Python Notes

- **About Python:**
  - Developed by Guido Van Rossum in the year 1991 at the Center for Mathematics and Computer Science managed by the Dutch government.
  - Python is a programming language that combines the features of both Java and C.
  - Python is a general-purpose programming language
  - Interpreted language
  - Python is a case sensitive language i.e., **a** is not equal to **A**

- **Features:**
  - **Simple:**
    - Python can perform complex task using only a few lines of code, while Java or C might take multiple lines
  - **Easy to learn:**
    - Python syntax is straightforward and shares similarity with the English language i.e., no use of {} or ; in programs
  - **Open Source:**
    - Python is an open-source software, meaning anybody can freely download it and use it to develop programs.
  - **High level language:**
    - Python uses English words in its programs and hence it's called a high-level programming language.
  - **Platform Independent:**
    - When a python program is compiled using a python compiler, it generates a byte code. Using a Python Virtual

Machine (PVM), anybody can run these byte code instructions on any computer system.

- **Portable:**
  - Python program will give the same result regardless of the platform they are being programmed on since they are platform independent
- **Procedure and Object Oriented:**
  - In python programs can be built either using functions and procedures or using classes and objects
- **Interpreted language:**
- **Integrated:**
- **Embeddable:**
  - We can insert python programs into a C or C++ program.

- **Installing Python:**
  - Go to: https://www.python.org/downloads/
  - Download the Software
  - Run the installer and follow the instructions

- **In Python there are two ways we can run our code:**
  - **Interactive mode:** Interactive mode is a command line shell which gives immediate output. It provides a quick way of running blocks or single line of code.
  - **Script Mode**: In script mode we can generally write multiple lines of command. It is used for writing programs

- **Variables in Python:**
  - Variable is a named location in the memory where data/values are stored. For Example:
    - A=10
    - Here A is the variable where the value 10 is stored
  - Variable name cannot start with a number
  - Variables are case sensitive name=2 & Name =4 would create two different variables

- **Identifiers in Python:**
  - An identifier is a name that is given to a variable or function or class etc.
  - Special symbols like: @, $, # cannot be used in our identifier
    - Name=input("Enter Your Name:")
    - Here **Name** is the identifier and the variable

- **Keywords in Python:**
  - Keywords are the reserved names which cannot be used as variable names, identifiers, or function names
  - Example: and, as, assert, break

- **Escape Sequence:**
  - An escape sequence is a set of special characters used in the form of backslash (\) followed by a character.
  - These characters are used to represent special meaning such as:
    - \n: Means New Line
    - \t: This Represents a Tab

- **Basic Data Types in Python:**
  - **Str:**
    - Represents string datatype (A group of characters) Example: a= "LBSIM"
    - Denoted by ' ' or " " quotes
  - **Int:**
    - The int datatype represents integer number, an integer number is a number without any decimal or fraction point. Example:
    - a =12,100
  - **Float:**
    - The float datatype represents decimal number. Numbers that contain decimal points. Example:
    - a = 12.11, 4.3

- **Bool:**
  - The bool datatype represents Boolean values. Only two values TRUE, FALSE can be represented by this datatype
- **List:**
  - Contains group of elements that can be of different types
  - Main difference between tuples and list is that lists are mutable
  - Values are separated by comma and enclosed in []
  - Example: a= ['hello',13,232,89,'True']
- **Tuple:**
  - Contains group of elements that can be of different types
  - Tuples are immutable by nature i.e., their values cannot be changed or updated
  - Values are separated by comma and enclosed in ()
  - Example: a= (1,2,7,5)
- **Dictionaries:**
  - Dictionaries are used to store data in key: value pairs
  - Dictionaries items are changeable meaning we can add or remove a key: value pair
  - Defined with {}
  - Cannot have two items with the same key.
  - Example: dict_a= {"Key1": values,"Key2": Value}

- **Type Conversion in Python: val=12345, txt="7343"**
  - Str(val) ->"12345": converts to string datatype
  - Int(txt) -> 7343: converts to int datatype
  - Float(txt) -> 7343.0: converts to float datatype
  - Bool(val) -> True: converts to bool datatype
  - List("hi") -> ['h','i']: convert to list

- **<u>Operators In Python:</u>**
  - **Assignment Operators:**
    - Addition -> +: a+b

- Subtraction -> -: a-b
- Multiplication -> *: a*b
- Division -> /: a/b
- Floor Division -> //: a//b
    - (Answer in integer: 10//3 =3, instead of 3.33)
- Remainder -> %: a%b
- Exponential (Power) -> **: a**b

- **Boolean Or Logical Operators:**
    - Works with two operands
    - A + B: Here A and B are the two operands
        - **AND: (a and b)**
            - when both conditions are satisfied returns True else False.
        - **OR: (a or b)**
            - when either one condition is satisfied returns True else False.
        - **NOT: (not a)**
            - True if operand is false

- **Relational Operators:**
    - Used to compare values
    - Returns a Boolean value
        - Equal To: ==
        - Not Equal To: !=
        - Less Than: <
        - Greater Than: >
        - Less Than Equal To: <=
        - Greater Than Equal To: >=
- **Bitwise Operator:**
    - Those operators that work on individual bit
        - & -> AND
        - | -> OR

- ^ -> Bitwise Exclusive OR
- ~ -> Inversion (0 to 1, 1 to 0)
- << -> Shifts the bit to left
- >> -> Shifts the bit to right

- **Membership Operators:**
  - In
  - Not In

```
a=2
items=[2,3,4]
if(a in items): # Using In
    print("Number Present")
else:
    print("Not present")
```

```
Number Present
```

- **Identity Operator:**
  - Is: checks whether the variables have same memory location
  - Is not

```
a=20
b=20
if(a is b):
    print("a and b have same identity")
else:
    print("a and b do not have same identity")
```

```
a and b have same identity
```

- **<u>Python Input:</u>**
  - To accept input from the keyboard, python provides the input() function.
  - Input functions takes the user input and return it as a string value

- Value = input("What is your Name?")
  - What is your Name? -> Saksham
  - Print(value)
  - 'Saksham'

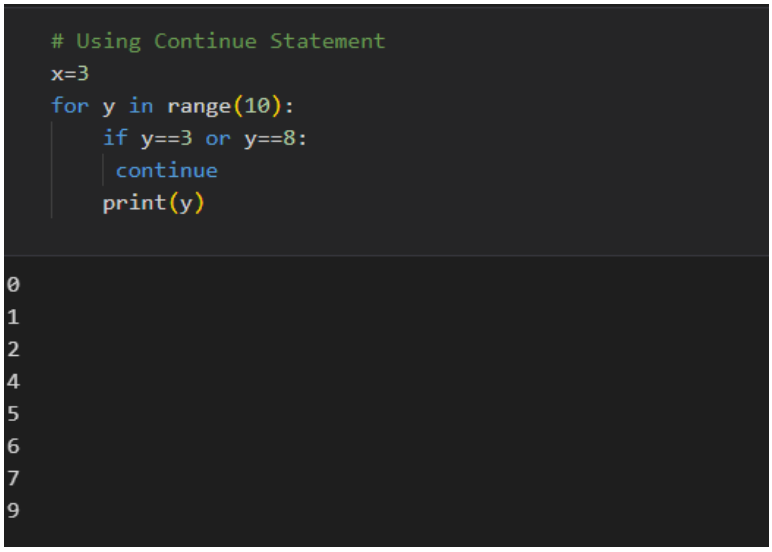
- **Python Output:**
  - To display outputs, python provides the print() function
  - Value = input("What is your Name?")
    - What is your Name? -> Saksham
    - Print(value)
    - 'Saksham'

- **Files in Python:**
  - **Opening Files in Python:**
    - To open files in python, there is a built in function open(). This function accepts filename and openmode in which to open the file
    - File=open('filename.txt','r')
      - Here r denotes that we wish to read data from the file
  - **Closing Files in Python:**
    - A file once opened should be closed using the close() method
      - File.close().
    - If the file is not closed the memory utilized by the file is not freed by, which may lead to problems in the future
  - **Writing in Files using Python:**
    - In order to write in a file using python, we need to first open it using either w: write mode or a: append mode
      - **w:** if any data exists in the file, it would be deleted
      - **a:** it will add to the end of the data. If the file does not exist, it will create a new file,
    - **Example:**
      - file=open("filename.txt","a")
      - **file.**write("hello")
      - **file.**close()

- **Reading Files in Python:**
  - To read files in python, we must open it in reading mode
  - File=open("filename.tx","r")
  - File.read()

- **Control Statements:**
  - **If:**
    - Used to execute statement depending whether the given condition is true or not.
    - If 5>2:
      - Print("True")
  - **if-else:**
    - executes statements when a condition is true otherwise it will execute other statements
    - if 5>2:
      - print("True")
    - else:
      - print("False")
  - **if-elif-else:**
    - extension of if-else, it is used when we have to test multiple conditions and execute statements depending on those conditions.
  - **While loop:**
    - Used to execute a statement several times depending on whether the condition is true or not
    - While a<b:
      - print(a)
      - a+=1
  - **For loop:**
    - Used to iterate over the elements of a sequence
    - For i in range(0,10,1):
      - Print(i)
  - **Break:**

- Used to come out of a loop prematurely when a condition is met
- for x in range(0,10):
    - if x==9:
    - break
- print(x)

- **Continue:**
  - Anything beyond the continue statement will not be executed for the given condition.
  - Allows us to go the beginning of the next iteration of the block
  - 
```
# Using Continue Statement
x=3
for y in range(10):
    if y==3 or y==8:
        continue
    print(y)
```
```
0
1
2
4
5
6
7
9
```

- **Pass:**
  - Does nothing. Used to fil syntax criteria in some cases
  - For i in range(10):
    - pass
- **Assert:**
  - Checks for condition -> assert>0,"Error wrong value entered"