**Python**

Python is a popular programming language.

Python can be used on a server to create web applications.

**What is Python?**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

Web development (server-side), software development, mathematics, system scripting.

**What can Python do?**

It can be used on a server to create web applications.

It can be used alongside software to create workflows.

It can connect to database systems. It can also read and modify files.

It can be used to handle big data and perform complex mathematics.

It can be used for rapid prototyping, or for production-ready software development.

**Why Python?**

It works on different platforms.

It has a simple syntax similar to the English language.

It has syntax that allows developers to write programs with fewer lines than some other programming languages.

Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

Python can be treated in a procedural way, an object-oriented way or a functional way.

In Python, variables are created when you assign a value to it.

Example:-

x = 5 , y = "Hello, World!"

Python has no command for declaring a variable.

**Comments:-**

One Line Comment using #

Multiline comment using ‘’’ ‘’’’ and """ """.

**Casting:-**

If you want to specify the data type of a variable, this can be done with casting.

Example

x = str(3) # x will be '3'

y = int(3) # y will be 3

z = float(3) # z will be 3.0

**Get the Type :-**

You can get the data type of a variable with the type() function.

Example

x = 5

print(type(x))

**Case-Sensitive:-**

Variable names are case-sensitive.

**Single or Double Quotes?**

String variables can be declared either by using single or double quotes.

**Variable Names :-**

**Camel Case :-**

Each word, except the first, starts with a capital letter.

myVariableName = "John"

**Pascal Cass :-**

Each word starts with a capital letter:

MyVariableName = "John"

**Snake Case :-**

Each word is separated by an underscore character:

my\_variable\_name = "John"

**Assign Multiple Values:-**

**Many Values to Multiple Variables :-**

Python allows you to assign values to multiple variables in one line:

Example

x, y, z = "Orange", "Banana", "Cherry"

print(x)

print(y)

print(z)

**One Value to Multiple Variables :-**

And you can assign the same value to multiple variables in one line:

Example

x = y = z = "Orange"

print(x)

print(y)

print(z)

**Unpack a Collection:-**

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called unpacking.

Example

fruits = ["apple", "banana", "cherry"]

x, y, z = fruits

print(x)

print(y)

print(z)

**Data Types :-**

**Numbers :-**

Python has three number types: integers, float and complex.

**Random Number Generation :-**

import random

print(random.randint(1, 10))

**Strings :-**

Strings are Arrays :-

Slicing Strings :-

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Modify Strings :-

Python has a set of built-in methods that you can use on strings.

upper()

lower()

strip()

replace("H", "J")

split(",")

join("myList") :- join() is used to concatenate multiple strings.

**String Concatenation :-**

To concatenate, or combine, two strings you can use the + operator.

To add a space between them, add a " ".

**F-Strings :-**

F-String is now the preferred way of formatting strings.

To specify a string as an f-string, simply put an f in front of the string literal, and add curly brackets {} as placeholders for variables and other operations.

**Placeholders and Modifiers :-**

A placeholder can contain variables, operations, functions, and modifiers to format the value.

A placeholder can include a modifier to format the value.

A modifier is included by adding a colon : followed by a legal formatting type, like .2f which means fixed point number with 2 decimals.

Example :-

price = 59

txt = f"The price is {price:.2f} dollars"

print(txt)

A placeholder can contain Python code, like math operations.

Perform a math operation in the placeholder, and return the result.

Example:-

txt = f"The price is {20 \* 59} dollars"

print(txt)

**Escape Characters:-**

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

**String Methods:-**

Python has a set of built-in methods that you can use on strings.

capitalize()

center()

count()

endswith()

find()

index()

upper()

lower()

replace()

split()

swapcase()

format() :-

**Booleans :-**

A boolean is a variable that can have one of two values: True or False.

Evaluate Values and Variables:-

The bool() function allows you to evaluate any value, and give you True or False in return,

**Operators :-**

Operators are used to perform operations on variables and values.

we use the + operator to add together two values {print(10 + 5)}

**Arithmetic Operators :-**

Arithmetic operators are used with numeric values to perform common mathematical operations. Types +,-,\* /,%,\*\*,//

**Assignment Operators :-**

Assignment operators are used to assign values to variables. Types =, += , -= , \*= , /= , %= , //=, \*\*=, &=, |=, ^=, >>=, <<=, :=

**Comparison Operators :-**

Comparison operators are used to compare two values. Types ==, !=, >, <, >=, >=

**Logical Operators :-**

Logical operators are used to combine conditional statements are and, or, not

**Identity Operators :-**

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location. Types is and is not.

**Membership Operators :-**

Membership operators are used to test if a sequence is presented in an object. Types in and not in.

**Bitwise Operators :-**

Bitwise operators are used to compare (binary) numbers. Types &,|,^,~,<<,>>

**Python Lists**

Lists are used to store multiple items in a single variable.

List items are ordered, changeable, and allow duplicate values.

**Append Items :-**

To add an item to the end of the list, use the append() method.

listName.append(Items We Want to Append)

**Insert Items :-**

To insert an item at a specific position in the list, use the insert() method.

listName.insert(index, "Item")

**Extend List :-**

To append elements from another list to the current list, use the extend() method.

listName1.extend(listName2)

**Remove Specified Item :-**

The remove() method removes the specified item.

listName.remove("Item We Want to Remove")

**Remove Specified Index :-**

The pop() method removes the specified index.

listName.pop(1)

**Clear the List :-**

The clear() method empties the list.

listName.clear()

**List Comprehension :-**

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

List comprehension is used to create a new list from an existing list or other iterable by applying an expression

**Expression :-**

The expression is the current item in the iteration, but it is also the outcome, which you can manipulate before it ends up like a list item in the new list.

Sort Lists Alphanumerically :-

listName.sort()

Sort Descending :-

listName.sort(reverse = True)

copy() method :-

listName.copy()

list() method :-

Another way to make a copy is to use the built-in method list().

list(thislist)

**Use the slice Operator :-**

You can also make a copy of a list by using the : (slice) operator.

mylist = thislist[:]

**Join Lists :-**

There are several ways to join, or concatenate, two or more lists in Python.

One of the easiest ways are by using the + operator.

list3 = list1 + list2

join two lists is by appending all the items from list2 into list1, one by one:

for x in list2:

list1.append(x)

use the extend() method, where the purpose is to add elements from one list to another list:

list1.extend(list2)

**Python Tuples**

Tuples are used to store multiple items in a single variable.

Tuples are ordered and immutable, meaning that once a tuple is created, you cannot change its elements.

Tuples are defined by enclosing elements in parentheses ().

A tuple is a collection which is ordered and unchangeable.

**Tuple Length**

To determine how many items a tuple has, use the len() function:

print(len(thistuple))

**tuple() Constructor :-**

The tuple() constructor returns a tuple object from a given iterable (like a list, string, or tuple).

thistuple = tuple(("apple", "banana", "cherry"))

**Update Tuples :-**