Contents

[1 Installation Python 3](#_Toc93709095)

[Step 1 3](#_Toc93709096)

[Step 2 3](#_Toc93709097)

[Step 3 3](#_Toc93709098)

[1 basic command 3](#_Toc93709099)

[2 take user input as string 4](#_Toc93709100)

[String slicing 4](#_Toc93709101)

[3 set 4](#_Toc93709102)

[4 if elif and loops 5](#_Toc93709103)

[5 list 6](#_Toc93709104)

[6 operators 6](#_Toc93709105)

[7 function 7](#_Toc93709106)

[8 map filter and reduce 7](#_Toc93709107)

[9 String formatting 8](#_Toc93709108)

[10 Decorator 8](#_Toc93709109)

[2 OOP in python 9](#_Toc93709110)

[1 Class 9](#_Toc93709111)

[2 Constructor 9](#_Toc93709112)

[3 exception handling 9](#_Toc93709113)

[Raise exception 10](file:///C:\Users\universal\Desktop\destop\python\intro.docx#_Toc93709114)

[4 Class methods and Static methods 10](#_Toc93709115)

[5 Abstraction & Encapsulation 11](#_Toc93709116)

[6 Inharitance 11](#_Toc93709117)

[Single Inheritance 12](file:///C:\Users\universal\Desktop\destop\python\intro.docx#_Toc93709118)

[Multiple Inheritance 12](file:///C:\Users\universal\Desktop\destop\python\intro.docx#_Toc93709119)

[MultiLevel Inheritance 12](file:///C:\Users\universal\Desktop\destop\python\intro.docx#_Toc93709120)

[Public Private and Protected Acess Modifier 13](#_Toc93709121)

[7 Polymorphism 13](#_Toc93709122)

[Super class and overriding 13](#_Toc93709123)

[Operator overloading 14](file:///C:\Users\universal\Desktop\destop\python\intro.docx#_Toc93709124)

[8 Operator Overloading & Dunder Methods 14](#_Toc93709125)

[Dunder methods 14](file:///C:\Users\universal\Desktop\destop\python\intro.docx#_Toc93709126)

[\_\_str\_\_ and \_\_repr\_\_ functions 14](file:///C:\Users\universal\Desktop\destop\python\intro.docx#_Toc93709127)

[9 Abstract Base Class 14](#_Toc93709128)

[10 Object Introspection 15](#_Toc93709129)

[3 external memory management 16](#_Toc93709130)

[1 to deal with external files 16](#_Toc93709131)

[2 import 16](#_Toc93709132)

[3 Cache creation 17](#_Toc93709133)

[4 Modules 18](#_Toc93709134)

[Inbuild module 18](#_Toc93709135)

[1 Time 18](#_Toc93709136)

[2 random 18](#_Toc93709137)

[3 gingerIt 18](#_Toc93709138)

[4 Virtual Environment 18](#_Toc93709139)

[5 Os Module 19](#_Toc93709140)

[6 Requests Module 20](#_Toc93709141)

[7 JSON module 20](#_Toc93709142)

[8 pickle module 21](#_Toc93709143)

[9 Pyinstaller 21](#_Toc93709144)

[10 Command Line utility 22](#_Toc93709145)

[11 Python Package 22](#_Toc93709146)

[5 Frameworks 24](#_Toc93709147)

[1 webdevelopment 24](#_Toc93709148)

[2 GUI(Graphic User Interface) 24](#_Toc93709149)

[3 Machine Learning and data science 24](#_Toc93709150)

[6 Miscellaneous 26](#_Toc93709151)

[\_\_iter\_\_() and \_\_next\_\_() method 26](file:///C:\Users\universal\Desktop\destop\python\intro.docx#_Toc93709152)

[1 Generator 26](#_Toc93709153)

[2 Python Comprehensions 27](#_Toc93709154)

[3 Coroutines 27](#_Toc93709155)

[4 Regular Expressions 27](#_Toc93709156)

[5 Python 2 to python 3 28](#_Toc93709157)

# 1 Installation Python

Step 1 pip /\* Pip is a package manager for Python, i.e., pip command can be used to download any external module in Python. It is something that helps us to get code written by someone else. \*/ install /\*module name here\*/ flask the above command is used to install flask from internet which is used to design website\*/

Step 2 python /\*and run command\*/

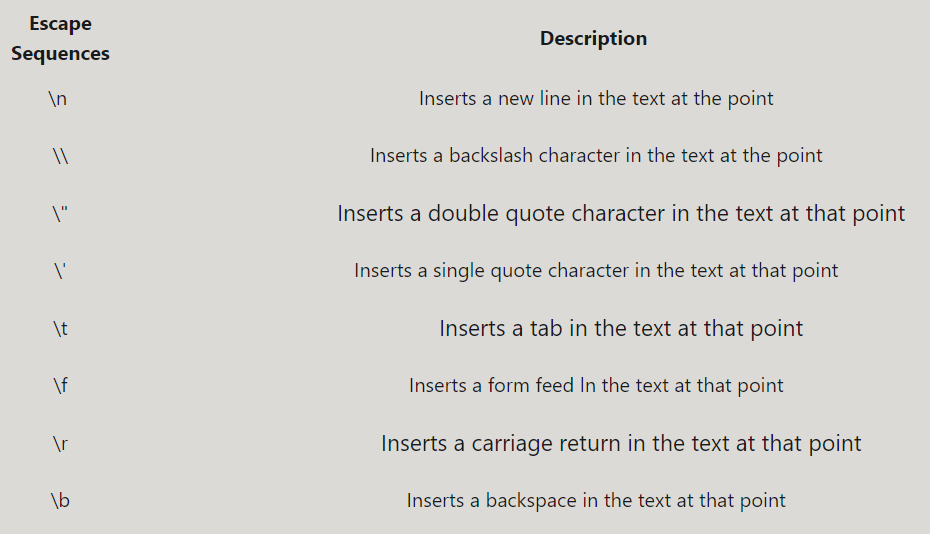
Step 3 import flask

## 1 basic command

Print(“ poppy ”) /\*as sout println in java\*/ /\*to make all code in one line write\*/ print(“poppy”,end= “ ”) /\*type python in cmd and use it directly for basis fns\*/ /\*eg calculator\*/ #poppy /\*# used for comment ie means poppy was commented out\*/ “““ poppy aao””” /\*triple colon like this used to comment out multiple lines\*/

var1= “poppy” or 1 or1.1 /\*store directly as a variable\*/ /\* to check type of float understood by python type\*/ print(type(var1) /\*casting is allowed in python eg\*/ int(“66”) /\*it will convert string to int similarly float() and str() for string\*/

/\*in python string used as block hence it can be added or subtracted\*/



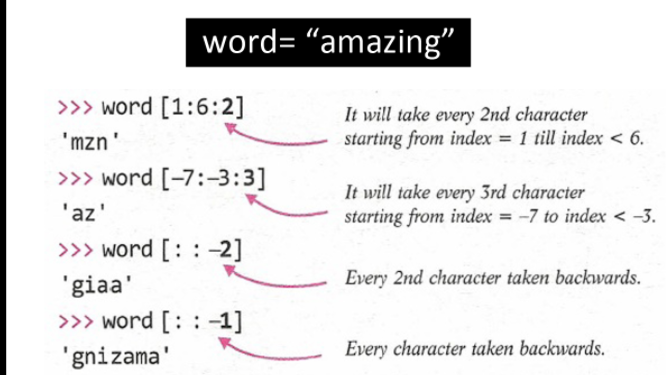
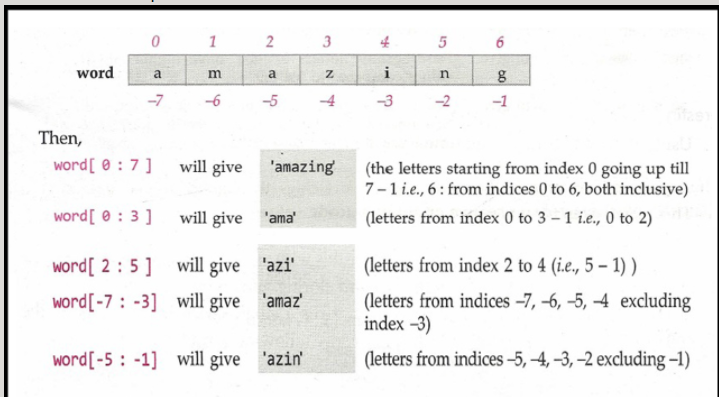
## 2 take user input as string

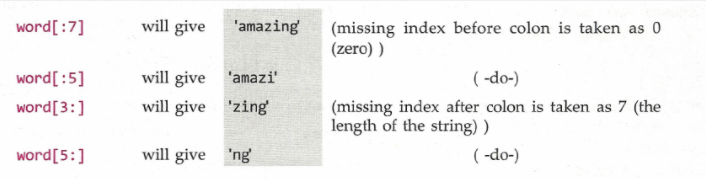
Input(); / \*take input from user in form of string it had to be converted into int of float in order to use it as for other operation to take specific int input type\*/ a=int(input()) /\*to print a it must be converted back to string by\*/ str(a)

Input(“enter a”) /\*it will give syntax guess the no\*/ /\*given number will be stored in reference\*/

/\*string.endswith(): eg\*/ print(str.endswith("ho")) /\*It returns True or False.\*/ /\*string.count(): eg count of in str\*/ str.count("o") /\*return int which is count of o in str\*/ /\*string.capitalize(): it will capitalized first letter of the string\*/ /\*string.upper(): It returns the copy of the string converted to the uppercase.\*/ /\*string.lower(): above for lower case.\*/ /\*string.find(): I will return first occurance of word or char stored in string\*/ /\*string.replace(“old\_word”, “new\_word”): This function replaces the old word or character with a new word or character from the entire string.\*/ /\*empty input() will execute other command only after pressing some key\*/

### String slicing



a is list if we write c=a[:] /\*all variable of b copied in c\*/

## 3 set

s=set(); /\*it will create a set of a used in maths it will only take unique values it can take values from list or using {} waale brackets\*/ s.add(2) /\*it will add unique 2 in set s\*/ s.remove(2) /\*it will remove 2 from set\*/

## 4 if elif and loops

str(a).isnumeric() /\*to filter numeric object from list\*/

for a1 in a: print(a1) /\*it will print all object in side a\*/ /\*above for 2 obj list\*/ for a,b in a1: print(a+“and”+b) /\*else\*/ for a1 in a: for b1in a1: /\*to print 2d list\*/

while i<35: fn i +=35

if ip==5 and/or ip>=3:  
 print("equal")  
elif ip>5:  
 print("greater")  
else:  
 print("poppy")

/\*it is the syntax of if statement use and/or for not use ! \*/

/\*for list use in or not in eg

if 15 not/ in l1: where l1 be the list

for i in range(45): /\*to print for loop for i=1 to 45\*/

/\*short form to print list with a object as between part and l be list and b we new object storing that fn\*/ b=a.join(l)

is means /\*2 reference in stack refer same object in heap\*/

== means /\*2 objects have same value\*/

Break means break the loop and continue means jb wo condn aae uske baad loop ko chalao

while(1) or while(True) it is an infinite loop

print("true") if(a>50) else print("false") /\*short form of if else\*/

for i, item in enumerate(l1): if i%2==0: code /\*it is a another representation to traverse i in list and put condn as per req\*/

/\*when for loop used with else and loop break than else condn will get printed\*/

## 5 list

list1 = ['harry', 'ram', 'Aakash', 'shyam', 5, 4.85] /\*just like js python can take list of string float and int together\*/

numbers = [2, 7, 9, 11, 3] # numbers.remove(9) /\*it will remove 9 from numbers list\*/ # numbers.pop() /\*it will pop out last value from list here 3\*/ # numbers.sort() /\*it will sort list\*/ # numbers.reverse() /\*it will reverse list\*/ # numbers.append(1) /\*it will add number one at last 1\*/ # numbers.insert(2, 67) /\*it will insert 67 at last index\*/

# numbers[1] = 98 /\*if such fn is used than number at first index will be replaced by 98\*/ /\*to avoid such thing we will use tupple\*/ eg tp=(1,2,3,4,56) /\*here changing original no will give error\*/ /\*to create const we will use\*/ # tp = (1,)

/\*To swap tu number we can do it by this way\*/ a= 1 b = 8 a, b = b,a /\*a,b=b,a it will swap a with b\*/ /\*isme ek baar me ek fn run krta he wo be outside print\*/ /\*list cannot be converted into string but it character could be\*/

len(n) /\*to print length of n\*/

/\*dictionary collection of words created by\*/ a={} /\*syntax of dicionary\*/

d2 = {"Harry":"Burger", "Rohan":"Fish", "SkillF":"Roti", "Shubham":{"B":"maggie", "L":"roti", "D":"Chicken"}}

/\*above will create dictionary with special fn like object in js\*/ print(d2[“Harry”] /\*it will print burger similarly Shubham will print whole dictionary so for shubhem to print Chicken\*/ print(d2[“Shubham”][“D”])

d2[420] = "Kebabs" /\*It will add 420: “Kebab” in the dictionary d2\*/ # del d2[420] /\*it will delete 420 object from d2\*/ # d3 = d2.copy() /\*it will make copy of dictionary at reference d3 if changes made in d3 copy will not reflected in d2\*/ /\*but in case of\*/ d2=d3 /\*as both pointers points at same object changes made in d3 will also be reflected in d2\*/

# d2.update({"Leena":"Toffee"}) /\*it will update item of object Leena into d2\*/ # print(d2.keys()) /\*it will print keys eg Shubham\*/ # print(d2.items()) /\*it will prints items eg Toffee\*/

## 6 operators

Athematic operators / will give value in float but // give values in integer less than actual value\*/ /\* \* will give multiply but \*\* will give fn of power eg 2^6 could be written as\*/ 2\*\*6

logical operator /\***is** could be used as **==** and **is not** could be used as **!=\*/**

## 7 function

result = lambda n1, n2, n3: n1 + n2 + n3; /\*the above is lambda fn short form of single form fn\*/ /\*result is fn name\*/ /\*n1, n2,n3 are input of lambda\*/ /\*fn work after : which **return** will fn will give (n1+n2+n3) as return value\*/

/\*array and list in python\*/ /\* use **\*args** to generate list and use **\*\*kwargs** to use dictionary in fn as input note fn will also take similar argument while using it in main programme\*/

def funargs(n, \*args, \*\*kwargs): print(n) for i in args: print(i) for k, v in kwargs.i (): print(f"{k} is a {v}") har = ["x", "y", "z”] normal = "u" kw = {"a":"1", "b":"2"} funargs(normal, \*har, \*\*kw)

def function2(a, b):

"""This is a function which will calculate average of two numbers

this function doesnt work for three numbers""" /\*w/o return act as void fn\*/ /\* else like non void fn\*/

average = (a+b)/2 # print(average) return average /\*return is optional it not present reference will give value none\*/

print(function2.\_\_doc\_\_) /\*it will print Docstrings of function 2 which was taken 2 variable and output as variable \*/

/\*note docstring was used to print all needed information for function in order to print use it more effectively\*/

/\***Local variable** are defined inside fn whose original variable is defined in main body\*/ /\* the variable inside main body is **global variable**\*/ /\*local variable can be changed inside fn as per its convenience but change will not be reflected in global variable\*/ /\*to solve problem above problem we will declare \*/ global x /\*inside fn where x is global variable\*/ /\*hence changes made inside fn for x will be reflected for outside fn for global variable x\*/

## 8 map filter and reduce

map(): "A map function executes certain instructions or functionality provided to it on every item of an iterable

items = [1, 2, 3, 4, 5] a=list(map((lambda x: x \*\*3), items)) /\*first part could be int parsing and in form of lambda x:x(i)\*/ /\*list is essential to pars list in map, first part is fn and second part is list need to be iterated\*/ print(a) #Output: [1, 8, 27, 64, 125]

filter():- **"A filter function in Python tests a specific user-defined condition for a function and returns an iterable for the elements and values that satisfy the condition or, in other words, return true."**

filter(function, iterable)a = [1,2,3,4,5] b = [2,5,0,7,3] c= list(filter(lambda x: x in a, b)) print(c) # prints out [2, 5, 3] /\*it give output in se when argument in filter is true\*/

reduce(): **"Reduce functions apply a function to every item of an iterable and gives back a single value as a resultant". /\*note to use it functools must be used to import reduce package\*/**

from functools import reduce a=reduce( (lambda x, y: x \* y), [1, 2, 3, 4] ) print(a) #Output: 24

## 9 String formatting

# 10 Decorator

**Decorator,** it is modification to the external layer of function, w/o changing its structure by adding same fn by itself. This is also known as **metaprogramming**because a part of the program tries to modify and add functionality to another part of the program at compile time.

A **wrapper**is a function that provides a wrap-around another function. and host fn act as argument for wrapper fn. The purpose of the wrapper function is to assist us. It is used when same command in fn had to run many time along with decorator.

* We can pass our function to the decorator as an argument, thus defining a function and passing it to our decorator.
* We can simply use the @ symbol before the function we’d like to decorate.

/\*@fn1\*/ def fn2(): a = f1(f2) a() /\*to print fn as decorator by method1 \*/ /\*second way is used after commenting a and uncommenting @fn1 note fn1 act as wrapper fn of fn2\*/

/\*First by %operator\*/ name="Jack” n="%s My name is %s” %name print(n) Output: "My name is Jack." /\*not good for large string\*/

/\*Second by Tuple ()\*/ s=”%s is in class %d”%(name,class) print(s) Output: Jack is in class 5 /\*not good for large string\*/

/\*Third by String Formatting (str.format)\*/ str = "This article is written in {} " print (str.format("Python")) Output: This article is written in Python. /\*  braces { } used in place of % better for multiscale operation

/\*Fourth Using f-Strings ( f )\*/ str1="Python” str2="Programming” print(f"Welcome to our {str1}{str2} tutorial”)

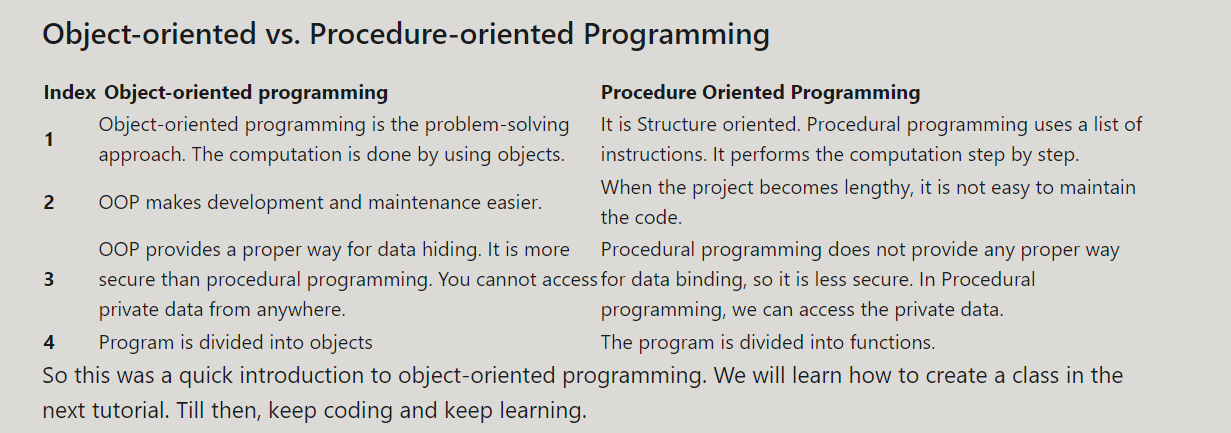
Output: Welcome to our Python Programming tutorial. /\*most recommended as it is modification of third with dynamic usuage\*/

# OOP in python

SubPart: 1 Instances 2 Constructor 3 Methods 4 Abstraction 5 Inheritance

one class can be easily accessible by another, and we can also restrict the access of a class so other classes can not use its functions. This concept comes in handy while working on bigger projects.

## 1 Class



## 2 Constructor

class a: pass /\*it will help to create class\*/ /\*note we will write pass if class is empty\*/ a1=a() a2=a() /\*a1 and a2 are objects which was created for class a\*/ /\*we can set object in variable in objects of classes eg\*/ a1.b= “poppy” /\*if we print a1.b it will give string poppy\*/ /\*variable could be string list dictionary\*/ /\*if I mention c=4 in class a\*/ class a: c=4 pass x=a() /\*c variable act as default variable for all objects\*/ /\*same object print x.c it will give 4 as o/p\*/ x.c=5 /\*it will create new instance for object c which will change value for c variable for x only while other value eg \*/ a1.c /\*it will give 4 as o/p\*/ a1.\_\_Dict\_\_ /\*it will print all non default variables and instances which belong to object a1\*/ /\*but if I change default value eg\*/ a.c=5 /\*change in class variable will be reflected in all other non instances objects variables (c) \*/

def \_\_init\_\_(self, b1, b2, b3): self.b1 = b1 self.b2 = b2 self.b3 = b3 /\*\_\_init\_\_ inside class will create constructor of class with b1 b2 b3 as variable taken as ip\*/ /\*note self-had to be taken separately as this in java to take input from object and replace it with constructor by self had to essentially taken as an arguments\*/

## exception handling

try: print("The sum of these two numbers is", int(num1)+int(num2))

except Exception as e: print(e) /\*Note in case of Exception any error can be printed\*/

/\*in order to avoid error this method is used and code is run forward without breaking\*/

/\*try will first run the programme if error comm using exception which stored as e will be and function will be run for e and further code will not get disrupt\*/

if x < 10:

raise ValueError('x <10')

else: /\*No Exception. Run this code \*/ finally: /\*Always run this code\*/

### Raise exception

if test\_condition: raise EXCEPTION\_CLASS\_NAME

A few of these exceptions include:

**KeyError:** Raised when a mapping key is not found in the set of existing keys.

**ValueError**: Raised when a function receives an argument with the right type but an inappropriate value.

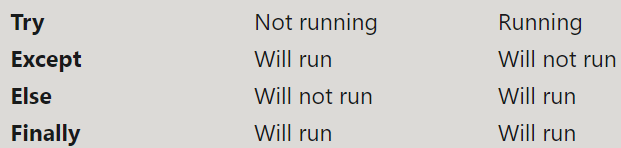
**EOFError (End Of File Error):** Raised when the input() function hits an end-of-file condition without reading any data.

**ImportError:** Raised when the import statement has trouble trying to load a module.

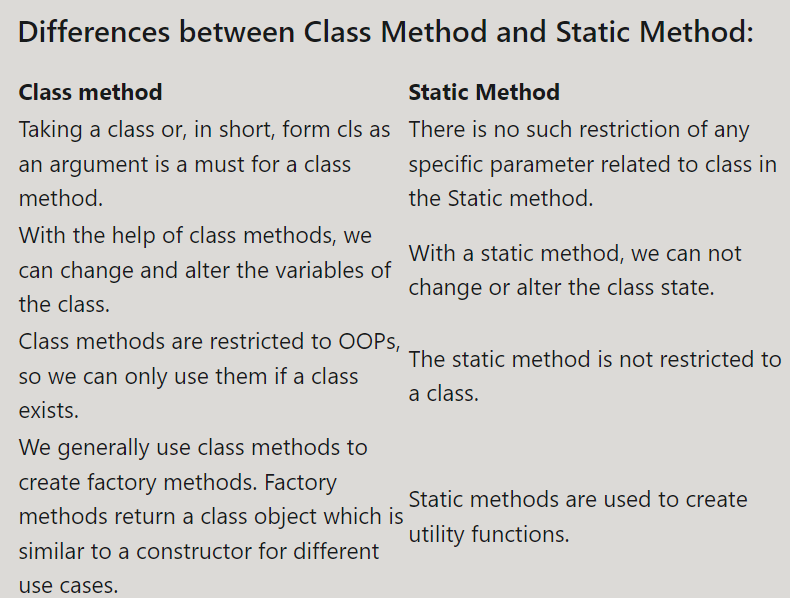
**NameError:**Raised when a local or global name is not found.

**ZeroDivisionError:** Raised when the second argument of a division is zero.

[more](https://docs.python.org/3/library/exceptions.html)



## Class methods and Static methods



* 1. **Class Methods**

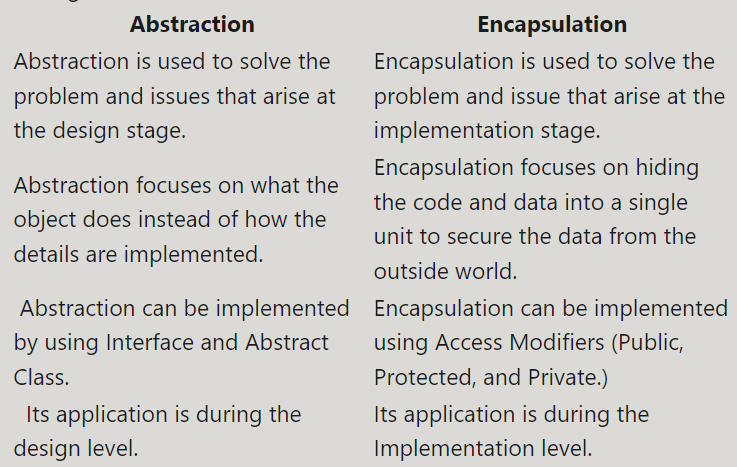
class a: @classmethod def b (cls, arg1, arg2, ...): cls.c=arg1 /\* cls is short form to represent class as whole similar to self for constructor and @classmethod is decorator to override class\*/ /\*the above method was called class method which can override both class fn and instances eg\*/ a1=a() a1.b(6) will change value of b variable directly\*/

@classmethod def b1(cls,string): return cls(\*string.split(" ")) a1=a.b1(tum ek pogli aurat ho) /\*above class method will split string to list as per define by previous constructors and variable could be taken as required\*/ /\* (\*) act as argument one line for above code\*/

**4.2 Static Methods**

@staticmethod def d(string): print("oye" + string) /\*the above fn act as static fn which will act similar to normal fn defined in a it will not take self as arguments and will increase code efficiency\*/ /\*for executing above fn we used it as class fn not object fn \*/ a.d(“poppy”) /\*it will print oye poppy but if a1 is used it will through an error\*/

## 5 Abstraction & Encapsulation



***Abstraction*** refers to hiding unnecessary details to focus on the whole product instead of parts of the project separately. It is a mechanism that represents the important features without including implementation details.

***Encapsulation*** means hiding under layers. ie, the internal representation of an object is generally hidden from the outside to secure the data. It improves the maintainability of an application and helps the developers to organize the code better.

## 6 Inharitance

/\*Hence python was more preferred for such cases over other language\*/

/\*In public, all the functions, variables, methods can be used publicly. Meaning, every other class can access them easily without any restriction.\*/ /\*it was written normally eg\*/ p=3

/\*In the case of a protected class, its members and functions can only be accessed by the classes derived from it, i.e., its child class or classes. No other environment is permitted to access it.\*/ /\*it was written with single underscore and can only used by host and inherent classes eg\*/ \_p=3

/\*In the case of private access modifiers, the variables and functions can only be accessed within the class.\*/ /\*it is used with double underscore and can only accessed by parent class only eg\*/ \_\_p=3

### Single Inheritance

class p: /\*parent class\*/ class c(p): /\*child class name c who had inherent its properties from parent p\*/ /\*note child can take all properties of parent and use its own property which was not available to parent\*/

Advantages of Inheritance: 1. It increases the code’s reusability as we do not have to copy the same code again to our new class. 2. It makes the program more efficient. 3. We can add more features to our already built class without modifying it or changing its functionality. 4. It provides a representation of real-world relationships. /\*note python allow multiple inheritance which was not allowed in java\*/

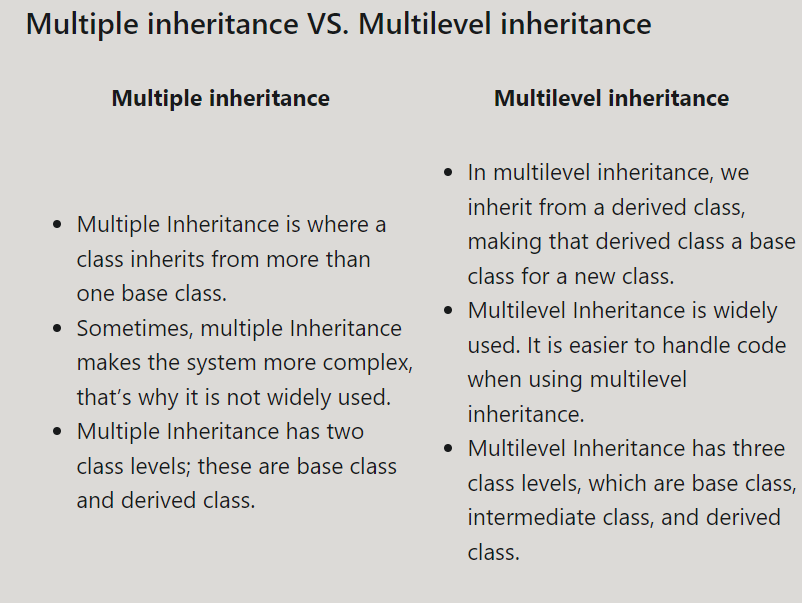
### Multiple Inheritance

class p1: class p2: class c(p1 , p2): /\*class child c will inherent properties from both parent in p1 and p2 and so on with hagiarchy of p1 and p2 and so on\*/ /\*in such case class c had to take data from constructor of p1 to initialize it or from child itself\*/ /\*if common variable object or fn was defined for multiple inheritance \*/ /\* first child object will get priority than of p1 than p2 and so on\*/

### MultiLevel Inheritance

class a: class b(a): #inherit level1 class c(b): #inherit level2 /\*above case is of multilevel inheritance where class b take inheritance of class a and class c take inheritance of b hence class c had taken 2 level inheritance of a and had properties of both b and a\*/ /\*the priority of inheritance will be class c> class b> class c for objects of class c and so on\*/

Diamond Shape Problem= /\*where more than do parent have same object there is no priority for choosing any one in two or more parents to solve such problem it exist for java hence java avoid multiple inheritance but it don’t exist for python\*/



### Public Private and Protected Acess Modifier

## 7 Polymorphism

### Super class and overriding

class p(): def \_\_init\_\_(self): pass /\*parent class p with constructor inside it\*/ class c(p):

def \_\_init\_\_(self): super().\_\_init\_\_() /\*to child class c constructor had inherent properties of constructor of parents class by super()\*/

/\*Overriding avoids duplication of code, and at the same time, enhances or customizes the part of it. It is a part of the inheritance mechanism. \*/ /\*eg in above case first priority will be give to variable of constructor of child than of child variable than to parent class constructor than to its parent variable as other variable were overided\*/ /\*if a argument is overrided its further arguments will not be printed\*/ /\*to avoid above issue that we want to override parent class argument but not its all part we use super().\_\_init\_\_(self) constructors\*/

/\*note in above case the similar argument can be represented in many form which is called polymorphism\*/

## 8 Operator Overloading & Dunder Methods

### Operator overloading

Operator overloading means assign new functionality to an operator beyond its normal functioning. Eg + usage as addn but in case of string it concatenate (concatmation is + operator overloading\*/

### Dunder methods

Dunder methods (\_\_) is used for such cases eg \_\_init\_\_ it is an overloading method

Dunder methods are used for operator overloading and customizing some other function’s behavior.

### \_\_str\_\_ and \_\_repr\_\_ functions

Both of these built-in methods are used to return a presentable description of any object rather than the default one. The difference in them is the way of writing them.

The \_\_str\_\_ method is mainly written for the end-user, while \_\_repr\_\_ is written for a developer.It is overridden to return a printable string representation of any user-defined class.

priority of \_\_str\_\_ is greater than \_\_repr\_\_. This means that if we pass an object into a print statement, it will return us the \_\_str\_\_ string even if \_\_repr\_\_ is also present there. In such cases, if we want to print \_\_repr\_\_, we have to call it exclusively with the object name in the print statement.

def \_\_add\_\_(self, other): return self.a1 + other.a1 /\*to add variable a for 2 or more objects\*/

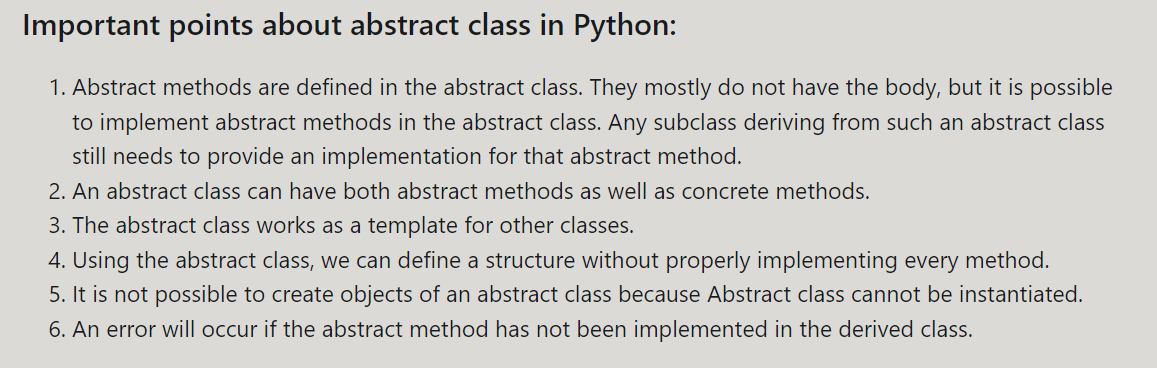
def \_\_repr\_\_(self): return f"a('{self.a1}', {self.a2}, '{self.a3}')" /\*if we print object b1 we will get this list of variable inside object b1 of we can use\*/ print(str(b1)) /\*or\*/ print(repr(b1))

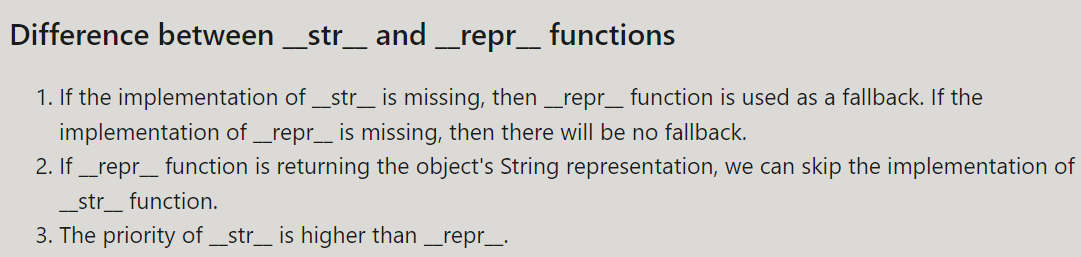
## 9 Abstract Base Class

/\*“An abstract class is a class that holds an abstract method.” And “An abstract method is a method defined inside an abstract class.”\*/ /\*they are essential when we want to engulf common rules for all other classes\*/

from abc import ABC, abstractmethod /\*ABCMeta for older version\*/ /\*it is the abc import to allows such initiation\*/

Class A(ABC): @abstractmethod def A1(self): /\*here A is abstract (essential class) with fn A1 compulsory for all other class to implement\*/ /\*now all other class must inherent from class A1 and they must have fn A1 inside them with their class speciality\*/

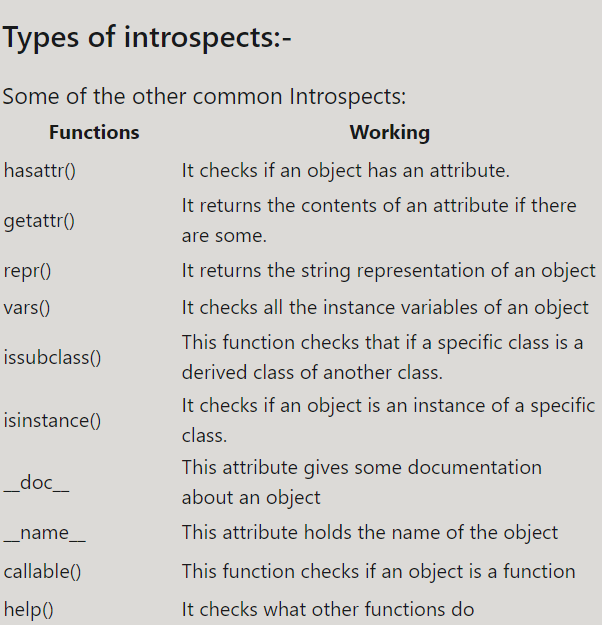




## 10 Object Introspection

/\*Introspection can be said as the ability to recognize the object along with all its details, such as id or location at runtime. \*/ type(object) /\* the type of our object, that whether it is int, float, or string.it also tell about object at runtime. Hence, by using introspection, we can inspect the Python objects dynamically. id(object) /\* Id provides us with the id allocated to the particular object. The id of each object is unique, meaning it is different, and no two objects can have the same id.  \*/

dir(o) /\* It returns us a list of attributes and methods associated with an object. It is mostly executed before and after updating our object by inserting more attributes or methods.



# 3 external memory management

## 1 to deal with external files

f = open("harry.txt", "rt") /\*open the file name harry.txt r for read t for text with syntax open("filename" ,"mode")\*/ print(f.readline()) /\*to print consiquentive line in harry.txt eg here first line than second than third and so on\*/ print(f.readlines()) /\*it will print all lines in form of list of all lines\*/ # content = f.read(345) /\*f.read will help to store readable data from f if 345 is present it will read only 345 characters from 0-345char if same fn run again next reading will start from 346th charcter\*/ # for line in f: /\*for loop to read line in this case if f was replaced by content it will read content character by character not line by line\*/ a=f.write(“poppy”) print(a) /\*it will pring no of character in f.write here 5 characters\*/ f.close() /\*open must come with closed statement\*/

f.tell() /\*it will tell no of character which were printed as op\*/ f.seek(5) /\*it will allow us to read line after char as it print line from char5 \*/

r : r mode opens a file for **read-only**. We do not have permission to update or change any data in this mode. It is default mode\*/

w : w mode does not concern itself with what is present in the file. It just **opens** a file **for writing** and if there is **already some data present** in the file, it **overwrites** it.

x : x is used to **create a new file**. It does not work for an **already existing file**, as in such cases the **operation fails**.

a : a stands for **append**, which means to **add something to the end of the file**. It does exactly the same. It just adds the data we like in write(w) mode but instead of overwriting it just adds it to the end of the file. It also does not have the permission of reading the file.

t : t mode is used to **open our file in text mode** and **only proper text files** can be opened by it. It deals with the **file data** as a **string**. /\*it is also default mode\*/

b : b stands for **binary** and this mode can only open the binary files, that are read in bytes. **The binary files include images, documents**, or all other files that **require specific software to be read**.

r+ : In plus mode, **we can read and write a file simultaneously**. The mode is mostly used in cases where we want to **update our file**.

## 2 import

/\*2 methods to import fns\*/

import sklearn /\* one is to import using an object. \*/ from flask import Flask /\*import the resource from another package or module. \*/ import pandas as pd /\*  choose to rename an imported resource, \*/

import file2 /\*to import local file type import file name hence variable and fn inside file can be imported\*/ print(file2.a)

/file2.printjoke("This is me") /\*inside file 2\*/ file 2 a =7 def printjoke(str): print(f"this function is a joke {str}") /\*when we import local fn it will also execute program inside imported file first before its actual use\*/ /\*to avoid above problem use syntax\*/ if \_\_name\_\_ == '\_\_main\_\_': /\*and put our code inside this part which we wont want user to use it\*/

## 3 Cache creation

***/\*Caching*** means storing the data in a place from where it can be served faster. In the case of data that has been frequently used, the computer assigns a cache memory, so it does not have to load it again and again from the main memory.\*/

/\*In Python, however, we have to do it all manually, as the program will not store anything in the cache itself.\*/

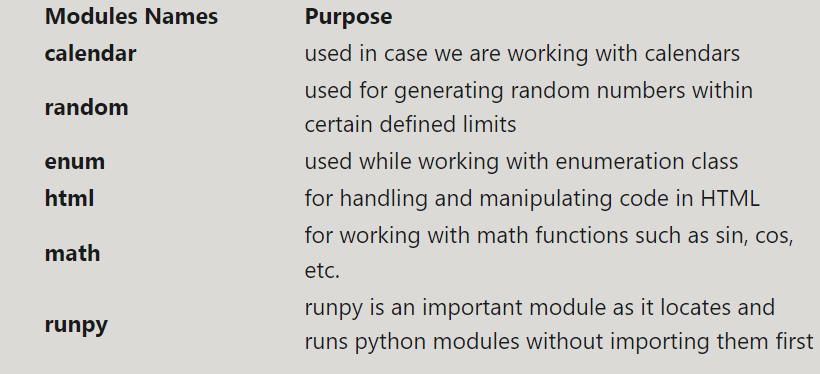
/\*After **Python 3.2**, cach was part **functools\*/**

from functools import lru\_cache /\*to import cache fn from functools library\*/

@lru\_cache(maxsize=32) def fn /\*it will first initialized fn than fn copy could be stored in memory till 32 times after 32 time programme will able to take extra memory\*/

# 4 Modules

## Inbuild module



## 1 Time

import time initial = time.time() time.time() – initial /\*initial is used to capture time at the moment and use to subtract with current capture to calculate actual loop\*/ time.asctime(time.localtime(time.time())) /\*to print local standard time\*/ time.sleep(2) /\*it will create delay of 2sec\*/

## 2 random

random.randint(1, 5) /\*to generate ramdom integer between 1 and 5\*/ random.random() /\*to generate float value in between\*/ l = random.choice(l1) /\*l will point random object from list l1\*/

## 3 gingerIt

from gingerit.gingerit import GingerIt /\*to import grammer corrector\*/ parser = GingerIt() /\*parser will initialize api\*/ text = parser.parse(text) /\*it will create list of text which is used to do all api work\*/ return str(text["result"]) /\*it will return corrected grammer syntax which is stored in text[“result”]

## 4 Virtual Environment

virtualenv v /\*cmd command to make a virtual envt inside folder v\*/ .\vScripts\activate /\*to activate virtual envt\*/ deactivate /\*to deactivate virtual envt\*/ pip**freeze > requirement.txt /\*to make track record file to install all virtual envt dependencies and imported module\*/** pip install package\_name == version /\*to install particular version mentioned in virtualenvt to run a particular code\*/ pip install -r .\requirements.txt /\*to install all code of virtual envt together\*/

## 5 Os Module

|  |  |
| --- | --- |
| Built-in Functions | Working |
| print(dir(os)) | It gives us a list of all the functions the OS module is composed of. |
| os.getcwd( ): | Here cwd is a short form for the current working directory. The function returns us the path of the directory we are currently in. It is important to know about our directory because when we are trying to import a file in python, the compiler searches for it in our current directory. |
| os.chdir( ): | It is used in case we want to change our directory. The new path is sent inside the parenthesis. If we want to access some files or folders from some other directory, we can use it. |
| os.listdir( ): | If we want to output the names of all the directories at a certain location, we can use this function. |
| os.mkdir( ): | To create a new directory or folder. The name is sent as a parameter inside the parenthesis. |
| os.makedirs( ): | To make more than on directory simultaneously. |
| os.rename( ): | To rename an already existing directory, we use this. We send the current name and new name of our directory as parameters. |
| os.rmdir( ): | It deletes an empty directory. |
| os.removedirs( ): | We can remove all directories within a directory by using removedirs(). |
| os.environ.get( ): | It will return us the environment variable. The environment variable must be set, or the function will return null. |
|  |  |
| os.path.join( ): | It joins one or more path components. We can join the paths by simply using a + sign, but the benefit of using this function is that we do not have to worry about extra slashes between the components. So less accuracy still provides us with the same result. |
| os.path.exists( ): | It returns us a Boolean value, i.e., either true or false. It is used to check whether a path exists or not. If it does, then the output will be true, otherwise false. |
| os.path.isfile( ): | It returns true if the path is an existing regular file. |
| os.path.isdir( ): | It returns true if the path is an existing directory. |

## 6 Requests Module

HTTP stands for the ***'Hypertext Transfer Protocol,'***. It is a set of protocols that are designed to enable communication between clients and servers. /\*From the name, we can detect that the get function returns us some information about the site we requested. All the information is stored in the object we used to send the request. Eg\*/ requests.get(URL, params={key: value}, args) import requests r2 = requests.post(url=url, data=data) r = requests.get("link")

**URL:** this is the URL of the website where we want to send the request.

**Params:** this is a dictionary or a list of tuples used to send a query string.

***Args:*** It is optional. It can be any named arguments offered by the get method.

|  |  |
| --- | --- |
| Methods | Working |
| put( ): | It is used to send a put request to a variable. It is usually used to update or completely change the resources of a specific URL. |
| delete( ): | Delete is used to delete the specific resource specified by URL. |
| head( ): | The head method returns a header for a specific resource. |
| post( ): | Post requests take with it the data to the server to update it with. |
| patch( ): | The patch is used to send patch requests. It is used to apply partial modifications to a resource. It carries with it the instructions for the modification. |

## 7 JSON module

  JSON stands for JavaScript Object Notation. JSON is a data-interchange format derived from JavaScript. It is mostly used for storing or transferring data. So, if we want our program to interact with the internet, we must be familiar with this module, even only to send or receive data through the internet.

**load():** This method is used to load data from a JSON file into a python dictionary.

**Loads( ):**This method is used to load data from a JSON variable into a python dictionary.

**dump()**: This method is used to load data from the python dictionary to the JSON file.

**dumps():** This method is used to load data from the python dictionary to the JSON variable.

import json data = '{"var1":"poppy", "var2": “pogli”}' parsed = json.loads(data)

## 8 pickle module

***Pickling*** is the process of serializing an object. Serializing means storing the object in the form of binary representation so it can be saved in our main memory. The object could be of any type.

import pickle file = "mycar.pkl" fileobj = open(file, 'wb') pickle.dump(cars, fileobj) fileobj.close()

file = "mycar.pkl" fileobj = open(file, 'rb') mycar = pickle.load(fileobj)

We can face some of the common pickle exceptions raised while dealing with the pickle module.

**Pickle.PicklingError:**If the pickle object does not support pickling, then Pickle.PicklingError exception is raised.

**Pickle.UnpicklingError:** This exception will raise if the file contains bad or corrupted data.

**EOF Error:**This exception will be raised if the end of the file is detected.

#### Disadvantages:

* There are some situations in which pickling is discouraged. For example, when we are working with multiple programming languages, pickle is discouraged.
* Pickle has been found slower than its alternatives.
* In some cases, it has also shown a few security vulnerabilities.
* When we update our program to a newer version, pickled data through the previous version can cause issues.

## 9 Pyinstaller

We create lots of Python programs and want to share them with the world. If we want our python program to run on any computer without the IDE or even installing Python itself, we must convert it to .exe. All the apps and programs we use on our computer are written using some language or code, but we do not have to install the particular language for the program to run.

python .\main.txt /\*to run txt file as python at terminal\*/ pyinstaller main.txt /\*to create exe file of python code \*/ /\*it could be txt file or python file\*/ pyinstaller --onefile file.py /\*to install python in one file\*/

## 10 Command Line utility

/\*Command line utility helps in using code from command line by taking input from python in terminal\*/

import argparse import sys /\*inbuild module to use command line utility\*/

/\*We can use the Python argparse module to create a command-line interface by following these steps:\*/ /\*Import the Python argparse module Create the parser Add optional and positional arguments to the parser

Execute .parse\_args() eg\*/ parser = argparse.ArgumentParser() parser.add\_argument('--x', type=float, default=1.0, help="helper fn") /\*--x will recognise command to be taken which is mention after it\*/ args = parser.parse\_args() sys.stdout.write(str(calc(args))) /\*to print calculated argument in terminal\*/

## 11 Python Package

/\*to create package file\*/ /\*create Licence.md readme.md setup.py outside main package\*/ /\*inside module name create \_\_init\_\_.py to write actual code setup.py will only used for installation perpose\*/

from distutils.command import install from setuptools import setup,find\_packages setup(name="packagepoppy", version="1.0.3", description="this is pogli aurat software package", long\_description="yaha pogli aurat software bana kr idhar udhar dolti he", author="poppy", packages=['packagepoppy'], install\_requires=[])

pip install wheel /\*package which will be used to make personal package\*/ python setup.py bdist\_wheel /\*write this command in directory where setup exist\*/ python setup.py sdist bdist\_wheel /\*to write above code more efficiently\*/

/\*if any changes were made in code \_\_init\_\_.py version in setup.py must be updated\*/ /\*than setup command must be rerun of same directory of setup.py\*/ python setup.py sdist bdist\_wheel

/\*hence using above methods local created packages can be run similar to original and imported packages by fn\*/ import package\_name

**name:**The name of the package. We can give our package any name of our choice.

**version:** The starting version should be 0.1 because, with any update, it automatically increases it to one decimal place.

**description:** Here, we give a brief description of our package and its functionalities and uses.

**long\_description:** In the long description we give an explanatory description of our package

**author:** We can specify the creator of the package here

**packages:** Here we give the name by which we want our package to be called or imported

**install\_requires:** If our package has a prerequisite package, then we have to specify that here so both of them can work simultaneously and can perform better.

# Frameworks

## 1 webdevelopment

1 Flask(fast and microframework) 2Django. (have inbuild module which are only needed to be used\*/

## 2 GUI(****Graphic User Interface)****

GUI is an **interface** with the help of which a user can interact with the device.

There are a lot of options available that could help us develop a GUI for our program. Some of the well-known are:

1.Tkinter  2. wxPython  3. JPython

## 3 Machine Learning **and data science**

**NumPy** library that includes support for large, multi-dimensional arrays and matrices along with higher-level mathematical functions written in predefined **C binaries**.(c language)

Now the other important feature that the **NumPy package** provides us with is**pandas**. It helps us store our data in a tabular form in the form of rows and columns, and while using a range of input and output data, it is beneficial.

In Python, we usually work with a large number of data, so we have to use a database to store such a large amount of data. For that purpose, we have to learn about **Python MySQL** that will serve our purpose of storing the data in the database for easy and fast retrieval.

TensorFlow to get prebuild libraries to acess and reuse ML modules [learn](https://scikit-learn.org/stable/)

# Miscellaneous

## 1 Generator

**/\* *Iterables*** are objects that can be placed inside a loop and can return one variable at a time. Examples list, string, tuple, etc.\*/  /\*An iteration can be defined as an object that does iterations on iterable.\*/ /\*Meaning that it will move from character to character doing iteration. \*/

### \_\_iter\_\_() and \_\_next\_\_() method

built-in method \_\_iter\_\_() that creates an object when called.

The object moves from character to character of iterable using the \_\_next\_\_() method. The \_\_next\_\_() method is what’s really behind the working of the loop.

**Generators** concept is also very similar as it is used to make an iterator. The only difference comes in the return statement. The generator does not use a return statement. Instead, it uses a yield keyword.

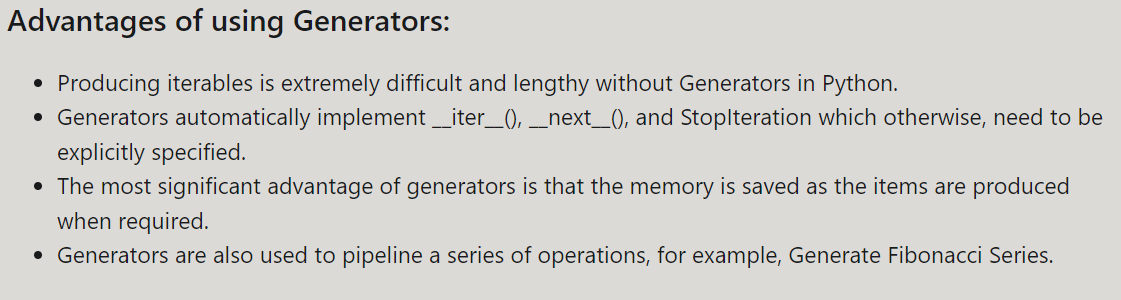
Yield functionality is very similar to return as it returns a value to the caller, but the difference is that it also saves the state of the iterator. Meaning that when we use the function again, the yield will resume the value from the place it left off.

Generators in Python are created just like the normal functions using the **‘def’**keyword. Generator functions do not run by their name, and they are run when the \_\_next\_\_() function is called.

A generator is very helpful in projects relating to memory issues because, like a simple iterator, it does not return all the values at a time;

instead, it produces, series of values over time. So a generator is generally used when we want to iterate over a series of values but do not want to store them completely in memory.

def getNum (x): for i in range(x): yield i /\* it is a generator fn which will only yield when called opon else it will not get printed\*/ seq = getNum (200) /\*to initialized created generator in object seq\*/ print(seq.\_\_next\_\_()) /\*it will only run when called upon\*/ till iterator react to n-1 here 1 value\*/ /\*note generators could be printed continuously for a period of value by using for loop similar to list



## 2 Python Comprehensions

## 3 Coroutines

def a(): fn /\*the first part of coroutines which will run only just after initialization\*/ while True: /\*code run after initialization\*/ value = (yield) /\*to take input from user in fn input being (yield)\*/ fn after initialization during running coroutine\*/

b =a() next(b) /\*run coroutine for first time\*/ /\*note next(b) should be run only one time after initialization of coroutine\*/ b.send("xyz") b.send(" send ") /\*to send data as ip\*/ b.close() /\*to close coroutine\*/

***Coroutines*** are mostly used in cases of time-consuming programs, such as tasks related to machine learning or deep learning algorithms.

In such situations, we do not want the program to read the file or data again and again, so we use coroutines to make the program more efficient and faster.

***Comprehensions in Python***  Comprehension in Python converts the four to five lines of code into a one-liner.

list = [a for a in range(50) if i%5==0] /\*list comprehension with [ ] to denote list\*/

alphaset = {alpha for alpha in ["a", "a", "b", "c", "d", "d"]} /\*set of alphabet to print unique element { } donate set\*/

dict = {i:f" and Item {i}" for i in range(5)} /\*to dictionary comprehension \*/ dict2 = {value:key for key,value in dict1.items()} /\*it can also help us to print reverse dictionary\*/

gen = (i for i in range(n)) a = gener(5) print(a.\_\_next\_\_()) /\*generator comprehension with ( ) denotes generator\*/

## 4 Regular Expressions

***/\**Regular expressions** are used to perform search-related tasks in Python. re” module is used for complex searching, using Metacharacters and special sequences. \*/ import re The module defines several functions and constants to work with RegEx. The “re” module is composed of five functions known as:

**findall**: It finds all searches for matches and prints resultant in the form of a list.

**search**: It works the same as a findall, but the resultant is a matched object if any is found.

**split**: The split function splits the string from every matched into two new strings.

**sub**: The sub-function works exactly like a replace function in notepad or MS Word. It replaces the original word with a word of our choice. **finditer**: The finditer yields an iterator as a resultant with all the objects that match the one we sent it) finditer supports more attributes than any other function defined above. It also provides more details related to the matched object. So, most of the examples we are going to see next will contain a finditer function in them.

Meta Characters  
[] A set of characters  
\ Signals a special sequence (can also be used to escape special characters)  
. Any character (except newline character)  
^ Starts with  
$ Ends with  
\* Zero or more occurrences  
+ One or more occurrences  
{} Exactly the specified number of occurrences  
| Either or  
() Capture and group  
Special Sequences  
\A Returns a match if the specified characters are at the beginning of the string  
\b Returns a match where the specified characters are at the beginning or at the end of a word r” ain\b.”  
\B Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word  
  
\d Returns a match where the string contains digits (numbers from 0-9)  
\D Returns a match where the string DOES NOT contain digits  
\s Returns a match where the string contains a white space character  
\S Returns a match where the string DOES NOT contain a white space character  
\w Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore \_ character)  
\W Returns a match where the string DOES NOT contain any word characters  
\Z Returns a match if the specified characters are at the end of the string

patt = re.compile(r'\d{5}-\d{4}') matches = patt.finditer(mystr) for match in matches: print(match)

## 5 Python 2 to python 3

/\*See python 2 to 3 documentation to make changes in codes\*/ eg changes print “a” == print(“a”)

raw\_input == input xrange() == range()

When a new update arrives, all the applications, software are not updated readily according to the new update, but it takes some time. Eg numpy for ml it take time to adjust itself for new update

