**What is Python?**

Python is an elegant & robust programming language that delivers both the power and application of traditional compiled languages with the ease of use of simpler scripting and interpreted languages.

Python was developed by Guido van Rossum in late 1989. It's High Level Language OOP (Object Oriented Programming) It is Scalable Extensible It's Portable It's Easy To Learn Robust & Effective Memory Manager Interpreted & (Byte) Complied

1. **Features:- It's High Level Language**

Python is HLL in other words It enables development of program in user friendly programming context.

Machine Code C, C++ & Java Python, Perl and Etc

Python has it's own high level built in data structures such as lists (resizable arrays), dictionaries ( hash tables) and lot more which we will study in this course.

1. **Features:- OOP**

OOP is a programming model that is totally based on classes and objects, It is used to structure Software program in simple reusable code which is usually a class and then we make a copy of this class which is known as object. Python is not only OOP Lang. It also has functional programming things borrowed from Haskell and Lisp. daws

1. **Features:- It is Scalable**

Python provides basic building blocks on which you can build an application, and as those needs expand and grow and further they can be easily maintained.

1. **Features:- It is Extensible**

You're working on some project, the amount of logic, files you write or the feature you made, you can still be able to use in just second whether it be in another file.

1. **Features:- It's Portable**

Python can be found on wide variety of systems and it is contributing to large parts in the CS domain.

1. **Features:- Robust & Effective**

When your Python crashes due to errors, the interpreter dumps out a "stack trace" full of useful information such as why your program crashed and where in the code (file name, line number, function call, etc.) the error took place.

1. **Features:- Memory Manager**

It is responsible for:- Allocation of Memory Retrieving and Storing the contents Effective Sharing Minimizing Memory Access Time

1. **Features:- Interpreted & (Byte) Complied**

Python is classified as Interpreted languages, Traditionally, purely interpreted languages are almost always slower than compiled languages because execution does not take place in a system's native binary language. BUT Python is byte compiled, it helps in boosting up the performance and having advantages of Interpreted languages.

**Why Learn Python?**

Python is currently the most widely used multi-purpose, high-level programming language, which allows programming in Object-Oriented and Procedural paradigms. Python programs are generally smaller than other programming languages like Java. Programmers have to type relatively less and the indentation requirement of the language, makes them readable all the time.

**What is a statement in Python?**

A statement is an instruction that a Python interpreter can execute. So, in simple words, we can say anything written in Python is a statement.

**Variable In Python**

You can think of a variable as a container or a storage location with a specific name.

In Python, you can create a variable by assigning a value to it using the equal sign (=). Here's a simple example: x = 10

In [4]: *# Examples*

X **=** 4

print(X)

4

In [6]: Student **=** ("Fahad")

print(Student)

Fahad

**Variable Type OR Data Type In Python**

Every variable in Python has a data type, which defines the type of data it can hold.

**Common data types in Python include:**

1. **Integers (int):**

Whole numbers without a decimal point. Example: 5, -10, 1000.

1. ***Floats (float):***

Numbers with a decimal point or in exponential form. Example: 3.14, 2.5e2 (which is equivalent to 250.0).

1. **Strings (str):**

Sequences of characters enclosed in single (') or double (") quotes. Example: 'Hello', "Python".

1. **Booleans (bool):**

Represents either True or False.

* *# Expamples of Integers:*
  + - a **=** 4
    - print(a)
    - 4
* *# Expamples of Floats*
  + - x **=** 0.4
    - print(x)
    - 0.4
* *# Expamples of Strings*
  + - b **=** ("Data Science and AI")
    - print(b)
    - Data Science and AI
* *# Expamples of Booleans*
  + - x **=** **True**
    - y **=** **False**
    - print(x,y)
    - True False

**Operators**

Operators are special symbols that perform specific operations on one or more operands (values) and then return a result. For example, you can calculate the sum of two numbers using an addition (+) operator.

**Types of operators**

* Arithmetic operator
* Relational operators
* Logical operators

**Arithmetic operator**

Arithmetic operators are the most commonly used. The Python programming language provides arithmetic operators that perform addition, subtraction, multiplication, and division. It works the same as basic mathematics.

1. +(Addition)
2. -(Subtraction)
3. \*(Multiplication)
4. / (Division)
5. // (Floor division)
6. ℅ (Modulus)
7. \*\* (Exponentiation)

**# Addition**

* + - x **=** 4
    - y **=** 5
    - print(x**+**y)
    - 9
* name **=** ("Fahad")
* surname **=** ("Siddiqui")
* print(name**+** " " **+**surname)
* Fahad Siddiqui

***# Subtraction***

* a **=** 10
* b **=** 30
* print(b**-**a)
* 20

***# Multiplication***

* a **=** 6
* b **=** 9
* c **=** 4
* print(a**\***b)
* print(a**\***b**\***c)
* 54
* 216
* Name **=** ("Fahad")
* print(name**\***3)
* FahadFahadFahad

***# Division***

* x **=** 2
* y **=** 8
* z **=** 10
* print(x**/**y)
* print(x**/**y**/**z)
* 0.25
* 0.025

***#Floor division***

* x **=** 2
* y **=** 4
* z **=** 2.2
* *# normal divion*
* print(y**/**x)

***# Floor division to get result as integer***

**print(y//x)**

* 2.0
* 2

***# Modulus***

*#the Modulus operator divides one value by a second and gives the remainder as a result.*

* x **=** 15
* y **=** 4
* print(x **%** y)
* 3

***# Exponent***

*# You can use it as a shortcut to calculate the exponential value.*

* num **=** 2
* print(num**\*\***2)
* 4

**Relational**

Relational operators are also called comparison operators. It performs a comparison between two values. It returns a boolean True or False depending upon the result of the comparison

In [38]:

* x **=** 10
* y **=** 5
* z **=** 2

***# > Greater than***

* print(x **>** y) *# True*
* print(x **>** y **>** z) *# True*

***# < Less than***

* print(x **<** y) *# False*
* print(y **<** x) *# True*

***# Equal to***

* print(x **==** y) *# False*
* print(x **==** 10) *# True*

***# != Not Equal to***

* print(x **!=** y) *# True*
* print(10 **!=** x) *# False*

***# >= Greater than equal to***

* print(x **>=** y) *# True*
* print(10 **>=** x) *# True*

***# <= Less than equal to***

* print(x **<=** y) *# False*
* print(10 **<=** x) *# True*
* True
* True
* False
* True
* False
* True
* True
* False
* True
* True
* False
* True

**Logical operators**

Logical operators are useful when checking a condition is true or not. Python has three logical operators. All logical operator returns a boolean value True or False depending on the condition in which it is used. and Denotetd by & or Denoted by |

In [41]:

a **=** **False**

b **=** **True**

Out[41]:

True

In [42]:

a **&** b *# False*

Out[42]:

False

In [43]:

b **&** a *# False*

Out[43]:

False

In [44]:

a **|** b *# True*

Out[44]:

True

In [45]:

b **|** a *# True*

Out[45]:

True

In [39]:

* print(**True** **and** **False**) *# False*

***# both are True***

* print(**True** **and** **True**) *# True*
* print(**False** **and** **False**) *# False*
* print(**False** **and** **True**) *# false*

***# actual use in code***

* a **=** 2
* b **=** 4

***# Logical and***

**if a > 0 and b > 0:**

*# both conditions are true*

print(a **\*** b)

**else:**

print("Do nothing")

* False
* True
* False
* False
* 8