**Features of Python:**

1. Simple and easy to learn: Python is a simple programming language. When we read Python program,we can feel like reading english statements. The syntaxes are very simple and only 30+ kerywords are available. When compared with other languages, we can write programs with very less number of lines. Hence more readability and simplicity. We can reduce development and cost of the project.

2. Freeware and Open Source: We can use Python software without any licence and it is freeware. Its source code is open,so that we can we can customize based on our requirement. Eg: Jython is customized version of Python to work with Java Applications

3. High Level Programming language: Python is high level programming language and hence it is programmer friendly language. Being a programmer we are not required to concentrate low level activities like memory management and security etc..

4. Platform Independent: Once we write a Python program,it can run on any platform without rewriting once again. Internally PVM is responsible to convert into machine understandable form.

5. Portability: Python programs are portable. ie we can migrate from one platform to another platform very easily. Python programs will provide same results on any paltform.

6. Dynamically Typed: In Python we are not required to declare type for variables. Whenever we are assigning the value, based on value, type will be allocated automatically.Hence Python is considered as dynamically typed language. But Java, C etc are Statically Typed Languages b'z we have to provide type at the beginning only. This dynamic typing nature will provide more flexibility to the programmer

7. Both Procedure Oriented and Object Oriented: Python language supports both Procedure oriented (like C, pascal etc) and object oriented (like C++,Java) features. Hence we can get benefits of both like security and reusability etc

8. Interpreted: We are not required to compile Python programs explcitly. Internally Python interpreter will take care that compilation. If compilation fails interpreter raised syntax errors. Once compilation success then PVM (Python Virtual Machine) is responsible to execute.

9. Extensible: We can use other language programs in Python. The main advantages of this approach are:

1. We can use already existing legacy non-Python code 2. We can improve performance of the application

10. Embedded: We can use Python programs in any other language programs. i.e we can embedd Python programs anywhere. 11. Extensive Library: Python has a rich inbuilt library. Being a programmer we can use this library directly and we are not responsible to implement the functionality.

**Identifiers** A name in Python program is called identifier. It can be class name or function name or module name or variable name.

**Identifier: 1. Alphabet Symbols (Either Upper case OR Lower case)**

**2. If Identifier is start with Underscore (\_) then it indicates it is private.**

**3. Identifier should not start with Digits.**

**4. Identifiers are case sensitive.**

**5. We cannot use reserved words as identifiers ρEg: def=10**

**6. There is no length limit for Python identifiers. But not recommended to use too lengthy identifiers.**

**7. Dollor ($) Symbol is not allowed in Python**

**Note: 1. If identifier starts with \_ symbol then it indicates that it is private 2. If identifier starts with \_\_(two under score symbols) indicating that strongly private identifier. 3.If the identifier starts and ends with two underscore symbols then the identifier is language defined special name,which is also known as magic methods.**

**Reserved Words** In Python some words are reserved to represent some meaning or functionality. Such type of words are called Reserved words. There are 33 reserved words available in Python.

True,False,None

• and, or ,not,is

• if,elif,else

• while,for,break,continue,return,in,yield

• try,except,finally,raise,assert

• import,from,as,class,def,pass,global,nonlocal,lambda,del,with

• Note:

1. All Reserved words in Python contain only alphabet symbols.

2. Except the following

3 reserved words, all contain only lower case alphabet symbols. True• False• None•

**Data Types**- Data Type represent the type of data present inside a variable. In Python we are not required to specify the type explicitly. Based on value provided,the type will be assigned automatically.Hence Python is Dynamically Typed Language. Python contains the following inbuilt data types 1. int 2. float 3.complex 4.bool 5.str 6.bytes 7.bytearray 8.range 9.list 10.tuple 11.set 12.frozenset 13.dict 14.None

**Note: Python contains several inbuilt functions 1.type() to check the type of variable 2. id() to get address of object**

**3. print() to print the value In Python everything is object**

**Slicing of Strings:** slice means a piece [ ] operator is called slice operator,which can be used to retrieve parts of String. In Python Strings follows zero based index. The index can be either +ve or -ve. +ve index means forward direction from Left to Right -ve index means backward direction from Right to Left

**Note: 1. In Python the following data types are considered as Fundamental Data types int• float• complex• bool• str• 2. In Python,we can represent char values also by using str type and explicitly char type is not available.**

**Type Casting** We can convert one type value to another type. This conversion is called Typecasting or Type coersion. The following are various inbuilt functions for type casting. 1. int() 2. float() 3. complex() 4. bool() 5. str()

**1.int():** We can use this function to convert values from other types to int

Note: 1. We can convert from any type to int except complex type. 2. If we want to convert str type to int type, compulsary str should contain only integral value and should be specified in base-10

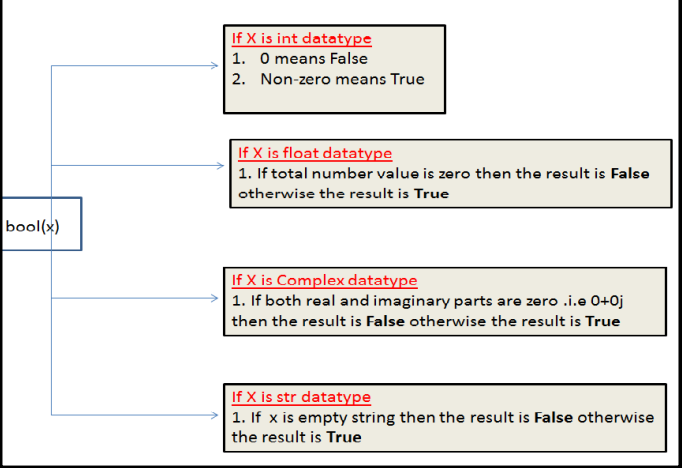
**2. float():** We can use float() function to convert other type values to float type.

Note: 1. We can convert any type value to float type except complex type. 2. Whenever we are trying to convert str type to float type compulsary str should be either integral or floating point literal and should be specified only in base-10.

**3.complex():** We can use complex() function to convert other types to complex type. Form-1: complex(x) We can use this function to convert x into complex number with real part x and imaginary part 0.

Form-2: complex(x,y) We can use this method to convert x and y into complex number such that x will be real part and y will be imaginary part.

**4. bool():** We can use this function to convert other type values to bool type.

****

5. str(): We can use this method to convert other type values to str type

**String Data Type**

**What is String?**

**Any sequence of characters within either single quotes or double quotes is considered as a String**

How to define multi-line String literals: We can define multi-line String literals by using triple single or double quotes.

Eg: >>> s='''durga software solutions'''

How to access characters of a String: We can access characters of a string by using the following ways**. 1. By using index 2. By using slice operator**

1. **By using index:**

Python supports both +ve and -ve index.

+ve index means left to right(Forward direction)

-ve index means right to left(Backward direction)

2. Accessing characters by using slice operator:

**Syntax: s[bEginindex:endindex:step]**

**bEginindex:From where we have to consider slice(substring)**

**endindex: We have to terminate the slice(substring) at endindex-1**

**step: incremented value**

**Note: If we are not specifying bEgin index then it will consider from bEginning of the string. If we are not specifying end index then it will consider up to end of the string The default value for step is 1**

Behaviour of slice operator: s[bEgin:end:step]

* step value can be either +ve or –ve
* if +ve then it should be forward direction(left to right) and we have to consider bEgin to end-1
* if -ve then it should be backward direction(right to left) and we have to consider bEgin to end+

\*\*\***Note: In the backward direction if end value is -1 then result is always empty. In the forward direction if end value is 0 then result is always empty**

**In forward direction:**

default value for bEgin: 0

default value for end: length of string

default value for step: +1

**In backward direction:**

default value for bEgin: -1

default value for end: -(length of string+1)

**Note: Either forward or backward direction, we can take both +ve and -ve values for bEgin and end index**

**Mathematical Operators for String: We can apply the following mathematical operators for Strings.**

1. + operator for concatenation

2. \* operator for repetition

len() in-built function: We can use len() function to find the number of characters present in the string.

Checking Membership: We can check whether the character or string is the member of another string or not by using in and not in operators

Comparison of Strings: We can use comparison operators (<,<=,>,>=) and equality operators(==,!=) for strings. Comparison will be performed based on alphabetical order.

Removing spaces from the string: We can use the following 3 methods

1. rstrip()===>To remove spaces at right hand side

2. lstrip()===>To remove spaces at left hand side

3. strip() ==>To remove spaces both sides

**Finding Substrings: We can use the following 4 methods**

**For forward direction: find() index()**

**For backward direction: rfind() rindex()**

1. find(): s.find(substring)-> Returns index of first occurrence of the given substring**. If it is not available then we will get -1**
2. By default find() method can search total string. We can also specify the boundaries to search. s.find(substring,bEgin,end) It will always search from bEgin index to end-1 index
3. index() method: index() method is exactly same as find() method except **that if the specified substring is not available then we will get ValueError.**

**Counting substring in the given String**: We can find the number of occurrences of substring present in the given string by using count() method.

1. s.count(substring) ==> It will search through out the string

2. s.count(substring, bEgin, end) ===> It will search from bEgin index to end-1 index

**Replacing a string with another string:**

s.replace(oldstring,newstring) inside s, every occurrence of oldstring will be replaced with newstring.

String objects are immutable then how we can change the content by using replace() method. Once we creates string object, we cannot change the content.This non changeable behaviour is nothing but immutability. If we are trying to change the content by using any method, then with those changes a new object will be created and changes won't be happend in existing object. Hence with replace() method also a new object got created but existing object won't be changed.

**Splitting of Strings:** We can split the given string according to specified seperator by using split() method.

**l=s.split(seperator)** The default seperator is space. The return type of split() method is List

**Joining of Strings:** We can join a group of strings(list or tuple) wrt the given seperator.

s=seperator.join(group of strings)

Changing case of a String: We can change case of a string by using the following 4 methods.

1. **upper()===>To convert all characters to upper case**
2. **lower() ===>To convert all characters to lower case**
3. **swapcase()===>converts all lower case characters to upper case and all upper case characters to lower case**
4. **title() ===>To convert all character to title case. i.e first character in every word should be upper case and all remaining characters should be in lower case.**
5. **capitalize() ==>Only first character will be converted to upper case and all remaining characters can be converted to lower case**

**Checking starting and ending part of the string:** Python contains the following methods for this purpose

Return true or false.

1. s.startswith(substring)

2. s.endswith(substring)

**To check type of characters present in a string: Python contains the following methods for this purpose.**

1) isalnum(): Returns True if all characters are alphanumeric( a to z , A to Z ,0 to9 )

2) isalpha(): Returns True if all characters are only alphabet symbols(a to z,A to Z)

3) isdigit(): Returns True if all characters are digits only( 0 to 9)

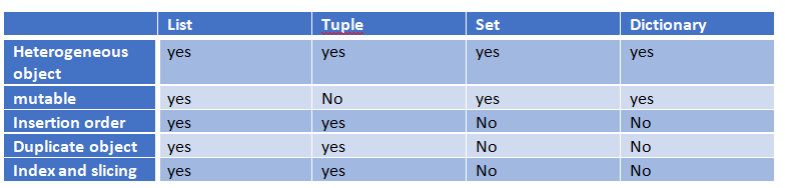
4) islower(): Returns True if all characters are lower case alphabet symbols

5) isupper(): Returns True if all characters are upper case aplhabet symbols

6) istitle(): Returns True if string is in title case

7) isspace(): Returns True if string contains only spaces

**Formatting the Strings**: We can format the strings with variable values by using replacement operator {} and format() method.



**List Data Structure** If we want to represent a group of individual objects as a single entity where insertion order preserved and duplicates are allowed, then we should go for List.

insertion order preserved. duplicate objects are allowed heterogeneous objects are allowed. List is dynamic because based on our requirement we can increase the size and decrease the size. In List the elements will be placed within square brackets and with comma seperator.

We can differentiate duplicate elements by using index and we can preserve insertion order by using index. Hence index will play very important role. Python supports both positive and negative indexes. +ve index means from left to right where as negative index means right to left

[10,"A","B",20, 30, 10] List objects are mutable.i.e we can change the content.

Creation of List Objects: 1. We can create empty list object as follows...

1) list=[]

1. If we know elements already then we can create list as follows list=[10,20,30,40]
2. With dynamic input: 1) list=eval(input("Enter List:")) 2) print(list) 3) print(type(list))
3. With list() function:
4. with split() function:

Accessing elements of List: We can access elements of the list either by using index or by using slice operator(:)

1. By using index: List follows zero based index. ie index of first element is zero. List supports both +ve and -ve indexes. +ve index meant for Left to Right -ve index meant for Right to Left
2. 2. By using slice operator: Syntax: list2= list1[start:stop:step]
3. start ==>it indicates the index where slice has to start default value is 0
4. stop ===>It indicates the index where slice has to end default value is max allowed index of list ie length of the list
5. step ==>increment value default value is 1

List vs mutability: Once we creates a List object,we can modify its content. Hence List objects are mutable

Traversing the elements of List: The sequential access of each element in the list is called traversal.

1. By using while loop:
2. By using for loop:

Important functions of List: I. To get information about list:

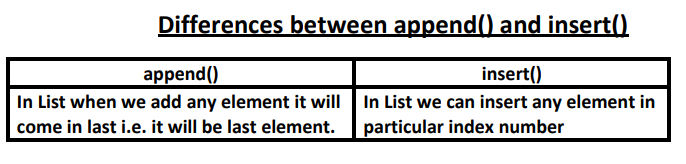
1. **len():** returns the number of elements present in the list
2. **count():** It returns the number of occurrences of specified item in the list
3. **index()** function: returns the index of first occurrence of the specified item.

**Note: If the specified element not present in the list then we will get ValueError.Hence before index() method we have to check whether item present in the list or not by using in operator.**

II. Manipulating elements of List:

1. append() function: We can use append() function to add item at the end of the list.
2. insert() function: To insert item at specified index position

Note: If the specified index is greater than max index then element will be inserted at last position. If the specified index is smaller than min index then element will be inserted at first position.



1. extend() function: To add all items of one list to another list

l1.extend(l2) all items present in l2 will be added to l1

1. remove() function: We can use this function to remove specified item from the list.If the item present multiple times then only first occurrence will be removed.

Note: Hence before using remove() method first we have to check specified element present in the list or not by using in operator.

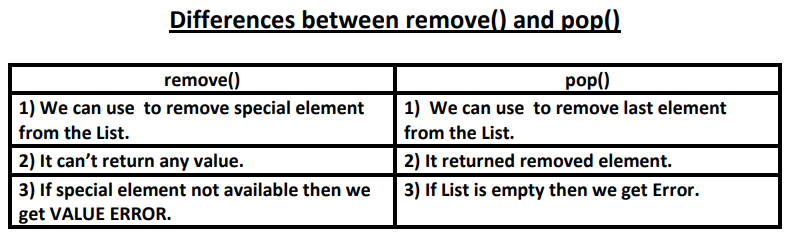
1. pop() function: It removes and returns the last element of the list. This is only function which manipulates list and returns some element.

**Note: 1. pop() is the only function which manipulates the list and returns some value**

**2. In general we can use append() and pop() functions to implement stack datastructure by using list,which follows LIFO(Last In First Out) order.**

In general we can use pop() function to remove last element of the list. But we can use to remove elements based on index. n.pop(index)===>To remove and return element present at specified index.

n.pop()==>To remove and return last element of the list



**Note: List objects are dynamic. i.e based on our requirement we can increase and decrease the size.**

**append(),insert() ,extend() ===>for increasing the size/growable nature**

**remove(), pop() ======>for decreasing the size /shrinking nature**

III. Ordering elements of List: 1. reverse(): We can use to reverse() order of elements of list.

2.sort() function: In list by default insertion order is preserved. If want to sort the elements of list according to default natural sorting order then we should go for sort() method. For numbers ==>default natural sorting order is Ascending Order For Strings ==> default natural sorting order is Alphabetical Order

**Note: To use sort() function, compulsory list should contain only homogeneous elements. otherwise we will get TypeError**

**Note: In Python 2 if List contains both numbers and Strings then sort() function first sort numbers followed by strings**

**To sort in reverse of default natural sorting order:** We can sort according to reverse of default natural sorting order by using reverse=True argument

Aliasing and Cloning of List objects: The process of giving another reference variable to the existing list is called aliasing.

1. By using slice operator:
2. By using copy() function:

Difference between = operator and copy() function = operator meant for aliasing copy() function meant for cloning

Using Mathematical operators for List Objects: We can use + and \* operators for List objects.

1. Concatenation operator(+):Note: To use + operator compulsory both arguments should be list objects,otherwise we will get TypeError.
2. Repetition Operator(\*): We can use repetition operator \* to repeat elements of list specified number of times

Comparing List objects We can use comparison operators for List objects.

**Note: Whenever we are using comparison operators(==,!=) for List objects then the following should be considered**

1. The number of elements

2. The order of elements

3. The content of elements (case sensitive)

Note: When ever we are using relatational operators(<,<=,>,>=) between List objects,only first element comparison will be performed.

**Membership operators:** We can check whether element is a member of the list or not by using memebership operators.

**in operator**

**not in operator**

clear() function: We can use clear() function to remove all elements of List.

**Nested Lists:** Sometimes we can take one list inside another list. Such type of lists are called nested lists.

**Note: We can access nested list elements by using index just like accessing multi dimensional array elements.**

**List Comprehensions**: It is very easy and compact way of creating list objects from any iterable objects(like list,tuple,dictionary,range etc) based on some condition.

Syntax: list=[expression for item in list if condition]

**Tuple Data Structure** 1. Tuple is exactly same as List except that it is immutable. i.e once we creates Tuple object,we cannot perform any changes in that object. Hence Tuple is Read Only version of List.

2. If our data is fixed and never changes then we should go for Tuple.

3. Insertion Order is preserved

4. Duplicates are allowed

5. Heterogeneous objects are allowed.

6. We can preserve insertion order and we can differentiate duplicate objects by using index. Hence index will play very important role in Tuple also. Tuple support both +ve and -ve index. +ve index means forward direction(from left to right) and -ve index means backward direction(from right to left)

7. We can represent Tuple elements within Parenthesis and with comma seperator. Parenethesis are optional but recommended to use.

Tuple creation: 1. t=() creation of empty tuple

2. t=(10,) t=10, creation of single valued tuple ,parenthesis are optional,should ends with comma

3. t=10,20,30 t=(10,20,30) creation of multi values tuples & parenthesis are optional

4. By using tuple() function:

Accessing elements of tuple: We can access either by index or by slice operator

1. By using index:
2. By using slice operator:

Tuple vs immutability: Once we creates tuple,we cannot change its content. Hence tuple objects are immutable.

Mathematical operators for tuple: We can apply + and \* operators for tuple

1. Concatenation Operator(+):
2. Multiplication operator or repetition operator(\*)

Important functions of Tuple:

1. len() To return number of elements present in the tuple
2. count() To return number of occurrences of given element in the tuple
3. index() returns index of first occurrence of the given element. If the specified element is not available then we will get ValueError.
4. sorted() To sort elements based on default natural sorting order. We can sort according to reverse of default natural sorting order as follows t1=sorted(t,reverse=True)
5. min() and max() functions: These functions return min and max values according to default natural sorting order.
6. cmp(): It compares the elements of both tuples.

If both tuples are equal then returns 0

If the first tuple is less than second tuple then it returns -1

If the first tuple is greater than second tuple then it returns +1

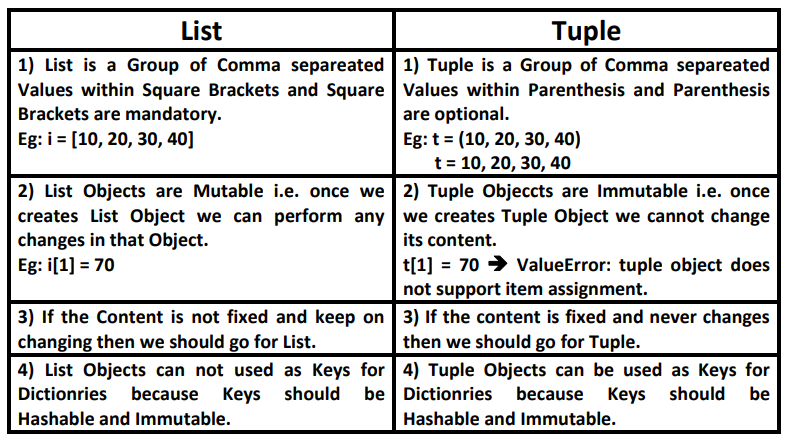
**Note: cmp() function is available only in Python2 but not in Python 3**

**Tuple Packing and Unpacking:** We can create a tuple by packing a group of variables. Tuple unpacking is the reverse process of tuple packing We can unpack a tuple and assign its values to different variables

**Note:** **At the time of tuple unpacking the number of variables and number of values should be same. ,otherwise we will get ValueError.**

**Tuple Comprehension**: Tuple Comprehension is not supported by Python. t= ( x\*\*2 for x in range(1,6))

Differences between List and Tuple: List and Tuple are exactly same except small difference: List objects are mutable where as Tuple objects are immutable. In both cases insertion order is preserved, duplicate objects are allowed, heterogenous objects are allowed, index and slicing are supported.



**Set Data Structure** If we want to represent a group of unique values as a single entity then we should goϖ for set. Duplicates are not allowed.

ϖ Insertion order is not preserved.But we can sort the elements.

ϖ Indexing and slicing not allowed for the set.

ϖ Heterogeneous elements are allowed.

ϖ Set objects are mutable i.e once we creates set object we can perform any changes in that object based on our requirement.

We can represent set elements within curly braces and with comma separation

ϖ We can apply mathematical operations like union,intersection,difference etc on setϖ objects.

Creation of Set objects: s=set(any sequence)

**Note: While creating empty set we have to take special care. Compulsory we should use set() function.**

s={} ==>It is treated as dictionary but not empty set.

Important functions of set:

1. add(x): Adds item x to the set
2. update(x,y,z): To add multiple items to the set. Arguments are not individual elements and these are Iterable objects like List,range etc. All elements present in the given Iterable objects will be added to the set.
3. copy(): Returns copy of the set. It is cloned object.
4. pop(): It removes and returns some random element from the set.
5. remove(x): It removes specified element from the set. If the specified element not present in the Set then we will get KeyError
6. discard(x): It removes the specified element from the set. If the specified element not present in the set then we won't get any error.
7. clear(): To remove all elements from the Set.

**Q. What is the difference between add() and update() functions in set?**

We can use add() to add individual item to the Set,where as we can use update() function to add multiple items to Set. add() function can take only one argument where as update() function can take any number of arguments but all arguments should be iterable objects.

**Q. Which of the following are valid for set s?**

1. s.add(10)

2. s.add(10,20,30) TypeError: add() takes exactly one argument (3 given)

3. s.update(10) TypeError: 'int' object is not iterable

4. s.update(range(1,10,2),range(0,10,2))

Mathematical operations on the Set:

1.union():x.union(y) ==>We can use this function to return all elements present in both sets x.union(y) or x|y

2. intersection(): x.intersection(y) or x&y Returns common elements present in both x and y

3. difference(): x.difference(y) or x-y returns the elements present in x but not in y

4.symmetric\_difference(): x.symmetric\_difference(y) or x^y Returns elements present in either x or y but not in both

Membership operators: (in , not in)

Set Comprehension: Set comprehension is possible.

set objects won't support indexing and slicing:

**Dictionary Data Structure** We can use List,Tuple and Set to represent a group of individual objects as a single entity. If we want to represent a group of objects as key-value pairs then we should go for Dictionary.

Eg: rollno----name phone number--address ipaddress---domain name

Duplicate keys are not allowed but values can be duplicated. Hetrogeneous objects are allowed for both key and values. insertion order is not preserved Dictionaries are mutable Dictionaries are dynamic indexing and slicing concepts are not applicable

Note: In C++ and Java Dictionaries are known as "Map" where as in Perl and Ruby it is known as "Hash"

How to create Dictionary--> d={} or d=dict() we are creating empty dictionary.

How to access data from the dictionary-> We can access data by using keys.

If the specified key is not available then we will get KeyError

We can prevent this by checking whether key is already available or not by using has\_key() function or by using in operator.

But has\_key() function is available only in Python 2 but not in Python 3. Hence compulsory we have to use in operator.

How to update dictionaries?

d[key]=value If the key is not available then a new entry will be added to the dictionary with the specified key-value pair

If the key is already available then old value will be replaced with new value

How to delete elements from dictionary-🡪

del d[key] - It deletes entry associated with the specified key. If the key is not available then we will get KeyError

d.clear() - To remove all entries from the dictionary

del d - To delete total dictionary.Now we cannot access d

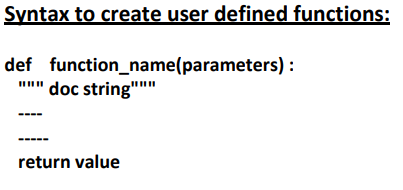
Important functions of dictionary:

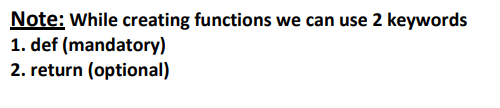
1. dict(): To create a dictionary
2. len() Returns the number of items in the dictionary
3. clear(): To remove all elements from the dictionary
4. get(): To get the value associated with the key d.get(key) If the key is available then returns the corresponding value otherwise returns None.It wont raise any error. d.get(key,defaultvalue) If the key is available then returns the corresponding value otherwise returns default value.
5. pop(): d.pop(key) It removes the entry associated with the specified key and returns the corresponding value If the specified key is not available then we will get KeyError
6. popitem(): It removes an arbitrary item(key-value) from the dictionaty and returns it.
7. keys(): It returns all keys associated eith dictionary
8. values(): It returns all values associated with the dictionary
9. items(): It returns list of tuples representing key-value pairs. [(k,v),(k,v),(k,v)]
10. copy(): To create exactly duplicate dictionary(cloned copy)
11. setdefault(): d.setdefault(k,v) If the key is already available then this function returns the corresponding value. If the key is not available then the specified key-value will be added as new item to the dictionary.
12. update(): d.update(x) All items present in the dictionary x will be added to dictionary d

Dictionary Comprehension: Comprehension concept applicable for dictionaries also.

**FUNCTIONS** 🡪 If a group of statements is repeatedly required then it is not recommended to write these statements everytime seperately.We have to define these statements as a single unit and we can call that unit any number of times based on our requirement without rewriting. This unit is nothing but function. The main advantage of functions is code Reusability. Note: In other languages functions are known as methods,procedures,subroutines etc Python supports 2 types of functions 1. Built in Functions 2. User Defined Functions

1. **Built in Functions:** The functions which are coming along with Python software automatically,are called built in functions or pre defined functions
2. **User Defined Functions:** The functions which are developed by programmer explicitly according to business requirements ,are called user defined functions.

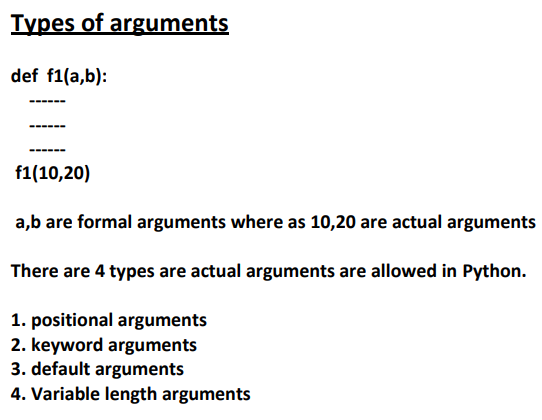




**Parameters** Parameters are inputs to the function. If a function contains parameters,then at the time of calling,compulsory we should provide values otherwise,otherwise we will get error.

**Return Statement:** Function can take input values as parameters and executes business logic, and returns output to the caller with return statement.

**Returning multiple values from a function:** In other languages like C,C++ and Java, function can return atmost one value. But in Python, a function can return any number of values.



1. positional arguments: These are the arguments passed to function in correct positional order.

The number of arguments and position of arguments must be matched. If we change the order then result may be changed. If we change the number of arguments then we will get error.

1. keyword arguments: We can pass argument values by keyword i.e by parameter name. Here the order of arguments is not important but number of arguments must be matched.

**Note: We can use both positional and keyword arguments simultaneously. But first we have to take positional arguments and then keyword arguments,otherwise we will get syntaxerror.**

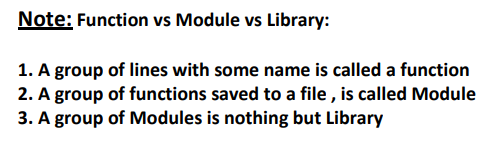
1. Default Arguments: Sometimes we can provide default values for our positional arguments. If we are not passing any name then only default value will be considered.

**\*\*\*Note: After default arguments we should not take non default arguments**

1. Variable length arguments: Sometimes we can pass variable number of arguments to our function,such type of arguments are called variable length arguments. We can declare a variable length argument with \* symbol as follows def f1(\*n): We can call this function by passing any number of arguments including zero number. Internally all these values represented in the form of tuple. **Note: We can mix variable length arguments with positional arguments.**

**Note: After variable length argument,if we are taking any other arguments then we should provide values as keyword arguments.**

**Note: We can declare key word variable length arguments also. For this we have to use \*\*. def f1(\*\*n): We can call this function by passing any number of keyword arguments. Internally these keyword arguments will be stored inside a dictionary.**



Types of Variables Python supports 2 types of variables. 1. Global Variables 2. Local Variables

1. **Global Variables** The variables which are declared outside of function are called global variables. These variables can be accessed in all functions of that module.
2. **Local Variables:** The variables which are declared inside a function are called local variables. Local variables are available only for the function in which we declared it.i.e from outside of function we cannot access.

global keyword: We can use global keyword for the following 2 purposes:

1. To declare global variable inside function

2. To make global variable available to the function so that we can perform required modifications

**Note: If global variable and local variable having the same name then we can access global variable inside a function as follows**

**Recursive Functions** :A function that calls itself is known as Recursive Function. The main advantages of recursive functions are: 1. We can reduce length of the code and improves readability 2. We can solve complex problems very easily.

**Anonymous Functions**: Sometimes we can declare a function without any name,such type of nameless functions are called anonymous functions or lambda functions. The main purpose of anonymous function is just for instant use(i.e for one time usage)

Normal Function: We can define by using def keyword.

def squareIt(n): return n\*n

lambda Function: We can define by using lambda keyword lambda n:n\*n

Syntax of lambda Function: lambda argument\_list : expression

**Note: By using Lambda Functions we can write very concise code so that readability of the program will be improved.**

**Note: Lambda Function internally returns expression value and we are not required to write return statement explicitly.**

**Note: Sometimes we can pass function as argument to another function. In such cases lambda functions are best choice. We can use lambda functions very commonly with filter(),map() and reduce() functions,b'z these functions expect function as argument.**

**filter() function:** We can use filter() function to filter values from the given sequence based on some condition. filter(function,sequence) where function argument is responsible to perform conditional check sequence can be list or tuple or string.

**map() function:** For every element present in the given sequence,apply some functionality and generate new element with the required modification. For this requirement we should go for map() function. Eg: For every element present in the list perform double and generate new list of doubles. Syntax: map(function,sequence) The function can be applied on each element of sequence and generates new sequence.

**reduce() function:** reduce() function reduces sequence of elements into a single element by applying the specified function. reduce(function,sequence) reduce() function present in functools module and hence we should write import statement.

**Note: In Python every thing is treated as object.Even functions also internally treated as objects only.**

**Function Aliasing:** For the existing function we can give another name, which is nothing but function aliasing.

**Note: In the above example only one function is available but we can call that function by using either wish name or greeting name. If we delete one name still we can access that function by using alias name**

**Nested Functions:** We can declare a function inside another function, such type of functions are called Nested functions.

In the above example inner() function is local to outer() function and hence it is not possible to call directly from outside of outer() function. Note: A function can return another function.

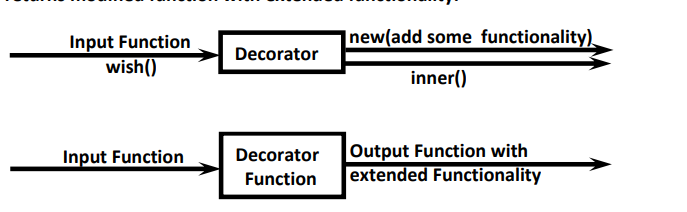
What is the differenece between the following lines? f1 = outer f1 = outer()

In the first case for the outer() function we are providing another name f1(function aliasing).

• But in the second case we calling outer() function,which returns inner function.For that inner function() we are providing another name f1

**Note: We can pass function as argument to another function**

**Function Decorators:** Decorator is a function which can take a function as argument and extend its functionality and returns modified function with extended functionality. The main objective of decorator functions is we can extend the functionality of existing functions without modifies that function.

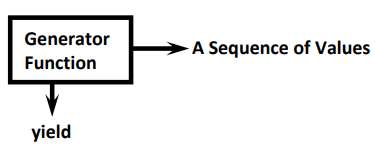


But we want to modify this function to provide different message if name is Sunny. We can do this without touching wish() function by using decorator.

**Decorator Chaining** -We can define multiple decorators for the same function and all these decorators will form Decorator Chaining. Eg: @decor1 @decor def num():

For num() function we are applying 2 decorator functions. First inner decorator will work and then outer decorator.

Generators Generator is a function which is responsible to generate a sequence of values. We can write generator functions just like ordinary functions, but it uses yield keyword to return values.



Advantages of Generator Functions:

1. when compared with class level iterators, generators are very easy to use

2. Improves memory utilization and performance.

3. Generators are best suitable for reading data from large number of large files

4. Generators work great for web scraping and crawling.

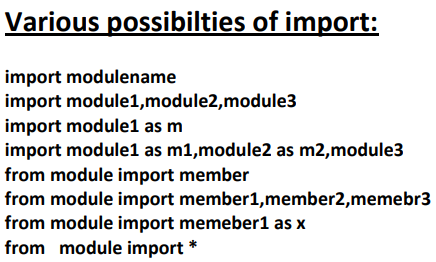
**Modules** A group of functions, variables and classes saved to a file, which is nothing but module. Every Python file (.py) acts as a module.

If we want to use members of module in our program then we should import that module. import modulename We can access members by using module name. modulename.variable ,modulename.function()

**Note: whenever we are using a module in our program, for that module compiled file will be generated and stored in the hard disk permanently.**

Renaming a module at the time of import (module aliasing): Eg: import durgamath as m here durgamath is original module name and m is alias name. We can access members by using alias name m

from ... import:-> We can import particular members of module by using from ... import . The main advantage of this is we can access members directly without using module name.



member aliasing: from durgamath import x as y,add as sum print(y) sum(10,20) Once we defined as alias name,we should use alias name only and we should not use original name

Reloading a Module: By default module will be loaded only once eventhough we are importing multiple multiple times.

**Note: In the above program, everytime updated version of module1 will be available to our program**

Finding members of module by using dir() function: Python provides inbuilt function dir() to list out all members of current module or a specified module. dir() ===>To list out all members of current module dir(moduleName)==>To list out all members of specified module

**Note: For every module at the time of execution Python interpreter will add some special properties automatically for internal use.**

**The Special variable \_\_name\_\_:** For every Python program , a special variable \_\_name\_\_ will be added internally. This variable stores information regarding whether the program is executed as an individual program or as a module. If the program executed as an individual program then the value of this variable is \_\_main\_\_ If the program executed as a module from some other program then the value of this variable is the name of module where it is defined. Hence by using this \_\_name\_\_ variable we can identify whether the program executed directly or as a module

Working with random module: This module defines several functions to generate random numbers. We can use these functions while developing games,in cryptography and to generate random numbers on fly for authentication.

1. random() function: This function always generate some float value between 0 and 1 ( not inclusive) 0<x<1
2. randint() function: To generate random integer beween two given numbers(inclusive)
3. uniform(): It returns random float values between 2 given numbers(not inclusive)
4. randrange([start],stop,[step]) returns a random number from range start<= x < stop start argument is optional and default value is 0 step argument is optional and default value is 1 randrange(10)-->generates a number from 0 to 9 randrange(1,11)-->generates a number from 1 to 10 randrange(1,11,2)-->generates a number from 1,3,5,7,9
5. choice() function: It wont return random number. It will return a random object from the given list or tuple.

**Packages** It is an encapsulation mechanism to group related modules into a single unit. package is nothing but folder or directory which represents collection of Python modules. Any folder or directory contains \_\_init\_\_.py file,is considered as a Python package.This file can be empty. A package can contains sub packages also.

The main advantages of package statement are 1. We can resolve naming conflicts 2. We can identify our components uniquely 3. It improves modularity of the application

Note: Summary diagram of library,packages,modules which contains functions,classes and variables.

**Exception Handling** In any programming language there are 2 types of errors are possible. 1. Syntax Errors 2. Runtime Errors

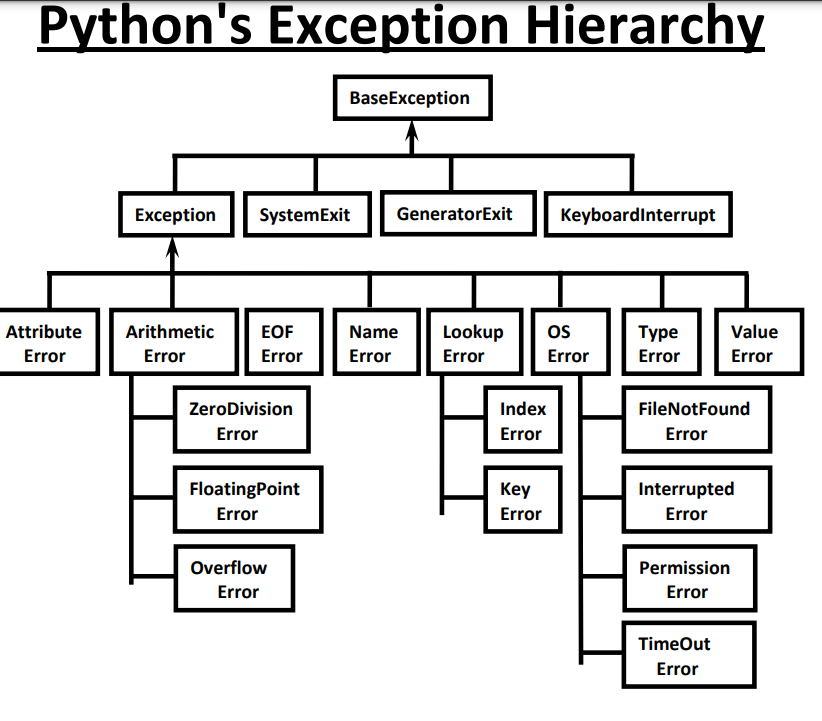
1. Syntax Errors: The errors which occurs because of invalid syntax are called syntax errors. Note: Programmer is responsible to correct these syntax errors. Once all syntax errors are corrected then only program execution will be started.
2. Runtime Errors: Also known as exceptions. While executing the program if something goes wrong because of end user input or programming logic or memory problems etc then we will get Runtime Errors. Note: Exception Handling concept applicable for Runtime Errors but not for syntax errors

What is Exception: An unwanted and unexpected event that disturbs normal flow of program is called exception.

It is highly recommended to handle exceptions. The main objective of exception handling is Graceful Termination of the program(i.e we should not block our resources and we should not miss anything) Exception handling does not mean repairing exception. We have to define alternative way to continue rest of the program normally.

**try:** read data from remote file locating at london except FileNotFoundError: use local file and continue rest of the program normally

**Default Exception Handing in Python:** Every exception in Python is an object. For every exception type the corresponding classes are available. Whevever an exception occurs PVM will create the corresponding exception object and will check for handling code. If handling code is not available then Python interpreter terminates the program abnormally and prints corresponding exception information to the console. The rest of the program won't be executed.



Every Exception in Python is a class. All exception classes are child classes of BaseException.i.e every exception class extends BaseException either directly or indirectly. Hence BaseException acts as root for Python Exception Hierarchy. Most of the times being a programmer we have to concentrate Exception and its child classes.

Customized Exception Handling by using try-except: It is highly recommended to handle exceptions. The code which may raise exception is called risky code and we have to take risky code inside try block. The corresponding handling code we have to take inside except block.

try:

Risky Code

except XXX:

Handling code/Alternative Code

Conclusions:

**1. within the try block if anywhere exception raised then rest of the try block wont be executed eventhough we handled that exception. Hence we have to take only risky code inside try block and length of the try block should be as less as possible.**

**2. In addition to try block,there may be a chance of raising exceptions inside except and finally blocks also.**

**3. If any statement which is not part of try block raises an exception then it is always abnormal termination.**

try with multiple except blocks: The way of handling exception is varied from exception to exception.Hence for every exception type a seperate except block we have to provide. i.e try with multiple except blocks is possible and recommended to use.

If try with multiple except blocks available then based on raised exception the corresponding except block will be executed.

If try with multiple except blocks available then the order of these except blocks is important .Python interpreter will always consider from top to bottom until matched except block identified.

Single except block that can handle multiple exceptions: We can write a single except block that can handle multiple different types of exceptions.

except (Exception1,Exception2,exception3,..): or except (Exception1,Exception2,exception3,..) as msg :

Parenthesis are mandatory and this group of exceptions internally considered as tuple.

Default except block: We can use default except block to handle any type of exceptions. In default except block generally we can print normal error messages.

Syntax: except: statements

**Note: If try with multiple except blocks available then default except block should be last,otherwise we will get SyntaxError.**

**Note: The following are various possible combinations of except blocks 1. except ZeroDivisionError: 1. except ZeroDivisionError as msg: 3. except (ZeroDivisionError,ValueError) : 4. except (ZeroDivisionError,ValueError) as msg: 5. except :**

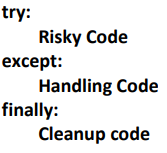
**finally block:**

1. It is not recommended to maintain clean up code(Resource Deallocating Code or Resource Releasing code) inside try block because there is no guarentee for the execution of every statement inside try block always.

2. It is not recommended to maintain clean up code inside except block, because if there is no exception then except block won't be executed.

Hence we required some place to maintain clean up code which should be executed always irrespective of whether exception raised or not raised and whether exception handled or not handled. Such type of best place is nothing but finally block.

Hence the main purpose of finally block is to maintain clean up code.



The speciality of finally block is it will be executed always whether exception raised or not raised and whether exception handled or not handled.

**\* Note: There is only one situation where finally block won't be executed ie whenever we are using os.\_exit(0) function. Whenever we are using os.\_exit(0) function then Python Virtual Machine itself will be shutdown.In this particular case finally won't be executed.**

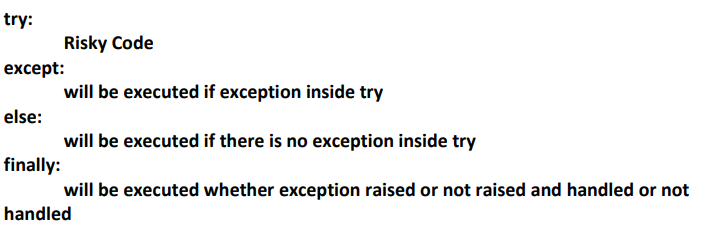
**Note: os.\_exit(0) where 0 represents status code and it indicates normal termination There are multiple status codes are possible.**

General Risky code we have to take inside outer try block and too much risky code we have to take inside inner try block. Inside Inner try block if an exception raised then inner

except block is responsible to handle. If it is unable to handle then outer except block is responsible to handle.

**Note: If the control entered into try block then compulsary finally block will be executed. If the control not entered into try block then finally block won't be executed.**

else block with try-except-finally: We can use else block with try-except-finally blocks. else block will be executed if and only if there are no exceptions inside try block.



Various possible combinations of try-except-else-finally:

1. Whenever we are writing try block, compulsory we should write except or finally block.i.e without except or finally block we cannot write try block.

2. Wheneever we are writing except block, compulsory we should write try block. i.e except without try is always invalid.

3. Whenever we are writing finally block, compulsory we should write try block. i.e finally without try is always invalid.

4. We can write multiple except blocks for the same try,but we cannot write multiple finally blocks for the same try

5. Whenever we are writing else block compulsory except block should be there. i.e without except we cannot write else block. 6. In try-except-else-finally order is important. 7. We can define try-except-else-finally inside try,except,else and finally blocks. i.e nesting of try-except-else-finally is always possible.

**File Handling** As the part of programming requirement, we have to store our data permanently for future purpose. For this requirement we should go for files. Files are very common permanent storage areas to store our data.

Types of Files: There are 2 types of files 1.

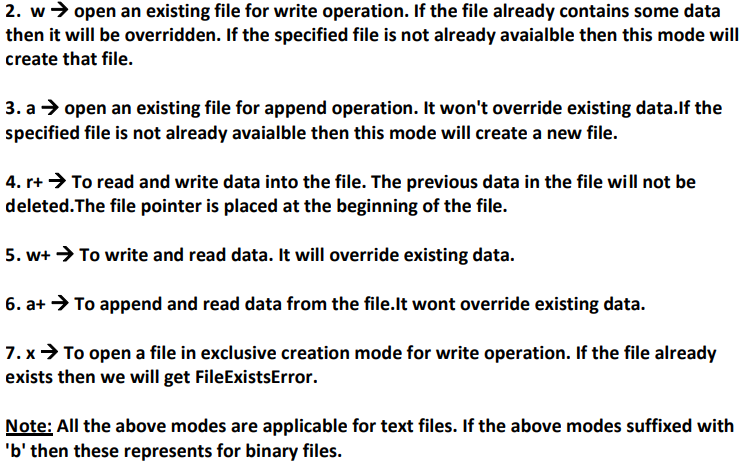
Text Files: Usually we can use text files to store character data eg: abc.txt 2.

Binary Files: Usually we can use binary files to store binary data like images,video files, audio files etc...

Opening a File: Before performing any operation (like read or write) on the file,first we have to open that file.For this we should use Python's inbuilt function open() But at the time of open, we have to specify mode,which represents the purpose of opening file. f = open(filename, mode)

The allowed modes in Python are

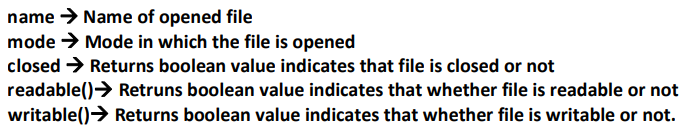
◊1. r ->open an existing file for read operation. The file pointer is positioned at the beginning of the file.If the specified file does not exist then we will get FileNotFoundError.This is default mode.



We are opening abc.txt file for writing data.

Closing a File: After completing our operations on the file,it is highly recommended to close the file. For this we have to use close() function. f.close()

Various properties of File Object: Once we opend a file and we got file object,we can get various details related to that file by using its properties



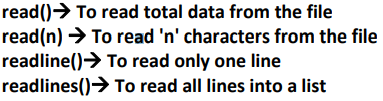
Writing data to text files: We can write character data to the text files by using the following 2 methods. write(str) writelines(list of lines)

Note: In the above program, data present in the file will be overridden everytime if we run the program. Instead of overriding if we want append operation then we should open the file as follows.

f = open("abcd.txt","a")

Note: while writing data by using write() methods, compulsory we have to provide line seperator(\n),otherwise total data should be written to a single line.

Reading Character Data from text files: We can read character data from text file by using the following read methods.

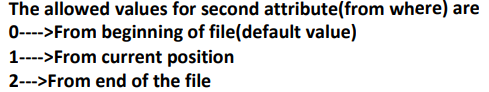


The with statement: The with statement can be used while opening a file.We can use this to group file operation statements within a block. The advantage of with statement is it will take care closing of file,after completing all operations automatically even in the case of exceptions also, and we are not required to close explicitly.

The seek() and tell() methods:

tell(): ==>We can use tell() method to return current position of the cursor(file pointer) from beginning of the file. [ can you plese telll current cursor position] The position(index) of first character in files is zero just like string index.

seek(): We can use seek() method to move cursor(file pointer) to specified location. [Can you please seek the cursor to a particular location] f.seek(offset, fromwhere) offset represents the number of positions



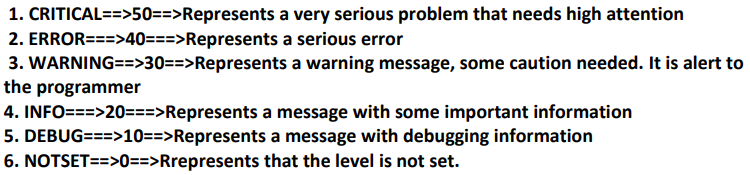
Note: Python 2 supports all 3 values but Python 3 supports only zero.

How to check a particular file exists or not? We can use os library to get information about files in our computer. os module has path sub module,which contains isFile() function to check whether a particular file exists or not? os.path.isfile(fname)

**PYTHON LOGGING** Logging the Exceptions: It is highly recommended to store complete application flow and exceptions information to a file. This process is called logging.

The main advantages of logging are: 1. We can use log files while performing debugging 2. We can provide statistics like number of requests per day etc To implement logging, Python provides one inbuilt module logging.

logging levels: Depending on type of information, logging data is divided according to the following 6 levels in Python.



By default while executing Python program only WARNING and higher level messages will be displayed.

How to implement logging: To perform logging, first we required to create a file to store messages and we have to specify which level messages we have to store. We can do this by using basicConfig() function of logging module. logging.basicConfig(filename='log.txt',level=logging.WARNING)

The above line will create a file log.txt and we can store either WARNING level or higher level messages to that file. After creating log file, we can write messages to that file by using the following methods.

logging.debug(message), logging.info(message) ,logging.warning(message), logging.error(message) ,logging.critical(message)

Note: We can format log messages to include date and time, ip address of the client etc at advanced level.

How to write Python program exceptions to the log file: By using the following function we can write exceptions information to the log file. logging.exception(msg)

**PYTHON DEBUGGING BY USING ASSERTIONS** Debugging Python Program by using assert keyword: The process of identifying and fixing the bug is called debugging. Very common way of debugging is to use print() statement. But the problem with the print() statement is after fixing the bug,compulsory we have to delete the extra added print() statments,otherwise these will be executed at runtime which creates performance problems and disturbs console output. To overcome this problem we should go for assert statement. The main advantage of assert statement over print() statement is after fixing bug we are not required to delete assert statements. Based on our requirement we can enable or disable assert statements. Hence the main purpose of assertions is to perform debugging. Usully we can perform debugging either in development or in test environments but not in production environment. Hence assertions concept is applicable only for dev and test environments but not for production environment.

Types of assert statements: There are 2 types of assert statements 1. Simple Version 2. Augmented Version

1. Simple Version: assert conditional\_expression
2. Augmented Version: assert conditional\_expression,message conditional\_expression will be evaluated and if it is true then the program will be continued. If it is false then the program will be terminated by raising AssertionError. By seeing AssertionError, programmer can analyze the code and can fix the problem.

Exception Handling vs Assertions: Assertions concept can be used to alert programmer to resolve development time errors. Exception Handling can be used to handle runtime errors.

**What is Class:**

⚽ In Python every thing is an object. To create objects we required some Model or Plan or Blue print, which is nothing but class.

⚽ We can write a class to represent properties (attributes) and actions (behaviour) of object.

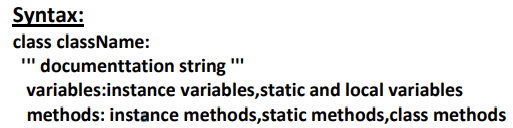
⚽ Properties can be represented by variables

⚽ Actions can be represented by Methods.

⚽ Hence class contains both variables and methods.

How to define a Class?

We can define a class by using class keyword.



Documentation string represents description of the class. Within the class doc string is always optional. We can get doc string by using the following 2 ways. 1) print(classname.\_\_doc\_\_) 2) help(classname)

Within the Python class we can represent data by using variables. There are 3 types of variables are allowed.

1) Instance Variables (Object Level Variables)

2) Static Variables (Class Level Variables)

3) Local variables (Method Level Variables)

Within the Python class, we can represent operations by using methods. The following are various types of allowed methods

1) Instance Methods

2) Static Methods

3) Class Methods

**What is Object: Pysical existence of a class is nothing but object. We can create any number of objects for a class.**

Syntax to Create Object: referencevariable = classname()

Example: s = Student() What is Reference Variable?

The variable which can be used to refer object is called reference variable. By using reference variable, we can access properties and methods of object.

Self Variable: self is the default variable which is always pointing to current object (like this keyword in Java)

By using self we can access instance variables and instance methods of object.

**Note: 1) self should be first parameter inside constructor def \_\_init\_\_(self):**

**2) self should be first parameter inside instance methods def talk(self):**

**Constructor Concept**: ☕ Constructor is a special method in python.

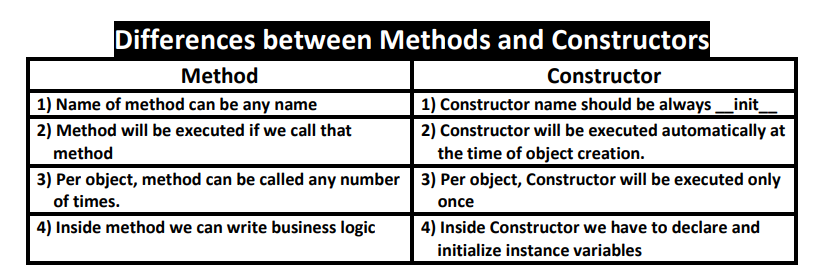
☕ The name of the constructor should be \_\_init\_\_(self)

☕ Constructor will be executed automatically at the time of object creation.

☕ The main purpose of constructor is to declare and initialize instance variables.

☕ Per object constructor will be exeucted only once. ☕ Constructor can take atleast one argument(atleast self)

☕ Constructor is optional and if we are not providing any constructor then python will provide default constructor.



Types of Variables: Inside Python class 3 types of variables are allowed.

1) Instance Variables (Object Level Variables)

2) Static Variables (Class Level Variables)

3) Local variables (Method Level Variables)

1)**Instance Variables:** If the value of a variable is varied from object to object, then such type of variables are called instance variables. For every object a separate copy of instance variables will be created.

**Where we can declare Instance Variables:**

**1) Inside Constructor by using self variable**

**2) Inside Instance Method by using self variable**

**3) Outside of the class by using object reference variable**

1) Inside Constructor by using Self Variable: We can declare instance variables inside a constructor by using self keyword. Once we creates object, automatically these variables will be added to the object.

2) Inside Instance Method by using Self Variable: We can also declare instance variables inside instance method by using self variable. If any instance variable declared inside instance method, that instance variable will be added once we call taht method.

3) Outside of the Class by using Object Reference Variable: We can also add instance variables outside of a class to a particular object.

**How to Access Instance Variables: We can access instance variables with in the class by using self variable and outside of the class by using object reference.**

How to delete Instance Variable from the Object:

1) Within a class we can delete instance variable as follows **del self.variableName**

2) From outside of class we can delete instance variables as follows del **objectreference.variableName**

**Note: The instance variables which are deleted from one object,will not be deleted from other objects.**

If we change the values of instance variables of one object then those changes won't be reflected to the remaining objects, because for every object we are separate copy of instance variables are available.

**2)Static Variables:**

☕ If the value of a variable is not varied from object to object, such type of variables we have to declare with in the class directly but outside of methods. Such types of variables are called Static variables.

☕ For total class only one copy of static variable will be created and shared by all objects of that class.

☕ We can access static variables either by class name or by object reference. But recommended to use class name.

Instance Variable vs Static Variable: Note: In the case of instance variables for every object a seperate copy will be created,but in the case of static variables for total class only one copy will be created and shared by every object of that class.

Various Places to declare Static Variables:

**1) In general we can declare within the class directly but from out side of any method**

**2) Inside constructor by using class name**

**3) Inside instance method by using class name**

**4) Inside classmethod by using either class name or cls variable**

**5) Inside static method by using class name**

How to access Static Variables:

**1) inside constructor: by using either self or classname**

**2) inside instance method: by using either self or classname**

**3) inside class method: by using either cls variable or classname**

**4) inside static method: by using classname**

**5) From outside of class: by using either object reference or classname**

Where we can modify the Value of Static Variable: Anywhere either with in the class or outside of class we can modify by using classname. But inside class method, by using cls variable.

If we change the Value of Static Variable by using either self OR Object Reference Variable: If we change the value of static variable by using either self or object reference variable, then the value of static variable won't be changed, just a new instance variable with that name will be added to that particular object.

How to Delete Static Variables of a Class: 1) We can delete static variables from anywhere by using the following syntax del classname.variablename 2) But inside classmethod we can also use cls variable del cls.variablename

\*\*\*\*Note:

⚽ By using object reference variable/self we can read static variables, but we cannot modify or delete.

⚽ If we are trying to modify, then a new instance variable will be added to that particular object.

⚽ t1.a = 70

⚽ If we are trying to delete then we will get error.

**3)Local Variables:**

⚽ Sometimes to meet temporary requirements of programmer,we can declare variables inside a method directly,such type of variables are called local variable or temporary variables.

⚽ Local variables will be created at the time of method execution and destroyed once method completes.

⚽ Local variables of a method cannot be accessed from outside of method

**Types of Methods: Inside Python class 3 types of methods are allowed 1) Instance Methods 2) Class Methods 3) Static Methods**

**1)Instance Methods:**

⚽ Inside method implementation if we are using instance variables then such type of methods are called instance methods.

⚽ Inside instance method declaration, we have to pass self variable. def m1(self):

⚽ By using self variable inside method we can able to access instance variables.

⚽ Within the class we can call instance method by using self variable and from outside of the class we can call by using object reference

Setter and Getter Methods: We can set and get the values of instance variables by using getter and setter methods.

Setter Method: setter methods can be used to set values to the instance variables. setter methods also known as mutator methods.

Syntax: def setVariable(self,variable):

self.variable=variable

Getter Method: Getter methods can be used to get values of the instance variables. Getter methods also known as accessor methods.

Syntax: def getVariable(self):

return self.variable

**3)Static Methods:**

⚽ In general these methods are general utility methods.

⚽ Inside these methods we won't use any instance or class variables.

⚽ Here we won't provide self or cls arguments at the time of declaration.

⚽ We can declare static method explicitly by using @staticmethod decorator

⚽ We can access static methods by using classname or object reference

Note: In general we can use only instance and static methods.Inside static method we can

access class level variables by using class name.

**2)Class Methods:**

⚽ Inside method implementation if we are using only class variables (static variables), then such type of methods we should declare as class method.

⚽ We can declare class method explicitly by using @classmethod decorator.

⚽ For class method we should provide cls variable at the time of declaration

⚽ We can call classmethod by using classname or object reference variable.

Class methods are most rarely used methods in python.

Inner Classes Sometimes we can declare a class inside another class, such type of classes are called inner classes. Without existing one type of object if there is no chance of existing another type of object, then we should go for inner classes.

**Garbage Collection**

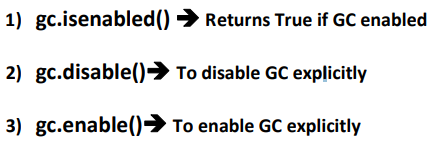
⚽ In old languages like C++, programmer is responsible for both creation and destruction of objects.Usually programmer taking very much care while creating object, but nEglecting destruction of useless objects. Because of his nEglectance, total memory can be filled with useless objects which creates memory problems and total application will be down with Out of memory error.

⚽ But in Python, We have some assistant which is always running in the background to destroy useless objects.Because this assistant the chance of failing Python program with memory problems is very less. This assistant is nothing but Garbage Collector.

⚽ Hence the main objective of Garbage Collector is to destroy useless objects.

⚽ If an object does not have any reference variable then that object eligible for Garbage Collection.

How to enable and disable Garbage Collector in our Program: By default Gargbage collector is enabled, but we can disable based on our requirement. In this context we can use the following functions of gc module.



**Destructors:**

⚽ Destructor is a special method and the name should be \_\_del\_\_

⚽ Just before destroying an object Garbage Collector always calls destructor to perform clean up activities (Resource deallocation activities like close database connection etc).

⚽ Once destructor execution completed then Garbage Collector automatically destroys that object.

Note: The job of destructor is not to destroy object and it is just to perform clean up activities.

Note: If the object does not contain any reference variable then only it is eligible fo GC. ie if the reference count is zero then only object eligible for GC.

How to find the Number of References of an Object: sys module contains getrefcount() function for this purpose. sys.getrefcount (objectreference)

Note: For every object, Python internally maintains one default reference variable self.

**Inheritance**

Using Members of One Class inside Another Class: We can use members of one class inside another class by using the following ways 1) By Composition (Has-A Relationship) 2) By Inheritance (IS-A Relationship)

1. By Composition (Has-A Relationship): By using Class Name or by creating object we can access members of one class inside another class is nothing but composition (Has-A Relationship). The main advantage of Has-A Relationship is Code Reusability.
2. ) By Inheritance (IS-A Relationship): What ever variables, methods and constructors available in the parent class by default available to the child classes and we are not required to rewrite. Hence the main advantage of inheritance is Code Reusability and we can extend existing functionality with some more extra functionality.

Syntax: class childclass(parentclass)

In the above example Parent class contains 10 methods and these methods automatically available to the child class and we are not required to rewrite those methods(Code Reusability) Hence child class contains 15 methods.

Note: What ever members present in Parent class are by default available to the child class through inheritance.

What ever methods present in Parent class are automatically available to the child class and hence on the child class reference we can call both parent class methods and child class methods.

IS-A vs HAS-A Relationship: If we want to extend existing functionality with some more extra functionality then we• should go for IS-A Relationship. If we dont want to extend and just we have to use existing functionality then we• should go for HAS-A Relationship. Eg: Employee class extends Person class Functionality But Employee class just uses Car• functionality but not extending

Composition vs Aggregation:

Composition: Without existing container object if there is no chance of existing contained object then the container and contained objects are strongly associated and that strong association is nothing but Composition.

Aggregation: Without existing container object if there is a chance of existing contained object then the container and contained objects are weakly associated and that weak association is nothing but Aggregation.

Conclusion: The relation between object and its instance variables is always Composition where as the relation between object and static variables is Aggregation.

Note: In the above example when ever we are creating child class object both parent and child class constructors got executed to perform initialization of child object.

Types of Inheritance:

1)Single Inheritance: The concept of inheriting the properties from one class to another class is known as single inheritance.

2)Multi Level Inheritance: The concept of inheriting the properties from multiple classes to single class with the concept of one after another is known as multilevel inheritance.

3)Hierarchical Inheritance: The concept of inheriting properties from one class into multiple classes which are present at same level is known as Hierarchical Inheritance

4)Multiple Inheritance: The concept of inheriting the properties from multiple classes into a single class at a time, is known as multiple inheritance.

5)Hybrid Inheritance: Combination of Single, Multi level, multiple and Hierarchical inheritance is known as Hybrid Inheritance.

6)Cyclic Inheritance: The concept of inheriting properties from one class to another class in cyclic way, is called Cyclic inheritance.Python won't support for Cyclic Inheritance of course it is really not required.

Method Resolution Order (MRO):

In Hybrid Inheritance the method resolution order is decided based on MRO algorithm. T

his algorithm is also known as C3 algorithm.Samuele Pedroni proposed this algorithm. It follows DLR (Depth First Left to Right) i.e Child will get more priority than Parent. Left Parent will get more priority than Right Parent. MRO(X) = X+Merge(MRO(P1),MRO(P2),...,ParentList)

Head Element vs Tail Terminology: Assume C1,C2,C3,...are classes.• In the list: C1C2C3C4C5....• C1 is considered as Head Element and remaining is considered as Tail.• How to find Merge: Take the head of first list• If the head is not in the tail part of any other list, then add this head to the result and• remove it from the lists in the merge. If the head is present in the tail part of any other list, then consider the head element• of the next list and continue the same process. Note: We can find MRO of any class by using mro() function. print(ClassName.mro())

**super() Method:** super() is a built-in method which is useful to call the super class constructors,variables and methods from the child class.

How to Call Method of a Particular Super Class: We can use the following approaches

1) super(D, self).m1() It will call m1() method of super class of D.

2) A.m1(self) It will call A class m1() method

**Various Important Points about super():**

**Case-1: From child class we are not allowed to access parent class instance variables by using super(), Compulsory we should use self only. But we can access parent class static variables by using super().**

**Case-2: From child class constructor and instance method, we can access parent class instance method, static method and class method by using super()**

**Case-3: From child class, class method we cannot access parent class instance methods and constructors by using super() directly(but indirectly possible). But we can access parent class static and class methods.**

**Case-4: In child class static method we are not allowed to use super() generally (But in special way we can use)**

**POLYMORPHISM**

poly means many. Morphs means forms. Polymorphism means 'Many Forms'.

Related to Polymorphism the following 4 topics are important

1) Duck Typing Philosophy of Python

2) Overloading **1) Operator Overloading 2) Method Overloading 3) Constructor Overloading**

3) Overriding **1) Method Overriding 2) Constructor Overriding**

**1) Duck Typing Philosophy of Python:** In Python we cannot specify the type explicitly. Based on provided value at runtime the type will be considered automatically. Hence Python is considered as Dynamically Typed Programming Language

What is the Type of obj? We cannot decide at the Beginning. At Runtime we can Pass any Type. Then how we can decide the Type? At runtime if 'it walks like a duck and talks like a duck,it must be duck'. Python follows this principle. This is called Duck Typing Philosophy of Python.

**Overloading ->**We can use same operator or methods for different purposes

There are 3 types of Overloading **1) Operator Overloading 2) Method Overloading 3) Constructor Overloading**

**1) Operator Overloading -**

⚽ We can overload + operator to work with Book objects also. i.e Python supports Operator Overloading.

⚽ For every operator Magic Methods are available. To overload any operator we have to override that Method in our class.

⚽ Internally + operator is implemented by using \_\_add\_\_() method.This method is called magic method for + operator. We have to override this method in our class.

**Method Overloading:** If 2 methods having same name but different type of arguments then those methods are said to be overloaded methods.

But in Python Method overloading is not possible. If we are trying to declare multiple methods with same name and different number of arguments then Python will always consider only last method.

How we can handle Overloaded Method Requirements in Python: Most of the times, if method with variable number of arguments required then we can handle with default arguments or with variable number of argument methods.

**Constructor Overloading:**

⚽ Constructor overloading is not possible in Python.

⚽ If we define multiple constructors then the last constructor will be considered.

**Overriding**

**Method Overriding**

⚽ What ever members available in the parent class are bydefault available to the child class through inheritance. If the child class not satisfied with parent class implementation then child class is allowed to redefine that method in the child class based on its requirement. This concept is called overriding.

⚽ Overriding concept applicable for both methods and constructors

**Abstract Method:**

Sometimes we don't know about implementation, still we can declare a method. Such types of methods are called abstract methods.i.e abstract method has only declaration but not implementation.

In python we can declare abstract method by using @abstractmethod decorator as follows.

@abstractmethod

def m1(self): pass

@abstractmethod decorator present in abc module. Hence compulsory we should• import abc module,otherwise we will get error. 

**Abstract class:** Some times implementation of a class is not complete,such type of partially implementation classes are called abstract classes. Every abstract class in Python should be derived from ABC class which is present in abc module.

Conclusion: If a class contains atleast one abstract method and if we are extending ABC class then instantiation is not possible. "abstract class with abstract method instantiation is not possible" Parent class abstract methods should be implemented in the child classes. Otherwise we cannot instantiate child class.If we are not creating child class object then we won't get any error.

Note: If we are extending abstract class and does not override its abstract method then child class is also abstract and instantiation is not possible.

**Interfaces** In Python: In general if an abstract class contains only abstract methods such type of abstract class is considered as interface.

Note: The inbuilt function globals()[str] converts the string 'str' into a class name and returns the classname.

Concreate class vs Abstract Class vs Inteface:

1) If we dont know anything about implementation just we have requirement specification then we should go for interface.

2) If we are talking about implementation but not completely then we should go for abstract class. (partially implemented class).

3) If we are talking about implementation completely and ready to provide service then we should go for concrete class.

Public, Protected and Private Attributes: By default every attribute is public. We can access from anywhere either within the class or from outside of the class.

Protected attributes can be accessed within the class anywhere but from outside of the class only in child classes. We can specify an attribute as protected by prefexing with \_ symbol.

But is is just convention and in reality does not exists protected attributes. private attributes can be accessed only within the class.i.e from outside of the class we cannot access. We can declare a variable as private explicitly by prefexing with 2 underscore symbols.

syntax: \_\_variablename=value

How to Access Private Variables from Outside of the Class: We cannot access private variables directly from outside of the class. But we can access indirectly as follows objectreference.\_classname\_\_variablename

\_\_str\_\_() method: Whenever we are printing any object reference internally \_\_str\_\_() method will be called which is returns string in the following format. To return meaningful string representation we have to override \_\_str\_\_() method.