**Introduction To Python:**

Python is a general-purpose, dynamic, high-level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is an easy-to-learn yet powerful and versatile scripting language, which makes it attractive for Application Development.

With its interpreted nature, Python's syntax and dynamic typing make it an ideal language for scripting and rapid application development.

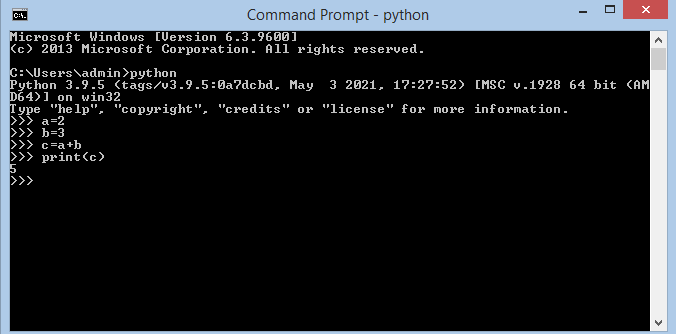
Python supports multiple programming patterns, including object-oriented, imperative, and functional or procedural programming styles.

Key features and characteristics of Python include:

1. **Readability:**
   * Python emphasizes readability and a clean syntax, making it easy for developers to write and maintain code. This is often referred to as the "Pythonic" way.
2. **Interpreted Language:**
   * Python is an interpreted language, which means that the source code is executed line by line by the Python interpreter, without the need for compilation. This contributes to its ease of use and rapid development.
3. **Versatility:**
   * Python is a general-purpose language and is widely used in various domains, including web development, data science, artificial intelligence, machine learning, automation, scientific computing, and more.
4. **Extensive Standard Library:**
   * Python comes with a large standard library that includes modules and packages for a wide range of tasks, reducing the need for external dependencies.
5. **Dynamic Typing:**
   * Python uses dynamic typing, allowing developers to create variables without specifying their data types explicitly. This enhances flexibility but requires careful attention to variable types during development.
6. **Community and Ecosystem:**
   * Python has a vibrant and active community that contributes to its growth and development. There are numerous third-party libraries and frameworks available, expanding the functionality of Python for different applications.
7. **Cross-Platform:**
   * Python is designed to be cross-platform, meaning that Python code can run on various operating systems, including Windows, macOS, and Linux.
8. **Object-Oriented:**
   * Python supports object-oriented programming (OOP) principles, facilitating the creation and organization of code through the use of classes and objects.
9. **High-Level Language:**
   * Python is a high-level language, which means it abstracts low-level details, providing a simpler and more accessible interface for developers.

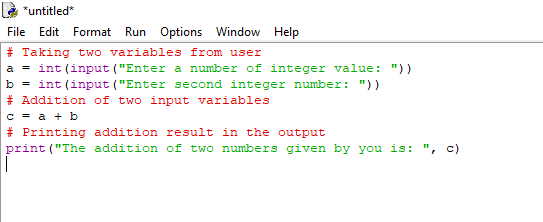
**Interactive Mode**

Interactive mode in Python refers to an environment where you can interact with the Python interpreter directly by entering one command or expression at a time. This is commonly used for quick experimentation, testing code snippets, and learning Python interactively. There are a few ways to enter interactive mode:

****

**Script Mode:**

Script mode in Python refers to running a Python script or program as a standalone file rather than interacting with the interpreter line by line. In script mode, you typically write a sequence of Python statements in a file with a **.py** extension, and then you execute the entire script.



**Data Type**

classifications that categorize and define the operations that can be performed on data. Python has a variety of built-in data types to represent different types of values. Here are some of the common data types in Python:

1. **Numeric Types:**
   * **int:** Integer type, e.g., x = 5
   * **float:** Floating-point type, e.g., y = 3.14
   * **complex:** Complex number type, e.g., z = 2 + 3j
2. **Sequence Types:**
   * **str:** String type, e.g., text = "Hello"
   * **list:** List type, e.g., numbers = [1, 2, 3]
   * **tuple:** Tuple type, e.g., coordinates = (4, 5)
3. **Set Types:**
   * **set:** Unordered collection of unique elements, e.g., my\_set = {1, 2, 3}
4. **Mapping Type:**
   * **dict:** Dictionary type, e.g., person = {"name": "John", "age": 30}
5. **Boolean Type:**
   * **bool:** Boolean type, represents True or False, e.g., is\_valid = True
6. **None Type:**
   * **NoneType:** Represents the absence of a value or a null value, e.g., result = None

**Type**

**Example:**

x = 5

print(type(x)) # Output: <class 'int'>

text = "Hello"

print(type(text)) # Output: <class 'str'>

**Variables in Python:**

A variable is a name that refers to a value stored in memory. Variables are used to store and manage data in your program. Unlike some other programming languages, you don't need to explicitly declare the type of a variable in Python; the interpreter dynamically determines the type based on the value assigned to it.

### **Variable Assignment:**

You assign a value to a variable using the assignment operator =.

**Example:** # Variable assignment

x = 10

name = "Alice"

pi\_value = 3.14

is\_valid = True

**Rules of variables:**

* Variable names in Python can contain letters (a-z, A-Z), numbers (0-9), and underscores (\_).
* Variable names cannot start with a number.
* Python is case-sensitive, so myVariable and myvariable are different variables.
* It's recommended to use descriptive variable names to enhance code readability.

### **Variable Reassignment:**

x = 10

print(x) # Output: 10

x = 20

print(x) # Output: 20

### Multiple Assignment:

You can assign values to multiple variables in a single line.

**Example:**

a, b, c = 1, 2, 3

print(a, b, c) # Output: 1 2 3

**Local Variable:**

Local variables in Python are those which are initialized inside a function and belong only to that particular function. It cannot be accessed anywhere outside the function.

**Example:**

**def** f():

    # local variable

    s **=** "I love Geeksforgeeks"

    print(s)

f()

**Global Variable:**

A global variable in Python is a variable that is defined outside of any function or block of code.

defmyfunc():globalxx = "fantastic"myfunc()print("Python is "+ x)

**NameSpace:**

A namespace in Python is a container that holds a collection of names (identifiers) and their corresponding objects (values). It provides a mapping between names and objects, allowing you to uniquely identify and access variables, functions, classes, and other entities within a program. Namespaces are a fundamental concept in Python for organizing and managing the scope of names.

There are several types of namespaces in Python:

1. **Local Namespace (Local Scope):**
   * Created inside a function or a method.
   * Contains local variables, parameters, and any other names defined within that function or method.
   * The local namespace is temporary and is created when the function is called, ceasing to exist when the function completes.
2. **Global Namespace (Global Scope):**
   * Includes names defined at the top level of a script or module.
   * Contains global variables, functions, and classes.
   * The global namespace is created when the script or module is executed and persists until the program terminates.
3. **Built-in Namespace:**
   * Contains names that are built into the Python language.
   * Includes built-in functions and objects such as print(), len(), str(), etc.
   * The built-in namespace is always available and does not need to be explicitly imported.

**Identifiers:**

**Identifier**is a user-defined name given to a variable, function, class, module, etc. The identifier is a combination of character digits and an underscore. They are case-sensitive i.e., ‘num’ and ‘Num’ and ‘NUM’ are three different identifiers in python.

## Rules for Naming Python Identifiers

* It cannot be a reserved python keyword.
* It should not contain white space.
* It can be a combination of A-Z, a-z, 0-9, or underscore.
* It should start with an alphabet character or an underscore ( \_ ).
* It should not contain any special character other than an underscore ( \_ ).