Table of Contents

Module 1- Introduction to Python Programming Language

Python is a high-level, general-purpose programming language. The latest version of Python is 3. Python 3 has the ability to offer the web development environment, Machine Learning applications, Natural Language Processing (NLP) and Big data Analytics.

Python language is used by corporate tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber etc. in very highly manner.

A. Installation and setup

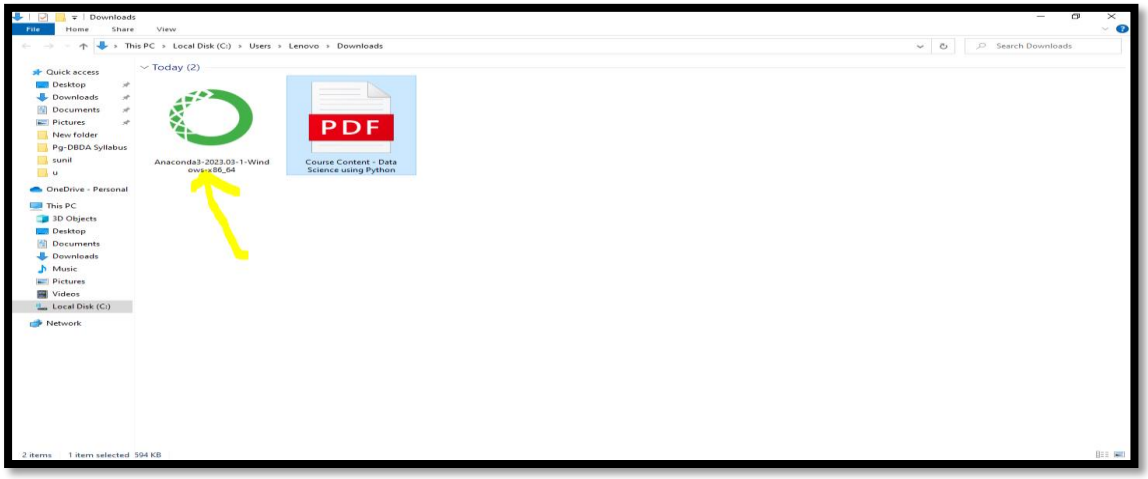
The Python language is a good choice for both beginners and experienced corporate industry developers. Python has the ability in scripting, automation, big data analysis, machine learning, and web- development. Python installation can be done by using the following steps-

A1. Prerequisites

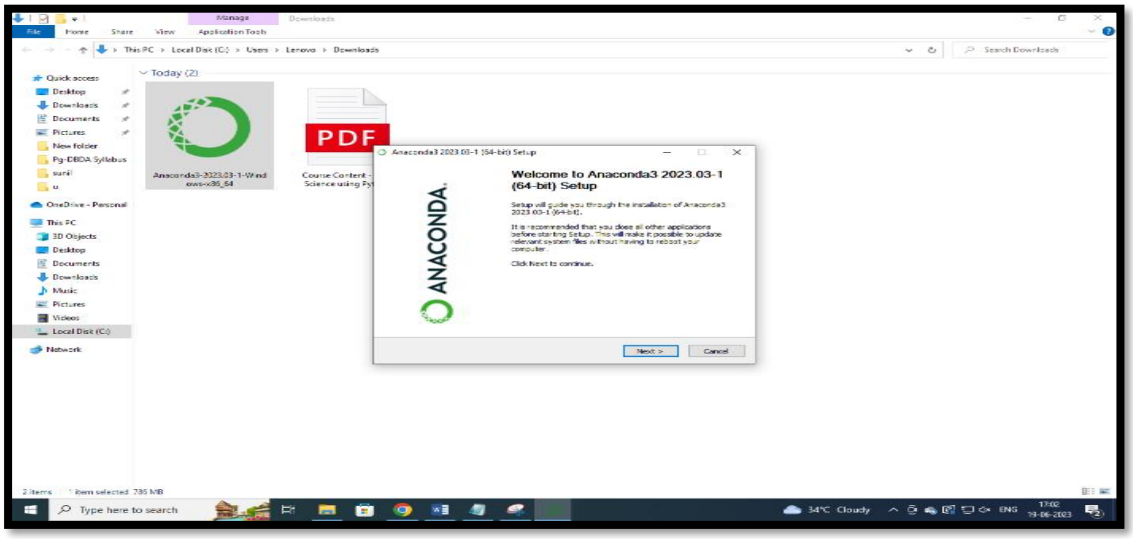
Computer system with Windows 10 and I3 processor with an internet connection.

Step 1- Download the Anaconda installer by using the URL **https://www.anaconda.com/**

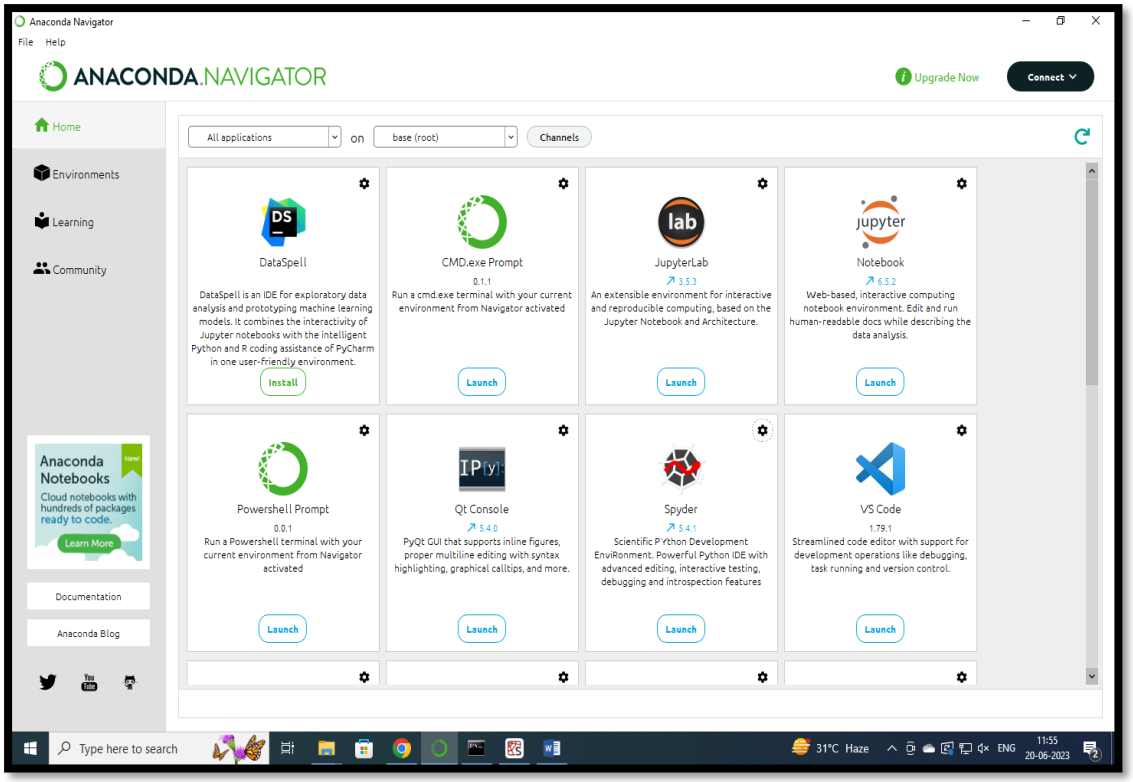
Step 2- Go to your Downloads folder and double-click the installer to launch.



Step 3- Click on the Next Button and follow the installation guidelines as per installation wizard-



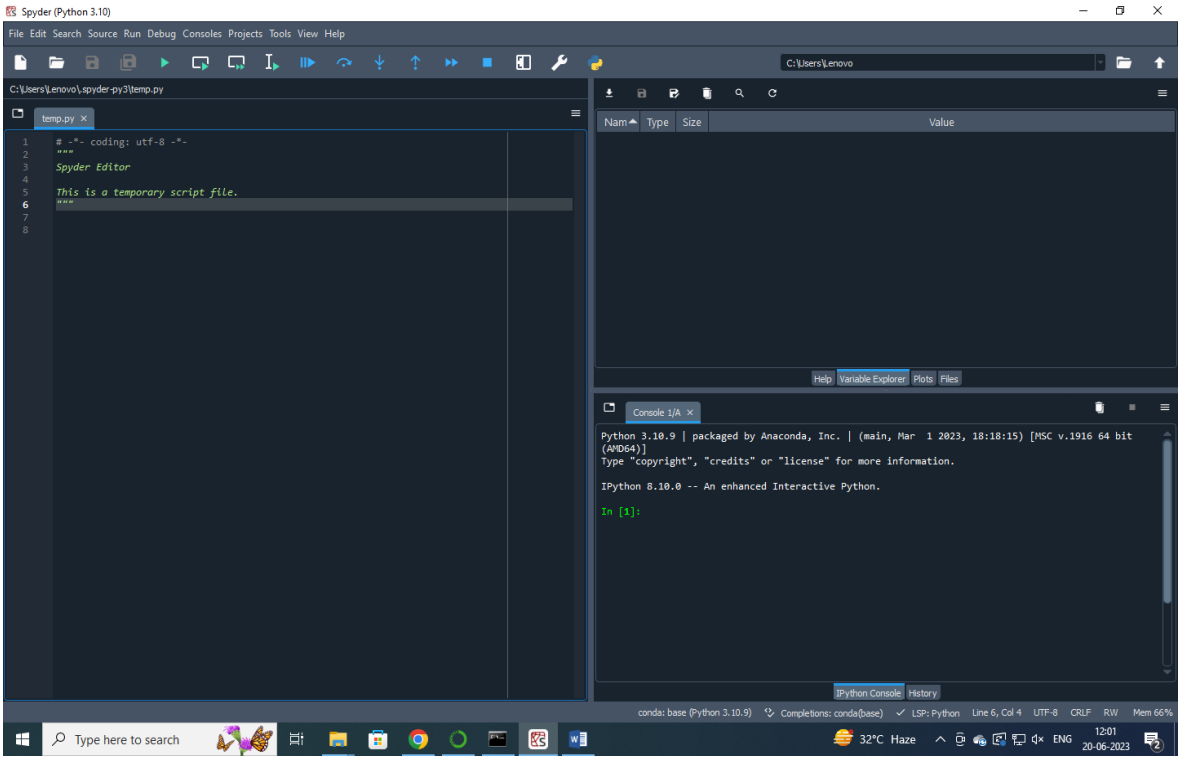
Step 4- After completion of installation of Anaconda this will be the Home screen -



Now Your Anaconda Platform is installed and ready for use and you can launch for the use of Spyder/ Jupyter tool for Python programming as per your interest-

B. Creating and saving a script file -

First Screen Layout of the Spyder tool. You can create any file by using the file options.



C. Basics of Python

Python is a very popular key programming language in today time. It was created by Guido van Rossum, and released in 1991. Python language can be used in following key domains for:

- Creating the web applications.
- Python can connect to database systems. We can perform CRUD operation on data base.
- Python can be used to handle big data and perform complex mathematical analysis.
- Python can be used for rapid prototyping, or for production-ready software development.
- Python can works on various platforms like Windows, Mac, Linux, etc.

C1. Features offered by the Python Programming Language-

There are many features of Python programming as mentioned below-

- **Python is Easy to use and learn:** Python has a simple and easy-to-understand syntax, unlike traditional languages like C, C++, Java, etc., making it easy for beginners to learn.
- **Expressive Language:** It allows programmers to express complex concepts in just a few lines of code or reduces Developer's Time.
- **Interpreted Language:** Python does not require compilation, allowing rapid development and testing. It uses Interpreter instead of Compiler.
- **Object-Oriented Language:** It supports object-oriented programming, making writing reusable and modular code easy.
- **Open Source Language:** Python is open source and free to use, distribute and modify.
- **Extensible:** Python can be extended with modules written in C, C++, or other languages.
- **Learn Standard Library:** Python's standard library contains many modules and functions that can be used for various tasks, such as string manipulation, web programming, Data Analysis, Natural Language Processing etc.
- **GUI Programming Support:** Python provides several GUI frameworks, such as Tkinter and PyQt, allowing developers to create desktop applications easily.
- **Integrated:** Python can easily integrate with other languages and technologies, such as C/C++, Java, and . NET.
- **Embeddable:** Python code can be embedded into other applications as a scripting language.

- **Dynamic Memory Allocation:** Python automatically manages memory allocation, making it easier for developers to write complex programs without worrying about memory management.
- **Wide Range of Libraries and Frameworks:** Python has a vast collection of libraries and frameworks, such as NumPy, Pandas, Django, and Flask, that can be used to solve a wide range of problems.
- **Versatility:** Python is a universal language in various domains such as web development, machine learning, data analysis, scientific computing, and more.
- **Large Community:** Python has a vast and active community of developers contributing to its development and offering support. That is useful for beginners to get help and learn from experienced industry experts.
- **Career Opportunities:** Python is a highly popular language in the job market. Learning Python can open up several career opportunities in data science, artificial intelligence, web development, and more.
- **Increased Productivity:** Python has a simple syntax and powerful libraries that can help developers write code faster and more efficiently.
- **Big Data and Machine Learning:** Python has become the go-to language for big data and machine learning. Python is very popular among the data scientists and machine learning engineers due to his advanced libraries like NumPy, Pandas, Scikit-learn, Tensor Flow etc.

C2. Applications areas of Python Language -

Python is a general-purpose, popular programming language, and it is used in almost every technical field. There are many applications area of python in real time -

- **Data Science:** Python is an important language for this field because of its simplicity, ease of use, and availability of powerful data analysis and visualization libraries like NumPy, Pandas, and Matplotlib.
- **Desktop Applications:** PyQt and Tkinter are useful libraries that can be used in GUI - Graphical User Interface-based Desktop Applications.
- **Console-based Applications:** Python offers the features to create command-line or console-based applications.
- **Mobile Applications:** While Python is not commonly used for creating mobile applications, it can still be combined with frameworks like Kivy or BeeWare to create cross-platform mobile applications.

- **Artificial Intelligence:** AI is an emerging Technology, and Python is a perfect language for artificial intelligence and machine learning because of the availability of powerful libraries such as Tensor Flow, Keras, and PyTorch.
- **Web Applications:** Python is commonly used in web development on the backend with frameworks like Django and Flask and on the front end with tools like JavaScript and HTML.
- **Enterprise Applications:** Python can be used to develop large-scale enterprise applications with features such as distributed computing, networking, and parallel processing.
- **3D CAD Applications:** Python can be used for 3D computer-aided design (CAD) applications through libraries such as Blender.
- **Machine Learning:** Python is widely used for machine learning due to its simplicity, ease of use, and availability of powerful machine learning libraries.
- **Computer Vision or Image Processing Applications:** Python can be used for computer vision and image processing applications through powerful libraries such as OpenCV and Scikit-image.
- **Speech Recognition:** Python can be used for speech recognition applications through libraries such as Speech Recognition and PyAudio.
- **Scientific computing:** Libraries like NumPy, SciPy, and Pandas provide advanced numerical computing capabilities for tasks like data analysis, machine learning, and more.
- **Testing:** Python is used for writing automated tests, providing frameworks like unit tests and pytest that help write test cases and generate reports.
- **Gaming:** Python has libraries like Pygame, which provide a platform for developing games using Python.
- **IoT:** Python is used in IoT for developing scripts and applications for devices like Raspberry Pi, Arduino, and others.
- **Networking:** Python is used in networking for developing scripts and applications for network automation, monitoring, and management.
- **DevOps:** Python is widely used in DevOps for automation and scripting of infrastructure management, configuration management, and deployment processes.
- **Finance:** Python has libraries like Pandas, Scikit-learn, and Stats models for financial modelling and analysis.

- **Audio and Music:** Python has libraries like Pyaudio, which is used for audio processing, synthesis, and analysis, and Music21, which is used for music analysis and generation.
- **Writing scripts:** Python is used for writing utility scripts to automate tasks like file operations, web scraping, and data processing.

C3. Popular Frameworks and Libraries of Python Programming Language

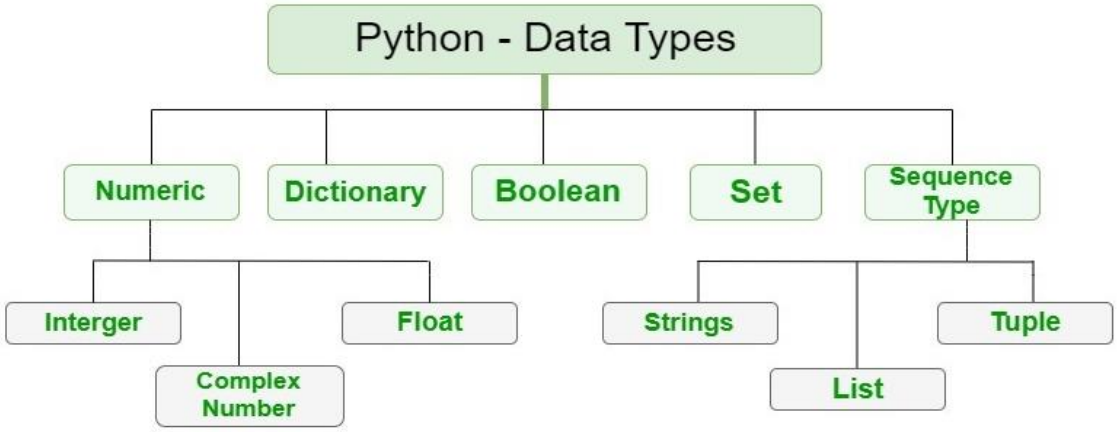
Python 3.10. 4 is the latest stable version published on March 24, 2023. Python has a wide range of libraries and frameworks that can be used in various fields such as machine learning, artificial intelligence, web applications, etc. Some of the popular frameworks and libraries of Python are given below.

- **Web development (Server-side) -** Django Flask, Pyramid, CherryPy (used by YouTube, Instagram, Dropbox)
- **GUIs based applications –** PYQT,PYGTK, PyJs, etc.
- **Machine Learning -** Tensor Flow, PyTorch, **Scikit-learn**, Matplotlib, Scipy, etc.
- **Mathematics -** NumPy, Pandas, etc.
- **Image processing (like OpenCV, Pillow)**
- **Beautiful Soup:** a library for web scraping and parsing HTML and XML
- **Requests:** a library for making HTTP requests
- **SQLAlchemy:** a library for working with SQL databases
- **Kivy:** a framework for building multi-touch applications
- **Pygame:** a library for game development
- **Pytest:** a testing framework for Python
- **Django REST framework:** a toolkit for building RESTful APIs
- **FastAPI:** a modern, fast web framework for building APIs
- **Streamlit:** a library for building interactive web apps for machine learning and data science
- **NLTK:** a library for natural language processing

D. Data Types

Data types is a key aspect in the programming language. It defines what type of data we are going to store in a variable. We use Variables to store different types of data as per real time user requirements.

For example- The age of an employee is 27 Years then I will stored as a numeric value and his address is Delhi then his address will be stored as in characters. Python has by default built-in data types as below-



D1. Numeric Data Type

Python numeric data types store numeric values. Python has four type of numerical data types −

- Int
- Long
- Float
- Complex

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

```
a=25                    # integer variable.
Print ("Variable having the value" , a, " is ", type(a))
b=70.245                # float variable.
Print ("Variable holds the value", b, " is ", type(b))
c=5+7j                  # complex variable.
Print ("Variable having the value", c, " is ", type(c))
```

D2. Python Sequence data type

Sequence Data Types are used to store data in containers in the Python language. The different types of containers are used to store the data **List, Tuple**, and **String.**

D3. Python Dictionary

Python dictionaries are kind of hash table type. It work on the terminology of key-value pairs.

D4. Python Boolean Data Types

Python Boolean type is also built-in data types that represents results in two values either True or False. Python has bool () function to evaluate the value of any expression.

```
a = True                # the value of a
Print (a)
Print (type (a))        # Find the data type of a
```

D5. Set

Set is also a data type used to store the data. It is an unordered collection of the data type. Set is iterable, mutable i.e can modify after creation and did not allow the duplicate items in the set.

The order of the elements is also undefined and it may return the changed sequence of the element. The set can be created by using the built-in function **set ().**

```
set1 = set ()                       # Set Creation
set2 = {'James', 2, 3,'Python'}     #Putting elements in set and Printing.
print (set2)
set2.add (20)                       # Adding element to the set
print (set2)
set2.remove (3)                     #removing element from the set
print (set2)
```

E. Variables

Variable is the fundamental concept in programming language which is use to refer the memory location where the data value will be store for the calculations/ manipulation. The variables acts like a containers and holds the data values. Equal operator **(=)** is used to assign values to the variables. Variables also known identifiers.

Example- John age is 25 years, here age is a variable and it's the value i.e. age = 25;

E1. Rules for declaration of Python variables

- Python variable name may be start from the alphabet character or underscore (_). It cannot be start with a number like '**1age'** is not a valid variable.
- In python variables declaration, the characters except the first character may be an alphabet of lower-case (a-z), upper-case (A-Z), underscore (_), or numbers (0-9).

- Variable name must not contain any white-space, special character (!, @, #, %, ^, &, *).
- Reserved keywords cannot be used as Python variable.
- Python Variables names are case sensitive i.e. **'age'** and **'AGE'** is not same.

E2. Declarations and value assignment to a Variable

We can declare any variable by using the below method-

Syntax: **Variable-name** = value i.e. [age = 25]

E3. In python we can assign single value to multiple variables

Syntax: **Variable1 = Variable2 = Variable3 =** Value i.e. **[a = b = c = 5]**

E4. In python we can assign different multiple values to multiple variables

```
a, b, c = 15, 25.9, "CDAC"
Print (a)
Print (b)
Print( c )
```

E5. Python also supports multi-word terminology to declare a variable

We can declare a variable by using more than one words by using the underscore (_) or without whitespace. There are three cases can be applied for such kind of declarations-

- **Camel Case -** In camel case, each word or abbreviation in the middle of begins with a capital letter. There is no intervention of whitespace.

  For example – **'ageOfStudent'** = 20, here '**ageOfStudent**' is a variable name.
- **Pascal Case –** Pascal case is same as the Camel Case, but here the first character of each word will be in capital. For example – '**AgeOfStudent' = 20**
- **Snake Case –** In snake case word are connected through underscore (_) character**.**

  For example – **'age_of_Student'** = 20, here '**age_of_Student**' is a variable name.

E6. Types of Variable

There are two types of variables in python programming language:

E6.1. Local Variable: Local variables the variables that's are initialized inside a function and belong only to that particular function or block. They cannot be accessed anywhere outside the function.

```
# declaring a local variable inside a function
def add():
 # defining local variables that has scope only inside the function.
  a = 20
  b = 30
  c = a + b
print("The sum is:", c)
# calling a function
```

add()

E6.2. Global Variable: These variables can be used throughout the program and they have its scope in the entire program. We can use global variables inside or outside the function.

A variable can be declared outside the function in python by using **'global'** keyword. Python provides the **global** keyword to use global variable inside the function. If a variable did not declared by using the **global** keyword, variable shall be treated as local variable.

```
x = 101                              # Declare a variable and initialize it
def myfun():                         # Global variable in function
global x                             #using a global variable
print(x)
x = 'Welcome to CDAC-Delhi'          # Changing the value of a global variable
print(x)
myfun()                              # Calling a function
print(x)
```

E7. Deleting a variable

A variable can be delete by using the **'del'** keyword.

Syntax: del <variable-name>

For Example:

```
x = 6
print(x)
del x                    # deleting a variable.
print(x)
```

F. Syntax and Comments in Python

Python is very popular programming language. It offers key features to write and execute the program. Python indentation is the fundamental features that has to follow while we are writing any python program.

F1. Indentation

Indentation refers to the spaces at the beginning of a code line. Python uses indentation to indicate a block of code. Python interpreter will indicate an error if you skip the indentation.

For Example:

```
if 20 > 12:                                    # Code with indentation
  print(" Twenty is greater than twelve!")
```

```
if 20 >12:                                          # Code without indentation
print(" Twenty is greater than twelve!")   [Error will be generated by the interpreter]
```

F2. Comments

Comments are the instructions or statement or line of code that will be ignored by the python interpreter during the execution of code. Comments used by the developer to explain the detailed about the code or flow inside the code.

Comments may be categorised in two types –

A. Single Line comment: Single comment can be put by using the **#** (Hash).

For Example: # I am working on Python [Single Line comment]

print (" I am studying")

B. Multiline comment: Multiline comment can be inserted by using the (#) in the beginning of the code lines or triple quotes **['''……..''']**.

For Example: ''' I am working [Multiline comments]

On my project

In java language'''

G. Python string manipulation

Python has string data type to store the text data like address, name etc. There are many built-in function has been offered by the python –

G1. String data type

String data type is used to store the text data. It store the real time information by using the single quotes or double quotes. Like 'CDAC' or "CDAC" is the same.

Assigning single line string value to a variable:

Syntax: Variable-name = 'data' [name = 'CDAC']

Assigning Multiline strings to a variable:

Python offers a key features in which we can assign a multiline string to a variable by using three quotes ("""……..""") or (''' ……..''').

Syntax:

```
details = """Big data technology,
Can use for analytics,
Python is used for making,
The data analytics."""
print (details)
```

G2. String Indexing

Strings are ordered sequences of character data. Indexing allows you to access specific characters in a string directly by using a numeric value. String indexing is zero-based, the first character in the string has index 0, and the next is 1 etc. You can use square brackets [ ] to specify the index position.

```
job = "I am working in CDAC."
x = job. Index ("working")
print(x)
```

G3. Slicing

Slicing is a powerful feature in Python that allows you to extract a portion of a string by specifying a range of indices.

Slicing can be done in python by two ways:

- By using the **slice()** method
- By using the **array slicing [::]** method

The syntax for slicing a string: string [**start: end**], where **start** is the starting index, **end** is the ending index (exclusive).

```
mytext = "Big data, Technology"
substring = mytext[5:10]     # after Slicing
print(substring)  # Output: 'ata,'
```

G3.A Slicing from the Start of the string:

If we want to slice any string from the start then no need to mention **start range**. The syntax is mentioned below:

Syntax: [: endrange] **i.e [2 : 5]**

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Values	W	E	L	C	O	M	E	S

G3.B Slicing from the end of the string

We can slice any string from the end also. No need to mention **end range**. The syntax is mentioned below:

Syntax: [start_range:] **i.e [5:]**

G3.C Negative Indexing

Negative indexing is also a fundamental method to slice a string. In this method the string will be slice from the end and indexing will be start from last i.e. -1.

Syntax: (String_name [start_range:-endrange])

G4. Concatenation

String concatenation is the fundamental method to merging or adding more than strings. It return a new string after concatenation operation.

There are following method are available to concatenate the strings:

I. Using (+) operator

We can use (+) operator to perform the concatenation on the strings as below:

Syntax: (string1 + string2)

For Example:

```
name = "CDAC”                /* string1
location = "Delhi"           /* string2
add = (name + location)      /* concatenation using (+) operator
print (add)
```

II. Using join() method

Join() method also can be used to perform the concatenation operation:

Syntax: join([var1, var2])

```
name = "CDAC"
loc = "Delhi"
det = "  ".join([name, loc])        # join() method is used for merging the string
 print(det)                        # [“_”] we used before join method for giving space between the string after merging operation.
```

{Key Point to be remember: The time complexity of (+) and join methods are **O (n),** where n is the total length of the strings.}

III. Using % operator

This method also can be used to perform the concatenation operation:

Syntax: ("% s % s" % (name, loc))

```
name = "CDAC"
loc = " Delhi"
 # % Operator is used here to combine the string
print("%  s % s" % (name, loc))
```

IV. Using format() function

This method also can be used to perform the concatenation operation as below:

```
name = "CDAC"
loc = "Delhi"
```

```
print ("{} {}".format(name, loc))          /* use of format function
 # we can also store the result in another variable
var3 = "{} {}".format (var1, var2)
 print (var3)
```

V. Using , (comma) separator

(,) comma separator method also used for marking the concatenation operation:

Syntax: (var1, var2)

For Example:

```
var1 = "big"
var2 = "data"
print(var1, var2)                    /* (,) comma separator for merging the strings.
```

G5. Formatting

String formatting is a technique that allow to input the data in a fixed format and obtain the output in the specified or required form. It is also known **String interpolation.**

Syntax: string. **Format (value1, value2...)**

#named indexes:

```
det1 = "My name is {fname}, I'm {age}".format(fname = "smith", age = 26)
print(det1)
```

A list of formatting characters is given below, which we can use for the string formatting:

:<	Left aligns the result (within the available space)
:>	Right aligns the result (within the available space)
:^	Center aligns the result (within the available space)
:=	Places the sign to the left most position
:+	Use a plus sign to indicate if the result is positive or negative
:-	Use a minus sign for negative values only
:	Use a space to insert an extra space before positive numbers (and a minus sign before negative numbers)
:,	Use a comma as a thousand separator
:_	Use a underscore as a thousand separator
:b	Binary format

:c	Converts the value into the corresponding unicode character
:d	Decimal format
:e	Scientific format, with a lower case e
:E	Scientific format, with an upper case E
:f	Fix point number format
:F	Fix point number format, in uppercase format (show inf and nan as INF and NAN)
:g	General format
:G	General format (using a upper case E for scientific notations)
:o	Octal format
:x	Hex format, lower case
:X	Hex format, upper case
:n	Number format
:%	Percentage format

H. Introduction to Python Operators

In Python programming, Operators are used to perform the arithmetic and logical operations on values and variables. There are reserved keywords / symbols which are used for making the logical and arithmetic operations. There are two key entities we used –

OPERATORS: These are the special symbols that is used to perform operation. Eg.- + , * ,%, /etc.

OPERAND: It is the value on which the operator is applied.

There are following types of operators in python as below mentioned:

H1. Arithmetic Operators

Python arithmetic operators are used to perform mathematical operations on numerical values. These operations are Addition, Subtraction, Multiplication, Division, Modulus, Exponents and Floor Division.

There are following list of operators and their symbols are given with their working for making the logical and mathematical operations on the value.

Syntax: operands 1 (operator) operands 2

Operator symbol	Description	Example
+	Addition	20 + 50 = 70
-	Subtraction	50 – 30 = 20
*	Multiplication	20 * 3 = 60
/	Division	30 / 10 = 3
%	Modulus	33 % 11 = 3
**	Exponent	3**2 = 9
//	Floor Division	7//3 = 1

For example:

a = 21

b = 10

```
print ("a + b : ", a + b)          # Addition
print ("a - b : ", a - b)          # Subtraction
print ("a * b : ", a * b)          # Multiplication
print ("a / b : ", a / b)          # Division
print ("a % b : ", a % b)          # Modulus
print ("a ** b : ", a ** b)        # Exponent
print ("a // b : ", a // b)        # Floor Division
```

Output:

```
a + b :  31
a - b :  11
a * b :  210
a / b :  2.1
a % b :  1
a ** b :  16679880978201
a // b :  2
```

H2. Comparison Operators or Relational Operators

Python has comparison operator is also known as relational operator. They are used to compare the values.

List of relational operator has been given below:

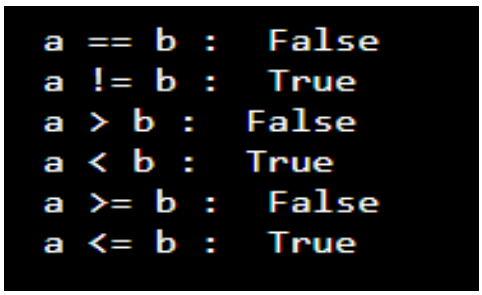
==	Equal	4 == 5 is not true.

!=	Not Equal	4! = 5 is true.
>	Greater Than	4 > 5 is not true.
<	Less Than	4 < 5 is true.
>=	Greater than or Equal to	4 >= 5 is not true.
<=	Less than or Equal to	4 <= 5 is true.

For Example:

```
a = 4
b = 5
print ("a == b : ", a == b)          # Equal
print ("a != b : ", a != b)          # Not Equal
print ("a > b : ", a > b)            # Greater Than
print ("a < b : ", a < b)            # Less Than
print ("a >= b : ", a >= b)          # Greater Than or Equal to
print ("a <= b : ", a <= b)          # Less Than or Equal to
```

Output:



H3. Logical Operators

Logical operators is the special types of operators which are used to combine conditional statements. These are the following logical operators:

Operator	Description	Syntax
AND	Logical AND: True if both the operands are true	x and y
OR	Logical OR: True if either of the operands is true	x or y
NOT	Logical NOT: True if the operand is false	not x

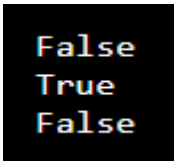
For example:

a = True

b = False

```
print (a and b)        # AND operator
print (a or b)         # OR operator
print (not a)          # Not operator
```

Output:



H4. Assignment Operators

Assignment operators are used to assign values to variables. These operators are used to perform various operations –

Operator	Name	Example
=	Assignment Operator	b = 10
+=	Addition Assignment	b += 5 (Same as b = b + 5)
-=	Subtraction Assignment	b -= 5 (Same as b = b - 5)
*=	Multiplication Assignment	b *= 5 (Same as b = b * 5)
/=	Division Assignment	b /= 5 (Same as b = b / 5)
%=	Remainder Assignment	b %= 5 (Same as b = b % 5)
**=	Exponent Assignment	b **= 2 (Same as b = b ** 2)
//=	Floor Division Assignment	b //= 3 (Same as b = b // 3)

For Example:

```
a =30
a += 4                          # Addition Assignment
print ("a += 4 : ", a)
a -= 6                          # Subtraction Assignment
print ("a -= 5 : ", a)
a *= 5                          # Multiplication Assignment
print ("a *= 5 : ", a)
```

```
a /= 5                          # Division Assignment
print ("a /= 5 : ",a)
a %= 3                          # Remainder Assignment
print ("a %= 3 : ", a)
a **= 2                         # Exponent Assignment
print ("a **= 2 : ", a)
a //= 3                         # Floor Division Assignment
print ("a //= 3 : ", a)
```

Output:

```
a += 4 :  34
a -= 5 :  28
a *= 5 :  140
a /= 5 :  28.0
a %= 3 :  1.0
a **= 2 :  1.0
a //= 3 :  0.0
```