PYTHON PROGRAMMING LANGUAGE

What is python?

1. Python is powerful programming language.

2. Specially used in data science,machine learning to solve daily life real-time problems and providing the solutions.

3. Python is a high-level,intrepeted,interactive and object oriented scripting langauge.

History of Python language:

1. Python was developed by "Guido Van Rossum" in late 1980's in National Research Institute for Mathematics and Computer Science in Netherlands.

2. Python was derived from many other languages including ABC,Modulo-3,C,C++,Algol=68,SmallTalk,Unix shell and other scripting languages.

Why python?

1.Easy to understand

2.Easy syntax

3.Beginner friendly

4.Supports multiple libraries

5.Supports OOP's

Applications of python

1.Web applications

2.Desktop applications

3.Database applications

4.Web scraping

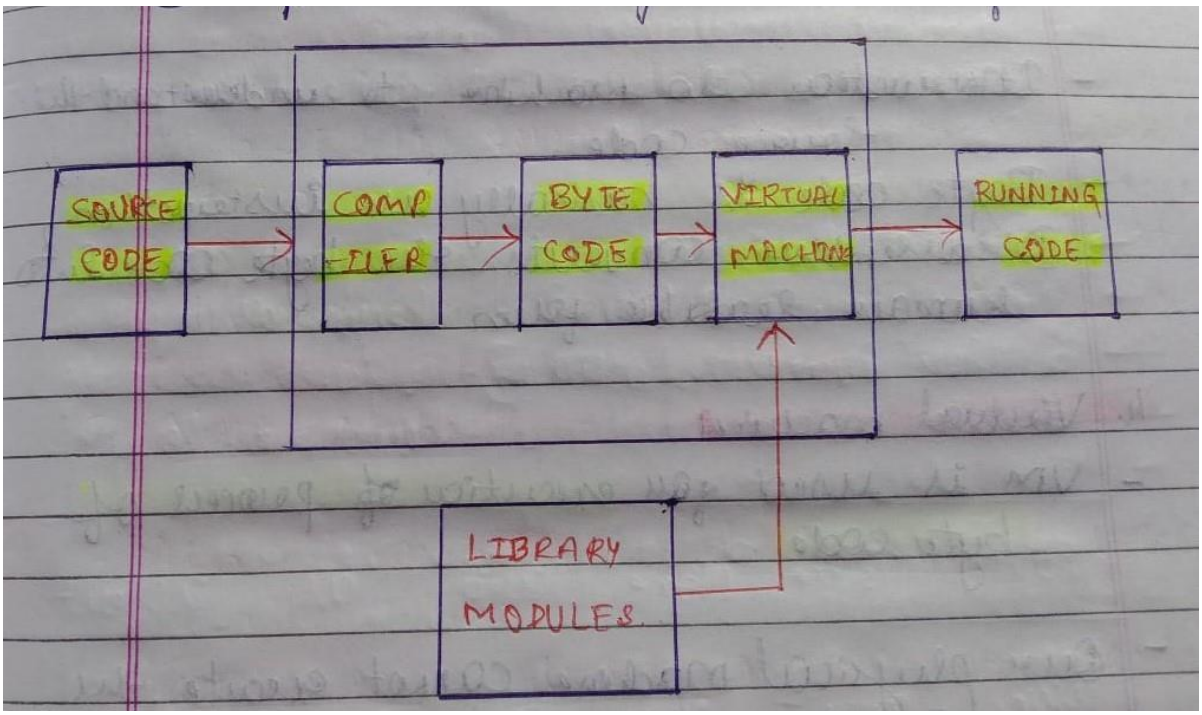
5.Wen mapping

6.Data analysis

7. Interactive web visualization

8. Computer vision and image processing etc.,

INTERNAL WORKING OF PYTHON



1.Python identifiers:

1.A Python identifier is a name used to identify variable,function,class ,module or other objects.

2.A identifier starts with a letter (a-z),(A-Z) or an underscore(_) followed by zero or more letters,and digits(0-9)

Naming conventions for identifiers:

1. class names start with an uppercase letter .All the other identifiers start with a lower case letter.

2. Starting an identifier with a single leading underscore indicates that the identifier is "private".

3.starting an identifier with two leading underscores indicates "strongly private".

4.If the identifier also ends with two trailing underscore ,the identifier is a language defined special name.

Python keywords:

1. Keywords are reserved words.

2. cannot use them as constant or varaible.

3. all the keywords except "True","False" and "None" are in lowercase.

4.they are 35 keywords present in python

In [42]:
```
#code to print all the keywords present in python
import keyword
print(keyword.kwlist)
a=len(keyword.kwlist)
print(a)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', '
class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'o
r', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
35
```

Indentation in python:

1. In C,C++,the indentation is specified by using the flower brackdets{}.

2.In python,indentation is done by 4 white spaces or tab.

Python comments:

In python we have 2 types of comments such as follow:

1.single line comment(specified using single or double quotes).

2. multi line comment(specified using doc sring).

In [43]:
```
#single line comment--->shows the single line comment.
print('hello my dear friends')
```

```
hello my dear friends
```

In [44]:
```
"""
multi line comment
how are you all
have a nice day guys
"""
print('hello CSE people')
```

```
hello CSE people
```

PYTHON VARIABLES:

1. Variables are nothing but reserved memory locations to store values.

2. When we create a varaible ,automatically some space in memory is reserved.

3.Based on the data types the interpreter allocates the memory.

4.Varaible can take any data type such as integer,float numbers,strings etc.

In [45]:
```
#example1:
#here 3 variables have been created and stored the values as shown below
a=50
b=50
c=a+b
print(c)

#example 2 on strings

x='hello'
x
```

```
100
```

Out[45]:
```
'hello'
```

Python Data types:

1.Integer data type

2.Float point numbers

3.Strings data type

4.Boolean data type

Python Data Structures

1. In python we have four types of data structures

2. These data structures are very useful and important while working with advance topics such as machine learning ,data science.

1.List

2.Tuple

3.Dictionary

4.Sets

1.LISTS IN PYTHON:

*List is a python data structure

*it is a combination of any data type.

*enclosed within [] and separated by commas(,).

*list is sequence of values called items or elements.

*list is similar arrays.

*list mutable.

*list is indexible.

In [46]:
```
#Creating a list with constructor()
a=([1,2,3,4,5])
b=([2.4,7.0,6.8])
c=(["hello","students"])
print(a)
print(b)
print(c)
```

```
[1, 2, 3, 4, 5]
[2.4, 7.0, 6.8]
["hello", "students"]
```

In [47]:
```
type(a)
```

Out[47]:
```
list
```

In [48]:
```
#creating list with heterogenous data:

a=(["hello",8,9,20.77])
a
```

Out[48]:
```
["hello", 8, 9, 20.77]
```

In [49]:
```
#create list using range function:
L=list(range(0,6))
print(L)
```

```
[0, 1, 2, 3, 4, 5]
```

In [50]:
```
#creating a list without constructor():

days=["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
days
```

Out[50]:
```
["monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday
"]
```

In [51]:
```
#accesing list items using index values:

days[1]
```

Out[51]:
```
"tuesday"
```

In [52]:
```
#negative indexing:

#creating a list without constructor():

days=["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
days[-1]
```

Out[52]:
```
"sunday"
```

In [53]:
```
days[:]
```

Out[53]:
```
["monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday
"]
```

In [54]:
```
#list slicing:
days[1:4]
```

Out[54]:
```
["tuesday", "wednesday", "thursday"]
```

In [55]:
```
#list slicing with step size:

days=["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
days[1:4:2]
```

Out[55]:
```
["tuesday", "thursday"]
```

List built-in function:

Function	Meaning
len()	Returns the number of elements in a list
max()	Returns element with greater value
min()	Returns the element with lowest value
sum()	Returns the sum of all elements in the list
random.shuffle()	Shuffle the elements in a list randomly -Here we make use of a random package.

In [56]:
```
#example:
days=['monday','tuesday','wednesday','thursday','friday','saturday','sunday']
len(days)
```

Out[56]: 7

In [57]:
```
numbers=[1,2,3,4,5,6,7,8]
max(numbers)
```

Out[57]: 8

In [58]:
```
numbers=[1,2,3,4,5,6,7,8]
min(numbers)
```

Out[58]: 1

In [59]:
```
numbers=[1,2,3,4,5,6,7,8]
sum(numbers)
```

Out[59]: 36

In [60]:
```
import random
numbers=[1,2,3,4,5,6,7,8]
random.shuffle(numbers)
print(numbers)
```

```
[1, 8, 6, 7, 3, 5, 4, 2]
```

List Operators

Operator	Meaning
"+"	Concatenates two lists with each other.
"*"	Replicates the elements of lists
"in"	-Used to determine whether the element is present in the list or not. -Returns true if the element is present in the list -Returns false if the element is not present.
"is"	Used to check whether two variables refer to the same object or not.
"del"	-Used to remove elements from the list.

In [61]:
```
#examples on operator:

a=[1,2,3]
b=[4,5,6]
c=a+b
print(c)
```

```
[1, 2, 3, 4, 5, 6]
```

In [63]:
```
a=[10,20]
b=3*a
print(b)
```

```
[10, 20, 10, 20, 10, 20]
```

In [64]:
```
days=["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
"wednesday" in days
```

Out[64]: True

In [65]:
```
days=["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
"wednesday" not in days
```

Out[65]: False

In [66]:
```
a="Arshiya"
b="Arshiya"
a is b
```

Out[66]: True

In [67]:
```
a="Arshiya"
b="Lubna"
a is b
```

Out[67]: False

In [69]:
```
days=["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
del[days[1]]
days
```

Out[69]: ['monday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']

List Methods:

Methods	Meaning
append()	Adds element at the end of the list
clear()	Clear the contents of the list
count()	Gives count of repeated items
copy()	Copy list from one list to another list
extend ()	Joins the elements of 2 lists

In [70]:
```
a=["a","b","c","d"]
print(a)
a.append("e")
print(a)
```

```
['a', 'b', 'c', 'd']
['a', 'b', 'c', 'd', 'e']
```

In [71]:
```
list=["red","orange","pink","black"]
print(list)
list.clear()
print(list)
```

```
["red", "orange", "pink", "black"]
[]
```

In [72]:
```
nums=[1,2,3,4,1,4,1,7,8,1]
print(nums)
nums.count(1)
```

```
[1, 2, 3, 4, 1, 4, 1, 7, 8, 1]
```

Out[72]:
```
4
```

In [73]:
```
list_1=[1,2,3,4]
print(list_1)
list_2=list_1.copy()
print(list_2)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4]
```

In [74]:
```
A=[1,3,2,4,5]
B=[5,6,7,3,6]
print(A)
print(B)
A.extend(B)
print(A)
```

```
[1, 3, 2, 4, 5]
[5, 6, 7, 3, 6]
[1, 3, 2, 4, 5, 5, 6, 7, 3, 6]
```

NOTE:we have many other methods such as index(),insert(),pop(),remove(),reverse(),sort() and so on.

2.PYTHON TUPLES

1.Tuple is another important data structures present in python.

2.Allow multiple data types.

3.Enclosed () and separated by commas(,).

4.Tuples are not mutable

5.Tuples are indexible

6.Tuples are ordered

In [79]:
```
#creating tuple with single element:

a=(5,)
print(a)
print(type(a))
```

```
(5,)
<class 'tuple'>
```

In [80]:
```
#creating tuple with integers
a=(1,2,3,4,5)
print(a)
```

```
(1, 2, 3, 4, 5)
```

In [81]:
```
#creating tuple with float numbers
b=(6.5,5.9,2.0)
print(b)
```

```
(6.5, 5.9, 2.0)
```

In [82]:
```
#creating tuple with strings
c=("one","two","three")
c
```

Out[82]:
```
('one', 'two', 'three')
```

In [84]:
```
#creating tuple multiple data types
mix=("hello",708,66.8,1,2)
mix
```

Out[84]:
```
('hello', 708, 66.8, 1, 2)
```

In [86]:
```
#accessing items of tuple

a=(1,2,3,4,"hello")
print(a[4])
```

```
hello
```

Tuple Methods

Method	Meaning
count()	Returns the number of times a specified value occurs in a tuple.
index()	Searches the tuple for a specific value and

index()	Searches the tuple for a specific value and returns the position of where it was found

3.Python Dictionary

1.Dictionaries are used to store the values in key-value pair.

2.It's a combination of keys and values

3. Elements are accessed based on the keys

4. It is ordered,mutable,and dont allow duplicate elements.

5.Enclosed within {} with key value pair

In [87]:
```
#create a dictionary
a={
    "name":"joy",
    "age":23,
    "education":"Engineer"
}
print(a)
```

```
{"name": "joy", "age": 23, "education": "Engineer"}
```

In [88]:
```
len(a)
```

Out[88]: 3

In [89]:
```
type(a)
```

Out[89]: dict

In [91]:
```
x=a.keys()
x
```

Out[91]: dict_keys(["name", "age", "education"])

In [92]:
```
y=a.values()
y
```

Out[92]: dict_values(["joy", 23, "Engineer"])

In [93]:
```
z=a.items()
z
```

Out[93]:
```
dict_items([('name', 'joy'), ('age', 23), ('education', 'Engineer')])
```

Dictionary methods:

Methods	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of dictionary
get()	Returns the value of specified key
items()	Returns the list containing tuple pair of keys and values
keys(),values()	key()-Returns the keys of the dictionary values()-Returns the values of the dictionary

4.Python Sets:

1.Sets are used to store multiple items in a single variable.

2.Sets are enclosed withing {}.

3. Sets are unordered,not mutable and unidexible.

4. Sets items are unchangable but we can remove and add new items.

In [94]:
```
#Example:
set1={1,2,3,4}
print(set1)
```

```
{1, 2, 3, 4}
```

In [95]:
```
3duplicates not allowed
set={1,2,3,4,5,6,7,1,1}
set
```

Out[95]:
```
{1, 2, 3, 4, 5, 6, 7}
```

In [96]:
```
len(set)
```

Out[96]: 7

In [97]:
```
type(set)
```

Out[97]: set

Set Methods:

Methods	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
Difference()	Returns a set containing difference between 2 or more sets
Discard()	Removes the specific item
Copy()	Returns a copy of the set

Note:We have still more methods in sets in the above table few are discussed.Please for more info refer this link(https://www.w3schools.com/python/python_sets.asp (https://www.w3schools.com/python/python_sets.asp))

Python Operators

In [ ]:
```
We have various python operators whcih are useful in performing various operat

Operators:
1.Arthematic operators
2.Comparison operators
3.Logical operators
4.Assignment operators
5.Special operators
a.Identity operators
b.Membership operators
```

1. Arthematic operators

Arthematic Operators	Description
+	Used for addition
-	Used for subtraction
*	Used for multiplication
/	Used for division
//	Used for floor division
%	Used for modules
**	Used for exponent or power

2. Comparison operators

a -first operand

b-second operand

Comparison operator	Description
>	Checks whether a is greater than b or not
<	Checks whether a is less b or not
>=	Check whether the operators are greater than equal to each other
<=	Check whether operands are less than equals to each others
==	Checks whether 2 operands are equal
!=	Check whether 2 operands are not equal

3. Logical operators

Logical operators	Description
and	True if both the operands are true
or	True if either of the operands is true

not	Returns the compliment of the operand

4. Assignemnt operators

Assignment Operators	Description
=	Assign value of right side of expression to left side operand.
+=	Add and assign: add right side operand with left side operand then assign to left operand
-=	Subtract and assign: subtract right operand with left operand and then assign to left operand
&	Perform AND operation on operands and assign value to left.
**	Perform exponent operation
%=	Perform modules AND operation
/=	Perform divide AND operation

5. SPECIAL OPERATORS

1. Identity operators:

*Used to check if the two values are located on the same part of the memory.

*Returns true or false depending on conditions.

*two identity operators are 'is' and 'is not'.

2. Membership operators:

*used to check whether a value is found in a sequence.

*two membership operators are 'in' and 'not in'.

CONTROL FLOW IN PYTHON:

1. if statement

2. if-else statement

3. nested if statement

LOOPS:

1.for loop

2.while loop

Python functions

1.Python functions is a block of statements that peforms a specific task.

2.Get executed when the function is called.

3. goal is to put dome repeatedly done task together and make aa function ,so instead of writing the same code againa nd again for different inputs ,we can perform the function calls to reuse the code contained in the function.

4. functions are defined by using "def" keyword.

5.We have 2 types of functions:

a.User-defined function

b.Built-in function

Syntax:

```
def function_name(parameters):
    ####statements
function_name(arguments)
```

In [ ]:
```
def add():
    a=2
    b=3
    sum=a+b
    return sum
add()
print(add())
```

1.Built-in function:

*python predefined functions that are readily available.

*such as min(),max(),print(),sum() etc

2.User-defined functions:

3.Anonymous function:

*function that we define without a name.

*anonymous functions are also called as lambda function.

*they are not declared without def keyword.

In [ ]:
```
function modularity:

def collectdata():
    "function to collect data"
    statement(s)

def cleandata():
    "function to clean data"
    statement(s)
def processdata():
    "function to pre-process the data"
    statement(s)
def exploredate():
    "function to explore data"
    statement(s)
def visualization():
    "function to viasualize data"
    statement(s)

#main program:
collectdata()
cleandata()
processdata()
exploredata()
visualization()
```

NOTE:

1.I request all the students to go through the basics of python.

2. This notebook just covers basics for refreshing the topics what you have studied in previous semester.

3. Requesting to be very thorough with python concepts

4.Python is essential and pre-requisit for machine learning.

In [ ]: