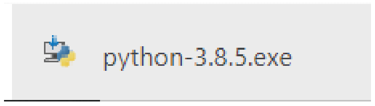
Installing Python: Before starting the programming in Python, one should install Python interpreter on the computer. To install python one must download the installation package of the required version from the link/ULR : https://www.python.org/. For windows there is a Windows x86-64 executable installer in both 32 bit and 64 bit versions. Latest python release as on 5th September 2020 is Python 3.8.5 .

1. Launch a web browser then navigate to downloads for Windows. Download the Python 3.8.5.

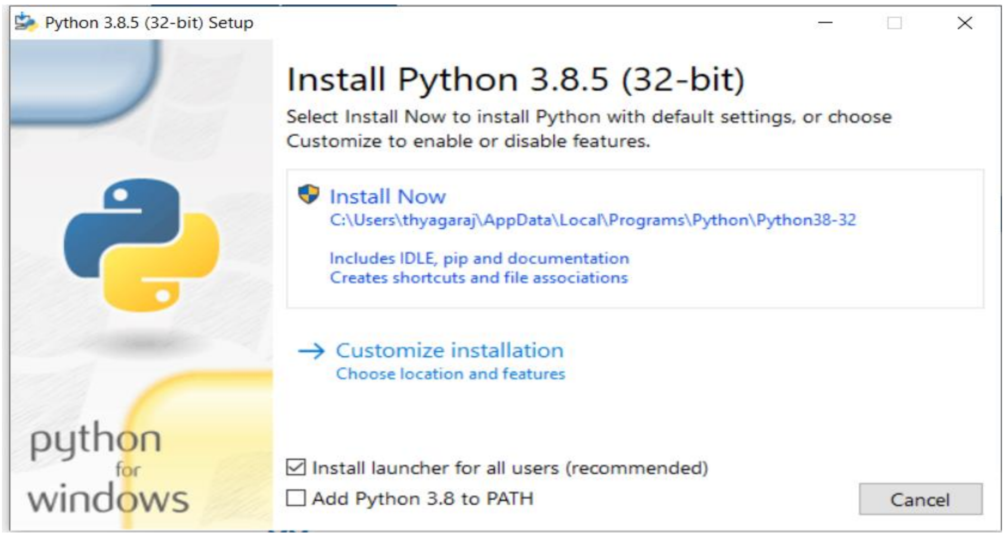


Note that Python 3.5+ *cannot* be used on Windows XP or earlier.
Not the OS you are looking for? Python can be used on many operating systems and environments.
View the full list of downloads.

2. When the download completes run the installer



3. Choose to install Now



Install Python 3.8.5 (32-bit)
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

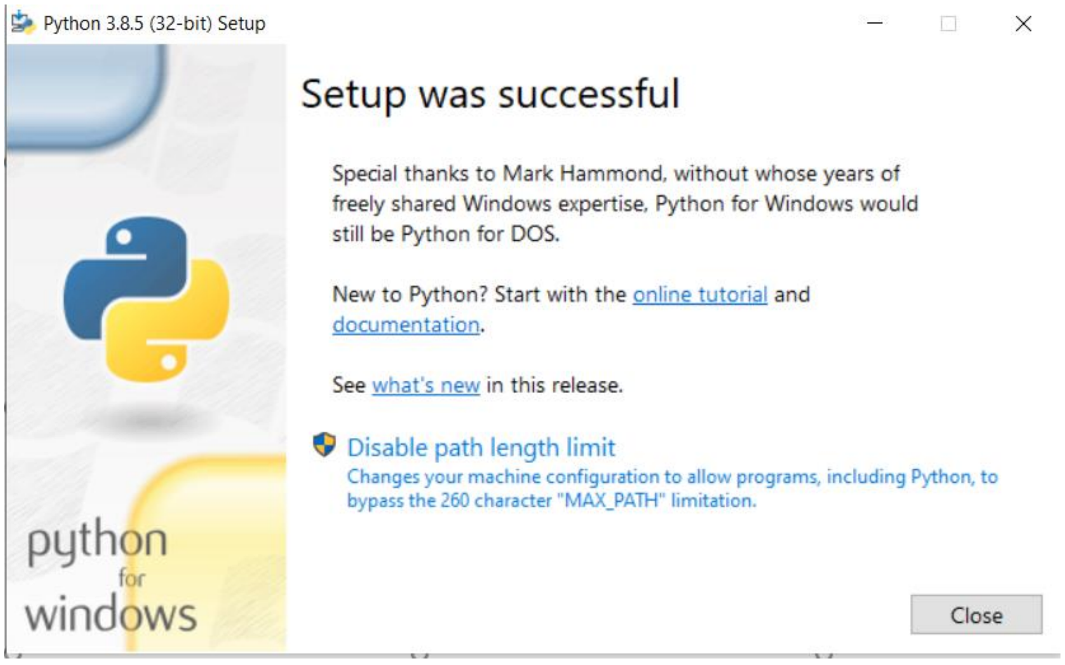
Install Now
C:\Users\thyagaraj\AppData\Local\Programs\Python\Python38-32
Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.8 to PATH

Cancel

After the setup is completely installed you will get Setup was successful message.



Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

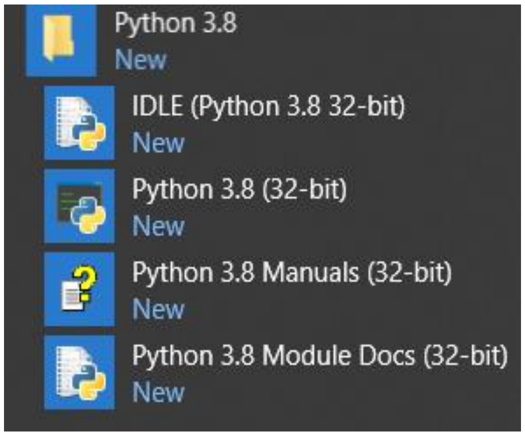
See what's new in this release.

Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

Interacting with Python IDLE (Integrated Development Learning Environment)

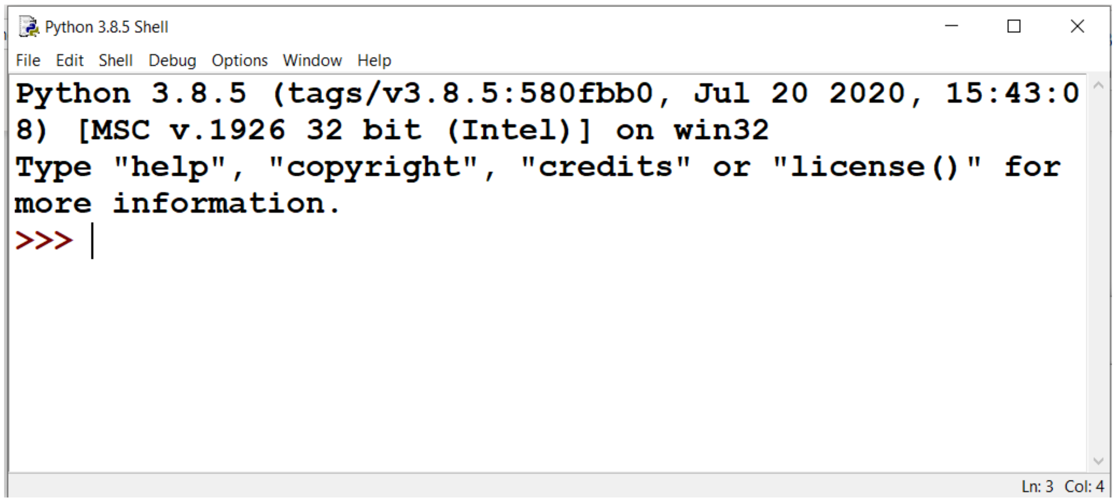
To write and run Python code interactively one can either use the command line window (Shell) or the IDLE. IDLE is Integrated Development Learning Environment that comes with Python, that can be used to edit, run, browse, and debug a Python program from a single interface.

Python Shell:
Python IDLE can started by clicking its icon created on the desktop or menu item on the **START menu -> Python3.8 -> IDLE (Python3.8 32- bit)** option as illustrated below:



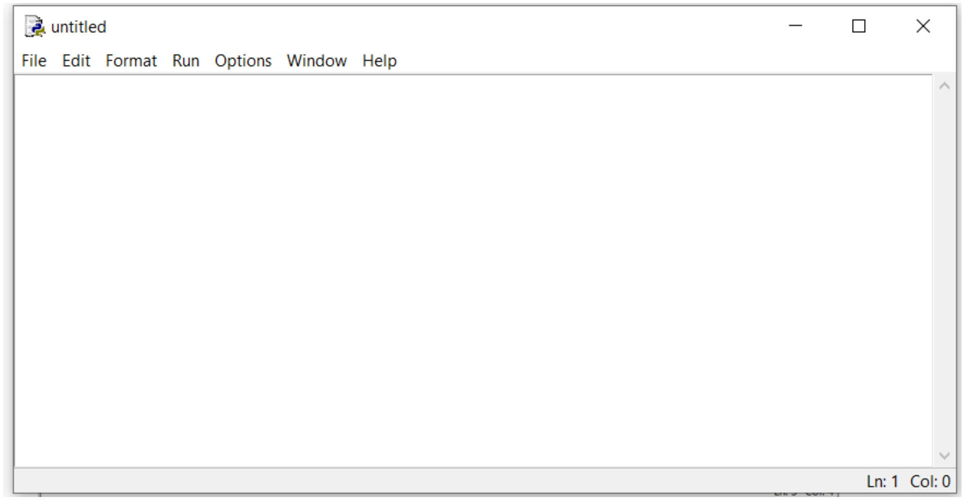
Python IDLE comprises Python shell **(Interactive mode)** and Python Editor (**Script mode**).

Python Shell:

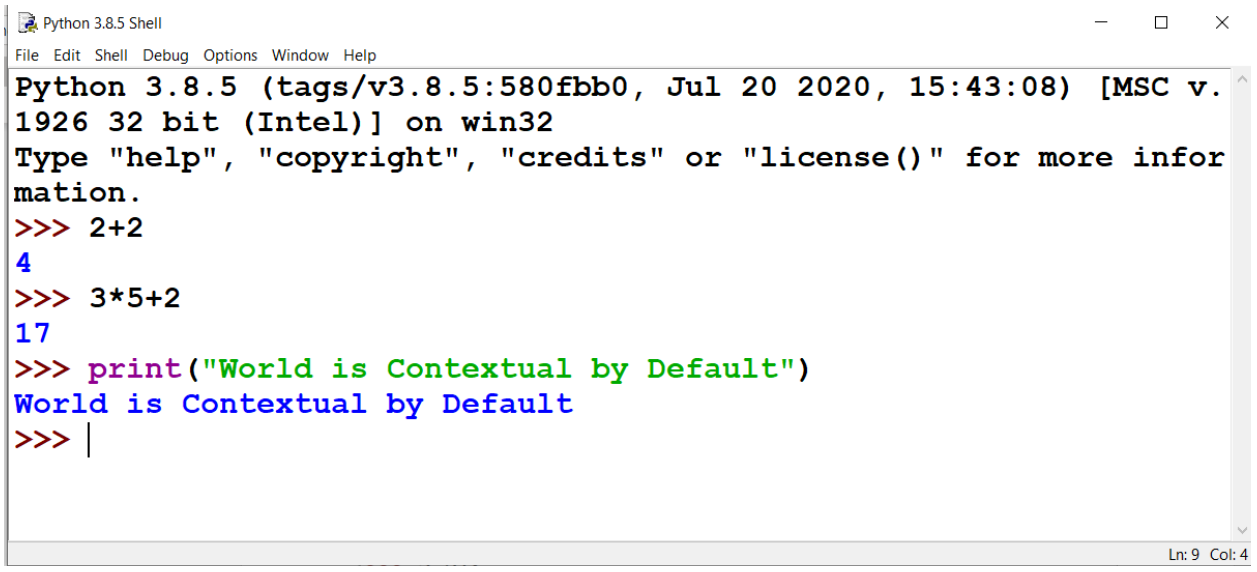


```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:0
8) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for
more information.
>>> |
```

Python Editor:



Python IDLE always starts by default with shell. Python shell shows a header message displaying its version and copyright notice. Soon after the header message, the command prompt (>>>) followed by blinking cursor gets displayed, indicating its readiness for accepting user Python commands/instructions /code. The three greater than symbols are called the prompt or Python Command prompt. Python Shell is an interactive window where we can type in the Python code and see the output in the same window as illustrated below:

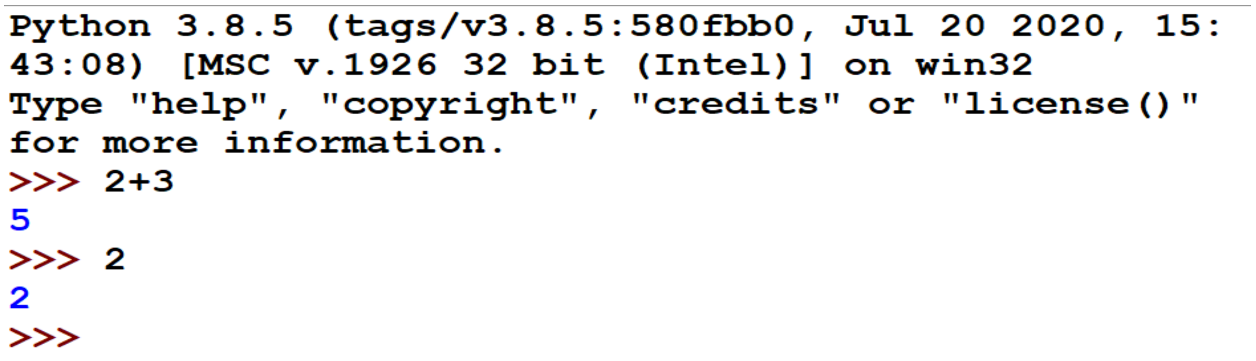


```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.
1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more infor
mation.
>>> 2+2
4
>>> 3*5+2
17
>>> print("World is Contextual by Default")
World is Contextual by Default
>>> 
```

Entering Expressions into the interactive Shell

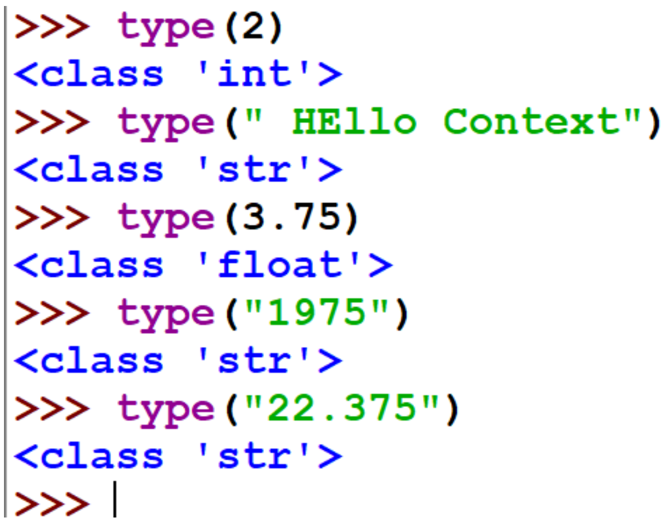
In Python, expression is the most basic kind of programming instruction in the language**.** An expression is a combination of ***values, variables, and operators***.

Examples: 17, x, x+17 ,1+2*2 , X**2, x**2 +y**2



```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:
43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>> 2+3
5
>>> 2
2
>>>
```

Value: A value is a letter or a number. Example: 1,2 and “Hello, World!”. Types are the data types to which the Values belong. **type(arg)** function returns the data type of the argument as illustrated below :



```
>>> type(2)
<class 'int'>
>>> type(" HEllo Context")
<class 'str'>
>>> type(3.75)
<class 'float'>
>>> type("1975")
<class 'str'>
>>> type("22.375")
<class 'str'>
>>>
```

Data types: Data types are the type of the values. Python Supports the following Data types:

1. **Numbers:** Number data types store numeric values. Number objects are created when you assign a value to them. Python recognizes several different types of data numbers like integers, floating point, and complex numbers.
   - 23 and −75 are *integers*,
   - 5.0 and −23.09 are floats *or floating point numbers.*
   - 2 + 3j is a *complex number*
2. **Strings:** Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows either ***pair of single or double quotes***.
   Example: "Hello ", "Contextual Artificial Intelligence" .
3. **Lists:** Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed ***within square brackets ([ ]).***
   Example: Sample_list = [1,"AI","ML",234]
4. **Tuples:** A tuple is another sequence data type that is similar to the list. A tuple consists of values separated by commas. Unlike lists, however, ***tuples are enclosed within parenthesis.***
   Example: sample_tuple = (1, "23", 777, "Eyes")
5. **Dictionary:** Python's dictionaries are kind of hash-table type. They work like associative arrays or hashes found in Perl and consist of **key-value** pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object. ***Dictionaries are enclosed within curly braces***.
   Example: d = {'Name': "Aana", 'Age': 23 , 'Roll_no' : '75TG123'}

Variables: A variable is a name that refers to a value. Example: x, si, area_of _Circle, etc. An assignment statement creates new variables and gives them values.

Examples:

Message = '*Python Programming* ',

p =1000, t= 2, r=3.142,

Si = p*t*r/100,

pi = 3.1415926535897931,
area_of _circle = *pi*r*r.*
To know the type of the variable one can use type () function. **Ex:** type(p)
To display the value of a variable, you can use a print statement:
Ex: print (Si) ; print(pi)

Rules for writing Variable names

1. **Variable names** can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (_).
2. **Variable names** cannot start with a number/digit.
3. **Keywords** cannot be used as Variable names .
4. We cannot use special symbols like **!, @, #, $, %** etc. in Variable names .
5. Variable names can be of any length.
6. Variable name must be of single word.

Table: Valid Variable Names and Invalid Variable Names

Valid Variable Names	Invalid Variable Names
python12	current- account(hyphens are not allowed)
Simple	savings account (spaces are not allowed)
interest_year	4freinds (can't begin with a number)
_rate_of_interest	1975 (can't begin with a number)
_spam	10April_$ (cannot begin with a number and special characters like $ are not allowed)
HAM	Principle#@(special characters like # and @ are not allowed)
account1234	'bear' (special characters like ' is not allowed)

Note:

Variable names are case-sensitive, meaning that velocity, VELOCITY, Velocity, and velocity are four different variables. It is a Python convention to start your variables with a lowercase letter.

Storing Values in a Variables : Values can be stored in a variable using Assignment statement. An assignment statement consists of a variable name , an equal sign and the value to be stored .

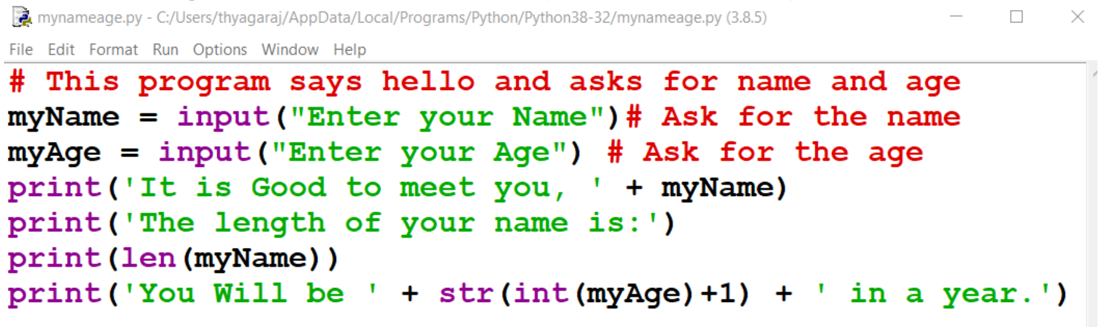
Example :

```
>>> ham = 40
>>> ham
40
>>> spam= 20
>>> ham+ spam
60
>>> spam = spam+25
>>> spam
45
```

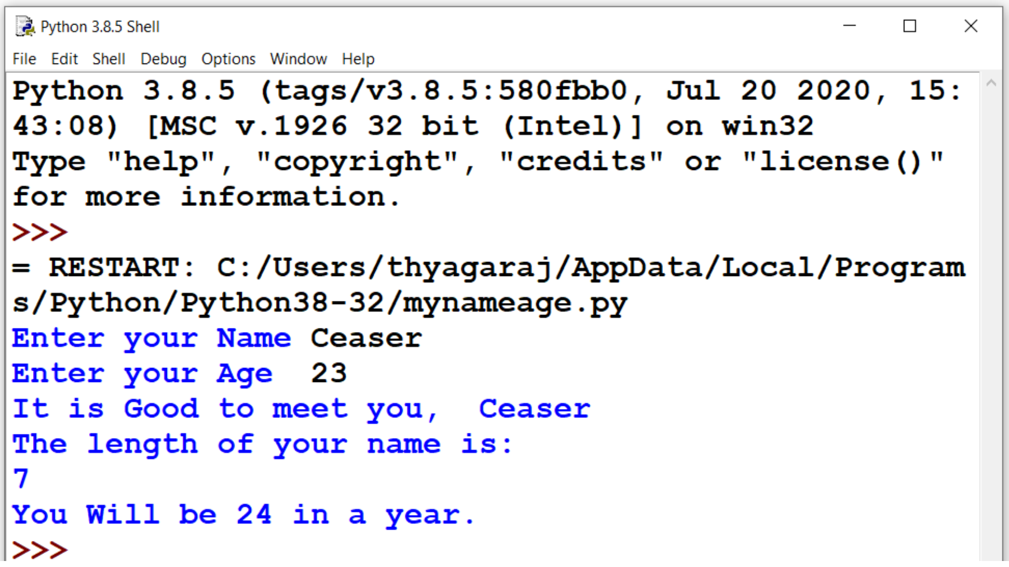
A variable **ham** is initialized (or created) the first time a value is stored in it. After that one can use it in expression with other variables and values like ***ham + spam***. When a variable is assigned a new value, old value gets erased or forgotten. This is called overwriting. Following code illustrates the overwriting of string

```
>>> a = "Delhi"
>>> a
'Delhi'
>>> a = "London"
>>> a
'London'
>>>
```

Sample Program: To read and print your name and age


mynameage.py - C:/Users/thyagaraj/AppData/Local/Programs/Python/Python38-32/mynameage.py (3.8.5)
File Edit Format Run Options Window Help

```
# This program says hello and asks for name and age
myName = input("Enter your Name")# Ask for the name
myAge = input("Enter your Age") # Ask for the age
print('It is Good to meet you, ' + myName)
print('The length of your name is:')
print(len(myName))
print('You Will be ' + str(int(myAge)+1) + ' in a year.')
```



```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:
43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
= RESTART: C:/Users/thyagaraj/AppData/Local/Program
s/Python/Python38-32/mynameage.py
Enter your Name Ceaser
Enter your Age  23
It is Good to meet you,  Ceaser
The length of your name is:
7
You Will be 24 in a year.
>>>
```

Dissecting the Sample Program

Sample program comprises executable statements containing comments and built in functions like print () , input () , len() , int() and str()

Comments: Comments are readable explanation or descriptions that help programmers better understand the intent and functionality of the source code. Comments are completely ignored by interpreter.

Advantages of Using Comments:

1. Makes code more readable and understandable.
2. Helps to remember why certain blocks of code were written.
3. Can also be used to ignore some code while testing other blocks of code.

Single Line Comments in Python:

The hash symbol #is used to write a single line comment.

Example:

```
# Printing a message
print(“ Enter your Name “)
myName = input (“ Enter Your Name”) # Read your name to myName
```

Multiline Comments in Python:

1. Using **#** at the beginning of each line of comment on multiple lines

Example:

```
# It is a
# multiline
# comment
```

2. Using String Literals **'''** at the beginning and end of multiple lines

Example:

```
'''
I am a
Multiline comment!
'''
```

The print() Function :

The print function is used to display the string value written within pair of double quotes inside the parentheses on the screen .

```
print('It is Good to meet you, ' + myName)
print('The length of your name is:')
print(len(myName))
print('You Will be ' + str(int(myAge)+1) + ' in a year.')
```

The line *print('The length of your name is:')* means “Print out the text in the string ‘'The length of your name is:’ . When Python executes the print statement, python interpreter calls the *print()*function and the string value is being *passed* to the function. The value within print() is called argument . Quotes within parentheses marks where the string begins and ends ; they are not part of the string value.

The input() function

This function is used to take the input from the user . Whatever the user enter as input, input() function convert it into a string . if you enter an integer value still input() function convert it into a string . The programmer is needed to convert it into an integer in your code using *typecasting*.

Myname = input(“Enter your name”)

Example:

```
>>> myName = input("Enter your Name")
Enter your Name Ceaser
>>> print(myName)
 Ceaser
>>> x = input("Enter a number")
Enter a number 230
>>> print(x)
 230
>>> type(x)
<class 'str'>
>>> x = int(input("Enter a number"))
Enter a number 240
>>> type(x)
<class 'int'>
```

Reading multiple values using input()

Programmer often want as user to enter multiple values in one line . In Python user can take multiple values or inputs in one line by using split() method . It breaks the given input by the specified separator. If separator is not provided then any white space is a separator. Generally, user use a split() method to split a Python string but one can used it in taking multiple input.

Example:

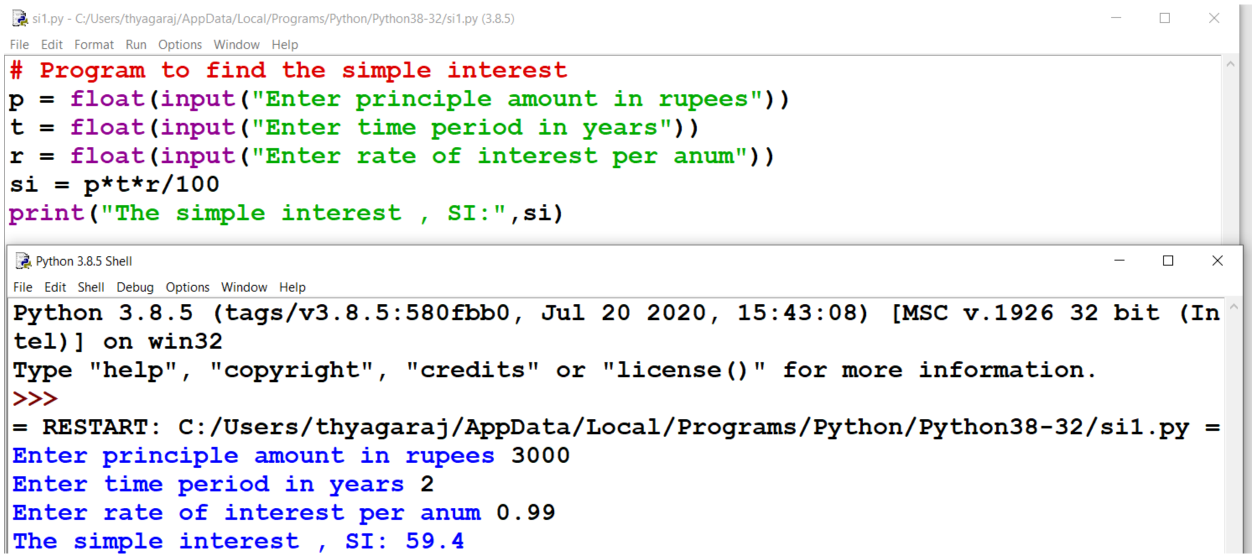
>>> **x, y,z = input ("Enter three values").split()**

Enter three values 2 3 4

```
>>> x
'2'
>>> y
'3'
>>> z
'4'
```

In order to get the input from the user through the keyboard Python provides a built-in function called input. When this function is called, the program stops and waits for the user to type something. When the user presses Return or Enter, the program resumes and input returns what the user typed as a string.

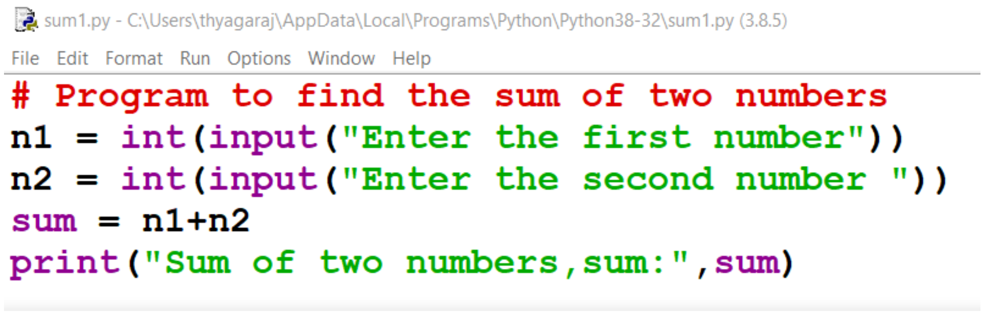
Example 1: Program to find the simple interest


si1.py - C:/Users/thyagaraj/AppData/Local/Programs/Python/Python38-32/si1.py (3.8.5)
File Edit Format Run Options Window Help
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help

```
# Program to find the simple interest
p = float(input("Enter principle amount in rupees"))
t = float(input("Enter time period in years"))
r = float(input("Enter rate of interest per anum"))
si = p*t*r/100
print("The simple interest , SI:",si)
```

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/thyagaraj/AppData/Local/Programs/Python/Python38-32/si1.py =
Enter principle amount in rupees 3000
Enter time period in years 2
Enter rate of interest per anum 0.99
The simple interest , SI: 59.4
```

Example 2 :Python Program to Add Two Numbers


sum1.py - C:\Users\thyagaraj\AppData\Local\Programs\Python\Python38-32\sum1.py (3.8.5)
File Edit Format Run Options Window Help

```
# Program to find the sum of two numbers
n1 = int(input("Enter the first number"))
n2 = int(input("Enter the second number "))
sum = n1+n2
print("Sum of two numbers,sum:",sum)
```

```
Enter the first number 3
Enter the second number  4
Sum of two numbers,sum:  7
```

The len() Function

In python len() function is used to find the number of characters in a given string as illustrated below :

```
>>> myName = input("Enter your Name")
Enter your Name Dr.KobaCeaser
>>> len(myName)
14
```

The str() , int() and float Functions :

str() function can be used convert integer or floating numbers into string data type.

```
>>> str(29)
'29'
>>> str(-3.14)
'-3.14'
```

Similarly **int()** and **float()** function will evaluate into integer and floating – point forms of the value you pass , respectively.

```
>>> int('-990')
-990
>>> int(1.25)
1
>>> int(2.99)
2
>>> int("325")
325
>>> float("32.45")
32.45
>>> float(10)
10.0
>>> float(32.75)
32.75
>>> float(10)
10.0
```

Note that if a values is passed to int() or float that they cannot evaluate as integer or float , Python will display an error message.

```
>>> int("Seven")
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    int("Seven")
ValueError: invalid literal for int() with base 10: 'Seven'
>>> float("thirty.five")
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    float("thirty.five")
ValueError: could not convert string to float: 'thirty.five'
```

The **int()** function can be used to round a floating point number down as illustrated below :

```
>>> int(23.45)
23
>>> int(8.8) + 22
30
```

Operators and operands

- Operators are special symbols that represent computations like addition and multiplication. The values the operator is applied to are called operands.
- The operators **+, -, *, /, and **** perform *addition, subtraction, multiplication, division, and exponentiation*, as in the following examples:

Table: Operators and Examples

Operator	Operation	Example	Evaluates to
**	Exponent	5**3	125
%	Modulus/Remainder	33%7	5
//	Integer Division/Floored quotient	33//5	6
/	Division	23/7	3.2857142857142856
*	Multiplication	7*8	56
-	Subtraction	8 – 5	3
+	Addition	7+ 3	10

Order of operations

- When more than one operator appears in an expression, the order of evaluation depends on the rules of precedence.

PEMDAS order of operation is followed in Python:

- **Parentheses** have the highest precedence and can be used to force an expression to evaluate in the order you want.
- **Exponentiation** has the next highest precedence,
- **Multiplication and Division** have the same precedence, which is higher than
- **Addition and Subtraction**, which also have the same precedence.
- **Operators with the same precedence** are evaluated from left to right.

Following examples illustrates the evaluation of expressions by Python interpreter. In each case the programmer must enter the expression , python interpreter evaluates the expression to a single value .

```
>>> 5+4*3
17

>>> (4+5)*3
27

>>> 12345678*45678
563925879684

>>> 3**5
243

>>> 22//7
3

>>> 22/7
3.142857142857143

>>> 27%5
2
```

```
>>>3 +  3
6
>>> (5-2)*((8+4)/(5-2))
12.0
```

Python interpreter evaluates parts of the expression as per the PEMDAS rule until it becomes a single value as illustrated below :

(5-2)*((8+4)/(5-2))

3 * ((8+4)/(5-2))

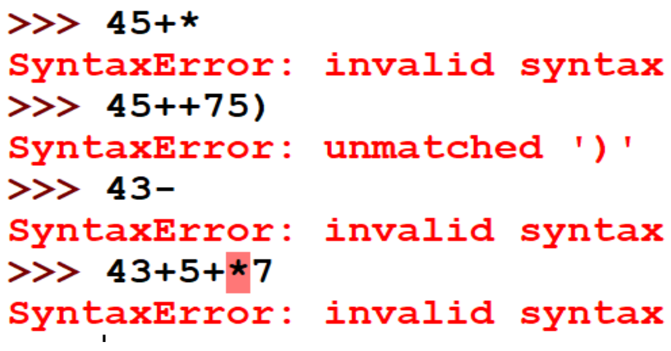
3*(12/(5-2))

3*(12/3)

3*4.0

12.0

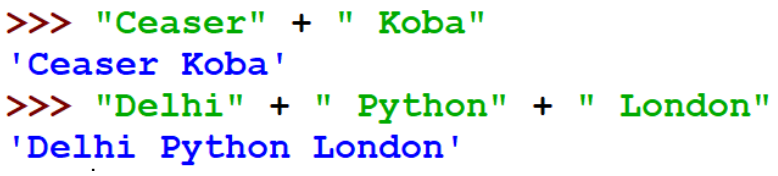
Note: If you type invalid expressions, python interpreter will not be able to understand it and will display a SyntaxError message as illustrated below:



```
>>> 45+*
SyntaxError: invalid syntax
>>> 45++75)
SyntaxError: unmatched ')'
>>> 43-
SyntaxError: invalid syntax
>>> 43+5+*7
SyntaxError: invalid syntax
```

String Concatenation and Replication

The functionality of the operator may change based on the data types of operand used . For example + which is the addition operator when it operated on two integers or floating – point values , can be used to concatenate two strings as illustrated below :



```
>>> "Ceaser" + " Koba"
'Ceaser Koba'
>>> "Delhi" + " Python" + " London"
'Delhi Python London'
```

Similarly * operator which is multiplication operator can be used as string replication operator when string is multiplied with a number as illustrated below :

```
>>> "Koba Bad"*3
'Koba BadKoba BadKoba Bad'
>>> "Ceaser Good"*5
'Ceaser GoodCeaser GoodCeaser GoodCeaser GoodCeaser Good'
```

Python Character Set :

The set of valid characters recognized by Python like letter, digit or any other symbol . The latest version of Python recognizes Unicode character set. Python supports the following character set:

- **Letters** : A-Z ,a-z
- **Digits** :0-9
- **Special Symbols** : space +-/***()[]{}//=!= == <> ,'"'",;:%!#?$&^⇔=@_
- **White Spaces** : Blank Space, tabs(->), Carriage return , new line , form feed
- **Other Characters** : All other 256 ACII and Unicode characters

Python Tokens:

A token (lexical unit) is the smallest element of Python script that is meaningful to the interpreter . Python has following categories of tokens: Identifiers, Keywords , Literals , operators and delimiters.

Identifiers: Identifiers are names that you give to a variable , class or Function. There are certain rules for naming identifiers similar to the variable declaration rules , such as : No Special character except_ , Keywords are not used as identifiers , the first character of an identifier should be _ underscore or a character , but a number is not valid for identifiers and identifiers are case sensitive .

Literals: A fixed numeric or non-numeric value is called a literal . Literals may be string, numbers (int, long, float and complex), Boolean (True or False), NONE and Operators.

Operators : A Symbol or a word that performs some kind of operation on given values and returns the result. There are 7 types of operators available for Python: Arithmetic Operator ,Assignment Operator, Comparison Operator, Logical Operator , Bitwise Operator , Identity Operator and Membership Operator .

Delimiters: Delimiters are the symbols which can be used as separators of values or to enclose some values. Examples of delimiters are () {} [],;:

Note : Comments and # symbol used to insert a comment is not a token.

Keywords: The reserved words of Python which have a special fixed meaning for the interpreter are called keywords. No keyword can be used as an identifier or variable names. There are 35 keywords in python as listed below:

Keyword	Description
and	Logical and operator
as	Alias
assert	Used for debugging
async	Used to make a function asynchronous by adding the async keyword before the function’s regular definition
await	Used in asynchronous functions to specify a point in the function where control is given back to the event loop for other functions to run. You can use it by placing the await keyword in front of a call to any async function
break	To break out of a loop
class	To define a class
continue	For skipping the statements and conitinuing the next iteration
def	For defining user defined functions
del	To delete an object
elif	Conditional statement, same as else if
else	Conditional statement
except	Used in exception handling
False	Boolean Value
finally	Used in exception handling , to execute a block of code no matter whether exception is there or not
for	Used to create for loop – iterative statement
from	Used to import specific parts of a module
global	Used to declare global variable
if	Conditional /decision making statement
import	Used to import a module or library
in	Used to check if a value if present in list, tuple, dictionaries , sets ,etc.
is	To check if two variables are equal
lambda	Used for defining an anonymous function

None	Used to represent a null value
nonlocal	To declare a non-local variable
not	A logical operator
or	A logical operator
pass	A null statement , a statement that will do nothing
raise	To raise an exception
return	To exit a function and return a value
True	Boolean Value
try	Used in exception handling
while	For creating a while iterative loop
with	Used to simplify exception handling
yield	To end a function , returns a generator

Following code segment can be used to obtain the list of python keywords :

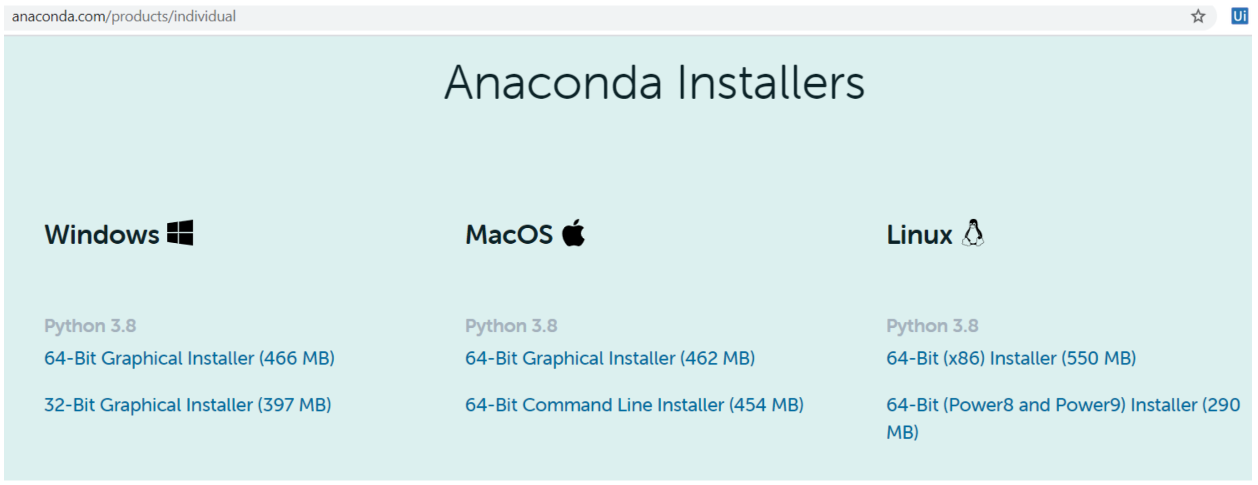
```
>>> import keyword
>>> len(keyword.kwlist)
35
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'async',
'await', 'break', 'class', 'continue', 'def', 'del', 'elif
', 'else', 'except', 'finally', 'for', 'from', 'global', '
if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'o
r', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yi
eld']
```

Installing Jupyter Notebook

[source : https://test-jupyter.readthedocs.io/en/latest/install.html]

Prerequisite: Jupyter requires Python (Python 3.3 or greater or Python2.7) for installing the Jupyter Notebook. Anaconda distribution is recommended for installing Python and Jupyter .

Installing Jupyter using Anaconda and conda:
To install anaconda, one must download the installation package of the required version from the link/ULR: https://www.anaconda.com/products/individual.



anaconda.com/products/individual

Anaconda Installers

Windows

Python 3.8
64-Bit Graphical Installer (466 MB)
32-Bit Graphical Installer (397 MB)

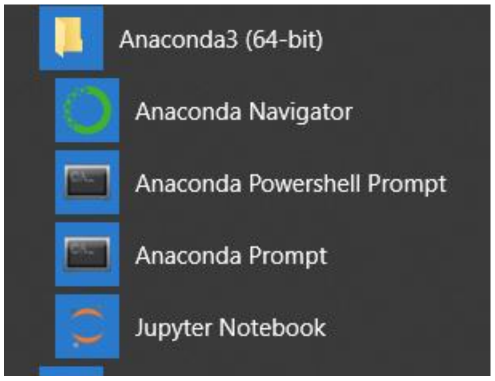
MacOS

Python 3.8
64-Bit Graphical Installer (462 MB)
64-Bit Command Line Installer (454 MB)

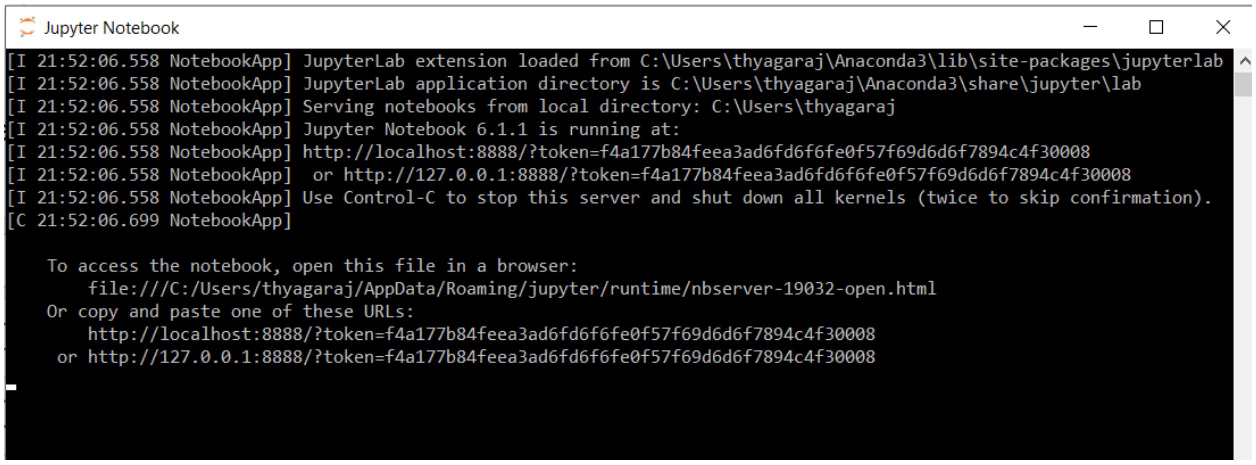
Linux

Python 3.8
64-Bit (x86) Installer (550 MB)
64-Bit (Power8 and Power9) Installer (290 MB)

For Windows and for Python 3.8 there is 64 – Bit and 32-Bit Graphical Installer. Once the appropriate Anaconda installer is downloaded and installed successfully , one can find the Jupyter appearing in the Anaconda folder as illustrated below :



Click the Jupyter Note Book , Server and all Kernels at background will open as illustrated below :

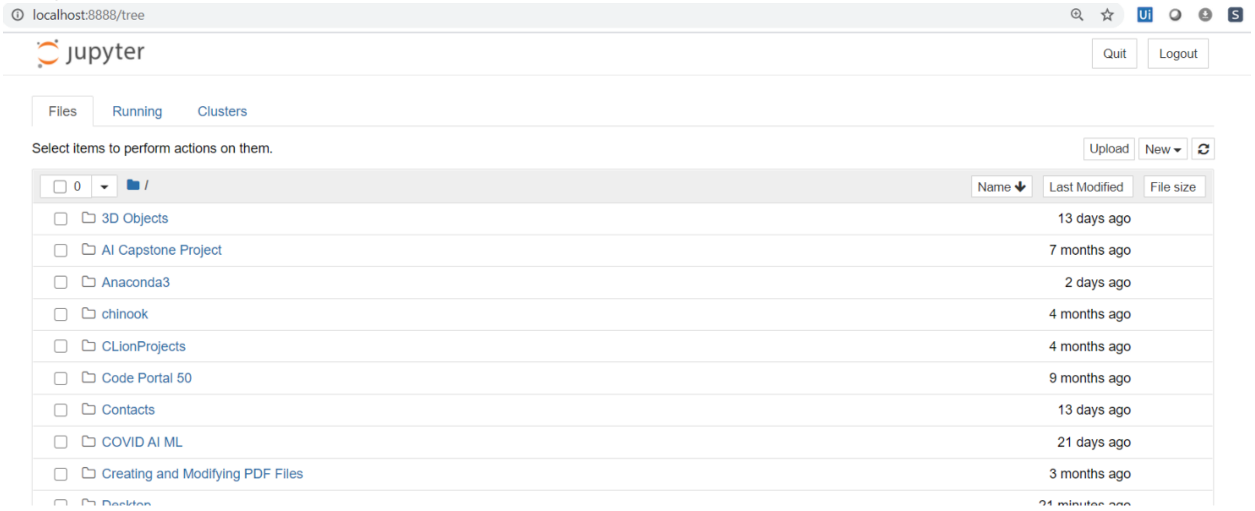


```
[I 21:52:06.558 NotebookApp] JupyterLab extension loaded from C:\Users\thyagaraj\Anaconda3\lib\site-packages\jupyterlab
[I 21:52:06.558 NotebookApp] JupyterLab application directory is C:\Users\thyagaraj\Anaconda3\share\jupyter\lab
[I 21:52:06.558 NotebookApp] Serving notebooks from local directory: C:\Users\thyagaraj
[I 21:52:06.558 NotebookApp] Jupyter Notebook 6.1.1 is running at:
[I 21:52:06.558 NotebookApp] http://localhost:8888/?token=f4a177b84feea3ad6fd6f6fe0f57f69d6d6f7894c4f30008
[I 21:52:06.558 NotebookApp]  or http://127.0.0.1:8888/?token=f4a177b84feea3ad6fd6f6fe0f57f69d6d6f7894c4f30008
[I 21:52:06.558 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 21:52:06.699 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/thyagaraj/AppData/Roaming/jupyter/runtime/nbserver-19032-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=f4a177b84feea3ad6fd6f6fe0f57f69d6d6f7894c4f30008
     or http://127.0.0.1:8888/?token=f4a177b84feea3ad6fd6f6fe0f57f69d6d6f7894c4f30008
```

Soon after the Server is launched, Jupyter notebook will get open. Also, one can open the Jupyter notebook by copy and pasting any one of the following URLs as displayed in the message box:
http://localhost:8888/?token=f4a177b84feea3ad6fd6f6fe0f57f69d6d6f7894c4f30008
or
http://127.0.0.1:8888/?token=f4a177b84feea3ad6fd6f6fe0f57f69d6d6f7894c4f30008

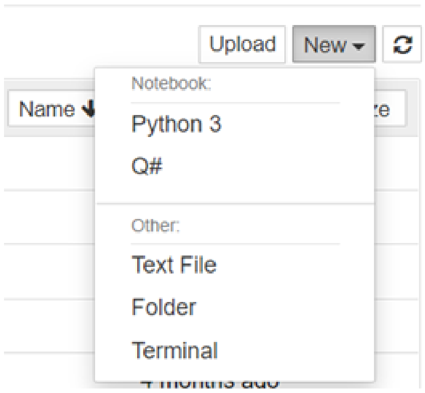
Jupyter Notebook editor will open as illustrated below:



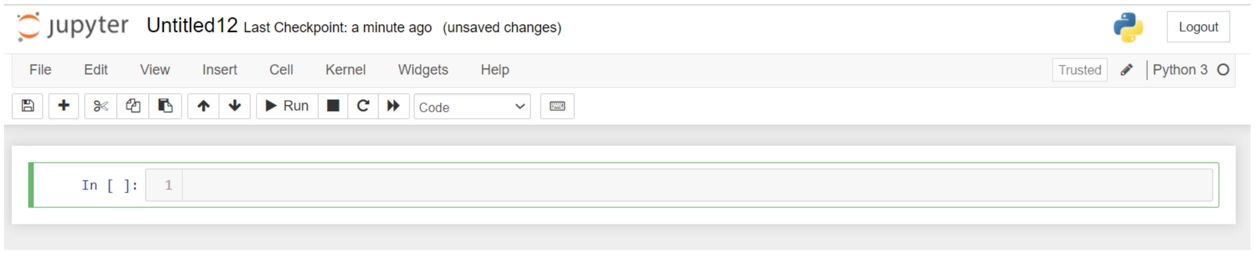
Select items to perform actions on them.

3D Objects	13 days ago
AI Capstone Project	7 months ago
Anaconda3	2 days ago
chinook	4 months ago
CLionProjects	4 months ago
Code Portal 50	9 months ago
Contacts	13 days ago
COVID AI ML	21 days ago
Creating and Modifying PDF Files	3 months ago

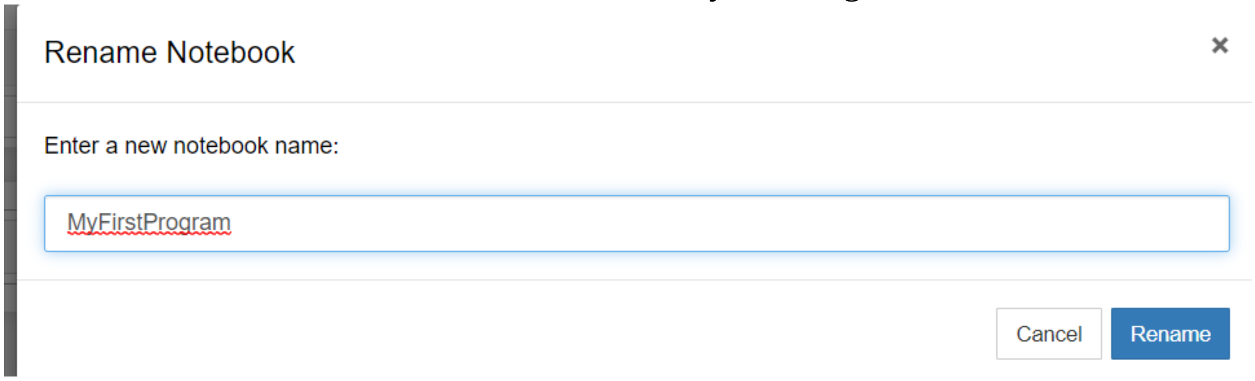
In order to open a new python file, one should click new button and select Python3 , as below :



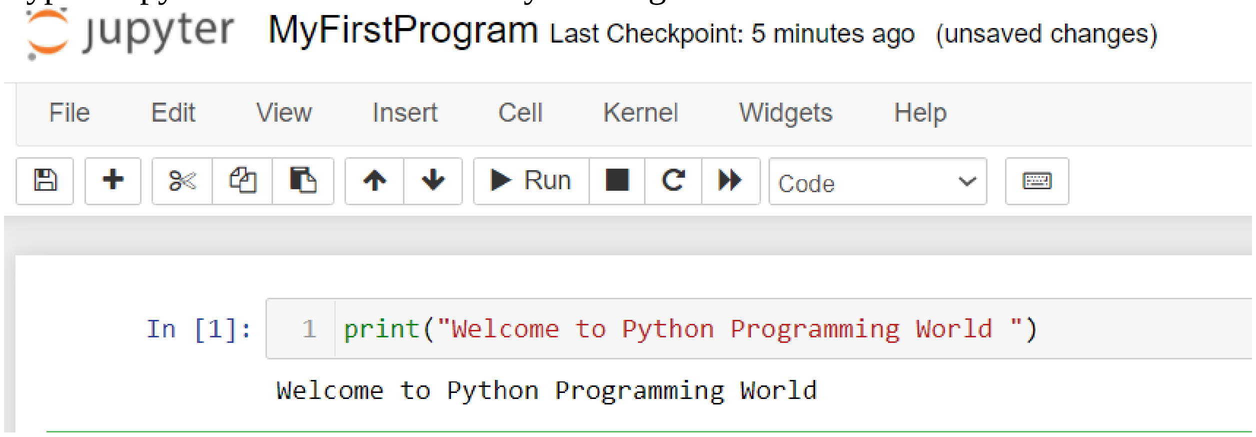
A new untitled Python file with one cell will get open:



Click Untitled12 and Rename the notebook as MyFirstProgram:



Type the python code and execute by clicking Run button :


jupyter
MyFirstProgram Last Checkpoint: 5 minutes ago (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
In [1]:  print("Welcome to Python Programming World ")

         Welcome to Python Programming World
```

Questions for Practice:

1. What is Python ? Who created Python?
2. What are the features of Python which makes it more popular?
3. List out the different jobs available for Python programmer.
4. What is the role of programmer? Lis two skills required to become good programmer.
5. Discuss, Why Should you learn to write Programs?
6. What is Zen of Python?
7. Discuss, with snapshots how to install latest version of IDLE Python and Jupyter Python.
8. Illustrate with examples how to interact with IDLE Python.
9. Explain how mathematical expressions can be executed in interactive shell.
10. Illustrate with example how write and execute programs in Jupyter Editor.
11. Explain the different components of Jupyter Editor.
12. Define Program. Differentiate between Compiler and Interpreter. Give Examples.
13. What are Python words and sentences? Explain with an example for each.
14. Classify the following list of items into variables, values, operators, strings, and keywords:
    List of items: *, + , - , **, < ,> , 'hello' , ‘ I am ok . How are you’, -88.8, /, 5, and, is , not , while , for, async, x, si, p , time , rate , velocity , speed, acc , % ,&, ! , ||
15. What are expressions? Illustrate the different types of expressions with examples.
16. What are data types? Classify the different data types in python with examples.
17. What are Python Variables? What rules one should follow to name the variables.
18. Give 5 examples for valid and invalid variables.
19. Discuss how to store values in a variable.
20. Write a sample program and dissect the program with explanation.
21. What are comments? What are the advantages of Comments? Explain the different ways of writing comments.
22. Give examples for single and multiline comments.
23. Explain the working and usage of print() function with examples.

24. Explain the working and usage of input() function with examples.
25. Give example to read multiple values using input().
26. Define operands and operators. Discuss PEMDAS rules with examples.
27. What are keywords? How many keywords are there in current version of Python?
28. Write a program to display all keywords in the current version of Python.
29. What are Python comments? Explain their importance with programming examples.
30. Explain print () , input() and split() functions with example.
31. Write a program for the following:
   1. To read and print a single value in a single line
   2. To read and print multiple values in a single line

32. Explain the following different types of errors: Syntax errors, Semantic errors and Logic Errors.
33. Explain the following functions with example: len() , str() , int() , float()
34. Predict the out put and justify your answer: (i) -11%9   (ii) 7.7//7 (iii) (200 – 70)*10/5   (iv) not "False"   (v) 5*|**2
35. List the rules to describe a variable in Python. Demonstrate at least three different types of variables uses with an example program.
36. Explain the following :
   1. Skills necessary for a programmer
   2. Interactive Mode
   3. Short circuit evaluation of expression
   4. Modulus operator.

37. Mention three types of errors encountered in python program.
38. Define the following with example : Values and Types , Variables , Expressions , Keywords , Statements , Operators and Operands, Order of Operations , Modulus Operators , String operations and Comments.
39. What three functions can be used to get the integer, floating-point number, or string version of a value?

Programs for Practice:

Write and execute python programs for the following :

1. To prompt a user for their name and then welcomes them.
2. To prompt a user for days and rate per day to compute gross salary.
3. To read the following input and display :
   a. Name :
   b. USN:
   c. Roll No:
   d. Mobile No:
   e. E-Mail Id:
   f. Percentage of Marks:
4. To find the simple interest for a given value of P, T and R. Program should take input from the user.
5. To find the compound interest.
6. To read two integers and find the sum, diff, mult and div.
7. To Convert given Celsius to Fahrenheit temperature.
8. To print ascii value of a character.
9. To display all the keywords.
10. To print the following string in a specific format :”"Twinkle, twinkle, little star, How I wonder what you are! Up above the world so high, Like a diamond in the sky. Twinkle, twinkle, little star, How I wonder what you are" .

    Output :

```
Twinkle, twinkle, little star,
   How I wonder what you are!
          Up above the world so high,
          Like a diamond in the sky.
Twinkle, twinkle, little star,
   How I wonder what you are
```

11. To get a python version.
12. To display the current date and time
13. To accept the radius of a circle from the user and compute the area.
14. To print the calendar of a given month and year.
15. To check whether a file exists .

16. To determine if a Python shell is executing in 32 bit or 64 bit mode on OS.
17. To get OS name , platform and release information .
18. To locate Python site packages.
19. To call an external command in python.
20. To get path and name of the file that is currently executing.
21. To parse a string to float or integer.
22. To list all files in a directory in Python.
23. To print without newline or space.
24. To determine profiling of Python programs.
25. To print to stderr.
26. To access environment variables.
27. To get the current username.
28. To find the local IP addresses using Pythons stdlib.
29. To get execution time for a python method.
30. To convert height in meters to centimeters.
31. To Convert all units of time to seconds
32. To convert the distance in feet to inches , yards and miles.
33. To calculate body mass index.
34. Given variables x = 15 and y = 30 , write a Python program to print "15+30=45".
35. To get the identity of the object
36. To check whether a string is numeric.
37. To get the system time .
38. To clear the screen or terminal
39. To calculate the time runs (difference between start and current time) of a program.
40. To input integer if not generate error.