**BASICS OF PYTHON**

### **Variable:**

In Python, a variable is a container for storing a value. You can create a variable by assigning a value to it using the **“=”** operator. The value can be of any data type, such as a string, integer, or floating-point number. Once a variable is assigned a value, it can be used throughout your program.

For example:  
  
x = 5  
y = “Hello”  
z = 3.14  
  
In this example, x is assigned the value of 5, y is assigned the value “Hello”, and z is assigned the value 3.14.

It’s important to note that variable names in Python must start with a letter or an underscore, and can only contain letters, numbers and underscores. They are case-sensitive.

You can also change the value of a variable by reassigning it to a new value. For example:  
  
x = 5  
x = 10  
  
In this example, x is first assigned the value of **5** and then reassigned the value of **10**.

### **Data Types:**

Python has several built-in data types, including strings, integers, and floating-point numbers. In Python, data types refer to the type of value a variable holds. Python has several built-in data types:

1. **Numbers:**This includes integers (e.g. 1, 2, 3) and floating-point numbers (e.g. 3.14, 1.23)
2. **Strings:**A string is a sequence of characters (e.g. “hello”, “world”). Strings can be enclosed in single or double quotes
3. **Lists:**A list is a collection of items that are ordered and changeable. Lists are written with square brackets and items are separated by commas
4. **Tuples:**A tuple is similar to a list, but it’s immutable (i.e. its items cannot be changed). Tuples are written with round brackets and items are separated by commas
5. **Dictionaries:**A dictionary is a collection of key-value pairs. Dictionaries are written with curly braces and keys and values are separated by colons
6. **Boolean:**A Boolean data type is either True or False
7. **None:**None is a special constant used to represent the absence of a value or a null value.

x = 5  
print(type(x))  
# Output: <class ‘int’>;  
y = “Hello”  
print(type(y))  
# Output: <class ‘str’>;  
z = [1, 2, 3]  
print(type(z))  
# Output: <class ‘list’>;

In this example, x is an integer, y is a string, and z is a list.

### **Operators:**

Python supports various operators for performing mathematical and logical operations, such as +, -, \*, /, and %. In Python, operators are special symbols that perform specific operations on one or more operands (i.e. the variables or values being operated on). Here are some common operators in Python:

1. **Arithmetic operators:**

These operators perform basic mathematical operations, such as addition (+), subtraction (-), multiplication (\*), division (/), and modulus (%).

x = 5  
y = 10  
  
# Using the + operator to add x and y

result = x + y  
print(result) # Output: 15

1. **Comparison operators:**

These operators compare two values and return a Boolean value (True or False) based on the comparison. Examples include equal to (==), not equal to (!=), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=).

Using the < operator to check if x is less than y  
  
result = x < y  
print(result) # Output: True

1. **Logical operators:**

These operators are used to combine multiple conditions. Examples include and (and), or (or), and not (not).

1. **Assignment operators:**

These operators are used to assign a value to a variable. Examples include =, +=, -=, \*=, /=, and %=.

Using the \*= operator to multiply x by 2  
  
x \*= 2  
print(x) # Output: 10

1. **Membership operators:**

These operators are used to test whether a value is in a sequence (e.g. a list or string). Examples include in and not in.

1. **Identity operators:**

These operators are used to compare the identity of two objects. **is** and **is not** are the identity operators in python.

### **Conditional Statements:**

It is used to execute different codes depending on certain conditions. The most common type of conditional statement is the if-else statement. The basic syntax of an if-else statement is as follows:

if condition:  
# code to be executed if the condition is true  
else:  
# code to be executed if the condition is false

The ‘condition’ in the if statement is a Boolean expression that evaluates to either True or False. If the condition is True, the code in the if block will be executed, otherwise, the code in the else block will be executed.

### **Loops:**

Python has two types of loops: for loops and while loops. These allow you to repeatedly execute a block of code. In Python, loops are used to repeatedly execute a block of code. There are two types of loops: for loops and while loops.

**1.For loops:** A for loop is used to iterate over a sequence of items (e.g. a list, string, or tuple) and execute a block of code for each item. The basic syntax of a for loop is as follows : for variable in sequence:  
# code to be executed for each item in the sequence

**2.While loops:** A while loop is used to repeatedly execute a block of code as long as a certain condition is true. The basic syntax of a while loop is as follows : while condition:  
# code to be executed as long as the condition is true.

### **Functions**

In Python, a function is a block of code that can be reused throughout your program. Functions are useful for breaking down a complex program into smaller, more manageable pieces.

The basic syntax of a function is as follows:

def function\_name(parameters):  
# code to be executed

Here, function\_name is the name of the function, and **‘parameters’** are the input values that are passed to the function. The code inside the function is executed when the function is called.

### **Libraries**

A library is a collection of pre-written code that you can use to perform various tasks. Libraries provide a way to add functionality to your program without having to write the code yourself.

There are many libraries available for Python, some of the most popular ones include:

**NumPy:** A library for working with numerical data in Python. It provides tools for working with arrays, matrices, and mathematical functions.

**Pandas:** A library for working with data in Python. It provides tools for data manipulation, data analysis, and data visualization.

**Matplotlib:** A library for creating visualizations in Python. It provides tools for creating charts, plots, and other types of graphics.

**Scikit-learn:** A library for machine learning in Python. It provides tools for classification, regression, clustering, and more.

**TensorFlow:** A library for building and training machine learning models in Python. It’s widely used for deep learning, computer vision, natural language processing, and more.

**OpenCV:** A library for computer vision in Python. It provides tools for image and video processing, feature detection and extraction, and more.

### **Classes and Objects**

In Python, a class is a blueprint for creating objects, and an object is an instance of a class. Classes provide a way to model real-world concepts and encapsulate data and behaviour in a single unit. **Object-oriented programming (OOP)** is a programming paradigm that uses classes and objects to represent real-world concepts and organize code in a way that is easy to understand, maintain, and extend.

A class is defined using the **‘class’** keyword, followed by the name of the class, and a colon. The class body is indented and contains the data and behaviour of the class.