

**1. INTRODUCTION**

* Python: what is it?
* What makes Python so popular?
* Python's history
* All, MLL, and HLL
* The difference between a compiler and an interpreter

**What is python??**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. multiple programming paradigms,

supports structured (particularly, procedural),object-oriented, and functional programming.

It is used for:

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

**Why is python so famous and the most recognized language?**

First and foremost reason why Python is very popular is because it is highly productive as compared to other programming languages like C++ and Java. It is much more concise and expressive language and requires less time, effort, and lines of code to perform the same operations.

The Python features like one-liners and dynamic type system allow developers to write very fewer lines of code for tasks that require more lines of code in other languages. This makes Python a very easy-to-learn programming language even for beginners and newbies. For instance, Python programs are slower than Java, but they also take very less time to develop, as Python codes are 3 to 5 times shorter than Java codes. Python is also very famous for its simple programming syntax, code readability and English-like commands that make coding in Python a lot easier and efficient.

**History of Python**

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system.

Python's name is derived from the British comedy group Monty Python, whom Python creator Guido van Rossum enjoyed while developing the language. Monty Python references appear frequently in Python code and culture

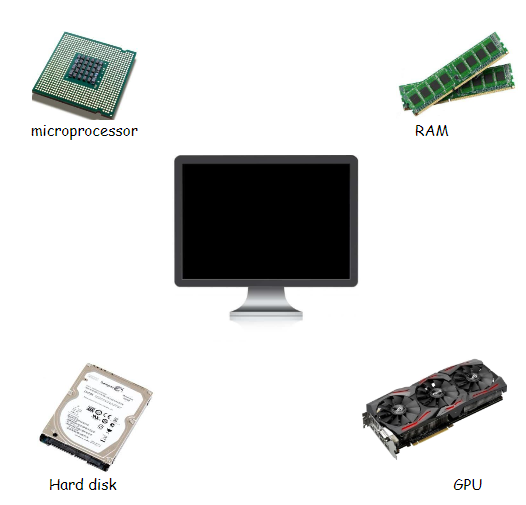
**for example** : the metasyntactic variables often used in Python literature are spam and eggs instead of the traditional foo and bar. The official Python documentation also contains various references to Monty Python routines.

**ALL, MLL, HLL**

Before you learn any programming language, it is important for one to understand some of the basics about computer and what are the languages that a computer can understand.

**Let's have a view on it.**

A computer is a collection of hardware components. Let us consider here few hardware components such as:



Out of these and many other hardware components, the most important or the heart of the computer is the

Microprocessor or CPU.

**Microprocessor or CPU**: A microprocessor is an electronic component that is used by a computer to do its work.

It is a central processing unit on a single integrated circuit chip containing millions of very small components

including transistors, resistors, and diodes that work together. They are created using a technology called as

Semiconductor technology.

**Semiconductor Technology??**

Any device which is made up of transistors is referred to as working in Semiconductor Technology.

A transistor is a device that regulates current or voltage flow and acts as a switch or gate for electronic signals. The

transistors have three terminals: emitter, base and collector.

There are two types of transistors:

1) NPN transistor.

2) PNP transistor.

Transistors can only store voltages. There are two levels of voltages:

Low Voltage referred to as OV

High Voltage referred to as— 5V

If we see the same in Software engineer's view, he/she looks the

two levels as:

Low Voltage referred to as —»O

High Voltage referred to as— 1

Therefore, in the perspective of a software engineer a Microprocessor or CPU can understand combinations of O and I.

**Programming Languages**

* Language is the main medium for communicating between the computer systems.
* A program is a collection of instructions that can be executed by a computer to perform a specific task.
* There were several programming languages used to communicate with the Computer.

**Case-I**

The world's first computer was invented in the year 1940s. During that time the task of a programmer was not simple.

For example, if they wanted a microprocessor to perform any operation then they had to use combinations of 0's and 1's.

During this time all the programs where written in the language called

110011010101100 as Machine Level Language. It is one of the low-level

programming languages.

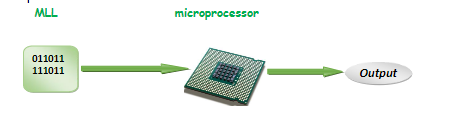
Codes written in 1940's as

To perform addition of two numbers: 0110110

To perform subtraction of two numbers: 1110111

To perform multiplication of two numbers: 1010101

To perform division of two numbers: 0100100



* The machine level code was taken as input and given to the microprocessor as the machine understood the binary value code and it gave the output.
* The main advantage of using Machine language is that there is no need for a translator to translate the code, as the Computers directly can understand.

The disadvantage was, it was difficult for a programmer to write the code or remember the code in this type of language.

**Case-2**

The problem with the Machine level code approach was decided to be changed in the 1950s.

They thought that instead of writing a long sequence of Os and 1's a single instruction can be given.

Codes written in 1950's as

To perform addition of two numbers: ADD

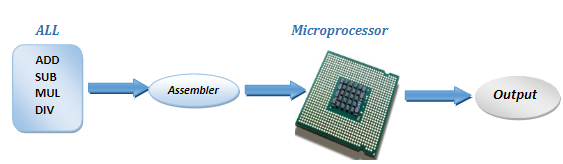
To perform subtraction of two numbers: SUB

To perform multiplication of two numbers: MUL

To perform division of two numbers: DIV

This approach of writing code is what is called Assembly Level Language. Instead of using numbers like in Machine

languages here we use words or names in English forms.



**An Assembler is software which takes Assembly Level Language (ALL) programs as input and converts it into Machine Level**

**Language (MLL) program.**

**Case-3:**

People always want things to be simple and easier, so in the 1960s they came up with the next type of language called High Level Programming Language.

High Level Languages are written in a form that is close to our human language, enabling the programmer to just focus on

the problem being solved.

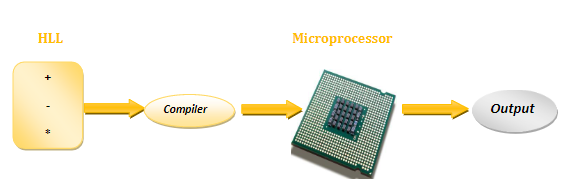
Codes written in 1960's as

To perform addition of two numbers: +

To perform subtraction of two numbers: -

To perform multiplication of two numbers: \*

To perform division of two numbers: /



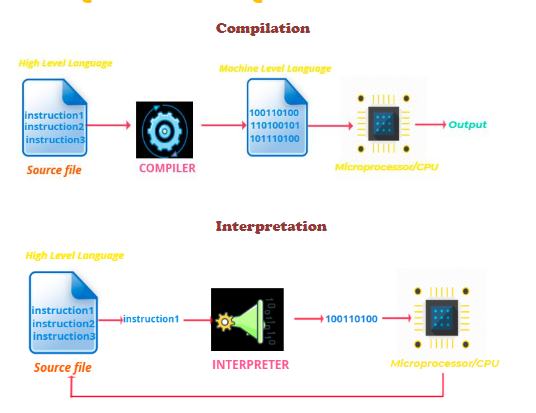
**A compiler is software which takes High Level Language (HLL) programs as input and converts it into Machine Level Language**

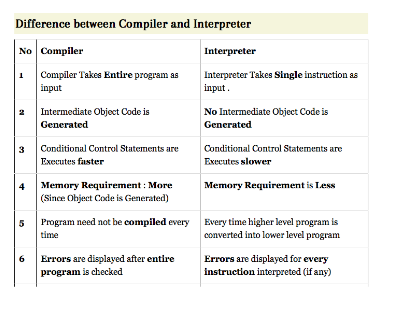
**(MLL) program.**

**Compiler vs Interpreter**

* Interpreters and compilers are very similar in structure. The main difference is that compilation is the process of converting a HLLprogram into MLL using software called a compiler.
* In compilation the entire HLL program is converted to MLL in one shot by the compiler and hence all the MLL instructions are readily available for CPU.
* Interpretation is the process of converting an HLL program into MLL using software called an interpreter.

In interpretation, at any given point of time only the Q single line of HLL program is converted to MLL which is then executed by CPU. Post execution the next line is converted and executed. This process repeats itself till all lines in the program are converted and executed. Because of this, the CPU does not have all lines of program readily available for execution. An interpreter will typically generate an efficient Machine Level representation and immediately evaluate it.





**Reasons to learn python**

1. Simple syntax

2. Usage in AI

*Father of the Python - Guido Van Rossum - 1989 - Netherland - He is one of the computer scientist working in most famous CWI - Amoeba OS(C language)*

Why Should We Learn Python ?

* EASY TO USE | Why is it easy to use ? - Python is compact and easy to use with simpe syntaxes.
* INTERPRETED LANGUAGE |
* CROSS PLATFORM LANGUAGE or PLATFORM INDEPENDENT LANGUAGE
* FREE & OPEN SOURCE

**disadvantages**

- Not the fastest language

- Lesser libraries than C, perl, java

- Not strong as Type binding

- Not easily convertible

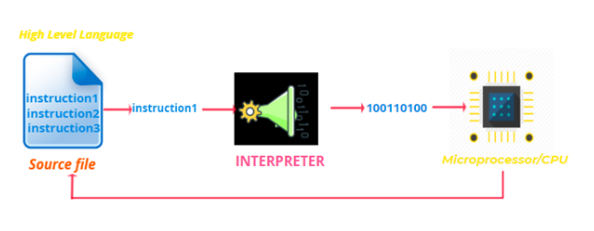
**Programming in Python**

* **Interactive mode & Script mode**
* **Difference b/w interactive & script mode**
* **Memory allocation**
* **Object Oriented Programming**
* **Principles of OOP**

**Programming in Python**

After knowing what Python is and different features of it and the history behind. Let us now get to know if python is an interpreted

programming language or if it is compiled programming language This is how a python file executes.



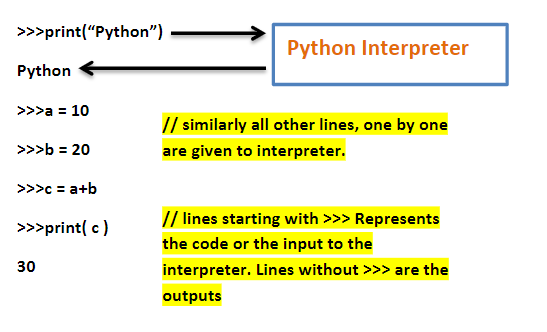
Each instruction is given to the interpreter which converts it into machine level and feeds to CPU to get the output. And again to get the next instruction we have to go back to the source file and continue the process. So no doubt python is an interpreted language.

**Interactive Mode & Script Mode**

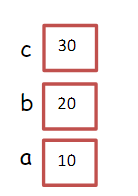
Let us now look into different modes of writing a code in python:

**Interactive mode**

* The interactive mode involves running your codes directly on the Python shell which can be accessed from the command line/ terminal of the operating system.
* Single line fed to the interpreter is executed at that moment of time, and waits for the next line to be entered. Each line is given to the interpreter which internally converts HLL to MLL and gives the respective output.



And whenever a variable with value is fed to the interpreter a container gets created internally with the value in it. As shown below:



And to exit this interactive mode you should give quit( ) as input.

**Pros and cons of Interactive Mode**

**pros :**

* It is great for single line or smaller codes.
* Interactive mode is a good way to play around and try variations on syntax.

**Cons:**

* Once you come out of interactive mode, you cannot revisit the previous code.
* It is harder to edit longer programs or even existing programs.
* Codes cannot be saved in interactive mode.

**Script Mode**

To overcome the drawbacks of interactive mode we have the second mode of writing a code called Script mode.

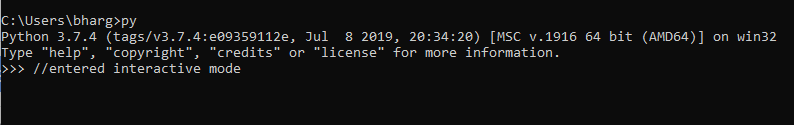
Let us know different platforms of typing a python code in script mode.

* Notepad //basic text editor
* Microsoft Word //basic text editor
* Different IDE (IDLE, Spider, PyCharm, Jupyter Notebook,VS-code)

Here let us get to know how to run a python script on the command line.

**Script mode in command line**

We know that if we type python and press enter we are into interactive mode like shown below:



Let us type the same addition code in python script and see how it works.

**print("Hello, Python" )**

**a = 10**

**b = 20**

**print(c)**

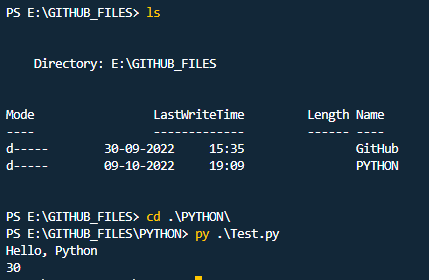
Let us see how to run a python script on the command line.

**Syntax:**



**Note**: Before running this command, make sure you are in the same directory as of the python script you want to run.

**Output:**



* In the script mode, you have to create a file, give it a name with a .py extension then run your code. The file which contains Python code that has to be executed is called a python script.
* The script mode is recommended when you need to create large applications.

As we have the python script in HLL, conversion must happen as below:



**Difference b/w interactive & script mode in python**

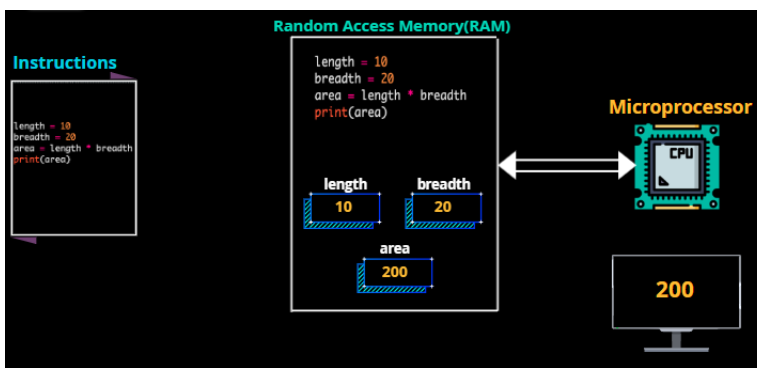
| **INTERACTIVE MODE** | **SCRIPT MODE** |
| --- | --- |
| 1. A way of using the python interpreter by typing commands & expressions at any prompt. | A way of using python interpreter to read & execute statements in the script |
| 1. Can’t Save and edit the code | Can save and edit the code |
| 1. If we want to experiment with the code, we can use interactive mode | If we clear about the code, use script mode |
| 1. We cannot save the statements for further use and we have to retype | We can save the statements for further use and no need to retype all the statements again |

**Memory allocation**

On one end we have the instructions and on the other end the microprocessor.

To give these instructions as the input to the microprocessor we need to store it so that the microprocessor has direct access to the instructions. To store these instructions we have something called RAM (Random Access Memory).

**RAM is a temporary memory directly connected to the microprocessor, and the main purpose of RAM is to store the instructions before it is given to the microprocessor. And then line by line instructions get converted to MLL and then given to the microprocessor.**

****

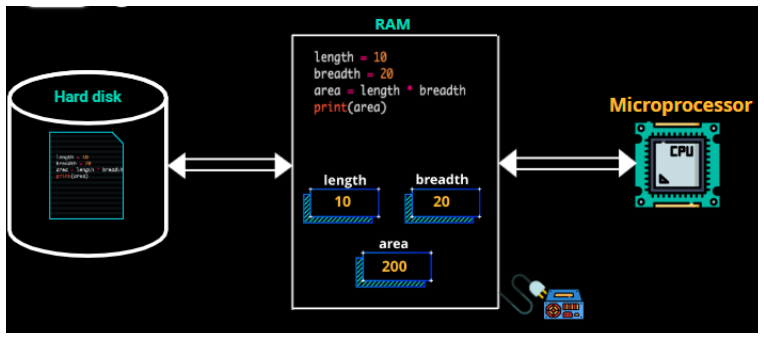
**RAM is the most important memory in the computer where all execution of the program happens on the RAM. In fact if the**

**instructions are on the RAM then only execution can happen if not execution cannot be performed.**

**RAM is a semiconductor technology device just like the microprocessor, which means it is made of transistors. So therefore**

**RAM always needs a constant supply of electricity for it to work. Once the supply is gone all the instructions and operations happening inside the RAM vanishes. That is the reason RAM is said to be volatile in nature.**

RAM is the main memory, as it is volatile in nature we have secondary memory which is called a Hard disk which is a magnetic device. To use the instructions again and again you need to make a copy of those instructions and store it in a hard disk. Those instructions are stored in a hard disk and are called a file



* The process of taking a copy from RAM and storing it in a hard disk is called saving.
* Loading is a task where we are taking a file and transferring it to RAM. All computers have these two memory devices namely hard disk and RAM. Where hard disks are used for permanent storage and RAM is used for temporary storage.

**Note**: Microprocessor is connected only to RAM. And RAM is connected to the hard disk.

.

**Object Oriented Programming**

We have two different types of programming languages, which are structural programming language and the second is object oriented

programming (OOP) language.

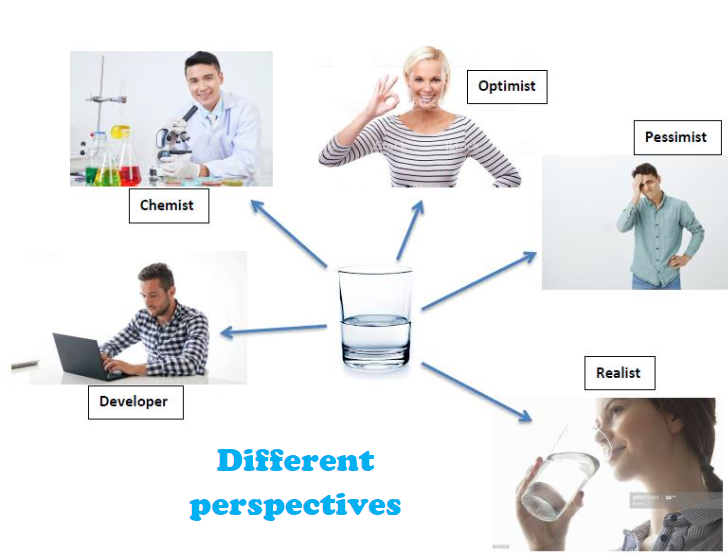
Here we will be focusing on OOP language.

Before going ahead with the object orientation, let us get to know

* What is orientation?

Orientation → Perspective → Way of looking at something

Let us take an example of glass of water and try to understand in better way:

****

A chemist will see the glass of water as liquid + gas, an optimistic person will look at it as half full glass, a pessimistic person will see it as half empty glass, a realist will see it as a glass of water, but a developer will always view it as an object.

Object Orientation is the way of looking at this world as a collection of objects. In this world no object is completely useless. All objects are in constant interaction with each other. No object exists in isolation.

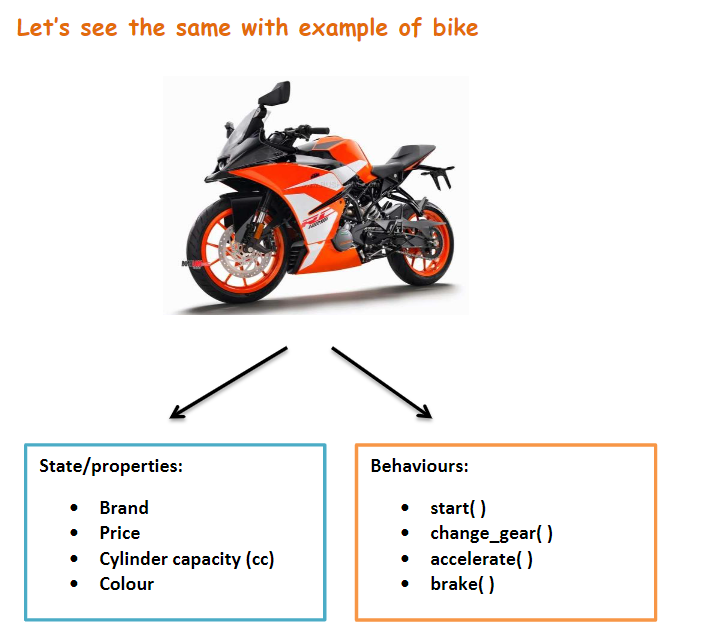
**Principles of OOP**

* View everything around you as objects.
* Every object belongs to a type. Where type does not exist but the objects of the type exist in reality.
* Every object has two parts:

1. State of an object /properties

2. Behavior of an object/actions

* To handle the properties of the states of an object we have a concept called as data types
* Similarly to handle behavior of an object we have a concept called as functions

****

**SETUP PROCESS OF PYTHON IN LOCAL SYSTEM**

* Download VS code in any website (https://code.visualstudio.com/)
* Then download these two extensions in VS code :(press ctrl+shift+x) --> it opens product store in left side panel with list of extensions installed in your system)

1. Python Extensions
2. Code Runner

========================================================================================================================================================================================================================

2. DATA TYPES INTRODUCTION & PYTHON MEMORY MANAGEMENT

=========================================================================================================================================================================================

\* OPERATIONS

2 + 2 = 4 ==> operands --> 2,2,4 operators --> +,=

- Arithmetic operators (+,-,\*,/,%,exp)

- Assignment operators (+=,-=,.......)

- Comparison operators (>,<,==,!=)

- Logical operators (and, or, not)

- Identity operators (is, is not)

- Membership operators (in , not in)

- Bitwise operators ()

=========================================================================================================================================================================================

\* VARIABLES

Variable is a container to store a values

- There is no command in the python to create a Variable (var can be created at the moment you assign a value to it)

- Variables do not need to be declared with any particular type and we can change type after they have been SETUP

- We can specify data type of a variable by using "Casting"

- We can get the data type of a variable by using "typeof()" operator

- Variable names always should declare in case-sensitive

Variable Names

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

Rules for Python variables

- A variable name must start with a letter or the underscore character

- A variable name cannot start with a number

- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )

- Variable names are case-sensitive (age, Age and AGE are three different variables)

ex : Legal variable names

myvar = "John"

my\_var = "John"

\_my\_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"

ex : iLLegal variable names

2myvar = "John"

my-var = "John"

my var = "John"

Multi Words Variable Names

- Variable names with more than one word can be difficult to read.

- There are several techniques you can use to make them more readable:

1. Camel Case

Each word, except the first word , remaining all words starts with a capital letter

myVariableName = "John"

2. Pascal Case

Each word starts with a capital letter:

MyVariableName = "John"

3. Snake Case

Each word is separated by an underscore character:

my\_variable\_name = "John"

Many Values to Multiple Variables

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

Ex : 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:

Ex : x = y = z = "Orange"

print(x)

print(y)

print(z)

=========================================================================================================================================================================================

\* DATA TYPES

Built-in Data Types

- In programming, data type is an important concept.

- Variables can store data of different types, and different types can do different things.

- Python has the following data types built-in by default, in these categories:

Text Type: str

Numeric Types: int, float, complex

Sequence Types: list, tuple, range

Mapping Type: dict

Set Types: set, frozenset

Boolean Type: bool

Binary Types: bytes, bytearray, memoryview

None Type: NoneType

-- NUMERIC TYPES

There are three numeric types in Python:

int

float

complex

Variables of numeric types are created when you assign a value to them:

x = 1 # int

y = 2.8 # float

z = 1j # complex

To verify the type of any object in Python, use the type() function:

Int

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

Complex

Complex numbers are written with a "j" as the imaginary part

Type Conversion

You can convert from one type to another with the int(), float(), and complex() methods:

x = 1 # int

y = 2.8 # float

z = 1j # complex

#convert from int to float:

a = float(x)

#convert from float to int:

b = int(y)

#convert from int to complex:

c = complex(x)

print(a)

print(b)

print(c)

print(type(a))

print(type(b))

print(type(c))

Random Number

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers

\* PYTHON CASTING

- There may be times when you want to specify a type on to a variable. This can be done with casting.

- Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

- Casting in python is therefore done using constructor functions:

int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)

float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)

str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

-- BOOLEAN TYPES

Booleans represent one of two values: True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

print(10 > 9)

print(10 == 9)

print(10 < 9)

When you run a condition in an if statement, Python returns True or False:

a = 200

b = 33

if b > a:

print("b is greater than a")

else:

print("b is not greater than a")

\_\_\_Evaluate Values and Variables\_\_

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

\_\_Most Values are True\_\_

Almost any value is evaluated to True if it has some sort of content.

Any string is True, except empty strings.

Any number is True, except 0.

Any list, tuple, set, and dictionary are True, except empty ones.

\_\_Some Values are False\_\_

In fact, there are not many values that evaluate to False, except empty values, such as (), [], {}, "", the number 0, and the value None. And of course the value False evaluates to False.

One more value, or object in this case, evaluates to False, and that is if you have an object that is made from a class with a \_\_len\_\_ function that returns 0 or False:

\_\_Functions can Return a Boolean\_\_

You can create functions that returns a Boolean Value:

def myFunction() :

return True

print(myFunction())

-- SEQUENCE TYPES

--> Python Lists

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

\* Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

\* Lists are created using square brackets[]

myList = ["apple","Banana","Cherry]

print(myList)

\* List Items

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

List items are indexed, the first item has index [0], the second item has index [1] etc.

Ordered : When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

changeable : The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

Allow duplicates : Since lists are indexed, lists can have items with the same value

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

print(thislist)