

History

How did the Python get its name?

- Got its name from a BBC comedy series “**Monty Python’s Flying Circus**”.

Who developed the Python?

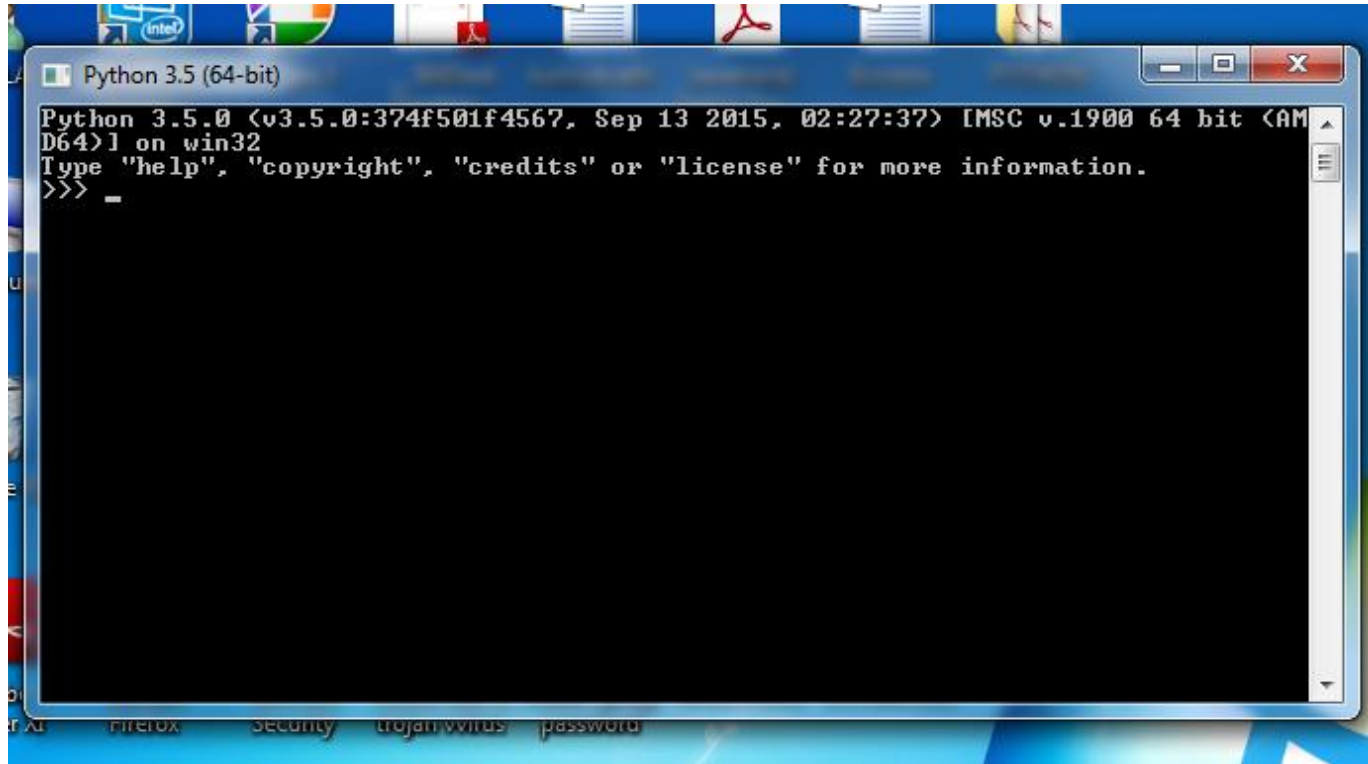
- Conceived in the late 1980, and its implementation began in **December 1989** by “**Guido Van Rossum**” at Centrum Wiskunde & Informatica (CWI) in Netherlands.
- First released in **1991** and presently administered by **Python Software Foundation**.

It is a successor to the **ABC** language with an extra features like **exception handling and extensibility**

ABC :

- Is an imperative **general purpose programming language** and programming environment developed at CWI.
- It is **interactive, structured, high level** and intended to be **used instead of BASIC, Pascal.**
- It is not meant to be systems programming language but is **intended for teaching or prototyping**

(i) Example USING COMMAND PROMPT



```
Python 3.5 (64-bit)
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit <AMD64>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

EX:

```
>>> print("Hello World!")
```

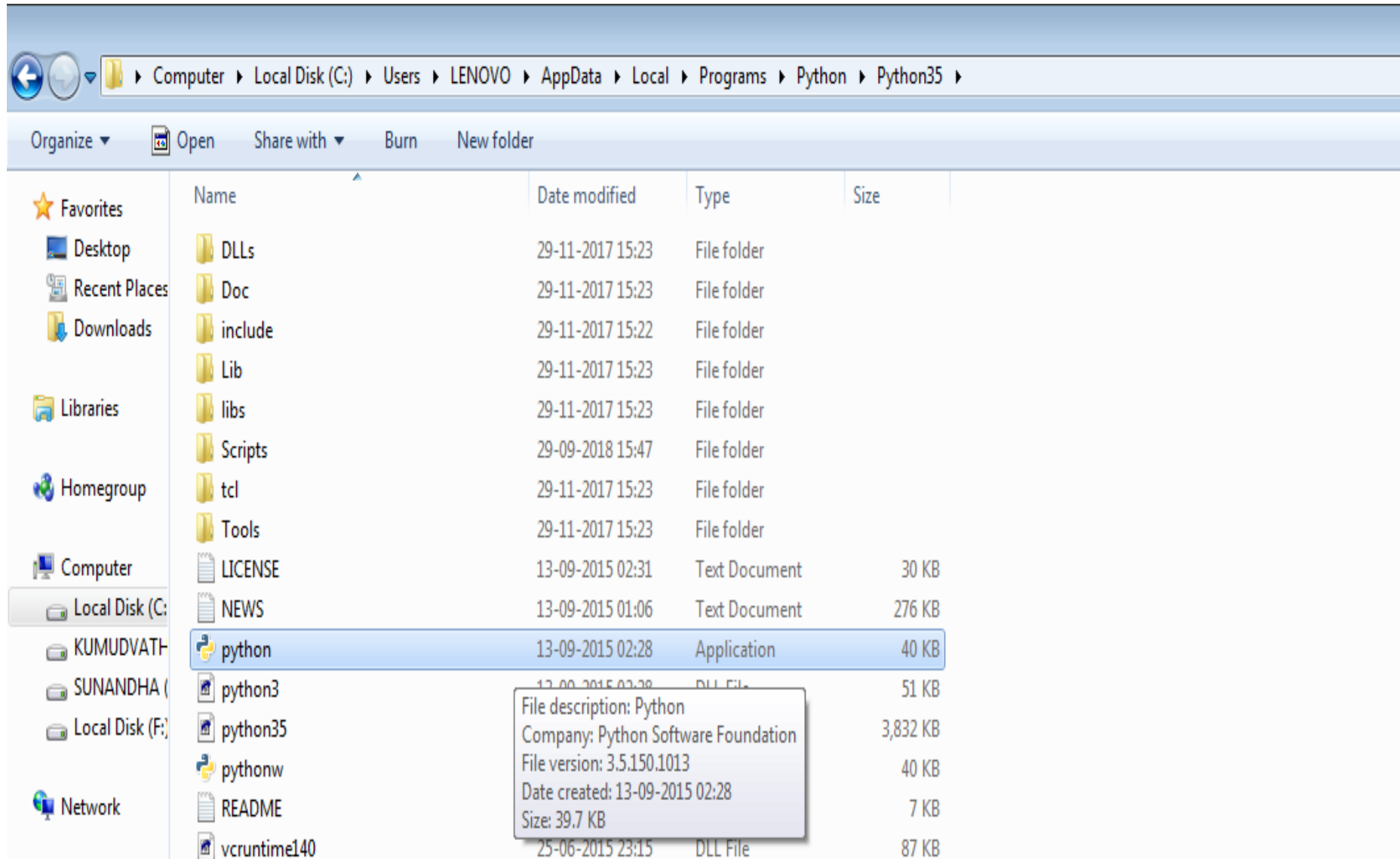
Hello World!

```
>>>quit() # to close the command prompt
```

```
>>>exit() # to close the command prompt
```

```
>>>Ctrl+z # then press enter to close command prompt
```

(ii). Go to the folder containing the **shortcut** or the **installed files** and click on the **Python command Line**.

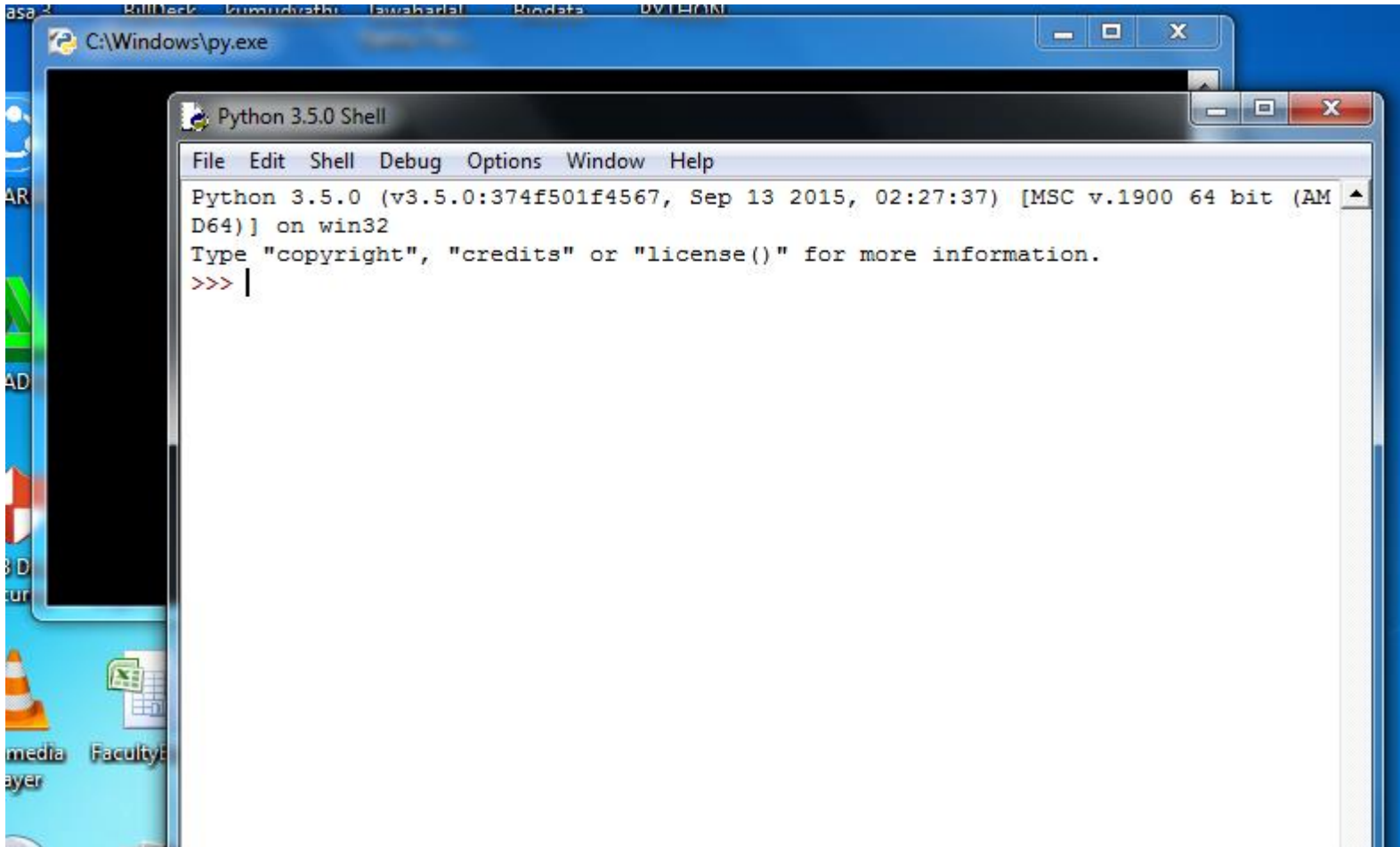


The screenshot shows a Windows Explorer window with the address bar set to `Computer > Local Disk (C:) > Users > LENOVO > AppData > Local > Programs > Python > Python35`. The left sidebar shows the navigation pane with 'Local Disk (C:)' selected. The main pane displays a list of files and folders. The 'python' application file is highlighted, and a tooltip is visible over it.

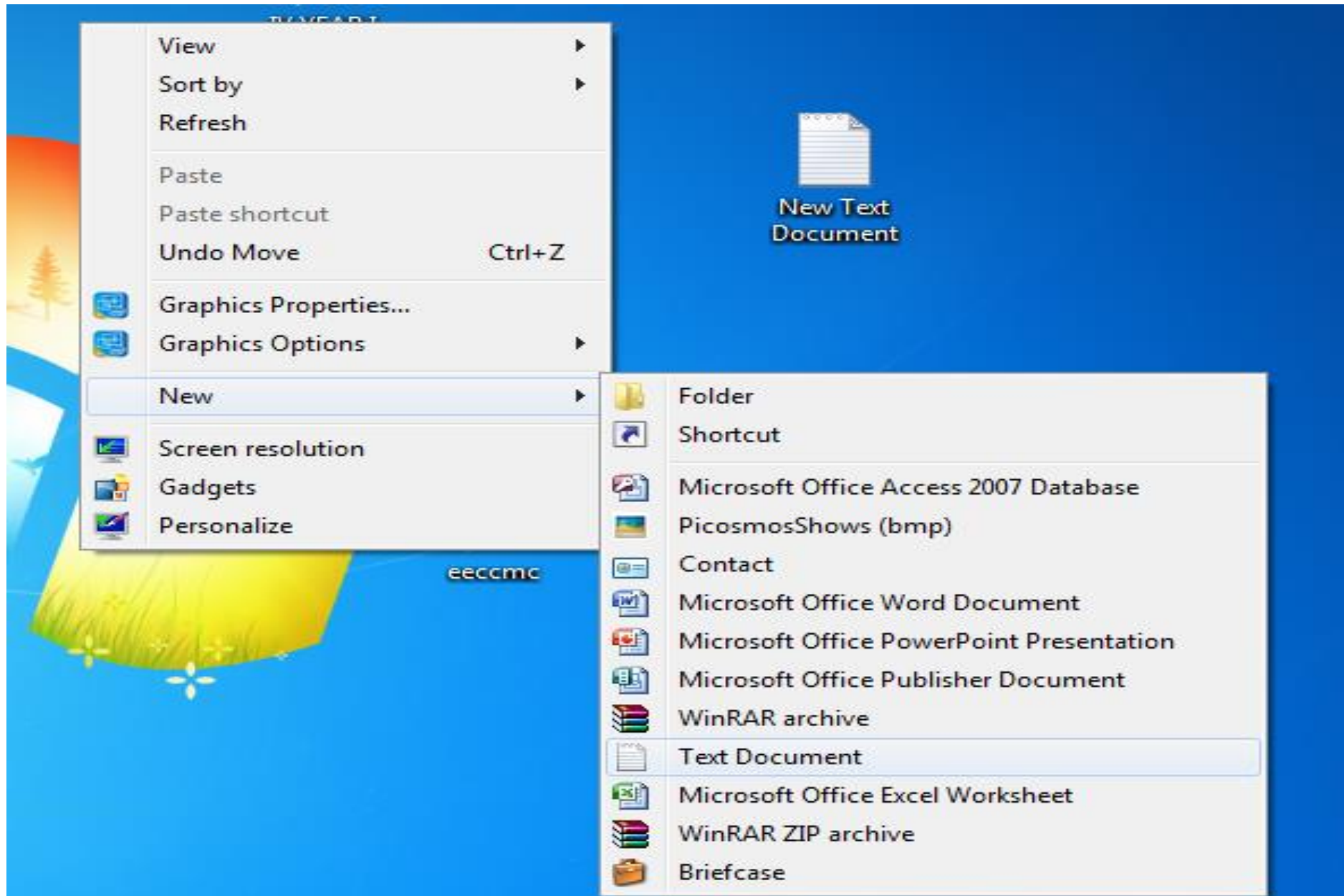
Name	Date modified	Type	Size
DLLs	29-11-2017 15:23	File folder	
Doc	29-11-2017 15:23	File folder	
include	29-11-2017 15:22	File folder	
Lib	29-11-2017 15:23	File folder	
libs	29-11-2017 15:23	File folder	
Scripts	29-09-2018 15:47	File folder	
tcl	29-11-2017 15:23	File folder	
Tools	29-11-2017 15:23	File folder	
LICENSE	13-09-2015 02:31	Text Document	30 KB
NEWS	13-09-2015 01:06	Text Document	276 KB
python	13-09-2015 02:28	Application	40 KB
python3	13-09-2015 02:28	DLL File	51 KB
python35			3,832 KB
pythonw			40 KB
README			7 KB
vcruntime140	25-06-2015 23:15	DLL File	87 KB

File description: Python
Company: Python Software Foundation
File version: 3.5.150.1013
Date created: 13-09-2015 02:28
Size: 39.7 KB

After double clicking on “idle” python file a Python Shell will be displayed for you with all menu items like File, Edit, Shell, Debug, Options, Window, Help.
Now you select “New File” from “File” menu



2. In Script Mode



sum - Notepad

File Edit Format View Help

```
a=1  
b=2  
c=a+b  
print("value of c is",c)
```

Some rules and certain symbols used with regard to statements in Python:

- Hash mark(#) indicates comments
- New line(\n) is standard line separator
- Backslash(\) continues the line
- Semicolon(;) joins two statements on a line
- Colon(:) separates header line from it's suite
- Statements are grouped as suite
- Suites are delimited via indentation
- Python file are organized as “modules”

Python Identifiers

- An identifier can be a combination of uppercase letters, lowercase letters, underscores, and digits (0-9). Ex: abc1_ or _abc1 .
- An identifier should not begin with a number.
- Special characters such as %, @, and \$ are not allowed within identifiers.
- You cannot use Python keywords as identifiers.
- Only Class identifiers begin with an uppercase letter
- You can use underscores to separate multiple words in your identifier.

Indentation

- Java, C, and C++ use braces to denote blocks of code.
- Indentation, by convention, is equivalent to 4 spaces to the right.
- EX:

firstif.py

```
v1=input("Enter the value of v1: ")
if(int(v1)==10):
    print("the value of v1 is: ",v1)
    print("in first if")
v2=input("Enter the value of v2: ")
if(int(v2)==100):
    print("the value of v2 is: ",v2)
    print("in second if")
```

Variables

- Container that stores values for accessing and changing.
- Is a way of pointing to a memory location.
- Other languages declare and bind a variable to a specific data type.
- Python is dynamically typed, meaning no pre-declaration is necessary.

```
>>>my_variable=10 --- indicates that my_variable is integer
```

```
>>>my_variable='hi hello' --- indicates string
```

- You can change the value and datatype.

Numbers

Python has 4 built-in numeric data types:

1.integer(int)

2.floating point numbers(float)

3.complex numbers

4. long----deprecated from python3

Strings

- Is a sequence of unicode characters.
- Is a combination of letters, numbers and special symbols.
- Ex: `s1=john123@gmail.com`
- We enclose string in single and double quotations.
- If string enclosed in single quotes has single quote(') in it then place a back slash before it
- Ex:
 - `S2='It doesn't look good at all'`
 - `SyntaxError: invalid syntax`
 - `>>>s2='It doesn\'t lookgood at all'`

- Strings may be indexed or subscripted

- In Python, indexing starts from zero

```
>>>s="Hello Python"
```

-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
H	e	l	l	o		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11

LISTS

Is similar to arrays in C

can store any type and any number of variables

Contains items separated by comma enclosed within square brackets[]

Ex:

```
>>>mylist=['abcd',20.5,(2+3j),0.05]
```

Methods

1. insert(index,obj)

2.append()

3. Access

4.count(obj)

5. index(obj)

6. Update or Assign

7.Slicing

8.sort()

9. reverse()

10. extend(seq)

11. remove()

TUPLES

Another sequence data type similar to list is tuple

can store any type and any number of variables

Contains items separated by comma enclosed within parentheses()

Ex:

```
>>>mytuple=('abcd',123,20.5,(2+3j),0.05)
```

Difference between List and tuple:

Lists	Tuples
Items are enclosed in <u>brackets[]</u>	Items are enclosed in <u>parentheses()</u>
Elements and size <u>can</u> be changed	Elements and size <u>cannot</u> be changed

- So, tuples are also called as read-only lists

Dictionary

Is a mutable and another container type that can store any number of python objects enclosed in { }.

Is different from sequence type containers like lists and tuples in the way the data is stored and accessed

Is a kind of hash table type

Consists of key-value pair and works like hash

Is like a list but instead of looking up an index to access values you'll be having a unique key which can be a number, string, tuple

Value can be anything

Colon separates a key from it's value

Creating and assigning dictionaries

Creating dictionaries involves simply assigning a dictionary a variable

```
>>> dict1={ } --- creating empty dictionary
```

```
>>> dict1['name']='john'
```

```
>>> dict1['branch']='CSE'
```

```
>>> dict1['subject']='PYTHON'
```

```
>>> dict1
```

```
{'subject': 'PYTHON', 'branch': 'CSE', 'name': 'john'}
```

```
---- assigning variables to the dictionary
```

Types of Operators

Python language supports the following types of operators –

1. Arithmetic Operators
2. Assignment Operators
3. Comparison (Relational) Operators
4. Logical Operators
5. Bitwise Operators
6. Membership Operators
7. Identity Operators

Loops

- In general, statements are executed sequentially – The first statement in a program is executed first, followed by the second, and so on.
- There may be a situation when you need to execute a block of code several number of times.
- Programming languages provide various control structures that allow more complicated execution paths.
- A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement –

Python programming language provides the following types of loops to handle looping requirements.

<p><u>while loop</u></p>	<p>Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.</p>
<p><u>for loop</u></p>	<p>Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.</p>
<p><u>nested loops</u></p>	<p>You can use one or more loop inside any another while, or for loop.</p>

Decision Making

- Decision-making is the expectation of conditions occurring during the execution of a program and specified actions taken according to the conditions.
- Decision statements evaluate multiple expressions, which produce True or False as the outcome.
- You need to determine which action to take and which statements to execute if the outcome is True or False otherwise.

- Python programming language assumes any **non-zero** and **non-null** values as True, and any **zero** or **null values** as False value.
- Python programming language provides the following types of decision-making statements.

S.No.	Statement & Description
<u>if statements</u>	An if statement consists of a boolean expression followed by one or more statements.
<u>if...else statements</u>	An if statement can be followed by an optional else statement , which executes when the boolean expression is FALSE.
<u>nested if statements</u>	You can use one if or else if statement inside another if or else if statement(s).

Loop Control Statements

The Loop control statements change the execution from its normal sequence.

When the execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Python supports the following control statements.

break statement

Terminates the loop statement and transfers execution to the statement immediately following the loop.

continue statement

Causes the loop to **skip the remainder of its body** and **immediately retest its condition** prior to reiterating.

pass statement

The pass statement in Python is used **when a statement is required syntactically but you do not want any command or code to execute.**

A function is

- a block of **organized, reusable code** that is used to perform a single, related action.
- Functions provide **better modularity** for your application and a high degree of code reusing.

Declaration vs. Definition:

`int fact(int);` ----> Declaration

```
int fact(int n)
```

```
{
```

```
    if(n==1)
```

```
        return n
```

```
    else
```

```
        return(n*fact(n-1))
```

Definition

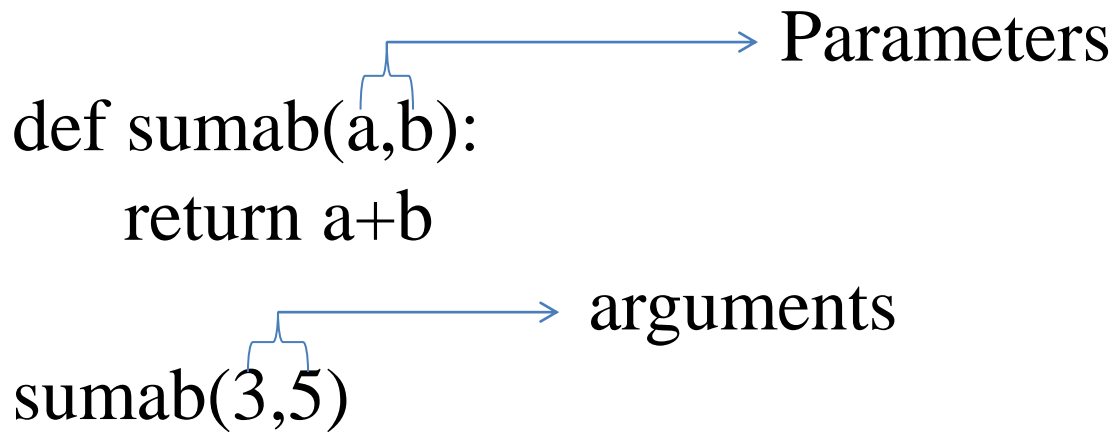
`fact(5);` ----> Calling

Difference between parameter and an argument:

Parameters comes as a part of function **definition**

Whereas **arguments** are part of function **call**

The arguments are the data we pass into the function's parameters



Function Arguments

The following types of formal arguments

1. Required arguments
2. Keyword arguments
3. Default arguments
4. Variable-length arguments

The Anonymous Functions

not declared in the standard manner by **using the def keyword**.

the **lambda** keyword to **create small anonymous functions**.

- can **take any number of arguments** but **return just one value** in the form of an expression.
- **cannot be a direct call** to print because lambda requires an expression.
- have their **own local namespace**
- appears as a one-line version of a function, they are not equivalent to inline statements in C or C++

Syntax

```
lambda [arg1 [,arg2,.....argn]]:expression
```

Example

```
# Function definition is here
```

```
sum = lambda arg1, arg2: arg1 + arg2
```

```
# Now you can call sum as a function
```

```
print ("Value of total : ", sum( 10, 20 ))
```

```
print ("Value of total : ", sum( 20, 20 ))
```

result –

```
Value of total : 30
```

```
Value of total : 40
```

Function vs. Procedure:

Functions concludes by **sending back a return value** to the caller.

procedures are treated as special cases of functions which **do not return a value**.

As python interpreter **implicitly return** a default value **None**, So, in Python **procedures are implied as functions**

Scope of Variables

All variables in a program may **not be accessible at all locations** in that program.

The scope of a variable **determines the portion** of the program where you can access a particular identifier.

There are two basic scopes of variables in Python –

- Global variables
- Local variables

Module :

a module is a **file consisting of Python code.**

It can **define functions, classes and variables.**

can also **include runnable code.**

example of a simple module

sum.py –

```
def sumab():
```

```
    a=int(input("Enter First Number: "))#20
```

```
    b=int(input("Enter Second Number: "))#10
```

```
    return a+b
```

The **import** Statement

The **import** has the following syntax –

```
import module1[, module2[,... moduleN]
```

When the interpreter encounters an import statement, it imports the module

Built-in Functions

1. `__import__()`
2. `dir()`
3. `globals()` and `locals()`
4. `reload()`

Overview of OOP Terminology

Class – A user-defined prototype for an object that **defines a set of attributes** called **data members** (class variables and instance variables) and methods, accessed via dot notation.

Class variable – Defined within a class

Instance variable – A variable that is **defined inside a method** and **belongs only to the current instance** of a class

Data member – A **class variable or instance variable** that holds data associated with a class and its objects.

Function overloading – The assignment of **more than one behaviour** to a particular function. The operation performed **varies by the types of arguments** involved.

Inheritance – The **transfer of the characteristics of a class to other classes** that are derived from it.

Instance – An **individual object of a certain class**.

Instantiation – The **creation of an instance** of a class.

Method – A special kind of function that is **defined in a class definition.**

Operator overloading – Using same operator in **different ways**

Like + for addition and concatenation

* for multiplication and repetition

Creating Classes

The name of the class immediately follows the **keyword *class*** followed by a colon as follows –

```
class ClassName:  
    'Optional class documentation string'  
    class_suite
```

- The *class_suite* consists of all the **component statements defining class members, data attributes and functions.**

EXCEPTION HANDLING

Difference between Error and Exception

Error---

- 1) Syntax Error – **errors with the construct** of the software
Cannot be executed or compiled correctly
Repaired before execution
Domain Failures
- 2) logical error – Caused by **lack of or invalid input**
executing in the **current flow is no longer possible**
Range Failures

Exceptions:

System errors and hardware interruptions

Detecting and Handling done by Operating System

Detection and Handling Exceptions:

detected by **incorporating them as part of a try statement**

Any code suite of try statement will be **monitored for exceptions**

Two main forms of try statements:

try-except-- allows to **detect and handle** exceptions

try-finally– allows to **detect and process** exceptions

A try statement is either accompanied by **one or more except clauses or exactly one finally clause**

Optional **else** for situations where code needs to run when no exceptions are detected

1. try-except:

syntax:

```
try:                                #watch for exceptions here
    try_suite
except Exception: #exception-handling code
    except_suite
```

2. try statement with multiple excepts:

To **handle different types of exceptions** with the same try

Syntax:

```
try:                                #watch for exceptions here
    try_suite
except Exception1: #exception-handling code
    except_suite_for_Exception1
except Exception2: #exception-handling code
    except_suite_for_Exception2
```


Assertions:

An assertion is a sanity-check that you can **turn on or turn off when you are testing the program.**

diagnostic predicates **must evaluate to Boolean True**, otherwise an exception is raised to indicate that the expression is False

an assertion **is like a raise-if** statement

works similar to Macros in C

assert expression[,args]

Exception

StandardError

SystemError

ArithmeticError AssertionError AttributeError EOFError EnvironmentError

FlotingPointError OverflowError ZeroDivisionError IOError OSError

ImportError KeyboardInterruptError LookupError MemoryError NameError

IndexError KeyError UnboundLocalError

RuntimeError

SyntaxError

TypeError

ValueError

NoImplementedError

IndentationError

UnicodeError