**Python**

Variable: when a name is declared for a variable it is stored in name space and it also has a scope

While taking the value of variable it uses legb rule,

L ; local variable : this is the variable which is declared within the function or lambda

E: enclosed function local : this is the variable that is declared outside the subfunction of the function declared.

G: global : this is the variable that is declared outside the function

B:built in function:this is the variable that are built in ex: range etc.

Eg:

Name = ‘ I am a global variable’

Def fun():

Name = ‘ I am enclosed function local’

Def func2();

Name=”I am local variable’

Aciton to be performed.

The global variable won’t be affected if we change the value inside the function but we can affect the global variable by using the key word global within the function to affect the global variable.

Data types:

-String : we can declare strings using single or double quotes and we can perform indexing and reverse indexing in python we can also perform slicing of the strings in python

-indexing can be done to pull any data in the string through indexing i.e ex : myString[8].we can also perform negative indexing also.

-slicing is done to pull the data from the from a start point to the end point in the string i.e myString[3:] this pulls the string from the starting 3 to all the other elements inside the string.

While slicing we can also include stepsize which jumps the inside a string through the string stepsize is declared at the end .

We can perform string concatenation by adding + to it

We can also perform multiplication of letter to display it many times by adding \* symbol(string repitation)

There are inbuilt function to remove the spaces in the string :

Rstrip : to remove right spaces

Lstrip: to remove the left spaces

Strip: to remove the spaces in between both left and right

Replace function is used to replace the existing values with new values.

A = ‘hello world’

a.replace(‘world’ , ‘there’)

output: hello there

we can also perform comparison on the strings , but the equality will be based on the alphabetical order.

We can find the substring in the strings using find() and index() in the forward direction and rfind() and rindex() in the backward direction.

Syntax: a = ‘string’

a.find(substring) , a.index(substring)

both find and index does the same it returns the index value of the substring , but find will return -1 if the substring is not present in the given string of characters , index will return a value error.

There is title case which capitalizes the letter of the first letter in the string

Swap case: converts all lower case characters to upper case and all upper case characters to lower case

There are methods for the string object.

The format method of the string is :

Syntax : the string { to be filled in} . format ( string to be inserted into the braces, if more than one braces are there then specify the string to be inserted into that )

Eg : print(‘hello there {}’.format(b)) where b is ‘my name is hari’

Output: hello there my name is hari

We can use place holder or the value to be printed out

Format method while printing : format( value:width:precison f)

Eg: print( string to be printed { variable: space from the string that is printed: value to be printed after the decimal point f)

Print(‘The value is {r}“.format{r=result:1:.3f}

We can also use %s to print out the string as in the c .

F string method: this is another type of representing the output

Print(f’hello there {b}’) where b is ‘my name is hari’

Output: hello my name is hari

Split function: variable.split(character)

Eg; a.split(‘l’) where a = hello

Output : [‘ he‘ “ “ , ‘o’]

Split function output is in the form of list.

Eval function : this function takes string of values as the input and evaluates the values according to sum or difference etc and provides the result.

List: list holds the collection of elements and they can also be indexed and sliced. We can change the list elements by indexing it and changing the value of the variable.

We can also add the last element to the list by using append method .

Variable name.append()

Append only adds one item , to add more than one item we can use extend and these functions only adds the items at the end but only one element can be added , if we add any list it unpacks it and adds it to the end of the existing list.we can insert at the beginning by using insert and also should specifiy the location to add sinceit takes two arguments.To remove the items we use clear command , it removes all the element in the list.

Pop: this method is used to pop out the last element in the list.

Variable.pop()

We can also index to remove the specified element

-sort : sort method is used to sort the number of the elements which are not in the order in a list , but the sort method returns nothing .

Variable.sort()

Reverse method: this method reverses the element in the list not based on any order .

Variable.reverse()

-Dictionaries ; These are unordered mappings for storing objects

Syntax; {‘key1’: value1: ‘key2’:value2}

Dictionaries values are retrieved by the keys name and they are unordered and cannot be sorted.

In for loop dictionaries can be accesed , for value can be accessed using .values() method and for keys can be accessed by .keys() method and both keys and values can be accessed by .items() method.The inbuilt methods for accessing the data are .fromkeys{} is usally done on empty string to create a dictinory , .get{key name } and .clear{} , .copy()

Dictionaries: d = dict(a=1,b=2,c=3)

Output: {‘a’: 1 , ‘b’: 2 , ‘c’: 3}

Fromkeys is used to declare the same values to all the keys.

B = {}.fromkeys([‘name ‘ , ‘age’ , ‘hobbies’], ‘unkown’)

Pop and popitem are used, we need to specify the key inside the brace for pop and not for pop item

Update can be used to add one dictionary to another.It will be updated at the end

Second.update(first)

-Tuple: This is similar to list but they are immutable and they can also be indexed and slicing.

Syntax: ( value, value , value)

Count and index are two inbuilt methods for tuple . The major difference between the tuple and the list are the list values can be changed dynamically but the tuple values cannot be changed.Tuples are immutables.

Count is used to count the number of duplicates present in the tuple i.e how many times the number is repeated in the tuple.

Sets : it is the collection of unique elements(it removes duplicates present while declaration of the set)

Declaration of set: a = {1,2,3,4,4,5,5,}

Output: a = {1,2,3,4,5}

.add(value) method is used to add elements to the sets

.remove(value) , .discard(value) , .copy and clear .

Math set is another type ,where we can use | to perform union between the two sets and & to perform intersection between them.

We cannot perform any slicing or indexing in sets as Sets do not have any order

Set comprehension:

Eg ; {num \* 2 for num in set}

-Boolean operator: true or false

-File IO : file can written using the write file command and read by using the read function and but once the data in the file is read the curser is at the end so inorder to reset the curser we should reset the curser to the top of the page this can be done through seek function.

The read function reads everything in the file as giant string, in order to read the file in a proper manner we should use readlines function, readlines reads the file as lists

To open the file that is in the different location we should use open and we should provide the full file path and close function is used to close the file

We can open the files in a specific modes as read only (r), writeonly(w) , read and write(r+) , append(a) and write and read(w+)

Variable = open(‘file name’) – syntax to open the file

File.read() – to read a file

File.seek(index value) – this method is used to reset the cursor to any point in the text.

File.readline() – This is used to print only single line at a time in the file.

File.readlines() –This method prints out the text inside the file in the form of lists

File.close() –This is used to close the opened file

File.closed –This is used to check whether the file is closed or not it returns a boolen value.

With open(filename , mode ) as variablename:

File action to perform

The above syntax automatically closes the file, the above method is more common to use the file operation.

File.write(string to be written): this is the syntax to write the value to the file, but the mode in the file open should be in w

Modes: w – write mode , r- read mode , a- append mode , r+ - read write mode

-Logical and comparison operators: these are similar to the operator in other languages

-Loops: if statement

Syntax ; if condition :

Execute the code.

Ifelse stateme Syntax : if condition :

Execute the code if true

Else :

Execute other code.

Elif statement :

Syntax : if(condition) :

Condition

Elif :(other condition)

Condition}

Else :

Condition

For loop: syntax: for variable name in declared variable

Do the action(variable name)

While loop: while (condition):

To be executed.

Break: This statement breaks from the current loop

continue: this statement skips the condition and again reruns the loop

Pass : this does not do anything

Functions:

Syntax : def func name ( args to be passed , or without any argument):

Program to be executed.

The order in which the parameters to be passed are parameters , \*args , default parameters , \*\*kwargs.

Complex functions: Here we declare functions inside another function , or by using enclosed local variable.

Operator:

-Range operator : this is used to print or display value within a given range.

Syntax : range(start, stop, step)

-enumerate : this function is used to get the index value with the given method and outputs in the form of multiple tuples.

A =[1,2,3,4,5]

For items in enumerate(a):

Print(items)

Output; (0,1) (1,2), (2,3) etc.

Syntax : enumerate(variable)

-zip : this function is used to zip two or more lists

Syntax : zip(list1, list2)

A = (1,2,3,4)

B = (5,6,7,8)

C = list(zip(a,b)) output [(1,5)(2,6) etc]

Dict(zip(a,b)) output {1:5 , 2:6 etc}

Tuple(zip(a,b)) output ((1,5)(2,6) etc)

-in and not in are known as membership operator: to check whether the element is there in the array or not and returns Boolean value

Syntax : element in string;

A = ‘hello’ eg: ‘a’ in ‘hello’ output False

Eg ; ‘a’ not in ‘hello’ outptut True

-min and max : this function is used to find the minimum and maximum value in the given list or etc

-importing libraries : syntax; from library name import function

-input: this function reads the value from the keyboard.

Syntax: input()

-comprehension: this function is used to minimize the for loop

Syntax : variable name =[variable for variable in other variable]

Eg for list comprehension : [ num \* 2 for num in lis]

Eg for dictionaries: {key: value for num in dict}

-we can use condtitional statements on them , check the videos for them

Eg of dict , it is similar to list , except no key should be used in dict

{key value : (num \* 2 if num %2==0 else ‘odd’ ) for num in lis}

Methods:They are useful to shorten the code, so that the function call can be made to repeat the function .

Syntax: def name\_of\_function():

Function execution

Return

\*args inside the () in the function helps to pass many values without any limit.

\*\*kwargs inside the () in the function helps to pass the values with keys .

Eg def function(\*\*kwargs):

for items in kwargs.items():

print(items)

function(a = 'hari' , b= 12 , c= 'bangalore')

Output : ('a', 'hari')

('b', 12)

('c', 'bangalore')

Capitalize method: this function is used to capitalize the first element in the string.

A = ‘hello world’

a.capitalize() output : ‘HELLO WORLD’

.join method= this method is to join the strings in a list

Syntax : ‘to be joined with’ .join (variable)

\*maps function: map can be used to connect the function with the lists or anything .

Syntax: map(function name, list name)

Eg : a = [1,2,3,4,5]

def double(x):

return x\*2

b = list(map(double , a))

print(b)

output: [2,4,6,8,10]

we can use with lambda expression as well, but I have used it with filter for example

\*filter: filter function is used to filter the value to a certain type as eg true or false etc.

While applying filter ,the function should return a Boolean type , since it filters the result and provides only on type of solution

Syntax: filter(name of the function , list name)

a = [1,2,3,4,5]

b = list(filter(lambda x: x%2==0 , a ))

print(b)

output: [2,4]

we use the filter method to filter the output based on some condition. We can use normal methods with the filter function , I have used normal method in map function.

\*lambda expression: this is also known as anonimus function

Syntax: function to be assigned to : lambda variable name: function to perform

Eg: square = lambda num : num \*\*2

Square(2)

-this method can be used in conjunction with the maps and filter as well ,and it is more efficient with them

-islower() and –isupper() : these are function used to check whether the string or list is upper or lower.

OOPS concept of python:

Class declaration:

Syntax: class ClassName:

Def \_\_init\_\_(self, parameter 1, parameter2):

Self.parameter1 = parameter1

Self.parameter2 = parameter2

Init method acts as constructor which runs every time the instance is created.

Self is the attribute that is used to connect the instance and the class method.

Eg : Basic Class declaration

class User:

def \_\_init\_\_(self, name , age):

self.name = name

self.age = age

print(f'The class is running and the my name is {name} and my age is {age}')

user1 = User('hari' , 23)

output: The class is running and the my name is hari and my age is 23

-class object attribute : this is declared outside the class method and this is same for all the instances created.

Eg; class dog():

Species = ‘mammal’

Def \_\_init\_\_(self, par1,par2):

Self.par1 = par1

Dog.speicies = any action

Mydog = dog(par1, par2)

Mydog.species

This indicates that the class object attribute need not be declared in the method , it remains same for every instance created. If we need to access the class attribute we need to use it with class name.

There are different types of using ‘\_’in python during the declaration of the variable inside the class.

\_name – this indicates that the variable is private but in python this variable can be accessed since there in is no methodology of private public and other in python it is just indicating to other developers to let them know that this variable should be held as private.

\_\_name – This is another type of variable, but this variable is specific to that class and this is declared inside the init method and there is another way of calling this method.

Syntax : while printing these king of methods

Print(objectname.\_classname\_varibale(this is the variable that is declared in \_format)

\_\_name\_\_ : this kind of declaration should not be done , these only should be used when it is required for overwriting similar kind of methods or variable that is already present in the python inbuilt.

Class method: This is used to pass the class to the method

Syntax :@classmethod

Def nameofthemethod(cls):

Method to perform.

-Inheritence:The advantage of using inheritance is to use the code that already worked on and to reuse the code and ability to reduce the complexity of the code.

Syntax: class classname(base classname):

Def \_\_init\_\_(self):

Super().\_\_init\_\_(value of the base class)

Eg: class Second(User):

def \_\_init\_\_(self, name , age):

super().\_\_init\_\_(name = name , age= age)

After inheriting the base class , we can append all the methods of the base class. And also we can define the same methods as the base class in the subclass and overwrite them.

Inorder to inherit the properites from the base class we have to use

Super().\_\_init\_\_(attributes of the base class neglecting self)

Properties for classes using generators(getters and setters)

Synatax : @propety

Def method name(self)

Return value

@abovemethodname.setter

Def samemethodname(self, value):

Function to perform

For the setter we can assign directly the value eg: instancename.namefthesetter = value to be assigned.

Eg : @property

def age(self):

return self.\_age

@age.setter

def age(self, value):

if value >= 10:

self.\_age = age

else:

self.\_age = 0

Here we are not altereing the age directly , but we are creating properties as we can alter it

Moderators:These are special type of inheritance which can do special things that a we don’t do it in the normal main class

Syntax; class Moderator(main class):

Rest is similar to the other inheritance but we can declare different methods here.

Multiple inheritance: here we can inherit the methods from more than two classes

Syntax: class name(class1, class2):

Rest is similar to the other inheritance class

Here the methods are inherited in the order of inheritance class, class 1, class2.

Polymorphism: polymorphism is having the same class in the different class , we use the class to access that method , since all the method are same.

Abstract class: these are classes ,where we don’t expect to create instance of it , but we want those methods to be overwritten by the inherited classes.

Magic methods or special methods:These are special methods that are created to use with the objects. We can create a method and then append it to the object eg of them are str method and len method.The some of the magic methods are \_\_Repr\_\_, \_\_add\_\_,\_\_mul\_\_

Eg: def \_\_str\_\_(self)

Action to be performed.

Iterators and iterables.

Iterators: An object that can be iterated , we can run a for loop on it and the object which returns a data when iterated.where we can run a next() function on it

Iterable: This is the object which returns the iterator when iter() is called upon

The value like ‘hello’ is iterable and we can iterate it using iterators.

We can use custom methods and classes with \_\_iter\_\_ and \_\_next\_\_ magic methods to create the custom loop methods

Modules and packages: modules is just like a file where we can import it to other files .

There are different types to import them.

i)import module name

ii)import module name as anyname to be used. ‘as ‘ is known as aliasing

iii)from module import specific module.

There are custom modules where we define it and import that into other files.

Packages are similar to that but contains additional \_\_init\_\_ file which indicates it is a package.

\_\_name\_\_ and \_\_main\_\_ : this is to indicate whether the python script is runned directly or with imported.

Error handling : Try block is used to check for the error and except block is used to handle that error when there is error and else can be combined with try and except to give options while error handling, Finally is the block which executes everytime. In the below syntax the else block will run if the except block does not run and finally block will run always.

Syntax:

Try:

Code

Except :

Code

Else:

Code

Finally :

Code

Different types of error :name error , valueerror , type error , zeroerror

Decorators:

These decorators are used to decorate a function , it is used to insert method inside another method with minimum effort.

Syntax: def bepolite(fn):

def wrapper():

print('hello world ')

fn()

print('end of the world')

return wrapper

@bepolite

def greet():

print('my name is hari')

greet()

Output ; hello world

my name is hari

end of the world

Generator: generator is a method which is used to generate values with effective memory usage , it is a regular function with yield keyword.In regular function return I called only once but in the generators yield can be called multiple times.Genertors is more memory efficient and fast when compared to the list methods

The keyword is yield which is used to generate values inside a for loop

Eg: For I in range(n):

Yield i

The main keywords of generators are iter and next

Iter is used to convert any string to be iterable and next is used to generate next value of the iterator.

Collections;

-counter: counter is a subclass of dictionary and used to calculate the number of occurrence of the a value in a list.It outputs in the form of dictionary

Syntax : counter (list name)

There are many sub classes , check the documentation

Eg; most\_common, items etc.

-default dictionary: This function is used to set the default value while declaring the value , it does not show any key error .

Syntax : defaultdict(default value)

Check Jupiter notebook for more.

-OrderedDict: this is used to retain the order in which the values are entered into the dictionary ,during the creation. Assign this value to the variable.

-namedtuple: it is similar to declaring the class , it is used to minimize the complexity of the tuple.

Syntax: class name = namedtuple(class name, attributes)

Datetime: this method is used specify the date time, and to indicate date time

Tuple unpacking :

When a tuple or list is passed into the function to execute somefunction it throws an error

We should perform tuple unpacking before sending it in the function .

Using \* we can make tuple unpacking.

Dictinoary unpacking:

Similar to tuple we can also perform dictionary unpacking :

Unsing \*\* we can make dictionary unpacking.

All and any function : all function returns true if all the elements in the list, string etc are truthy i.e all the elements satisfy the condition .

All([list comprehension])

Any function returns true if any element in the list or string etc meets the condition .

Any([list comprehension])

Sortable method and reversed method : this function is used to sort or reverse list , tuples etc , but this does not alter the declared variable but creates a copy. We can also use condition for sorting or reversing.

Syntax: sorted(variable name)

Reverse(variable name)

Python testing

-assert is the statement used for testing some method for validity

Syntax: assert condition to execute , statement to show for the error

Assertion warning neglection , if we use –O while running the python file python neglects the assertion line while running the testing

-Doctests: Doctests are used to run the test cases for which we need a expected result.

The syntax: “””

>>> use case

Result for the use case”””

Return ouput

Command to run the python file

Python –m doctest –v python file

-Unit test:This testing is used to run the testing on single functions or classes rather than running testing on whole application .

Syntax: import unittest

from basic2 import eat , nap

class name of the classs(unittest.TestCase):

def method name(self):

self.assertEqual{any assert statement for unity}(method to be tested), string to return if failed))

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()