**OOPS**

**Oops Principles:**

Oops principles are nothing but certain rules or guidelines which we apply for business applications.

Following are different oops principles

* Encapsulation
* Inheritance
* Polymorphism
* Abstraction

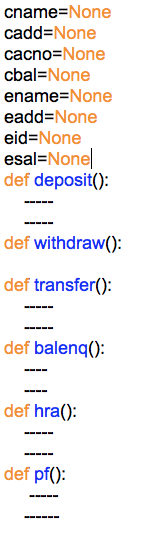
- Advantages of Oops principle are security, flexibility and reusability.

**Object Oriented Programming Language:**

Any programming language which can implement oops principle is known as Object Oriented Programming Language.

Ex: C++, Java, .NET, Python

If we implement the business application in python language using procedure orientated mechanism, it looks like below.



Global Variables

**Problems in procedure oriented mechanism:**

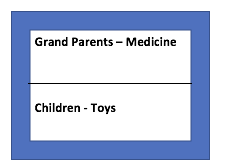
1. Missing security for global variables. Customer data is missing security from employee functions and vice versa
2. Whenever we run this program, memory gets allocated for all the variables. Even if you try to access customer functions but still memory gets allocated to all the variables
3. For all the global variables of the program, memory get create only once. So, we can store only 1 customer details only in the memory.

To overcome these problems, we need to go for 1 oops principle called ***Encapsulation***.

This entire scenario is something like, in a room grandparent’s medicines and children toy’s both kept. So, both will have access to both items

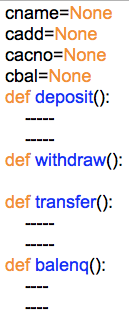
**Encapsulation:**

A concept of binding or grouping related data members along with its related functionalities is known as encapsulation.



**Group1**

**Group2**



This is one group (This is Customer)



This is another group (This is Employee)

- Here advantage is whatever group you execute for that particular group only memory gets create.

- This is roughly we have grouped, but practically this group we can implement using a concept called ***class***

**class:**

class is a syntax or structure which is used to group related data members along with its related functionalities

Data🡪var 🡪 1. Static Variables

class Classname:

“””-----“””

data

op

2. Non-Static Variables

3. Local Variables

Operations🡪Methods🡪1. Static Methods

2. Non-static methods

3. Class Methods

- We can define 3 types of variables with in the class. They are

(1) Static variables (Memory allocated automatically)

(2) Non-static variables (When object creates memory is allocated)

(3) Local variables (When method executes memory is allocated)

- With in the class we can represent the operations by using methods

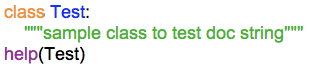
- We can define 3 types of methods with in the class they are

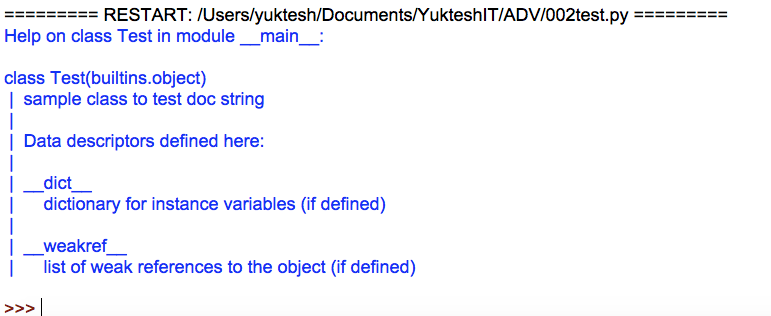
(1) Static Methods

(2) Non-static Methods

(3) Class Methods

Ex: Test class.





Let’s say there is customer class, in bank lakhs of employees will try to access class. This concept we make it happen by creating ***object***to the customer class

c1

class Customer:

“””-----“””

data

op

c2

c3

On the customer class objects, we can perform customer operations.

**Object:**

The concept of allocating memory space for non-static variables of a class at run-time dynamically is known as object.

or

Instance of a class is known as an object

Instance = dynamic memory allocation

**Reference variable = classname()**

object creation statement:

- After creating the object python interpreter takes the original address of the object, creates indirect address based on the original address and gives that indirect address to the variable.

-The variable which is storing indirect address of the object is known as a **reference variable**

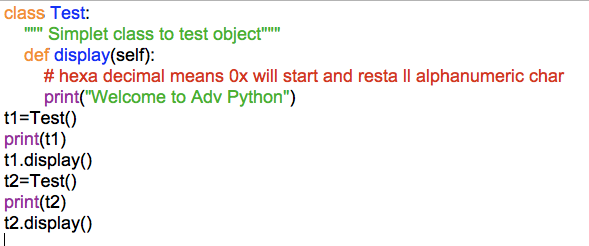
-The indirect address which is created by the python interpreter based on the original address of the object contains 2 parts they are **class name** and second one is **hexadecimal representation** of hash code.

-We can create ‘n’ of objects for a class.

-Through the reference variable we can put the data into the object, we can get the data from the object and we can call the methods on the object.

-Every object will have unique address

Ex1: Program to understand object creation and calling a method.



op:



**Static variables:**

-The variables which are declared with in the class outside of all the methods are known as “static variables”

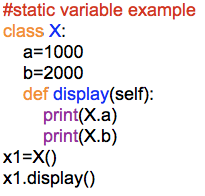
- The data which is common for all the objects is recommended to represents by using static variable.

- For all the static variables of a class memory will be allocated only once.

- Static variables of class, we can access with in the all methods of same class by using classname

- Static variables of 1 class we can access from outside of the class by using classname or by using reference variables

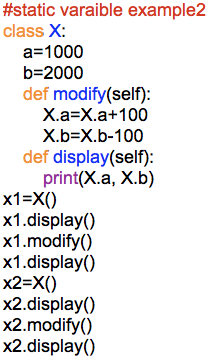
Ex1: Understand static variable



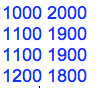
op:



Ex2: Understanding static variable



o/p:



**Non-Static Variables:**

-The variables which are declared with in the class by ***self.*** are known as non-static variables

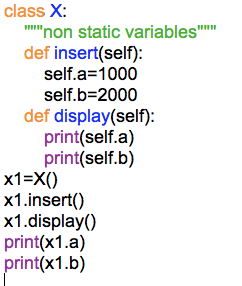
- The data which is separate for every object is recommended to represent by using non-static variables.

- For all the non-static variables of a class memory will be allocated with in the object.

- Non-static variables of 1 class we can access with in the same class by using self.

- Non-static variables of 1 class we can access from outside of the class by using reference variable.

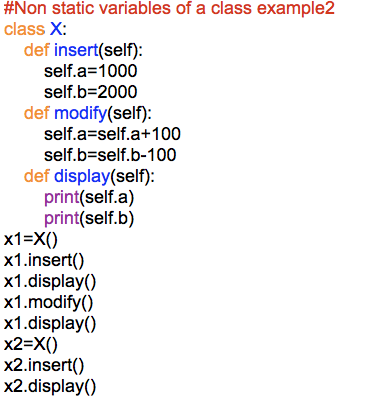
Ex: Program1 to understand non-static variables of a class.



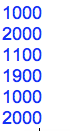
o/p:



Ex: Program2 understanding static variable

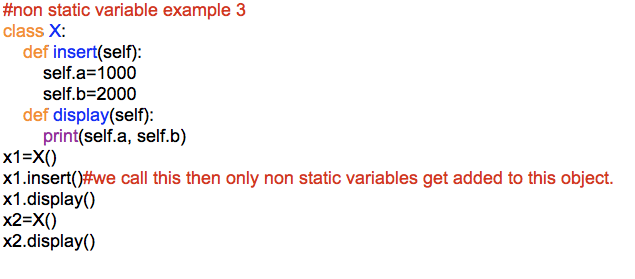


o/p:



If we define non-static variables with in the methods, those non-static variables will be added to the object whenever we call that method on the object.

Ex: Program 3 to understand non static variable



o/p:



To overcome this problem, we go for constructor:

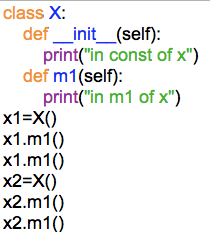
**Constructor:**

- Constructor is a special method which is used to define and initialize non-static variables of a class at the time of creating object.

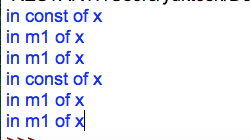
Differences between methods and constructors

|  |  |
| --- | --- |
| **Method** | **Constructor** |
| Method name can be any name | Constructor name must be \_\_init\_\_(self) |
| Method executes whenever we make a method call | Constructor will be executed whenever we create an object |
| With respective to 1 object 1 method can be called multiple times. | With respective to 1 object 1 constructor will be executed only once |
| Methods are used to represent the business logic to perform the operations | Constructors are used to define and initialize non-static variables of a class at the time of creation of the object |

Ex1: Program to understand constructor

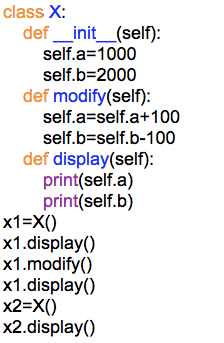


op:



Constructor executes only once for each object unlike methods. Actual, usage of constructor is to initialize non-static variables.

Ex:2 Initializing variables using constructor

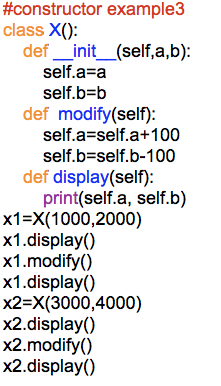


op:

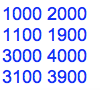


For every object, different values if we would like to give for non-static variables then recommended mechanism is giving values of non-static at the time of object creation.

Ex:3 Constructor example 3



op:



**Local Variables:**

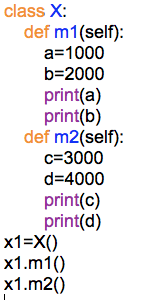
-The variables which are declared with in the method body are known as local variables.

- The data which is required to use with in a particular method is recommended to represent by using local variables.

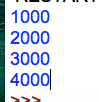
- Local variables of a method memory will be allocated whenever we make a method call.

- Local variables of 1 method cannot be accessed from outside of that method.

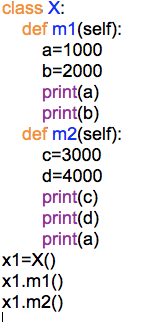
Ex1: Program to understand local variables



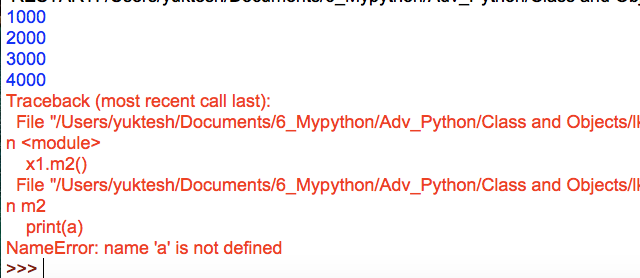
op:



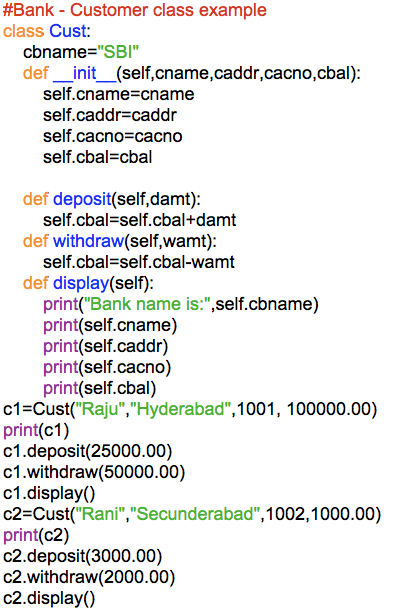
Ex:2 Access a in m2 and check the result



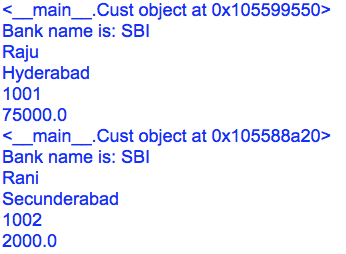
op:



Ex3: Understand Customer class with various variables



op:



**Garbage collector:**

- The concept of removing the unused or unreferenced objects from the memory location is known as a “Garbage Collection”

- At the time of execution of program if any object reference count becomes zero than python interpreter internally calls the garbage collector.

-The number of reference variables which are pointing an object is known as a reference count of the object.

- In any object the reference count becomes 0, then we call that object as an unused or unreferenced object

-Garbage collector is a predefined program or predefined background thread, which removes the unused objects from the memory locations

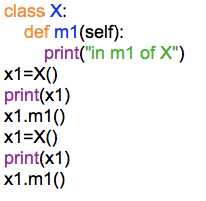
-After executing the garbage collector, more memory space is available to the program and rest of the program execution becomes faster.

x1=X()

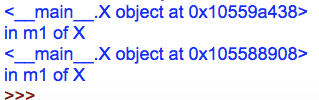
R=1, become 0 when second object gets created

x1=X()

Ex1: Program to understand reference variable and object reference count.



op:



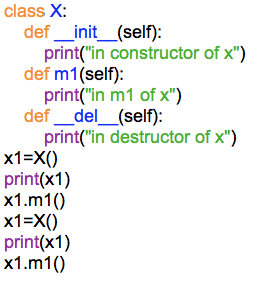
**Destructor:**

- Destructor is a special method and the name of the destructor must be \_\_del\_\_()

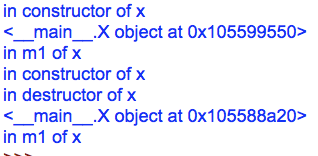
- Destructor of a class will be executed whenever that class object is deleting from memory location

- Resource releasing statements or resource deallocation statements are recommended to represent in destructor.

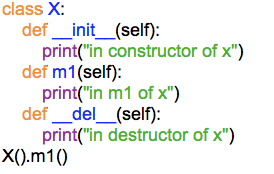
Ex:1



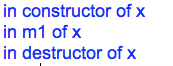
op:



Ex2: Let us do small change in the code and rewrite another program



op:

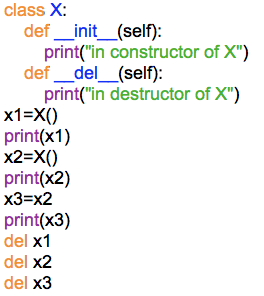


Garbage collector can only remove objects but not variables. To remove variables, we need to use ***del*** keyword.

**del:**

- del is a keyword, which removes given variable or variables from the memory location

- whenever del removes variable the reference count of the object which is pointed by that variable will be decreased by one.



op:



**Using properties of one class into another class:**

-We can use the properties of one class into another class in two ways.

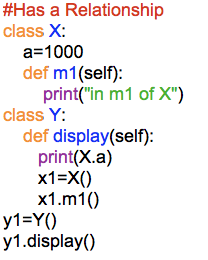
(i) Has a Relationship

(ii) Is a Relationship

**Has a Relationship:**

-The concept of using the properties of one class into another class by using class name or by using reference variable name is known as a “Has a Relationship”

Ex: Program to understand “Has a Relationship”



op:



**Is a Relationship (or) Inheritance:**

-The concept of using the properties of 1 class into another class directly is known as inheritance or Is a Relationship

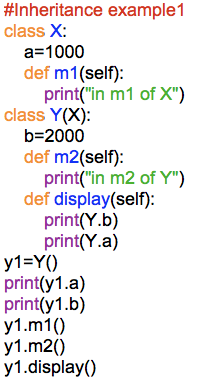
-A class which is extended by another class is known as a super class or base class.

-A class which is extending another class is known as a sub class or derived class

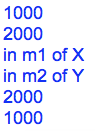
-Super class properties directly we can access with in the sub class like as a sub class members.

-We can access both super class and sub class properties by using sub class name or sub class reference variable.

Ex:1

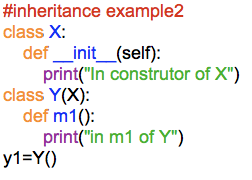


op:



-Constructor of class gets executed immediate after creating object.

Ex3:



op:



If we create Y class object internally how many class objects are getting create???

A)3 (Object, X, Y)

**Object class:**

-Object class is a predefined class, which is defining builtins module

-Object class is a super class for every class in python.

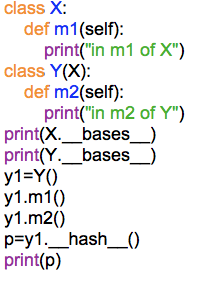
-Object class properties we can access directly in each and every class

-Object class properties we can access through any class reference variable.

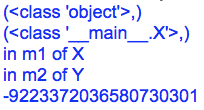
-There is predefined method in Object class \_\_hash\_\_() which we can use in our classes.

ex: p=\_\_hash\_\_() # this method executed means understanding is object class object is created.

Ex:1 Understanding object class

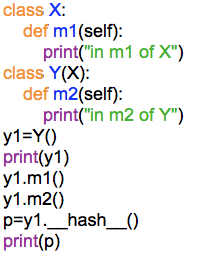


op:

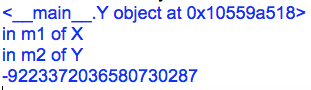


\_\_bases\_\_ variable helps in finding the base class of the current class

Ex2 Program2 to understand object class



op:



\_\_hash\_\_() return integer number.

**Types of Inheritance:**

1. Single Inheritance

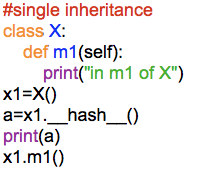
The concept of inheriting the properties from only 1 class to another class is called single inheritance

object

|

|

x



o/p:



**Multilevel Inheritance:**

The concept inheriting the properties from multiple classes to single class with the concept of 1 after the another is known as a multilevel inheritance.

Object

|

|

X

|

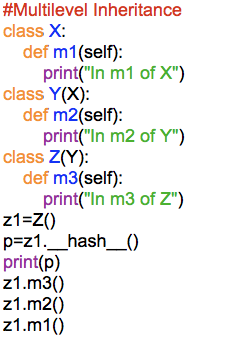
|

Y

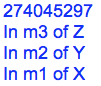
|

|

Z



o/p:



**Hierarchical Inheritance:**

The concept of Inheriting the properties from one class to multiple classes separately is known as hierarchical inheritance

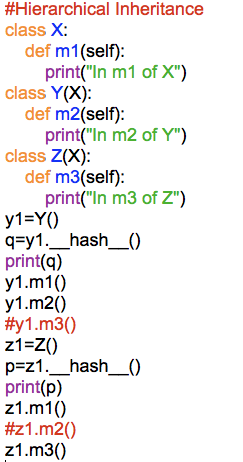
Object

|

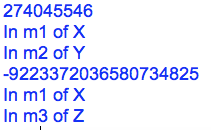
|

| |

X Y



op:



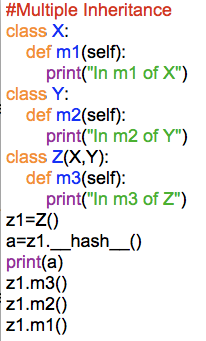
**Multiple Inheritance:**

The concept of Inheriting the properties from multiple classes into single class at a time is known as a multiple inheritance.

object object

X Y

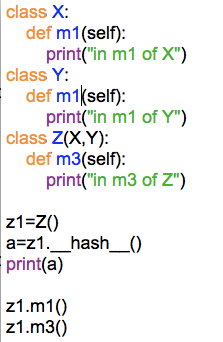
Z



o/p:



If both super classes have same function name means:

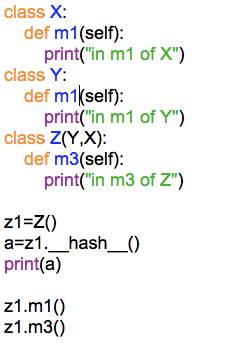


class Z(X, Y) # first it will verify in Z then X and Y

class Z(Y, X) #first it will verify in Z then Y and X

o/p:





OP:



Python does not support cyclic inheritance.

**Cyclic Inheritance:**

The concept of Inheriting the properties from sub class into super class is known as a cyclic Inheritance.

X

Y

Z

**Hybrid Inheritance**

Combination of Single, Multilevel, Multiple and Hierarchical Inheritances is known as Hybrid Inheritance.

object

X Y Z

A B

M

**Task for you**

Hint: Build m1() to m6() all methods for each class.

create object of M class and call all methods from M class object.

**Polymorphism:**

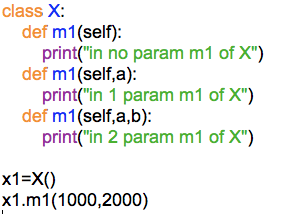
This Poly means many, Morphism means forms and forms means functionalities of logics.

-The concept of defining multiple logics to perform the same operation is known as Polymorphism

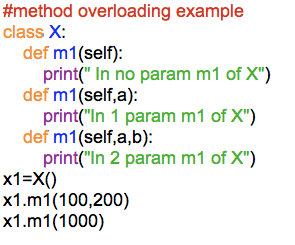
-Polymorphism can be implemented through method overloading or method overriding

**Method Overloading:**

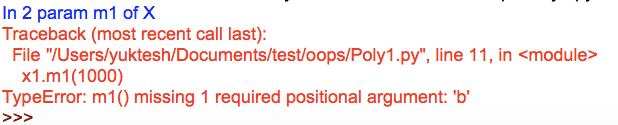
The concept of defining multiple methods with same name with same number of parameters or different no. of parameters with in a class is known as Method Overloading.



Whenever we define with multiple methods with the same name with the same number of parameters or different number of parameters with in a class then by default python interpreter recognizes last defined method only.



o/p:



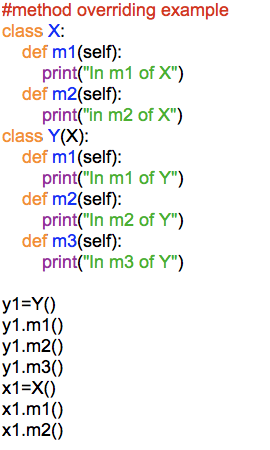
Method Overloading does not support in Python

All Rules of Method overloading in Python applies for same as for Constructor Overloading also.

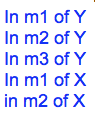
**Method Overriding:**

-The concept of defining multiple methods with the same name, same number of parameters or different number of parameters one is in super class and another one is in sub class is known as a method overriding

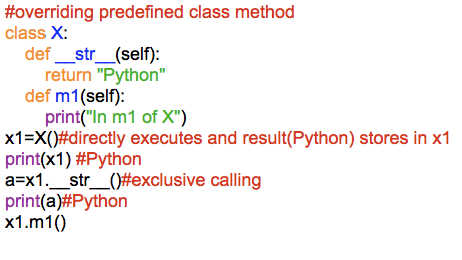
-Whenever we are overriding super class method in sub class then by default sub class method will be executed. If sub class object is created



o/p:



Predefined Class method we are overriding in below class

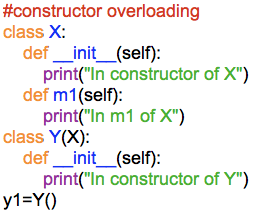


o/p:



Constructor Overriding is possible in python.

Ex:



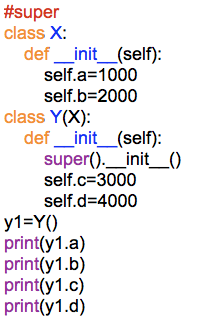
o/p:



super:

super class constructor variables can also access through sub class obj like below using super()

Program:



o/p:



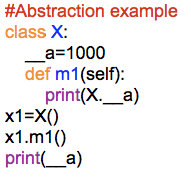
**Abstraction:**

or

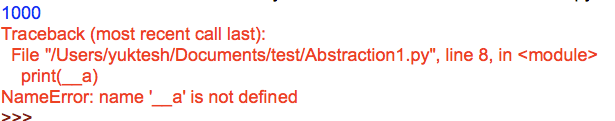
**Data Hiding:**

-The concept of hiding the properties of a class is known as an abstraction.

-We can hide the properties of a class by using \_\_

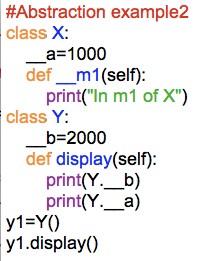


o/p:

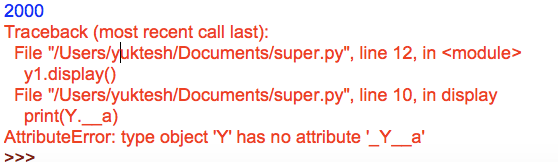


Hidden properties of a class are not available to its sub classes

Ex:

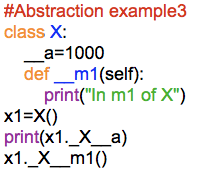


o/p:



Hiding properties of a class we can access from outside of the class by using some special syntax

Ex:



o/p:



**self:**

self is representing object address

self.a self.b means in the object address these variables a and b will go and store.

**Non-static method:**

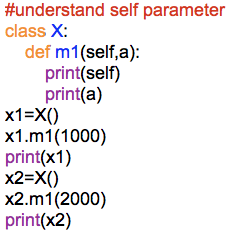
- Non-static() method of a class should contain atleast 1 parameter (self)

- Non-static method of a class should be called from outside of the class by using reference variable

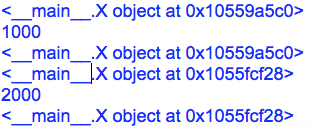
- Whenever we call a method by using reference variable the address which is present inside the reference variable will be passed to the first parameter of that method

- Above concept explained in below program

Program24:



o/p:



**Static Methods:**

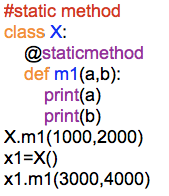
- A method which is preceded by the @staticmethod decorator is known as a static method.

- It is not mandatory to define the parameters to the static() methods

- Static methods of 1 class we can call from outside of the class by using or class name or reference variable

- Below program is example for static method

Program 25:



o/p:



**Class Methods:**

-A method which is preceded by the @classmethod decorator is called ***“class method”.***

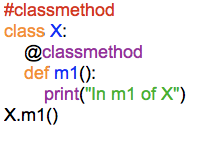
-class method should contain at least 1 parameter. Passing 1 variable is must else below error will come

-class method of 1 class we can call from outside of the class by using class name or reference variable

-Whenever we call a classmethod() by using class name or reference variable. class name will be password to the first parameter of that method. and executes that method

Below program shows error as we are not passing any parameter.

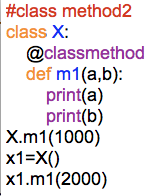
Ex1 Understanding class method



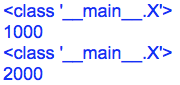
o/p:



Ex2: Understanding passing variables to class method



o/p:



**Abstract Classes and Interfaces**

**Concrete Method:**

- A method whose code is defined in the same class is called concrete method.

- Concrete method we write in general classes.

**Abstract Method:**

- An abstract method is a method whose code is defined in its sub class.

- Abstract methods are written without any body implementation. A method without any body is called Abstract method.

**Abstract Class:**

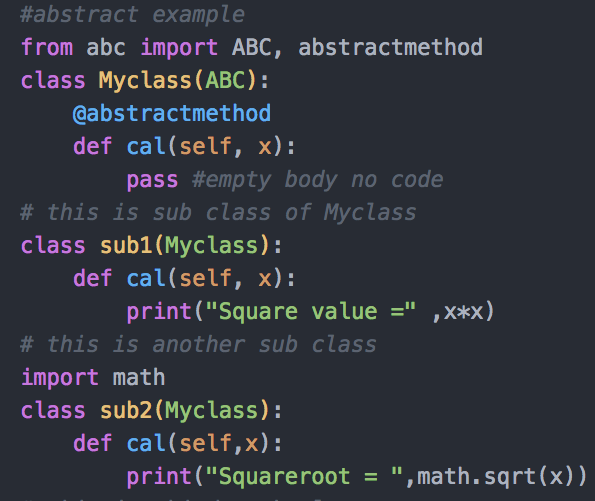
- A class in which at least 1 abstract method is present that class is called Abstract class.

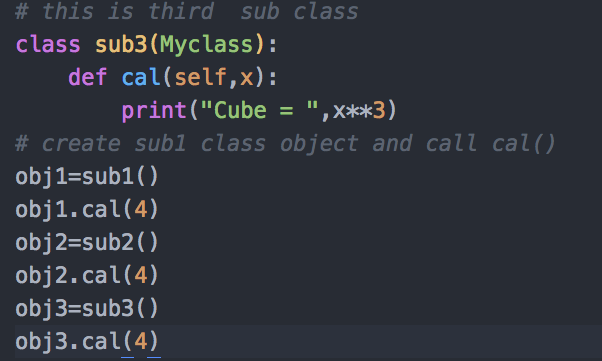
- An abstract class can contain concrete methods.

- An abstract class contains abstract methods whose implementation is done in subclasses. It is not possible to estimate the total memory required to create the object for the abstract class. Therefore, PVM cannot create object for abstract class.

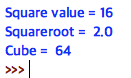
Program1:

Write a python program to create a abstract class and sub class which implement the abstract method in sub class.

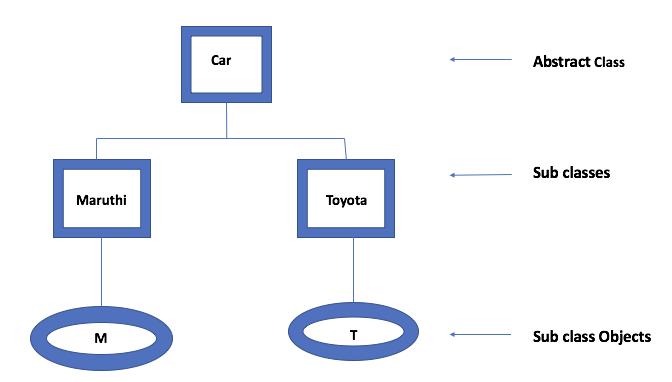




O/P:

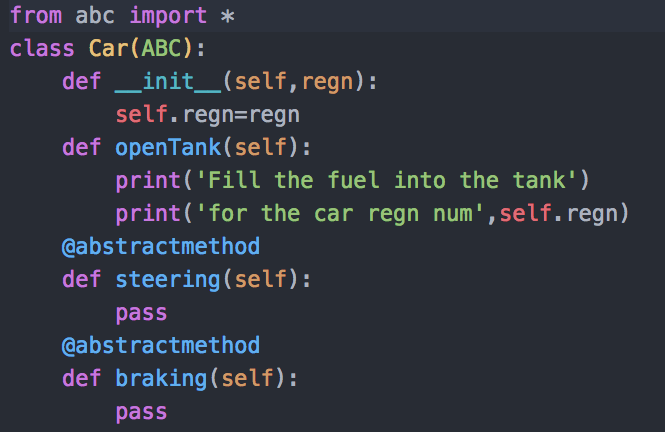


Let’s try to understand this concept with another real-time example.



Program2:

Write a python program to create a Car class that contains a concrete method and 2 abstract methods

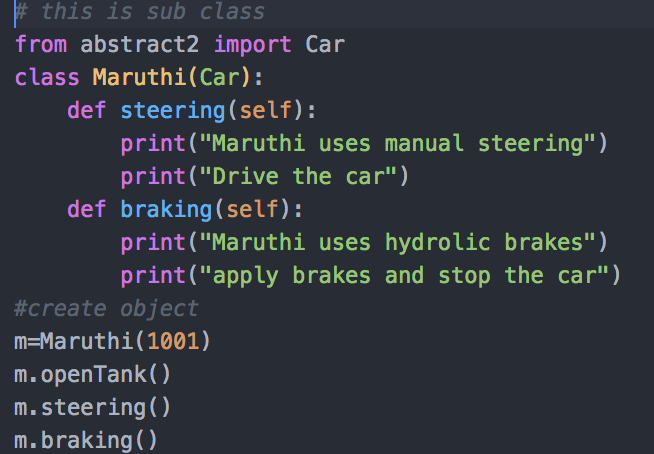


O/P: It will just execute

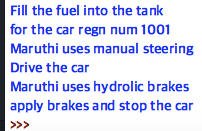
Now let us create a two different sub classes called Maruthi and Toyota and implement these abstract methods in them.

Program3:

Write a Python program implementing abstract methods in subclass of Car class.

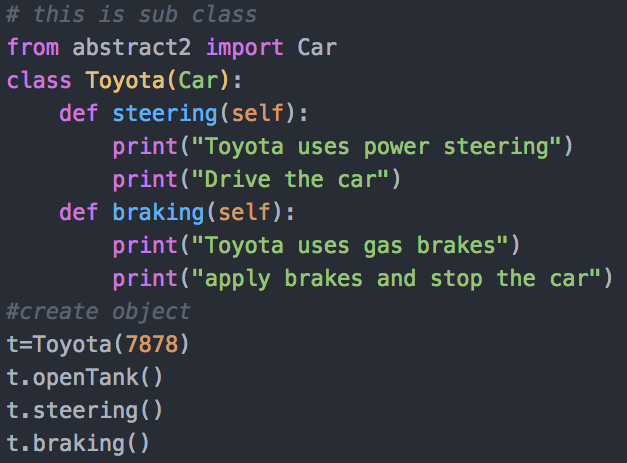


O/P:

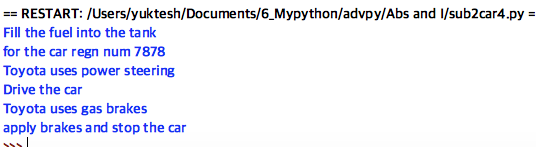


Program4:

Write another Python program implementing abstract methods in subclass of Car.



O/P:



**Interfaces in Python:**

-Python does not support Interfaces concept. To implements this concept we use abstract classes only.

-As we understood abstract classes i.e., if a class contains atleast 1 abstract method is an abstract class.

-It means abstract classes can have concrete methods also

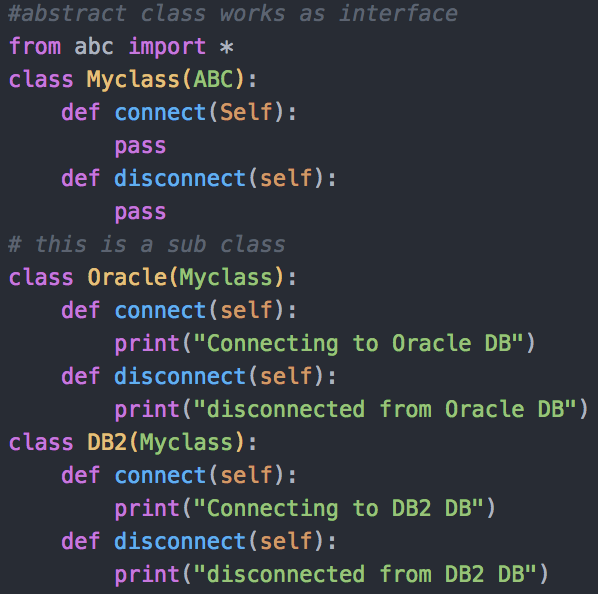
-Whereas an Interface is also an abstract class but only difference is, we should not write any concrete method in it.

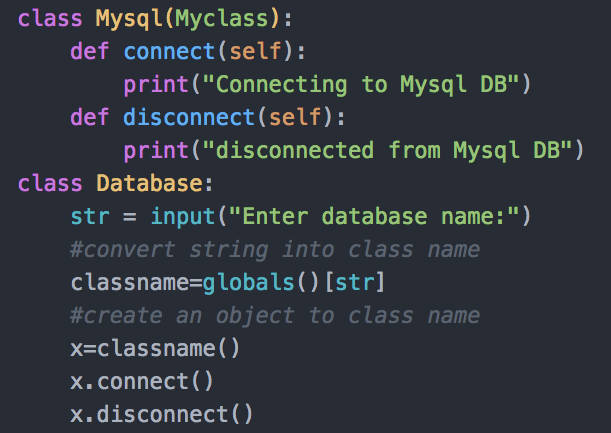
-In real interfaces and more flexible than abstract classes as 100% of the code can be built in sub classes on programmer choice.

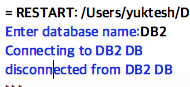
We shall understand interface concept using a program

Program5:

Write a Python program to develop an interface that connects database.







Generally, If you see Interfaces are provided by programming languages and these interfaces are been implemented by difference database software vendors. These are called Drivers.

Difference between the Abs and Interface is whenever if the there are common features we need to share then we should go for abstract class. Whenever 100% code if user wants to implement on by his own. This choice we need to provide by giving an interface to him.

**Exception Handling:**

**Errors in a Python program:**

Generally, there are 3 possibilities of errors can occur in a program

1. Compile-time error
2. Logical error
3. Run-time error

Compile-Time Error:

* These are syntactical error we make in code due to which program fails in execution

Ex: Missing colon (:) symbol in for loop or functions

* These are the errors detected by Python compiler and give clear description where the error is in which line and all

Logical Error:

* These are the errors performed by programmers by giving the wrong logic in the source code
* Here code will execute but results are not as expected

Ex: For addition program one may use – minus operator instead of +

* These errors are not detected by either Python Compiler or PVM programmer only should properly check the code and resolve it.

Run-Time Error:

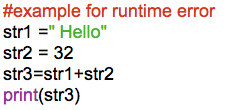
* Runtime errors are the errors detected by PVM at the time of execution of the program.

Ex: Insufficient memory to store a file in the disk, string concatenation of int and string type.

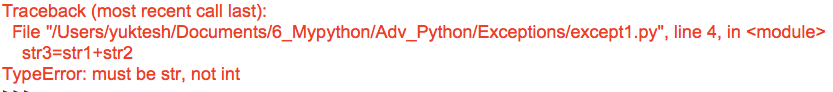
* These are not detected by Python compilers

Program1:

Write a python program to understand runtime error.



O/P:



It shows TypeError it means string cannot concatenate with two diff type of string variables.

Program2:

Write a program to understand runtime errors in better way



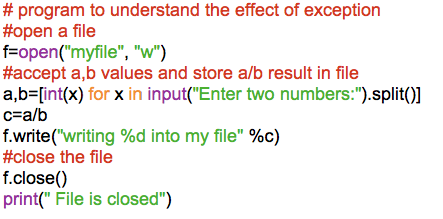
O/P:



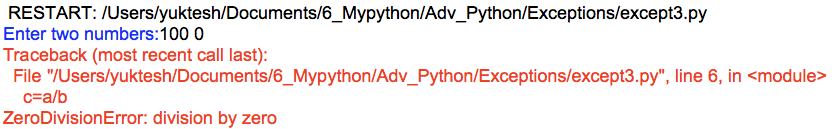
It is showing IndexError it means we are for index number which is not there in the list.

Program3:

Write a python program to understand the importance of exception



O/P:



- In the above program, 100/0 results in infinity which is a value cannot be stores in a variable therefore we are getting ZeroDivisionError

- When this error occurs PVM simply display error message and abnormally terminates the program.

- Due to this sudden termination next statements the program are not getting executed. In this case f.close() is not executed therefore file is not closed

- If file is not closed then there will be more chance to steal your file data.

- Therefore, it is programmer responsibility to handle this type of errors.

**Exceptions:**

- An Exception is a runtime error which should be handled by programmers.

- If a programmer can guess the error going to come in program execution and if he can able to handle that it is called Exception.

- All exceptions are represented as classes in Python

**Types of Exceptions**

- There are two types of exceptions in Python

1. Built-in Exceptions

2. User defined Exceptions

Built-in Exceptions: These are the exceptions which are already available in Python.

- For all built-in exceptions, Exception is the base class which is derived from BaseException class.

- Most of the class name end with Error word.

User-defined Exceptions: These are the exceptions which are defined by the user. User defined Exception classes must be derived from Exception class.

BaseException

|

Exception

|

| |

StandardError Warning

| |

1. ArithmeticError 1. DeprecationWarning
2. AssertionError 2. RuntimeWarning
3. SyntaxError 3. ImportWarning
4. EOFError
5. RuntimeError
6. ImportError
7. NameError

ArithmeticError – Represents base class for arithmetic operations like ZeroDivisionError, FloatingPointError etc

AssertionError – Raised when Assert statement gives error

SyntaxError – Raised when compiler encounters syntax error

TypeError – Raised when inappropriate datatype

EOFError – Raised when input() reach end of file condition without reading data

RuntimeError – Raised when error is detected that doesnot fall in any category

ImportError – Raised when an import statement fails to find the module being imported

NameError – Raised when an Identifier is not found locally or globally.

**Exception Handling:Er**

Why we need to handle exceptions because, to make our program strong and secured.

To handle the exceptions, programmer should follow below 3 steps

Step1: Try Block, Possibility of exception statements should mention in a try block

try:

statements # these statements may lead to exception

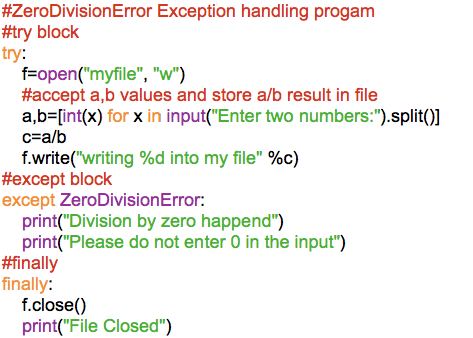
Step2: Except Block, Programmer should develop Except block and here they have to display exception details. Which helps users understand there is an error in the program

Step3: Finally Block, At last Programmer should write Finally Block. In this they should write mandatory execute statements like closing file in the above program.

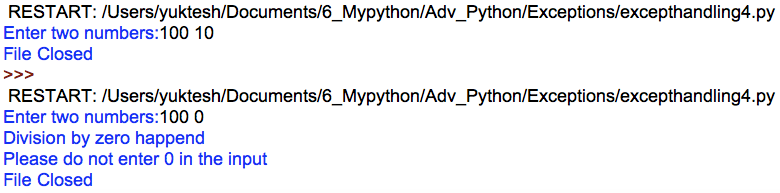
Finally block will executes at any cost though exception occurs or not. Performing above 3 tasks in any program is nothing but called Exception Handling.

Program4:

Write a python program to handle ZeroDivisionError



O/P:



**The complete Exception handling syntax is**

try:

statements

except Exception1:

handler1

except Exception2:

handler2

else:

statements

finally:

statements

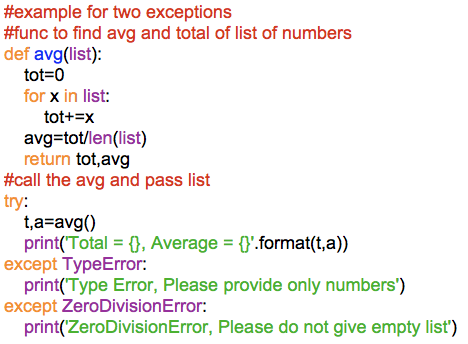
-Try block can yield 1 or more exceptions which we handle separately with different exception handlers. If no exception is raised, then else block will be executed and finally block executes all time.

Important points to note about Exceptions:

* 1 try block can be followed with many except block
* multiple except blocks are used to handle multiple exceptions
* we cannot write except block without a try block
* we can write try block without except block
* else and finally blocks are not mandate to implement
* else block executes only if there is no exception
* finally block executes always

Program5:

Write a python program to handle multiple exceptions.



O/P1:



Now, edit the above program by giving empty list to avg() like below



O/P2:



**Types of Except Blocks:**

Except block executes only when there is an exception in the try block. except block can be written in various formats

Type1:

To catch exception we can write except block with the ExceptionClass name.

Syn: except ExceptionClass:

Type2:

We can catch exception as an Object.

Syn: except ExceptionClass as Obj:

Type3:

To catch multiple exceptions, we can write multiple exception blocks or in single exception block write all exceptions as tuple inside braces

Syn: except (ExceptionClass1, ExceptionClass2, ExceptionClass3..):

Type4:

We can write exception block without mentioning any exception details

Syn: except:

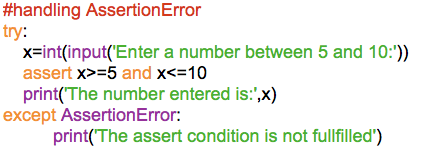
**The Assert Statement:**

The Assert statement is useful to ensure that a given condition is True. If it is not true, It raises AssertionError.

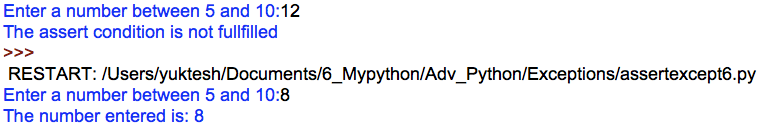
Syn: assert condition: message

Program6:

Write a python program to understand AssertionError.

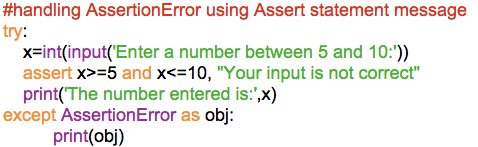


O/P:



Program7:

Write a python program to understand Assert statement with message



O/P:



**User Defined Exceptions:**

- Exceptions defined by programmer are called User Defined Exceptions.

- Why user should define exception though we have built-in exceptions?

- Though we have different types of built-in exceptions there may be some scenarios where none of exceptions available in python are useful. In such cases, we must build our own exceptions.

- Steps to create user defined exceptions:

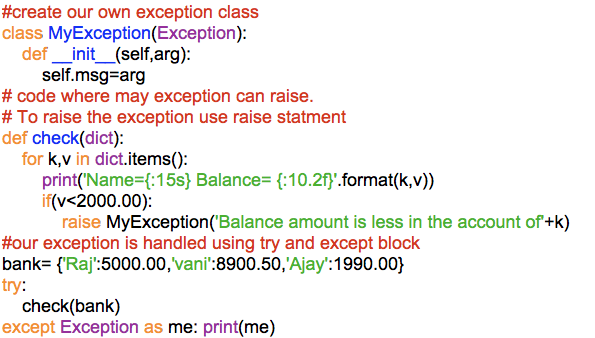
1. User exception class must be sub class of Exception class
2. Write a string parameterized constructor in the class
3. Raise the user defined exception using raise statement

Ex: raise MyException(‘message’)

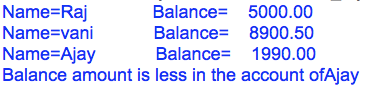
1. Handle the exception inside try and except.

Program8:

Write a Python program to understand user defined exceptions



O/P:



**Logging the Exceptions:**

-Logging is a concept of storing all the error messages raised by a program into a file. This file is called Log file

-Logging helps in debugging the software program

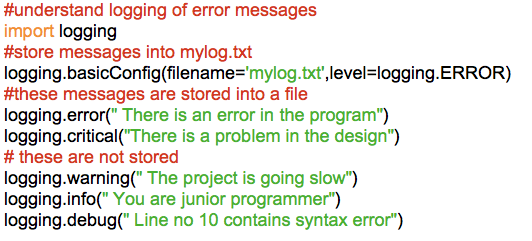
-Python provides a module ‘logging’ that is useful to create a file that stores all error messages of a program into a file

-Errors are classified into 6 levels in logging module depending upon the seriousness of the error.

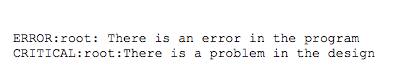
|  |  |  |
| --- | --- | --- |
| **Level** | **Numeric Value** | **Description** |
| CRITICAL | 50 | Very serious error need attention |
| ERROR | 40 | Some serious error |
| WARNING | 30 | Represents a warning message, some caution is needed |
| INFO | 20 | Represents message with some information |
| DEBUG | 10 | Represents a message with debug information |
| NOTSET | 0 | Represents that the level not set |

Program9:

Write a python program to understand the logging of error messages



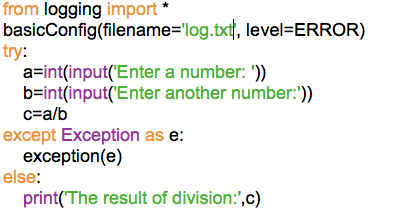
O/P:



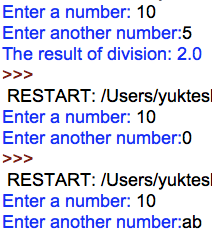
Got created mylog.txt and above message was stored.

Program10:

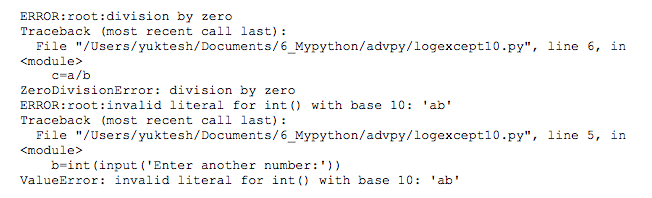
Write a python program to store the messages released by any exception into a log file.



O/P:



In log.txt you can see below content



**File Handling:**

- File is a named memory location in disk, which stores data in permanent manner.

- Python language provides various functions and methods to provide the communication between python programs and files

- Python programs can open the file, perform the read or write operations on the file and close the file.

- We can open the files by calling open() function of built-in’s module

-At the time of opening the file, we have to specify the mode of the file

- Mode of the file indicates for what purpose the file is going to be opened (r,w,a,b)

**File Modes:**

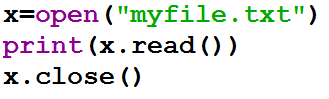
|  |  |
| --- | --- |
| Mode | Description |
| ‘r’ | Open a file for reading (default) |
| ‘w’ | Open a file for writing. Creates a file if it does not exist or truncates the file if it exists |
| ‘x’ | Open a file for exclusive creation. If the file already exists the operation fails |
| ‘a’ | Open for appending at the end of the file without truncating it. Creates a new file if it does not exists |
| ‘t’ | Open in text mode(default) |
| ‘b’ | Open in binary mode |
| ‘+’ | Open a file for updating (reading & writing) |

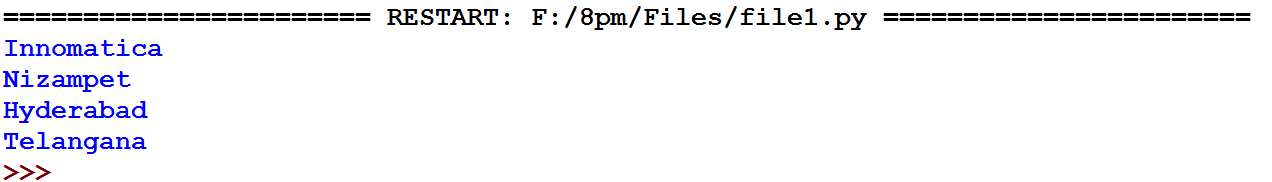
**Reading data from a file:**

-We have open() function which helps in opening the file in specific mode and creates file object.

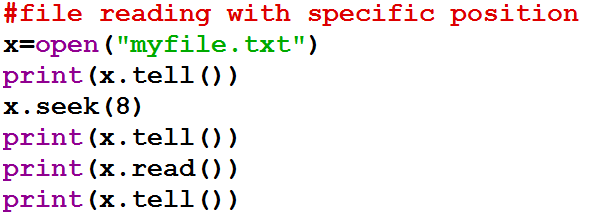
- File object provides various methods and by using those methods we can perform the read or write operations on the files and we can close the files.

Ex1: Open a file and read data.



Output: 

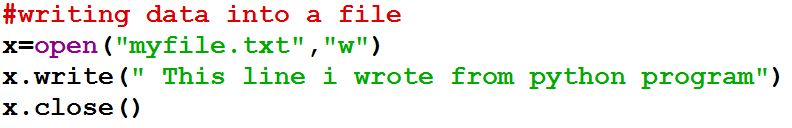
Ex2: file reading from specific position



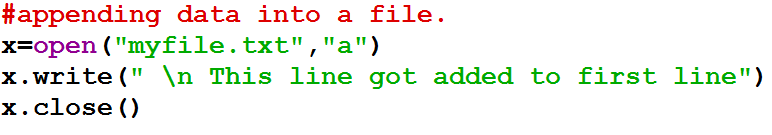
**Writing data into files:**

-We use write() function to write some data into a file

Ex3: Write a python program to write data into a file



Ex4: Write a python program to append data into a file

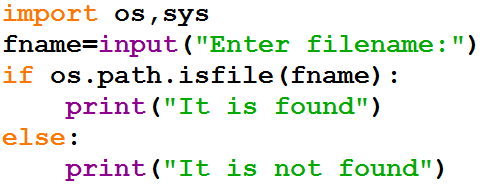


**Knowing a file exists or not:**

-os.path.isfile(fname) helps in finding out whether a file with name fname exists or not in the computer.

-Here os is module and path is sub module in os module and isfile() is method which takes input as file name which we are opening and returns true else it returns false.

Ex:5 Python program to know whether a file exists or not

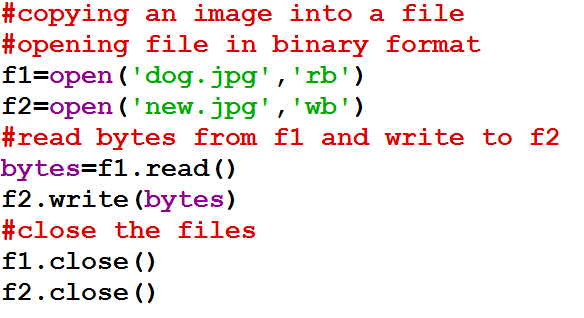


**Working with Binary Files:**

-Binary files handles data in the form of bytes. Here we can store images, av files etc

-We use ‘rb’ mode to read binary file data and ‘wb’ mode to write into binary file

Ex:6 Write a python program to copy an image into a file.

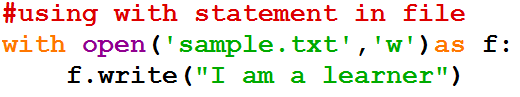


**With statement in files:**

-We can open files using with statements.

-Advantage of file open using with statement is it will takecare of close the file. Therefore it is not required to use close()

Ex:7 Write a python program to understand with statement in files



**Pickle an Unpickle in Python:**

-The concept of storing object data into a binary file is called pickle or serialization.

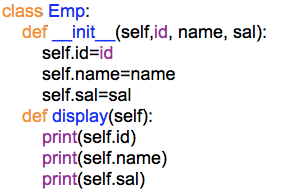
-This we implement by using dump() function of pickle module

- The concept of storing binary file data into object is called unpickling.

- This we implement by using load() function of pickle module

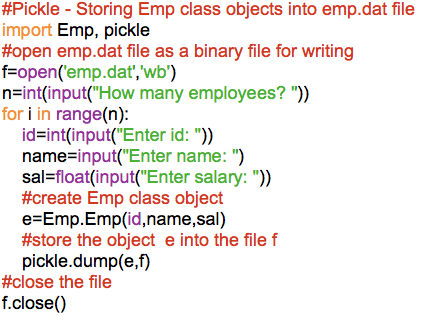
- Before pickle an object we need to create a class

Ex 7: Program to create a class and save it as Emp.py



- Now pickle the Emp class object into a file using dump()

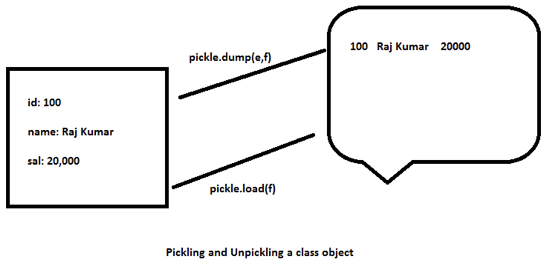
Ex 8: Program to pickle Emp class objects



Op:

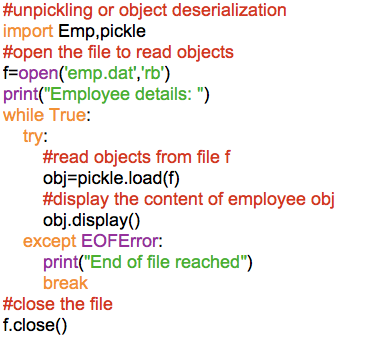


- Open the program directory you can see emp.dat binary file got generated with 2 employee obj details

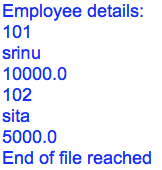


- Now to unpickle emp class object we should use load() function

Ex 9: Program to unpickle Emp class objects



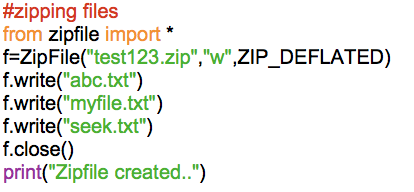
Op:



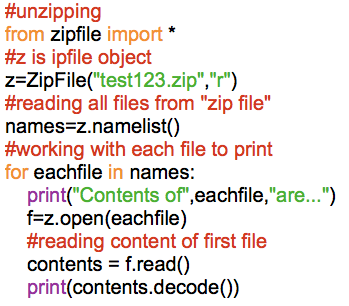
**Zipping and Unzipping of files:**

To implement this concept, we have zipfile module in python

Ex 10: Program to zip the files



Ex 11: Program to unzip the files



**Multithreading:**

- Thread is a functionality or logic which can execute simultaneously along with the other part of the program.

- Thread is a light weight process

- Any program which is under execution is known as process

- We can define the threads in python by overwriting run() method of thread class

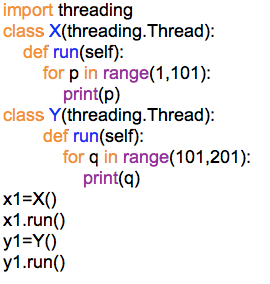
- Thread class is a predefined class which is defined in threading module

- ‘threading’ module is a predefined module is python

- If we call the run() method directly the logic of the run() method will execute as a normal method logic

- To execute the logic of the run() method as a thread, we call start() method of the Thread class.

Ex1: calling run() method of Thread class



op:

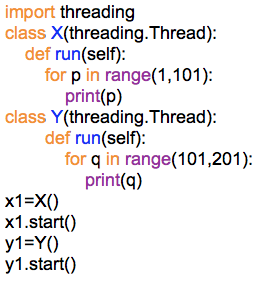
1

2

…

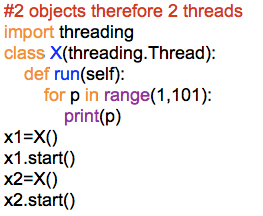
200

Ex2: calling start() method of Thread class



Op: Both threads simultaneously executes

Ex3: Calling start() method more than once



Op:

11

22

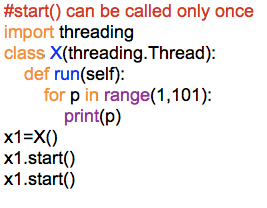
…

9999

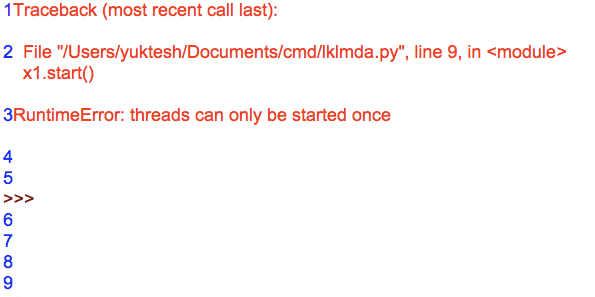
100100

again here run() method will execute as thread, therefore output will be parallel executes.

Ex4: Calling start() more than once

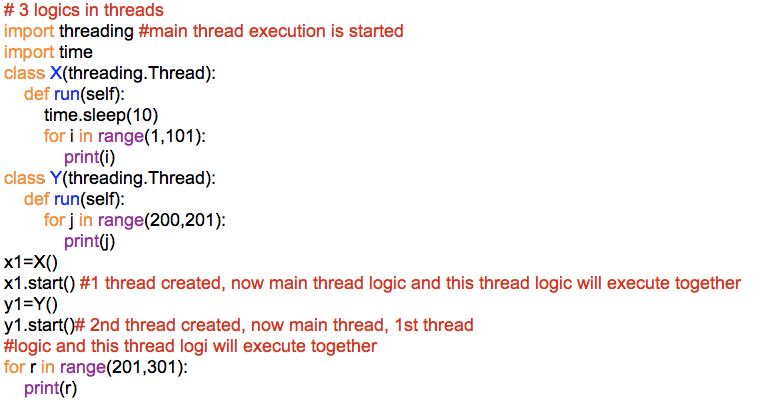


op:



From output we can understand start() method can be called once from every object.

Ex5: Program on multithread execution



op: All 3 thread executes parallel

Understanding detailed execution of threads

- Whenever we run the python program, by default python interpreter creates main thread.

- After creating the main thread, program execution starts from main thread

- In the middle of the execution of the main thread if any other threads are created those threads and main thread execution will takes place one with the other.

- All logics executes mixed together

- As of now what all programs we wrote, for all 1 thread got created i.e., called main thread.

- We prefer to work with threads to execute programming logic simultaneously.

- We know threads are executing in above program, but we never know in which fashion these threads are executing.

- Python Interpreter allocates some time to each thread to execute.

- While executing in between python interpreter can give chance for other thread to execute.

- This and all python interpreter does internally which we cannot able to know as a developer.

- In Multiple threads, which thread should start first and how much time it should execute and next what thread to execute, this python interpreter performs through scheduling algorithms of OS.

- Every operating system consists of scheduling algorithms which follow its own dynamic scheduling.

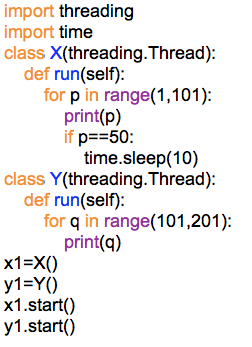
Suspending the execution of the threads in the middle of the execution temporarily.

- We can suspend the execution of the thread temporarily in the middle of its execution by calling sleep() function

- sleep() function is a predefined function which is defined in time module

- sleep() suspend the execution of the current thread until specified time is over.

Ex6: Suspending thread execution using sleep() function



Op:

First thread will print upto 50 and stops for 10 sec and then prints other 51 to 100. Whereas second thread will print 101 to 201 parallely.

Thread Synchronization:

- The concept of avoiding multiple threads to access the same logic at a time is called as Synchronization.

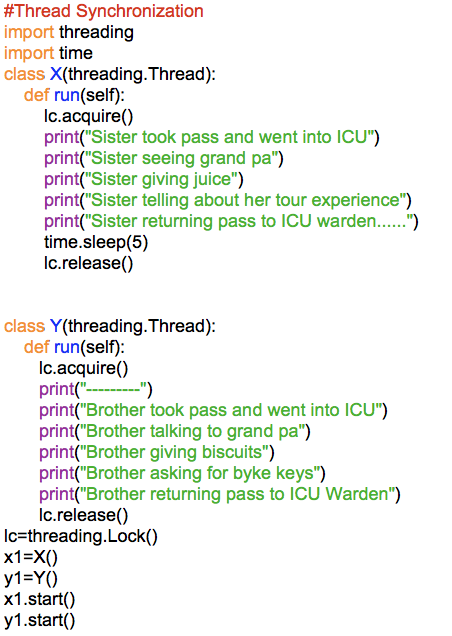
- Synchronization can be implemented by calling acquire() & release() methods of lock class

- lock class is present in threading module

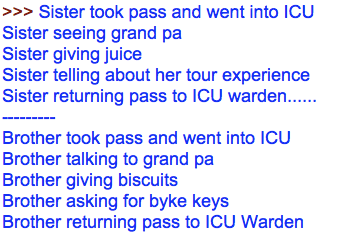
- Synchronization degrades the performance of the application

- If any specific business requirements need synchronization then only implement synchronization.

Ex7: Program to perform Thread Synchronization



Op:



Deadlock of Threads:

- Even we synchronize the thread there is possibility of other problems like deadlock.

- When a thread has locked an object, and waiting for another object to be released by another thread, and the other thread is also waiting for the first thread to release the first object, both the threads will continue waiting forever. This is called thread ’deadlock’.

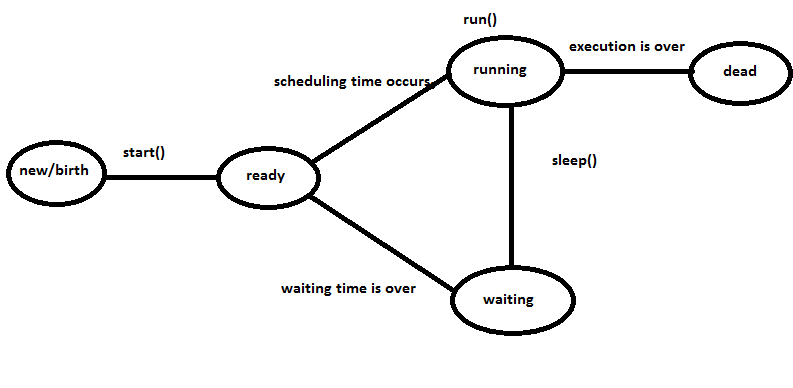
Avoiding Deadlocks in a program:

-There is no specific solution for the problems of deadlocks.

- It depends on the logic used in program by developer.

- The developer should design his program in such a way, that is does not occur any dead lock

Thread Life Cycle:



-Creating the object of a class which is over writing run() method of Thread class is known as creating a thread.

- Whenever thread is created then we call thread is in new state.

- Whenever we call the start() method on the new state threads then those threads will be forwarded for scheduling.

- The threads which are forwarded for scheduling are known as ready state threads

- Whenever scheduling time occurs, ready state thread starts execution

- Whenever sleep() is called on the running state threads then immediately those threads will wat

- The threads which are waiting will sent to wait state. They are called waiting state threads

- Whenever waiting time is over or specified thread execution is over then immediately waiting state threads are forwarded for scheduling.

- If running state threads execution is over then immediately those threads execution will be terminated

- The threads which execution is terminated is known as dead state threads.

**REGULAR EXPRESSION:**

- Regular expressions are nothing but some special characters

- Every regular expression character contains some special meaning.

- By using regular expressions characters, we can define regular expression patterns

- To work with the regular expression patterns, python provides one predefined module called *‘re’*

- By using regular expression patterns, we can extract the required information from the given data, we can perform the data format validations at client side and we can define the url patterns for the web pages.

Regular expression characters:

1. \* 🡪 it matches zero or more occurrences of its preceding character

Ex: ab\*c

ac

abc

abbc

abbbbbbbbc

ab #error

1. + 🡪 it matches 1 or more occurrences of its preceding character

Ex: ab+c

ac # error

abc

abbc

abbbbbbbbbc

1. ? 🡪 it matches zero or one occurrences of its preceding character

Ex1: ab?c

ac

abc

abbc # error

Ex2: pea?rl

perl

pearl

Ex3: Colou?r

Color

Colour

Colouur #error

1. . 🡪 It matches any single character

Ex1: a.c

a$c

a$c

a\_c

abbc # error

1. [] 🡪 It matches any single character in the given list

Ex1: b[aeiou]d

bad

bed

bid

bod

bud

bxd # error

b8d # error

1. [^] 🡪 It matches any single character other than in the given list.

Ex1: b[^aeiou]d

bad #error

bed #error

bid #error

bod #error

bud #error

bxd

b8d

1. [-] 🡪 It matches any single character in the given range.

Ex1: x[a-e]y

xay

aby

xcy

xdy

xey

xpy #error

xwy #error

Ex2: x[0-3]y -> any single digit

x0y

x1y

x2y

x3y

x4y #error

Ex3: [a-z]-> any 1 lowercase alphabet

Ex4: [A-Z] -> anyone upper case alphabet

Ex5: [a-zA-Z] anyone alphabet

Ex6: [a-zA-Z0-9] ->any alphanumeric

Ex7: [^0-9] -> any single non-digit

Ex8: [^a-z] -> any one non-lowercase alphabet

Ex9: [^A-Z] -> any one non-uppercase alphabet

Ex10: [^a-zA-Z] -> any one non-alphabet

Ex11: [^a-zA-Z0-9] -> any one non-alphanumeric

(special character)

1. (|) 🡪 It matches any one string in the list.

Ex1: (java|Hadoop|python)

1. {m} 🡪 It matches exact occurrence of its preceding character.

Ex1: Ab{3}c #here b repeats 3 times

Abc #error

Abbc #error

Abbbc

Abbbbbbc #error

10){m,n} 🡪 It matches min m occurrences and max ‘n’ occurrences of its preceding characters

Ex1: Ab{3,5}c

Abbc #error

Abbbc

Abbbbc

Abbbbbc

Abbbbbbbbc #error

11){m,} 🡪 it matches min m occurrences and max no limit of its preceding characters

Ex1: Ab{3,}c

Abbc #error

Abbbc

Abbbbbbbc

12)^ 🡪 start of the line

Ex1: ^hello, ^abc, ^[^abc]

13)$ 🡪 end of the line

Ex1: hello$,

[0-9]$ # single digit

14)\d or [0-9] 🡪 any single digit

Ex1: [0-9][0-9][0-9][0-9]

or

[0-9] {4}

or

\d \d \d \d

or

\d{4}

15)\D or [^0-9] 🡪 any single non-digit

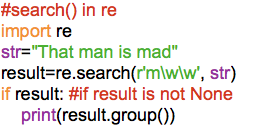
16)\w or [a-zA-Z0-9] 🡪 any alphanumeric

17)\W or [^a-zA-Z0-9] 🡪 any non-alphanumeric or special character

18)\s 🡪 ‘ ‘, ‘\t’, ‘\n’

* By using regular expression special character, we can define the regular expression patterns
* The sequence of regular expression characters represented in “ “ and it is preceded by the ‘r’ character is known as a regular expression
* by calling predefined func/method of re module we can extract the pattern matching information from the given data

Ex1: Program to search for the string start with ‘m’ and have total 3 characters.

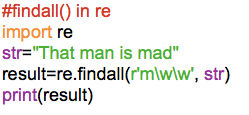


op:



search() method retrieves only first occurrence of string.

Ex2: Same above program using findall()

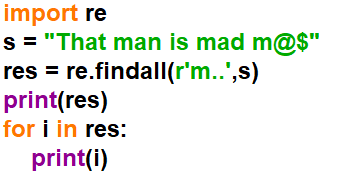


op:



findall() method retrieves all the possible occurrences.

Ex2.1#if we need all character possibility



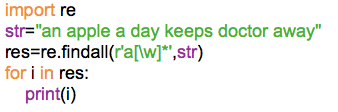
Ex3: Program to understand sub()



op:



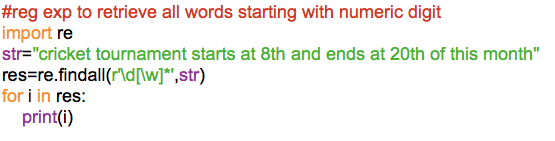
Ex4: Create regular expression to retrieve all words starting with a given string



op:



Ex5: Program to create regular expression to retrieve all words starting with a numeric digit.



op:



Ex6: Program to create regular expression to display month and date in required format.

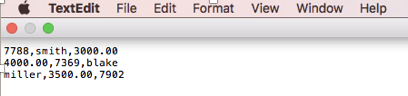


op:

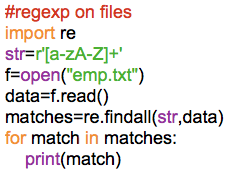


Regular expression on files:

create emp.txt file with below data



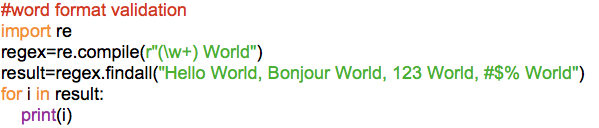
Ex7: Program to apply regular expression on this file data.



op:



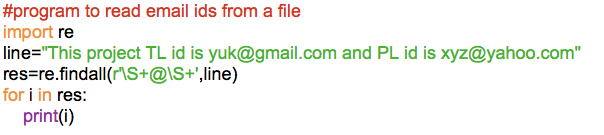
Ex8: Program to validate word format.



op:



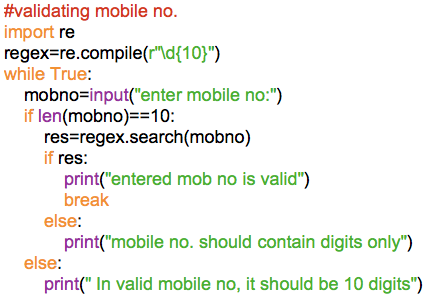
Ex9: Program to create a regular expression that reads email-ids from a line



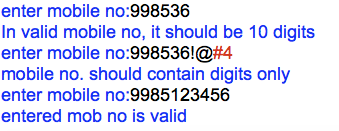
Op:



Ex10: Program to find the given mobile number is correct or not:



Op:



**Python Database Connectivity**

- Database is a software which helps in storing data in permanent manner.

- We prefer to store in database for the effective utilization and manipulation.

- Databases provide security for the data

- Different databases are Oracle, Mysql, DB2, SQLITE3, SqlServer and so on.

- To provide communication between python program and database, we use database related external modules.

- Database related external module we can install by using “pip application”

- Download & Install Oracle database 11g or 12c enterprise edition from the [www.oracle.com](http://www.oracle.com)

Configuring Database:

Open command prompt and type sqlplus

In cmd:

---------

c:\user:\db>sqlplus

sql>UID: sys as sysdba

password: oracle\_999 or what you gave while installation or directly click enter button for 2 times

sql>alter session set”\_oracle\_script”=true;

sql>create user **yuktesh** identified by **yuktesh123**

Note: In place of yuktesh and yuktesh123, you can give your own uid and pwd

sql>grant dba to Yuktesh;

sql>commit;

Now, a new user got created and granted required permission to work with database.

- Ensure latest version of python is running in your computer.

- Download and install the “cx\_Oracle” module by using following command in prompt.

Note: Your computer should connect to internet to install any module.

c:\user\hp>cd..

c:\user>cd

c:\cd python 36

c:\python36>cd scripts

c:\python36\scripts>pip install cx\_Oracle

-We can established connection with the database by calling connect function of cx\_Oracle module.

Syntax: import cx\_Oracle

con=cx\_Oracle.connect(‘dbusename’,’dbpwd’,’ipadd of the computer where database is installed: portno/database service name’)

- connect function established the connection with database with the given details, store the connection details into connection object and that object address will be stored into the given variable.

- After establishing the connection with database to send the sql queries to the database we have to create “cursor object”

- cursor object can be created by calling cursor() method of the connection object.

Syntax: cur=con.cursor()

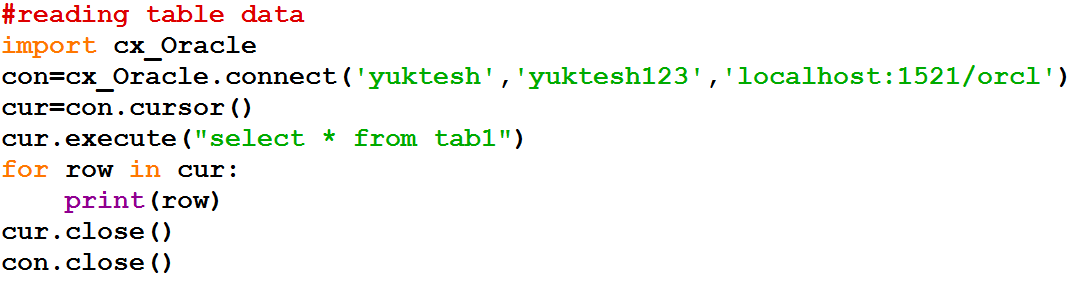
-By calling method of cursor object, we can execute the sql queries

- By calling the methods of cursor objects we can perform sql operations on the database

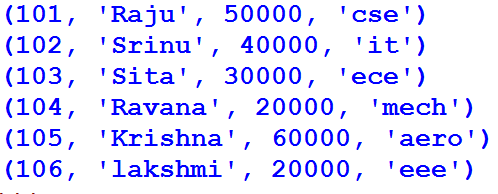
- After reading data from the database or after writing data into data base we need to close the cursor object by calling close() method of cursor object

- After closing the cursor object we have to close the connection object by calling close() method of connection object.

Program1: Python program to read a database table data



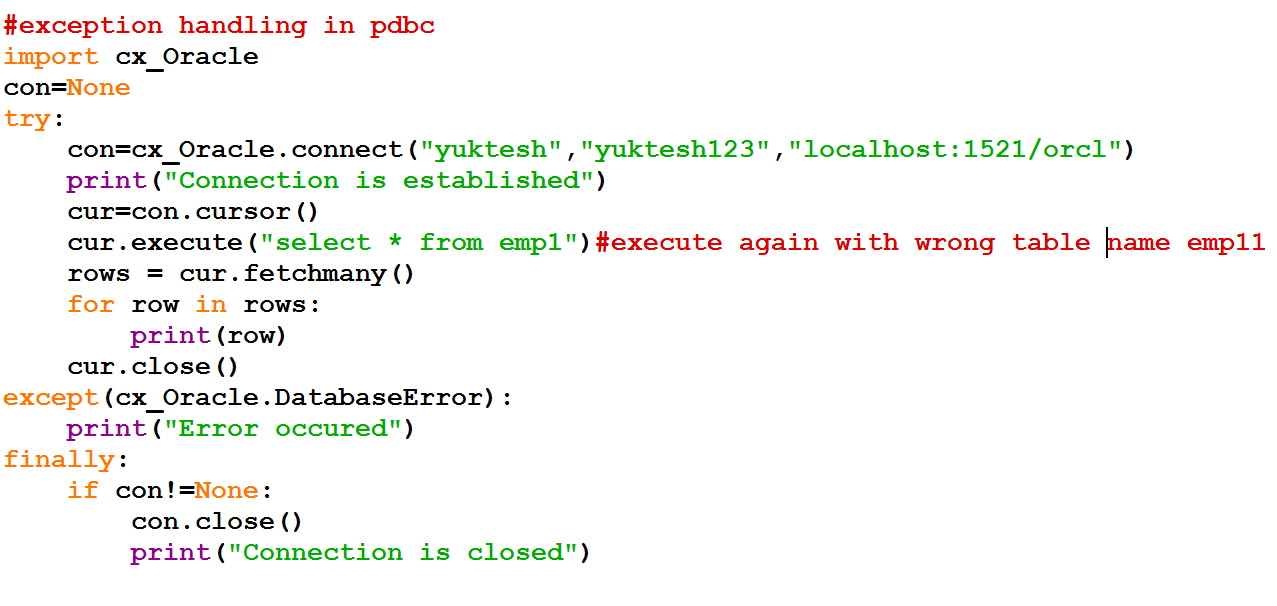
**Op:**



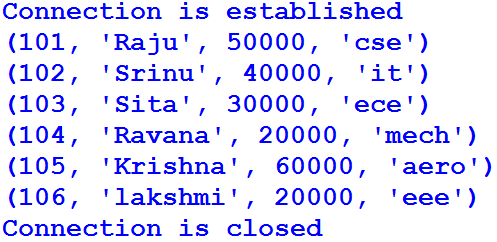
- Whenever connection is established with the database connection should get close before terminating the program.

- To close the connection if the connection is established we use exception handling techniques.

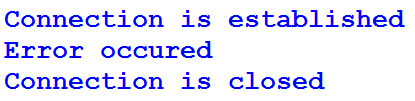
Program2: Handling exception while reading table data



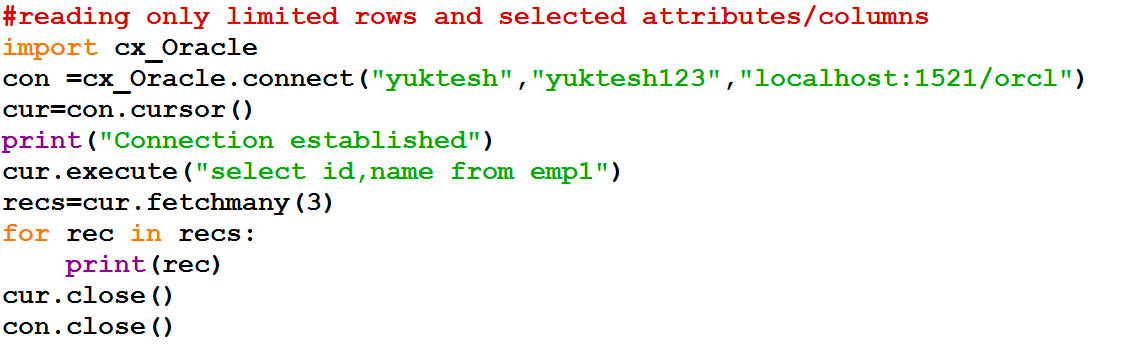
First time op:



Second time o/p after giving wrong table name as emp11



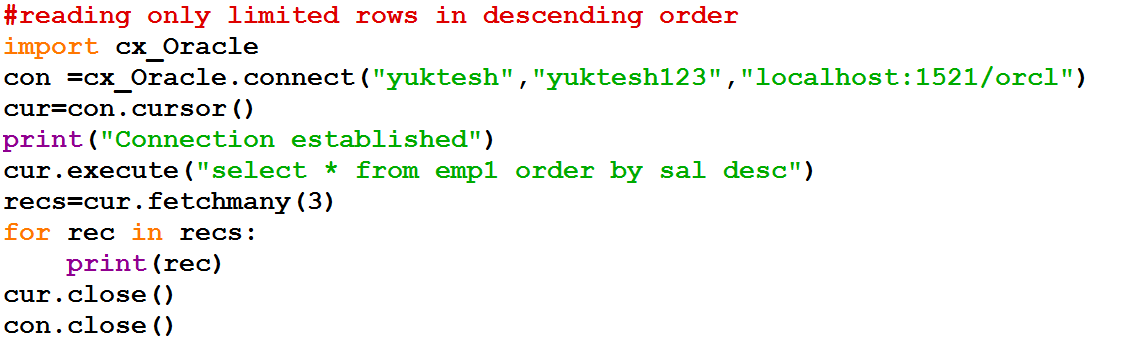
Program3: Python program to display limited rows of selected attributes



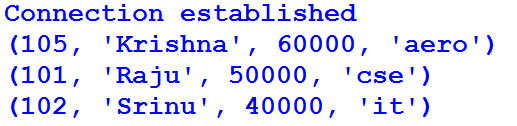
Op:



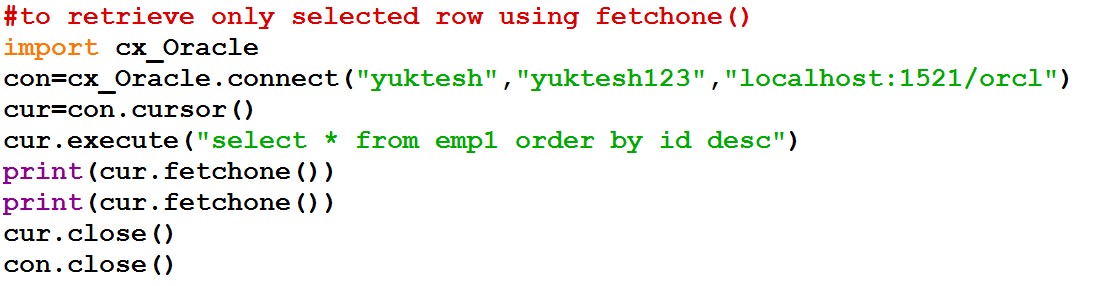
Program4: Modifying above program and displaying emp1 table data in descending order



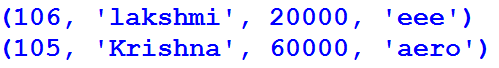
**Op:**



Program5: Python program toretrieve selected rows using fetchone()



Op:



**Executing dml commands through python programs:**

- dml commands are used to modify the data of the database objects.

- Whenever we execute dml commands, the records are going to be modified temporarily.

- Whenever we run “rollback” command the modified records will come back to its original state.

- To modify the records of the database objects permanently we use “commit” command.

- After executing the commit command even though we execute “rollback” command, the modified records will not come back to its original state.

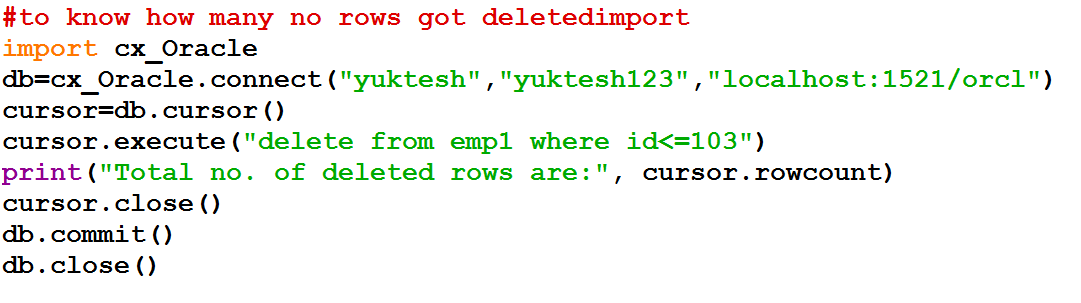
- Create the emp100 table in the database by using following command

create table emp100 as select \* from emp1;

- Whenever we run the dml command through the python program, then the no. of records which are modified because of that command will be stored into the rowcount attribute of cursor object.

- After executing the dml command through the python program we have to call commit() method of cursor object.

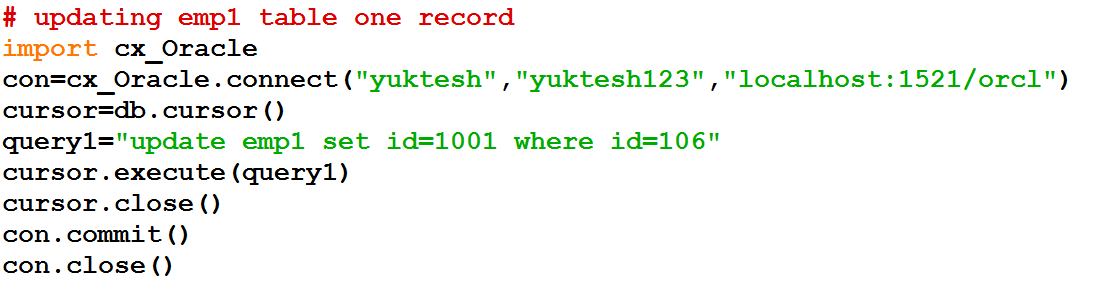
Program6: Python program to know how many rows got deleted from the table



Op:

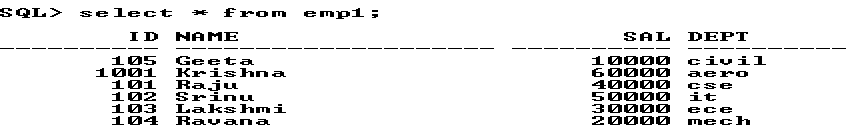


Program7: Python program to update a record in the table – dml command

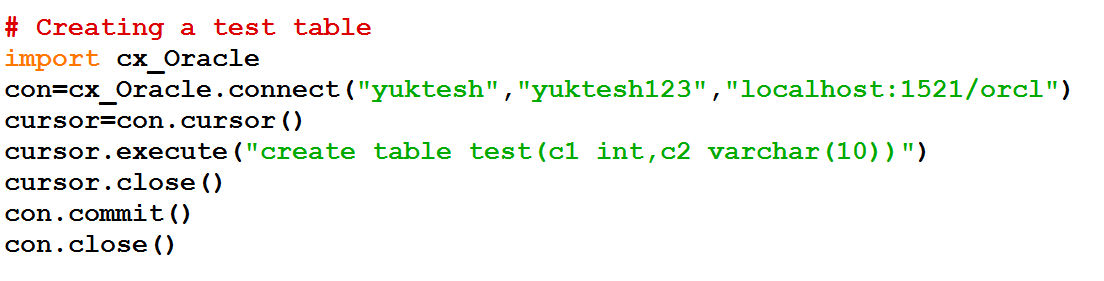


Op:

Nothing printing in output, but go to table and display emp1 data you can see the change



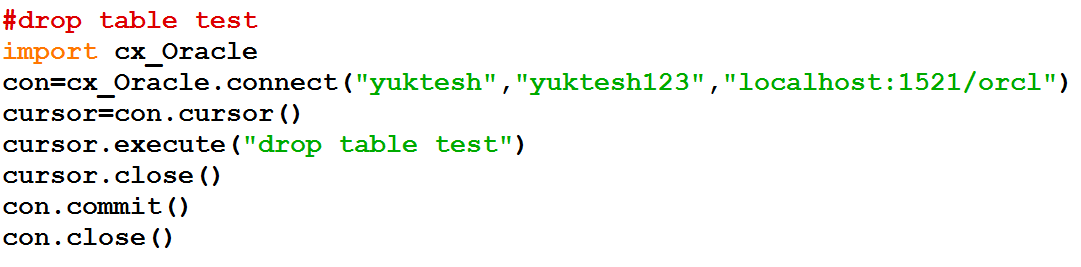
Program8: Program to send same sql query more than once.



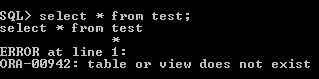
Op:



Program 9: Python program to delete a table from database



Op:

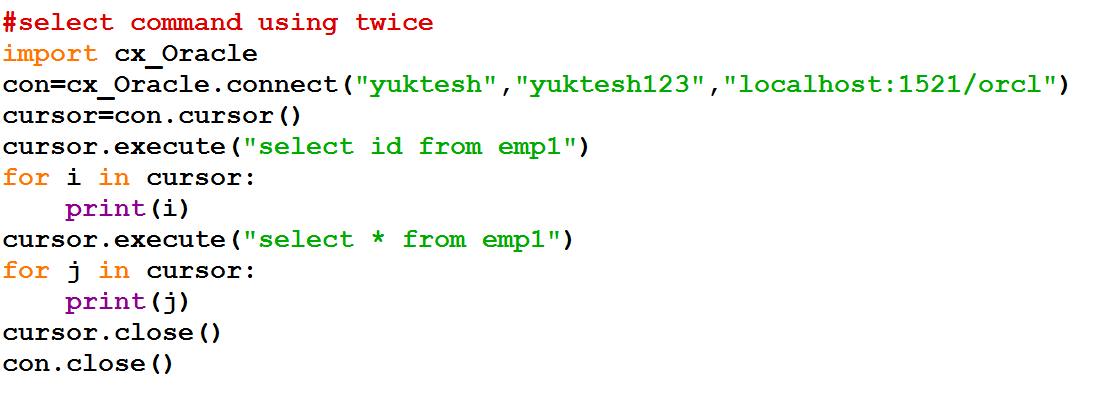


-Whenever we send same sql query for multiple times by changing the data by calling execute() method then internally two operations will take place at database with respect to everytime calling of execute() method.

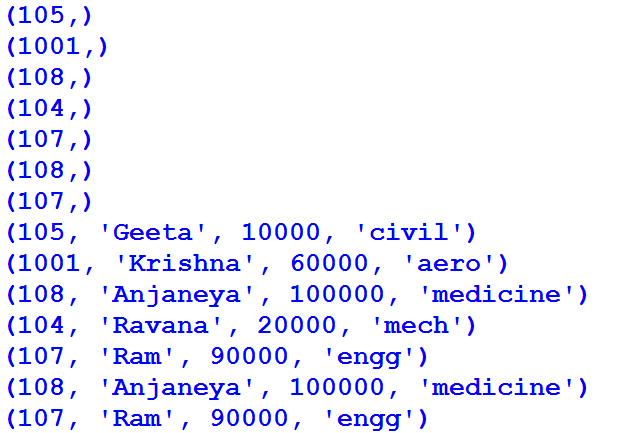
1. query compilation

2. query execution

Program10: Python program to display data 2 times from the table without prepare()



Op:



- Even though we send the same sql query to the database for multiple times by calling execute() method every time query compilation will be performed in the database so that performance of the application is degraded.

- To overcome the above problem we use prepare() of cursor object

- We can send the sql queries with bind variables to the database by calling prepare() method of cursor object.

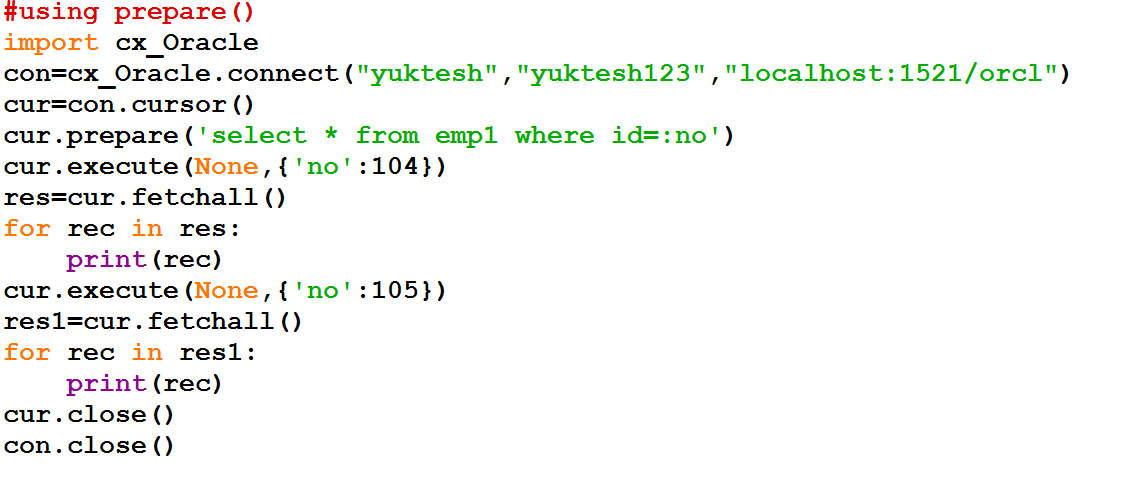
- Whenever query forwarded to the database through the prepare() method in the database that query will be compiled and compiled query will be stored in database

- Whenever values are submitted to the bind variables by calling execute() method of cursor object then query will be executed in the database

- We can submit values to the bind variables for ‘n’ no. of times

- With respect to every time submission of values to the bind variables, only query execution will take place in database.

Program



Op:



**DATA and TIME**

Date and Time are very important things. For example

1. Employee experience we count depending on the joining data and time
2. Age of an employee calculated is calculated using his birthday
3. Bank people pay interest amounts on the deposits depending up on the days lapsed
4. Employees work timings on the project depends on the release date and time of the project.

Therefore, it is important for us to know how to work with Data and Time in python.

Python provides 3 modules to work with date and time concept they are

1. datetime
2. time
3. calendar

datetime module is the important module that deals with dates along with times. There are 4 important class are available in datetime module. They are

1. datatime class
2. date class
3. time class
4. timedelta class

datatime class:

It handles the combination of date and time. attributes of this class are year, month, day, hour, minute, second and microsecond

date class:

It handles dates of calendar without taking time zone into consideration. Attributes of this class are year, month and date

time class:

It handles time in the format of 24hr\*60min\*60sec

attributes of this class are hour, minute, second,

microsecond

timedelta class:

It helps in handle the durations. The duration may be the difference between two dates, time or datetime instances.

Epoch:

It is a point where time starts.

In windows computers, this point is taken as 0.0 hours of January first of the current year

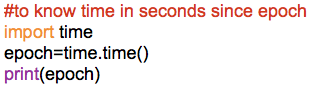
For Unix, the epoch is 0.0 hours of January 1st of 1970

Finding time in Seconds:

It is possible to measure time in seconds as we have time() method in time class.

**Program1:**

Write a python program to find time in seconds since epoch.



O/P:



Above result is number of seconds from the beginning of this year.

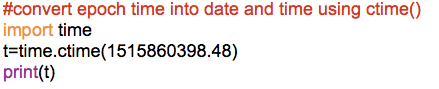
Above result is not much helpful as it is tough read.

Finding date and time from epoch time:

To find date and time using epoch time we have ctime() from the time module

**Program2:**

Write a python program to convert epoch time into corresponding date and time.



O/P:



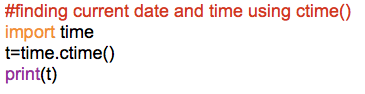
Finding Current Date and Time:

There are 3 ways to find the current date and time of our computer

1. using ctime() function of time module
2. now() method of datetime class of datetime module
3. today() method of datetime class of datetime module

**Program3:**

Write a python program to know the current date and time using ctime() method

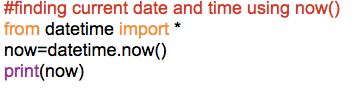


O/P:



**Program4:**

Write a python program to know date and time using now() method

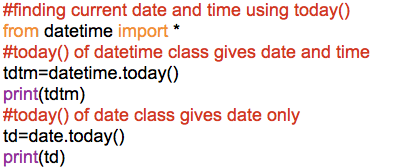


O/P:



**Program5:**

Write a python program to find date and time using today() method.



O/P:

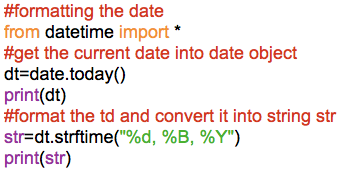


Formatting Dates and Times:

We can format the date and time outputs in our own required formats using a strftime()

**Program6:**

Write a python program to understand date formatting

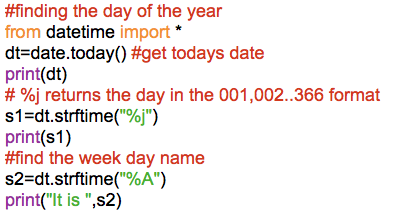


O/P:

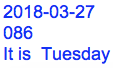


**Program7:**

Write a python program to find the day of the year and week day name

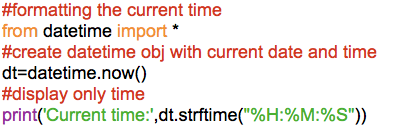


O/P:



**Program8:**

Write a python program to modify the time in our format



O/P:

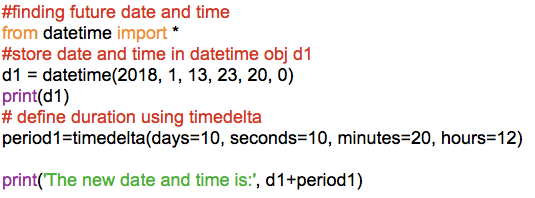


Finding Durations using ‘timedelta’:

timedelta class of datetime module is used to find difference between the two dates and finding the future date by adding some period to current date.

**Program9:**

Write a python program to find the future date and time from an existing date and time



O/P:

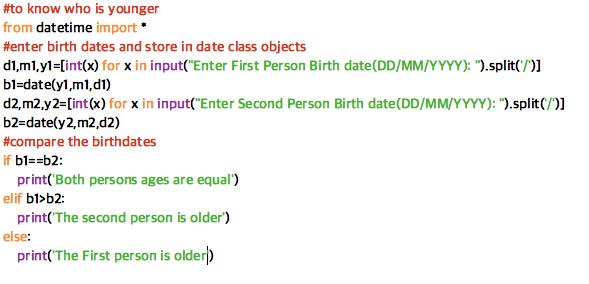


Compare Two Dates:

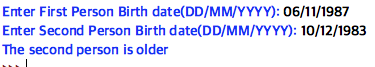
We can compare two dates just like numbers.

**Program10:**

Write a python program to find younger of two persons.



O/P:



Sorting Dates:

We can sort dates using sort(). Here we can take different date class obj in a list using append() and we can sort list.

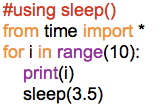
Stopping Execution Temporarily:

We can use sleep() to suspend the execution of the program. We need to pass seconds as parameter to sleep().

When this sleep() comes across by PVM it will stop that execution of the program that many seconds

**Program11:**

Write a python program to display 10 numbers with each gap of 3 seconds.



O/P:



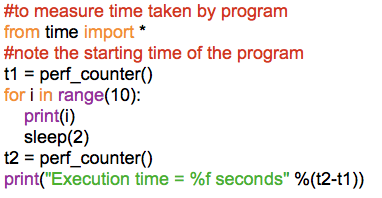
Finding the program execution Time:

There are two fuctions provided by time module which helps in measuring the time difference between two points of the program

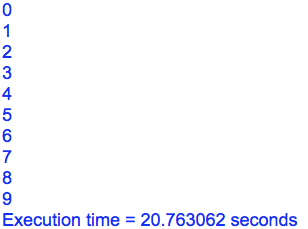
1. perf\_counter(): It helps in returning exact time how much program has taken to execute in fractional seconds
2. process\_time(): It also does same as perf\_counter along with it includes CPU processing time as well. It gets differ for CPU to CPU

**Program12:**

Write a python program to find the execution time of a program



O/P:

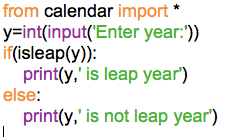


Working with Calendar Module:

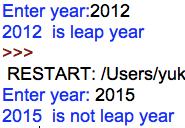
Using calendar module, we can print any month any year. Also, we can find weather a given year is leap year or not using isleap()

**Program13:**

Write a python program to find whether a given year isleap year or not.

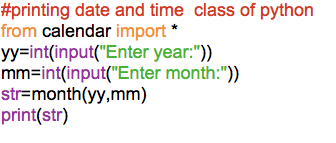


O/P:

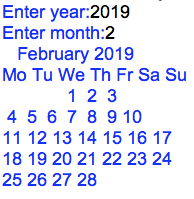


**Program14:**

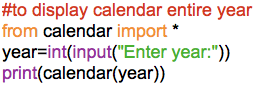
Write a python program to display the calendar for a given month and year.



O/P:

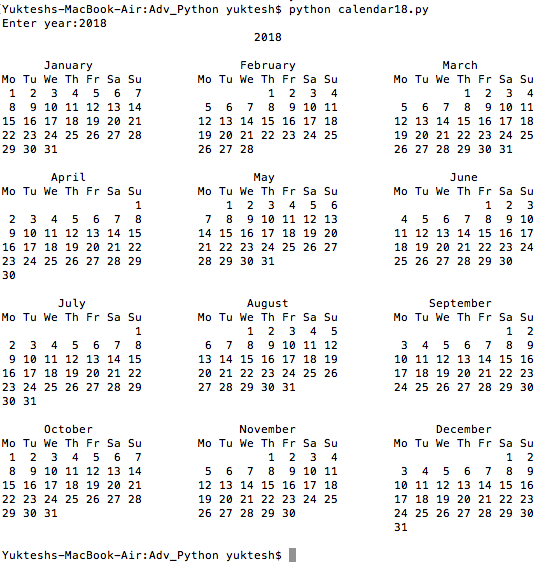


**Program15:** Write a python program to display the calendar year.



Op:

Enter year:2018



**GUI**

In 2 ways we can interact with applications

1. CUI (Character User Interface or Command User Interface)
2. GUI (Graphical User Interface)

- Interacting with applications using GUI has following advantages

1) It user friendly, users can easily interact with applications

2) It adds attraction and beauty to any application by adding different pictures

3) It is possible to simulate real life objects using GUI. Ex: Calculator

4) GUI helps to create graphical components like push buttons, radio buttons, check buttons, menus, etc

**GUI in Python:**

- Python provides tkinter module to develop GUI program. The tkinter represents toolkit interface for GUI

- Every python GUI program should follow these steps.

1. First of all, we should create the root window. The root window is the top level window that provides rectangular space on the screen where we can display text, colors, images, components etc

2. In the root window, we have to allocate space for our use. This is done by creating a canvas or frame. So, canvas and frame are child windows in the root window.

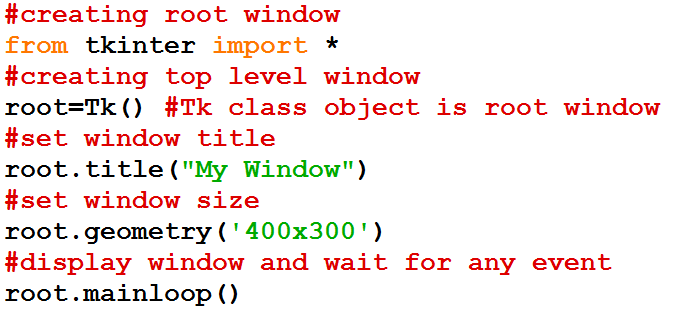
3. Generally, we use canvas for displaying drawings like lines, arcs, circles, shapes, etc. We use frame for the purpose of displaying components like push buttons, check buttons, menus, etc. These components are also called ‘Widgets’.

4. When the user clicks on the widget like push button, we have to handle that event. It means we have to respond to the events by performing the desired tasks.

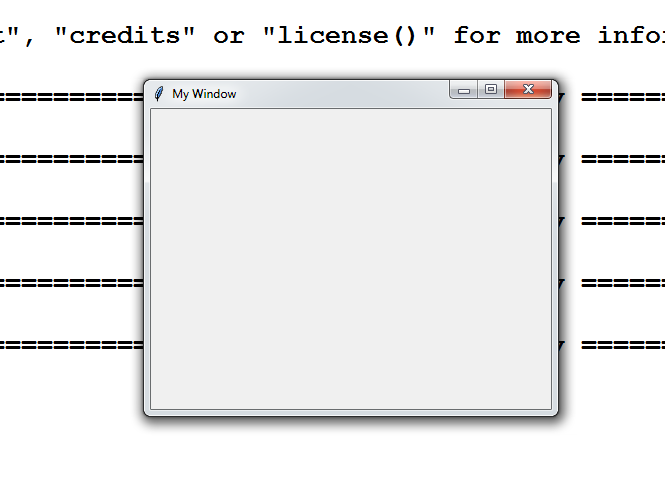
**The root window:**

To display the graphical output, we need space on the screen. This space that is initially allocated to every GUI program is called ‘top level window’ or ‘root window’. We can say that the root window is the highest level GUI component. We can reach this root window by creating an object to Tk class

Program1: Creating a root window with title and size.



Op:



**Frame:**

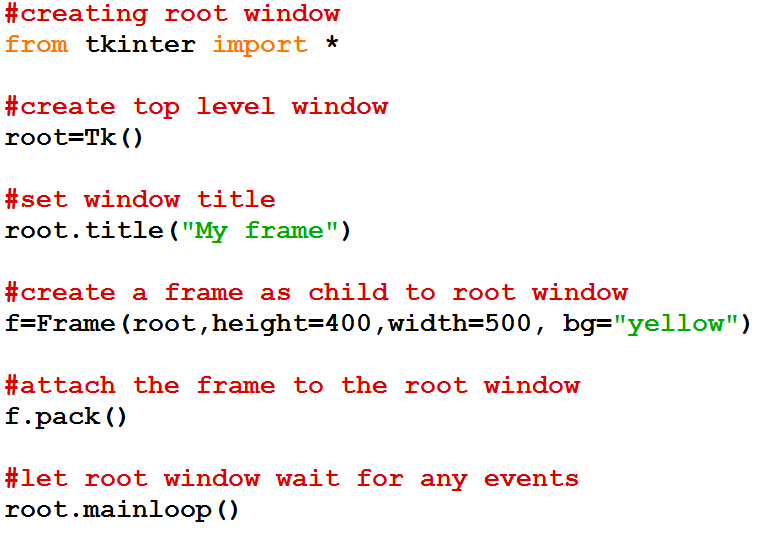
A frame is similar to canvas that represents a rectangular are where some text or widgets can be displayed. Our root window is in fact of frame. To create a frame, we can create an object of Frame class, as

Syn: f=Fram(root,height=400,width=500,bg=”yellow”)

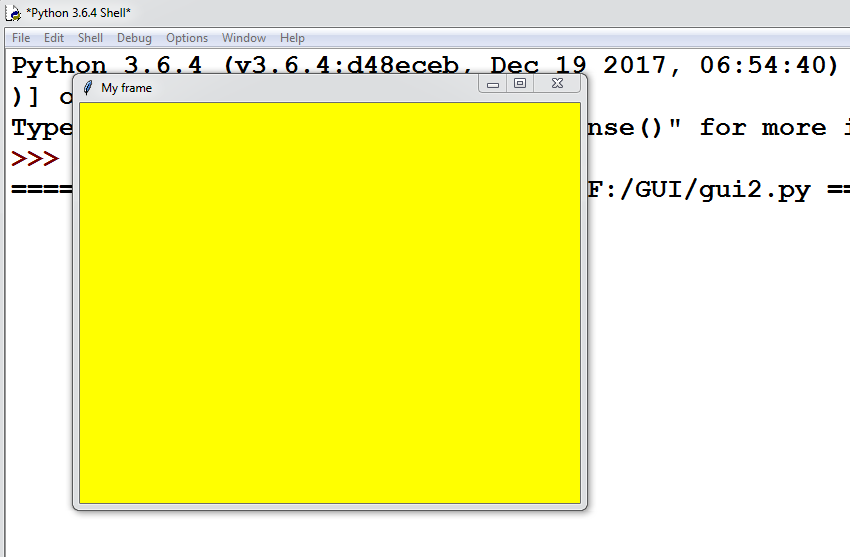
Once the frame is created it should be added to the root window using pack() method, as

f.pack()

Program2: To create frame with some background color and title.



Op:



**Widgets:**

A widget is a GUI component that is displayed in the frame and can perform a task as desired by the user. All widgets are objects. For example, a push button is a widget that is nothing but an object of Button class. Similarly, label is a widget that is an object of Label class. Once a widget is created, it should be added to frame. The following are important widgets in Python:

* Button
* Label
* Message
* Text
* Scrollbar
* Checkbutton
* Radiobutton
* Entry
* Spinbox
* Listbox
* Menu

In general, working with widgets takes the following 4 steps:

1.Create the widgets that are needed in the program. As an example, suppose we want to create a push button, we can create an object to Button class, as:

b=Button(f,text=’My Button’)

Here ‘f’ is Frame object to which the button is added. ‘My Button’ is the text that is displayed on the button.

2.When the user interacts with a widget, he will generate an event. For example, clicking on a push button is an event. Such events should be handled by writing functions or routines. These functions are called in response to the events. Hence they are called ‘callback handles’ or ‘event handlers’. Other examples for events are pressing the enter button, right clicking the mouse button etc. As an example, let us write a function that may be called in response to button click.

def buttonClick(self):

print(“You have clicked me”)

3. When the user clicks on the push button, that ‘clicking’ event should be linked with the ‘callback the handler’ function. Then only the button widget will appear as if it is performing some task. As an example, let us bind the button click with the function, as

b.bind(‘<Button-1>’,buttonClick)

Here, ‘b’ represents the push button.<Button-1> indicates the left mouse button. When the user presses the left mouse button, the ‘buttonClick’ function is called as these are linked by bind() method in the preceding code.

4. The preceding 3 steps make the widgets ready for the user. Now, the user has to interact with the widgets. This is done by entering text from the keyboard or pressing the mouse button. These are called ‘event loop’. These events are continuously monitored by our program with the help of a loop, called ‘event loop’. In our programs, the mainloop()

**Button:**

A push button is a component that performs some action when clicked. Push buttons are created as objects of Button class, as:

b=Button(f,text=’My Button’, width=15, height=12,bg=’yellow’,fg=’blue’, activebackground=’green’, activeforeground=’red’)

we can also display an image on the button as shown in the following two steps:

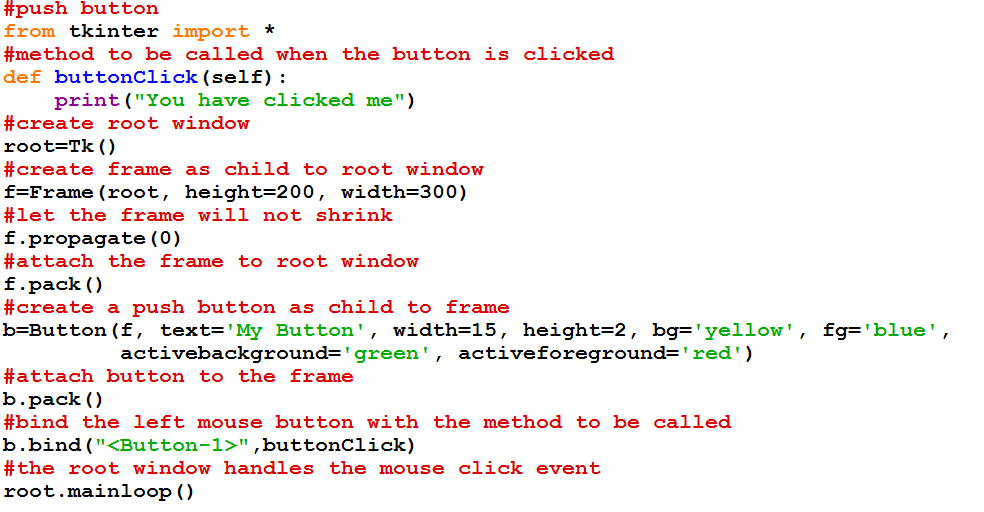
#first load the image into a file1

file1 = PhotoImage(file=”cat.gif”)

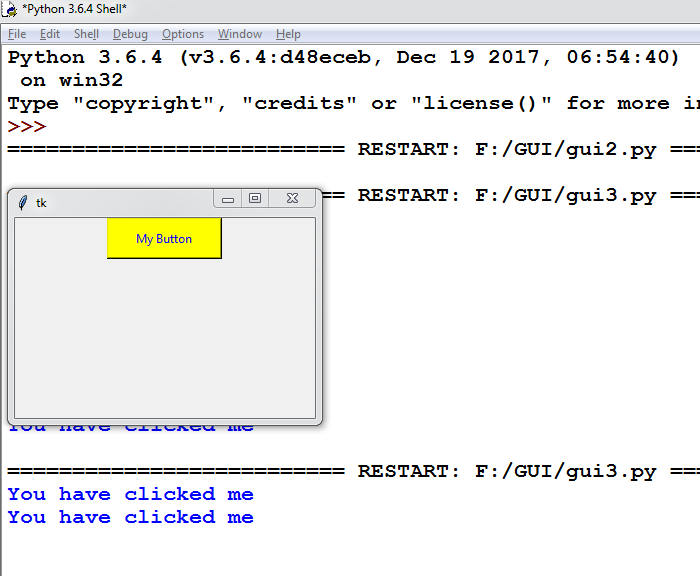
#create a push button with image

b=Button(f, image=file1, width=150, height=100, bg=’yellow’, fg=’blue’, activebackground=’green’, activeforeground=’red’)

Program3: To create a push button and bind it with an event handler function.



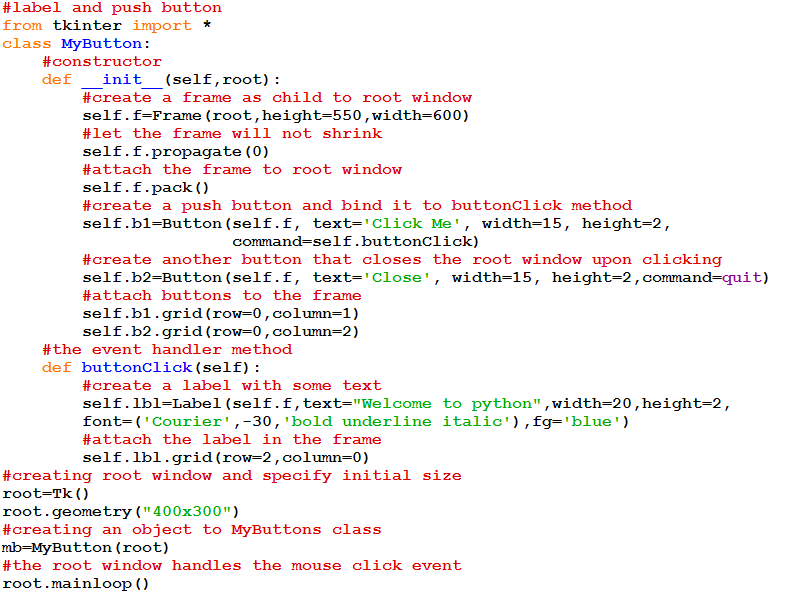
Op:



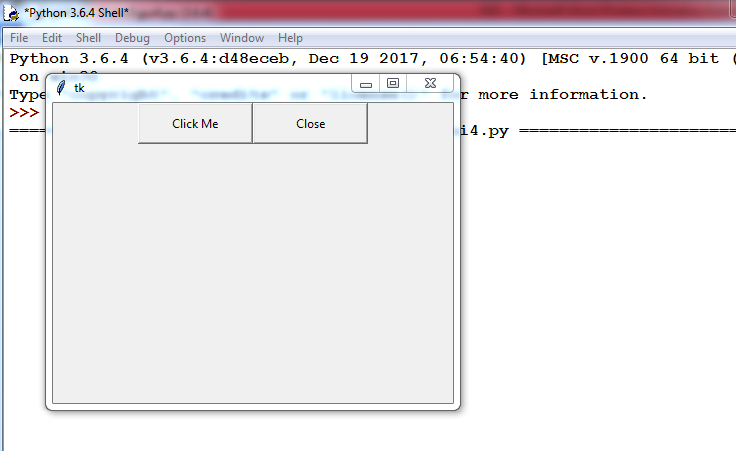
NOTE: We can eliminate bind() method and we will use ‘command’ option to link the push button with event handler function, as:

b=Button(f,text=’My Button’, width=15, height=2, bg=’yellow’, fg=’blue’, activebackground=’green’, activeforeground=’red’,command=buttonClick)

Program 4: Rewriting the above program to create a push button using command option and class concept.



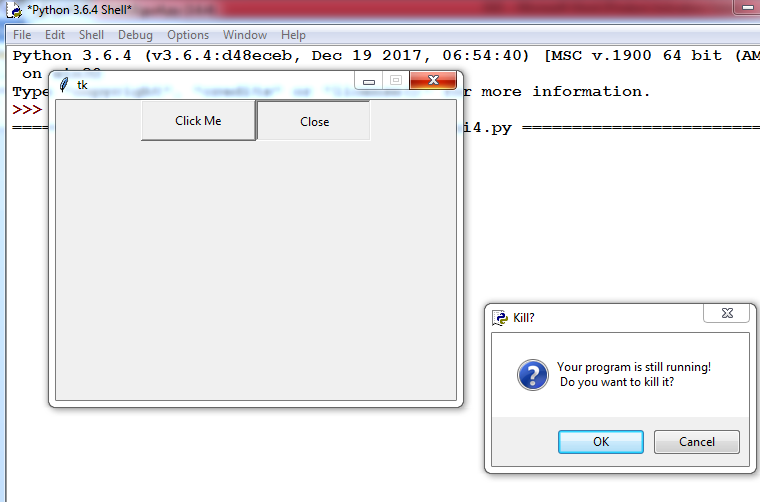
Op:

’

When I click on ‘click Me’ button below got displayed.



When I clicked on close button below message get displayed.



Clicking on ‘OK’ will close the window

**Message Widget:**

A message is similar to label. But messages are generally used to display multiple lines of text where as a label is used to display a single line of text. All the text in the message will be displayed using the same font. To create a message, we need to create an object of Message class, as:

m=Message(f, text=”This is a message that has more than 1 line of text.”, width=200, font=(‘Roman’,20,’bold italic’), fg=’dark goldenroad’)

Here, ‘text’ represents the text to be displayed in the message. The ‘width’ option specifies the message width in pixels. ‘font’ represents the font for the message. We can use options ‘fg’ for specifying foreground color and ‘bg’ for specifying background color for the message text.

Text widget:

Text widget is same as a Label or Message. But Text widget has several options and can display multiple lines of text in different colors and fonts. It is possible to insert text into a Text widget modify it or delete it. One can create a Text widget by creating an object to Text class,as:

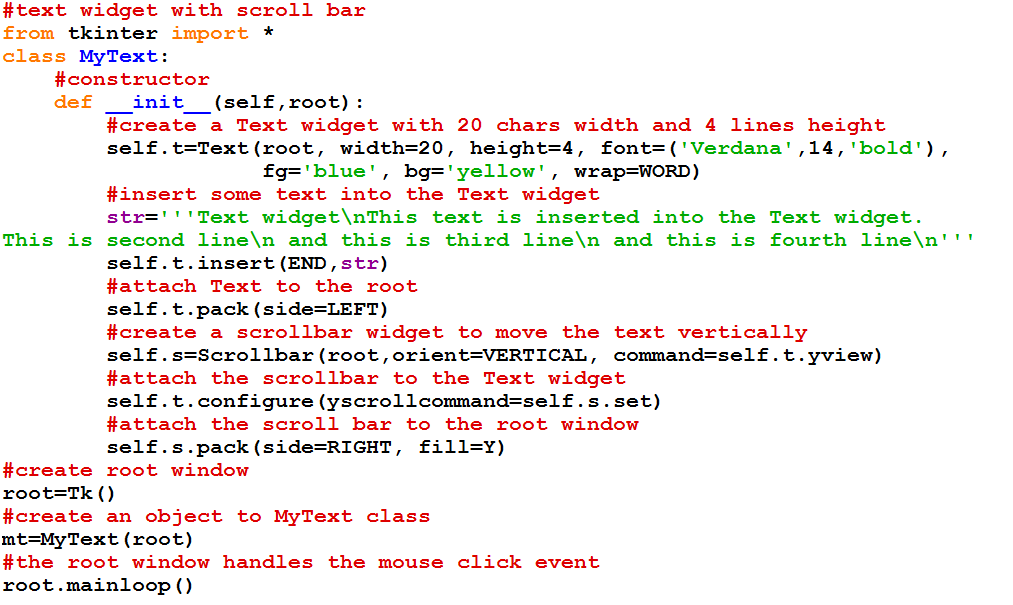
t=Text(root, width=20, height=10, font=(‘Verdana’, 14, ‘bold’), fg=’blue’, bg=’yellow’, wrap=CHAR/WORD/NONE)

Once the Text widget is created, we can insert any text using insert() method, as:

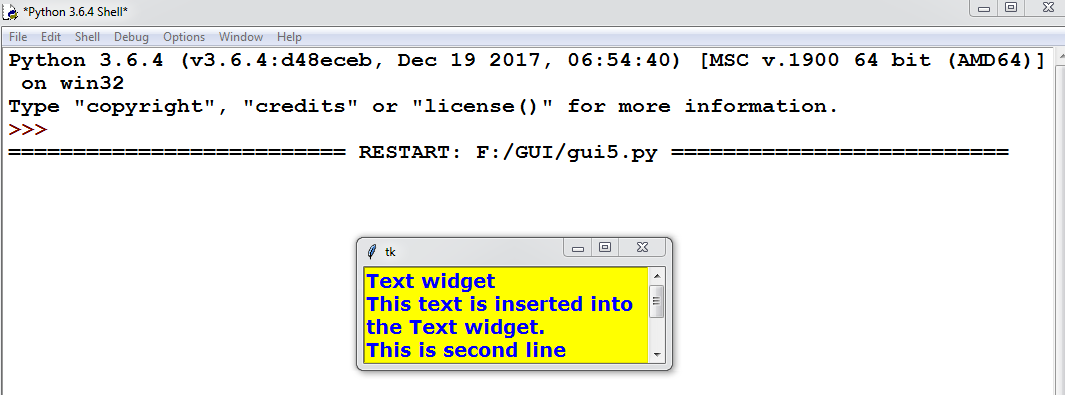
t.insert(END, ‘Text widget\nThis text is inserted into the Text widget.\n This is second line \n and this is third line \n’)

Here, the first argument END represents that the text is added at the end of the previous text. We can also use CURRENT to represent that the text is added at the current cursor position. The second argument is the text that is added to the Text widget.

Program 5: To create a Text widget along with a vertical scrollbar.



Op:

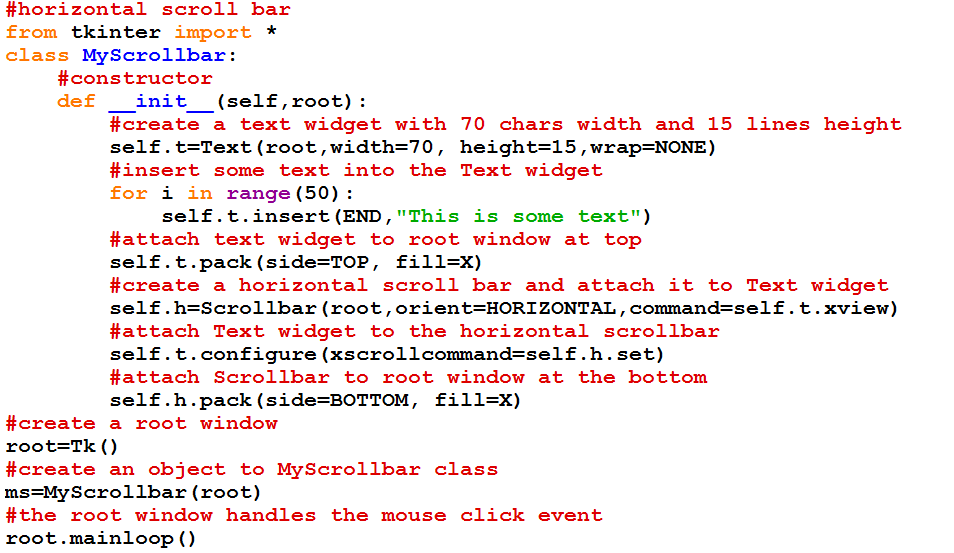


**Scrollbar widget:**

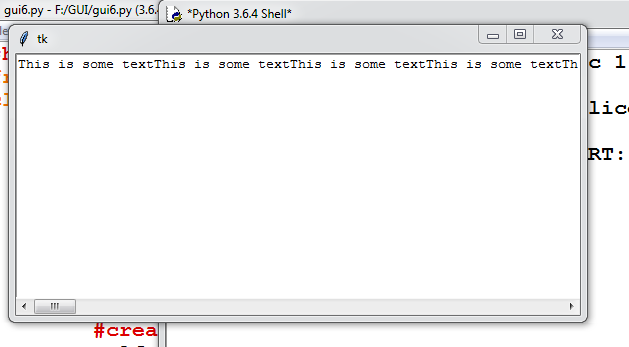
A scroll bar is a widget that is useful to scroll the text in another widget. For example, the text in the Text, Frame or Listbox can be scrolled from top to bottom or left to right using scroll bars. There are two types of scroll bars. They are horizontal and vertical. The horizontal scroll bar is useful to view the text from left to right. The vertical scroll bar is useful to scroll the text from top to bottom. To create a scroll bar, we have to create Scrollbar class object, as:

h=Scrollbar(root, orient=HORIZONTAL, command=t.xview)

Program 6: To create a horizontal scrollbar and attach to a Text widget to view the text from left to right.



Op:



**Checkbutton Widget:**

Check buttons, also known as check boxes are useful for the user to select one or more options from available group of options. Check buttons are displayed in the form of square shaped boxes. When a check button is selected, a tick mark is displayed on the button. We can create check buttons using Checkbutton class as:

c1=Checkbutton(f, bg=’yellow’,fg=’green’,font=(‘Georgia’,20,’underline’), text=’Java’, variable=var1, command= display)

Here option ‘variable’ represents an object of intVar() class. ‘command’ represents the method to be called when the user clicks the check button.

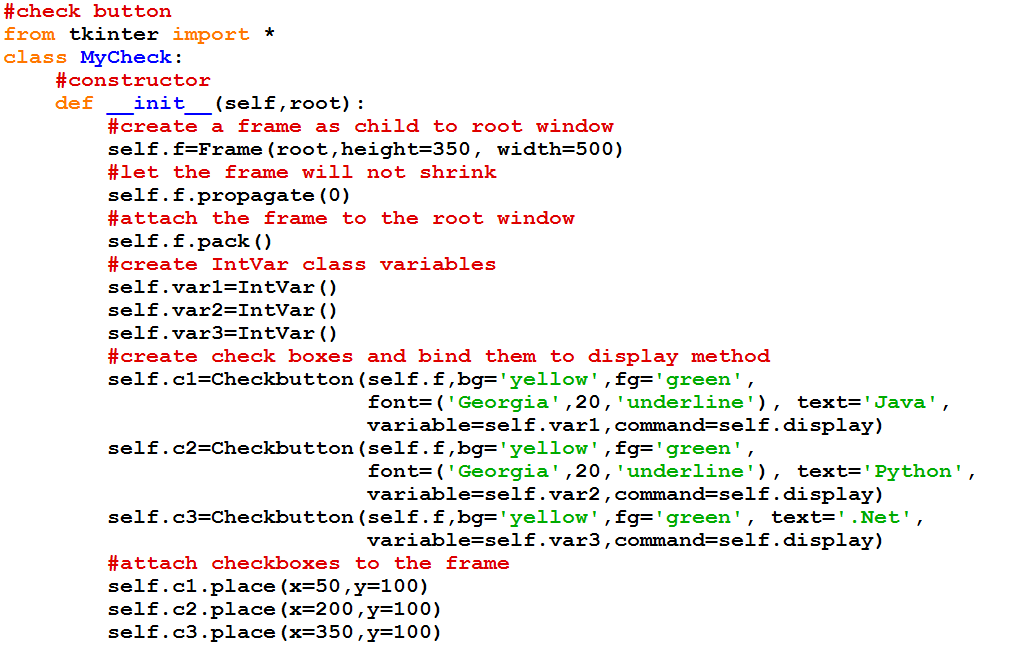
The class ‘IntVar’ is useful to know the state of the check button, whether it is clicked or not. IntVar class object can be created as:

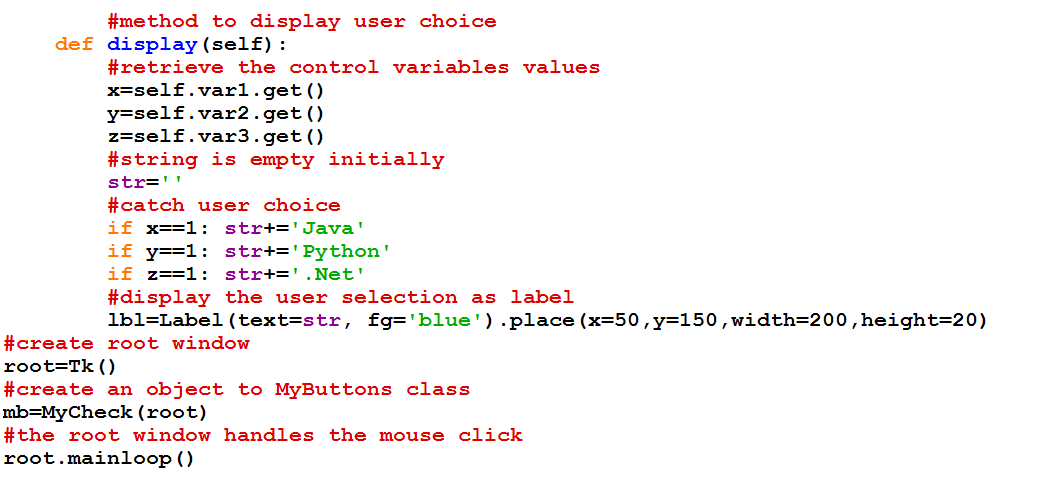
var1=IntVar()

When the check button is clicked or selected, the value of ‘var1’ will be 1, Otherwise its value will be 0. To retrieve the value from ‘var1’, we should use get() method, as:

x=var1.get() #x can be 1 or 0

Program 7: To create 3 check buttons and know which options are selected by user.





**Radioabutton Widget:**

A radio button is similar to a check button, but it is useful to select only one option from a group of available options. A radio button is displayed in the form of round shaped button. The users cannot select more than 1 option in case of radio buttons. When a radio button is selected, there appears a dot in the radio button. We can create a radio button as an object of Radiobutton class, as:

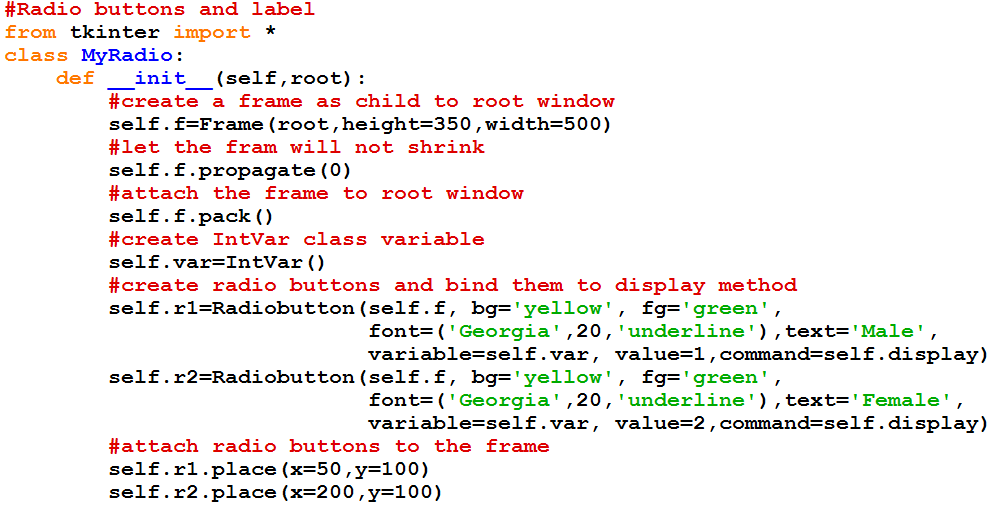
r1=Radiobutton(f,bg=’yellow’,fg=’green’,font=(‘Georgia’,20,’underline’), text=’Male’, variable=var, value=1,command=display

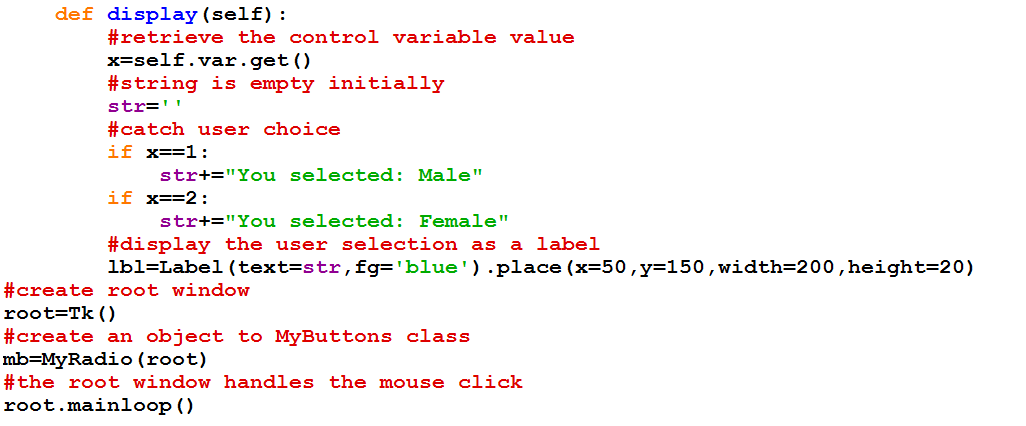
The option ‘text’ represents the string to be displayed after the radio button. ‘variable’ represents the object of IntVar class. ‘value’ represents a value that is set to this object when the radio button is clicked. The object of IntVar class can be created as:

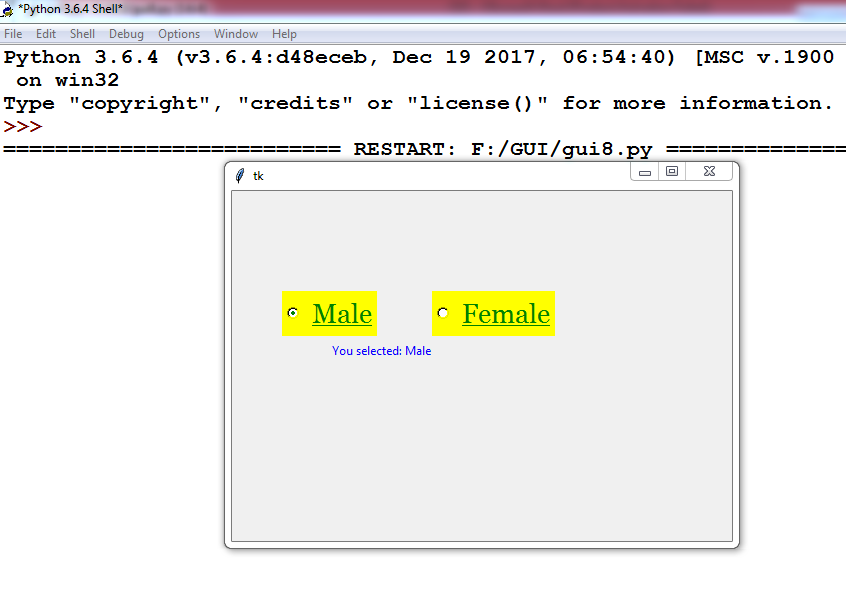
var=IntVar()

When the user clicks the radio button, the value of this ‘var’ is set to the value given in ‘value’ option, i.e.1. It means ‘var’ will become 1 if the radio button ‘r1’ is clicked by the user. In this way it is possible to know which button is clicked by the user

Program 8: To create 2 radio buttons and know which option is selected by the user





Op:

**Entry widget :**

Entry widget is used to create a rectangular box that can be used to enter or display one line of text. For example, we can display names, passwords or credit card numbers using entry widgets. A Entry widget is created as an object of Entry class as:

e1=Entry(f, width=25, fg=’blue’, bg=’yellow’,font=(‘Arial’,14),show=’\*’)

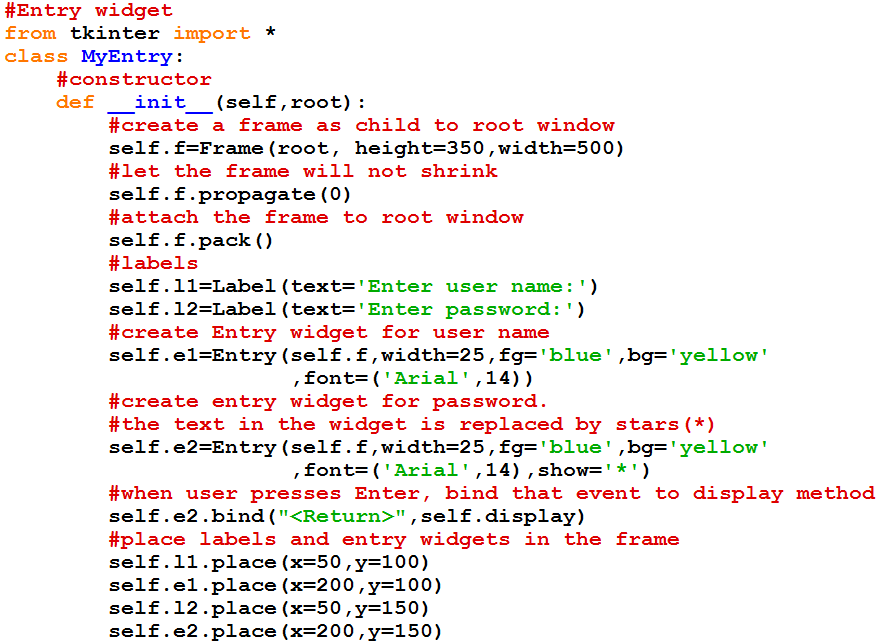
After typing text in the Entry widget user presses the Enter button. Such an event should be linked with the Entry widget using bind() method as:

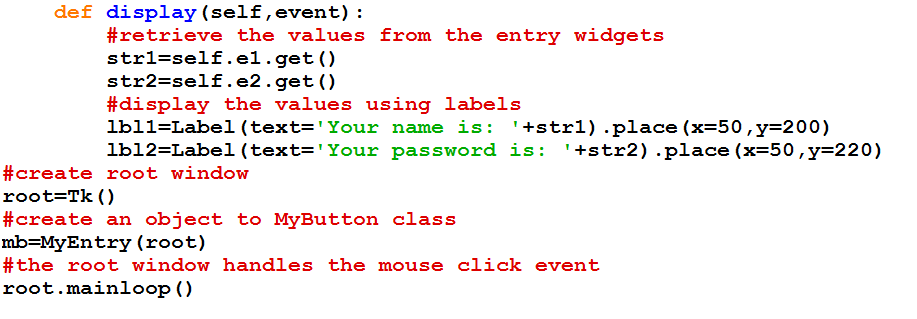
e1=bind(“<Return>”,self.display)

When the user presses Enter button, the event is passed to display() method. Hence, we are supposed to catch the event in the display() method, using the following statement:

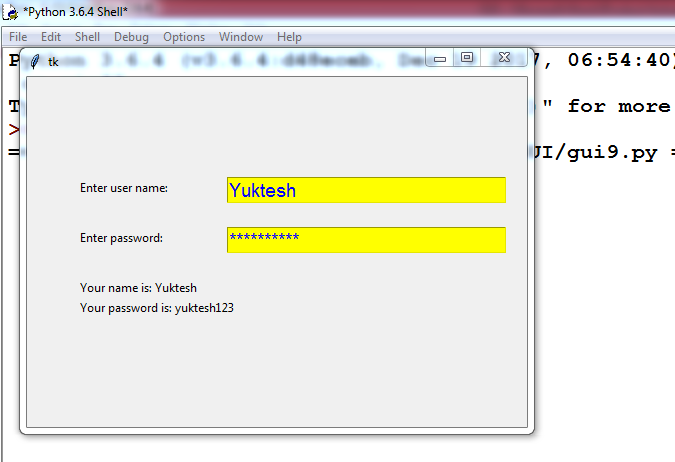
def display(self, event):

Program 9: To create an Entry widgets for entering user name and password and display the entered text.





Op:



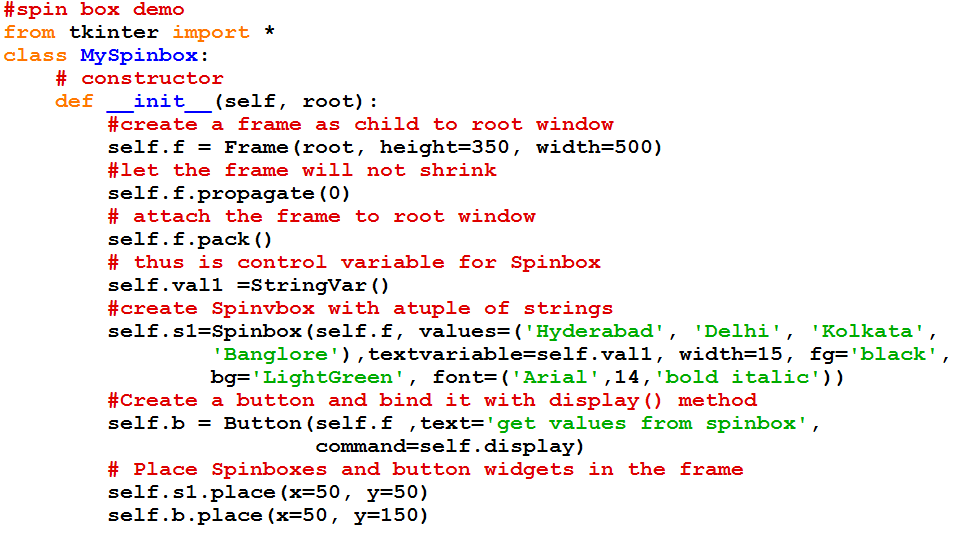
**Spinbox widget:**

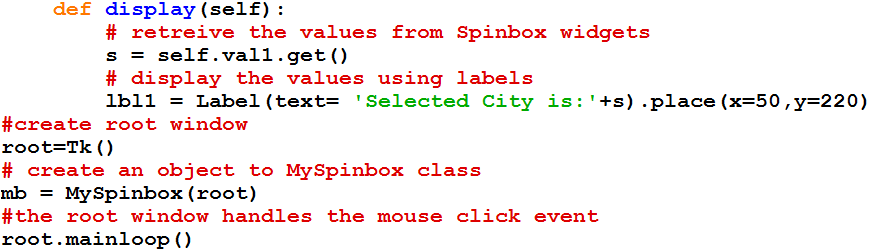
A Spinbox widget allows the users to select values from a given set of values. The values may be a range of numbers or a fixed set of strings.

The spin box is created as an object of Spinbox class. To create a spin box with ranging from 5 to 15, we can write the following statement:

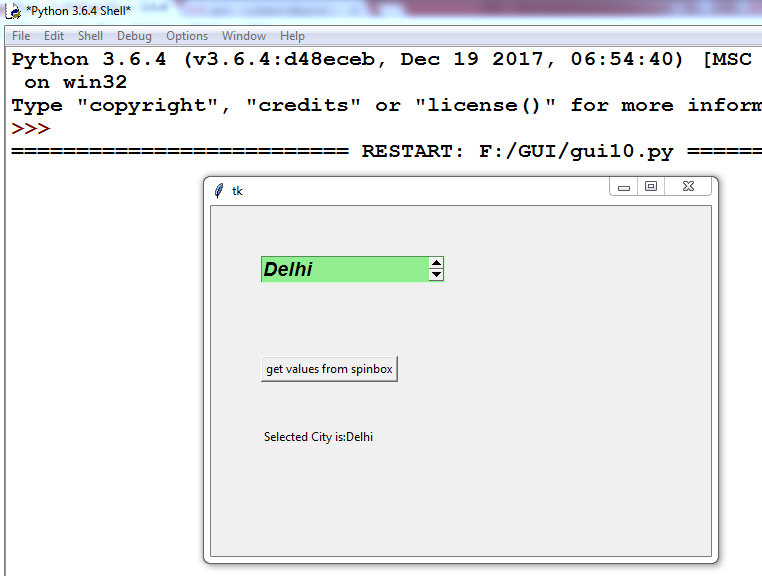
s1=Spinbox(f, from\_=5, to=15, textvariable=val1, width=15, fg=’blue’, bg=’yellow’,font=(‘Arial’,’14,’bold’))

**Program 10:**





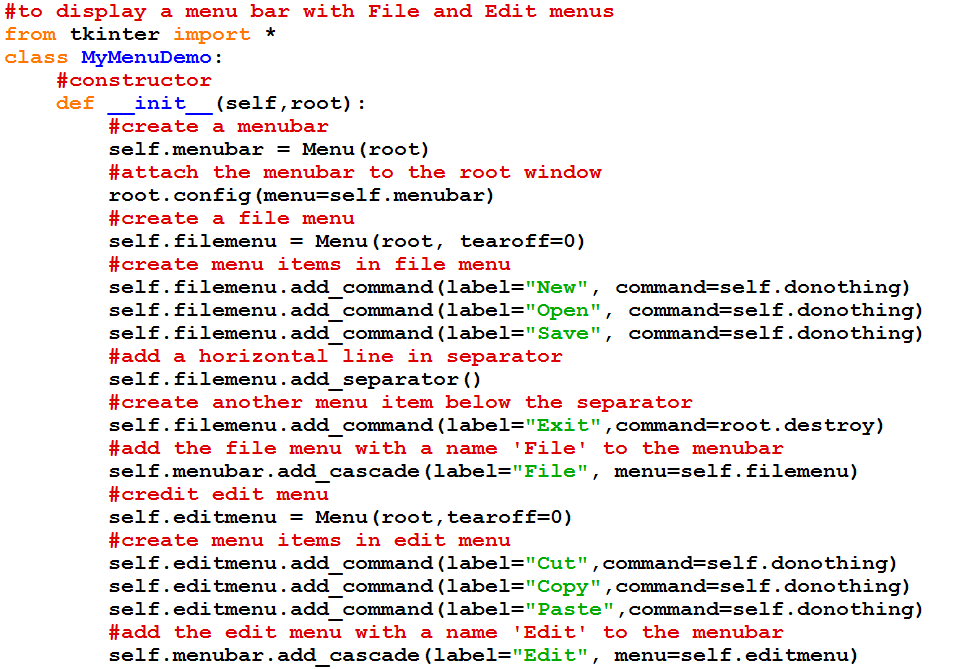
Op:

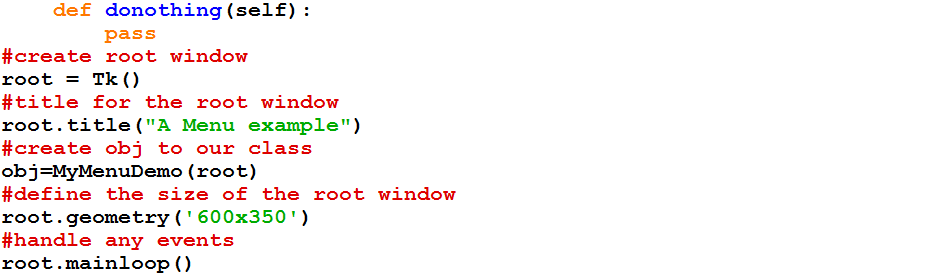


**Menu widget:**

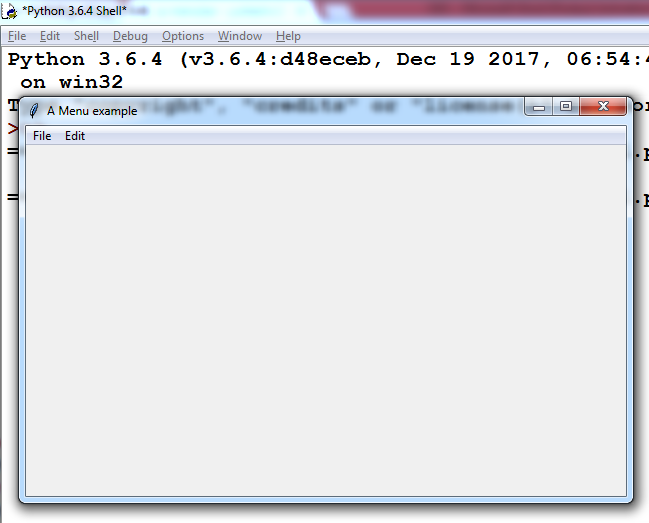
A menu is a group of items or options for user to select from.

Program 11: To create a menu bar and adding File and Edit menus with some menu items.





Op:



**Networking in Python**

Interconnection of computers is called a network. A simple network can be formed by connecting two computers using a cable. Thus a network can have two computers or two thousand computers. For example, Internet is the largest network on the earth where millions of computers are connected.

Major 3 requirements to establish a network:

1. Hardware – physical devices ex: computer, cables, moderns, routers, hubs etc
2. Software – programs to communicate between client and server
3. Protocol – Represents set of rules to establish connection

Protocals we use in Networking:

Following are the 2 standard protocols which we use in transfer of data in a network

1. TCP/IP
2. UDP

TCP/IP:

- It is the standard model used in all networks included internet is based on this protocal.

- There are 5 layers this protocal has. They are

i. Application layer

ii. TCP

iii. IP

iv. Data Link layer

v. Physical layer

UDP:

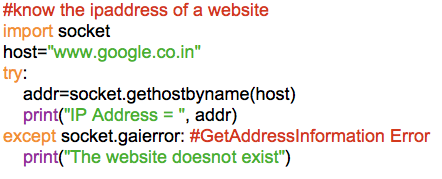
-UDP is another protocal that transfers data in a connection less manner.

SOCKET:

- Socket helps in establishing connection on the network.

- To perform network operation using python program we should use socket module

Program1: To find the IP Address of a computer



Op:



Program2: Downloading an image from website.



op:

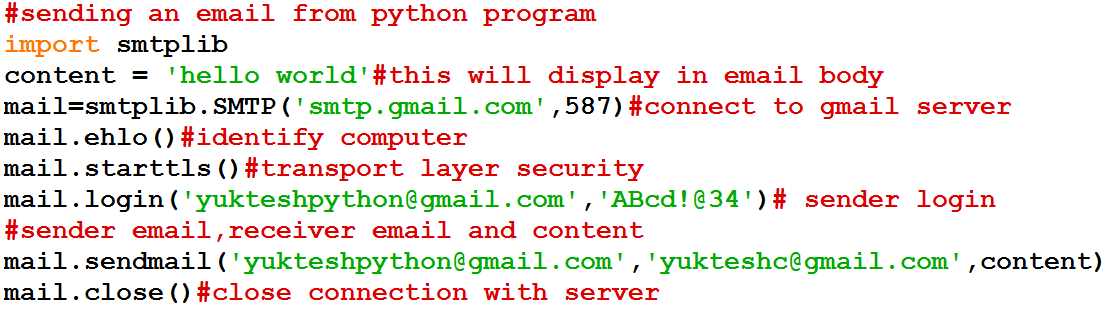
myimage.jpg will saved in the program directory

**Sending a simple mail:**

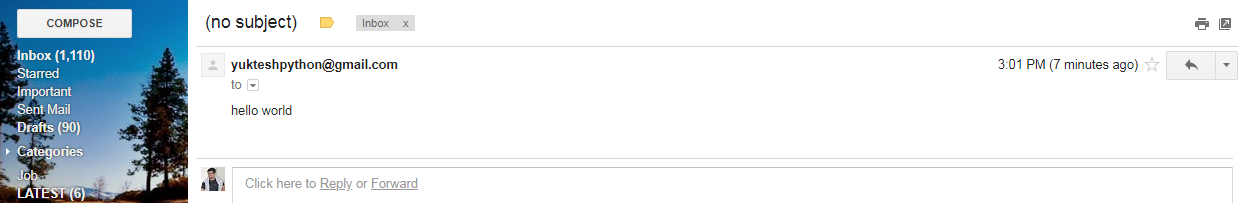
- We can send email using python program using SMTP class of smtp module

- SMTP class object creation means we are connecting to smtp protocol (simple mail transfer protocol) which helps in sending email.

Program3: program to send an email.



Op:



- Due to security reasons some times gmail will not allow to receive emails. In this case we need to inform gmail.com server to allow “less secure apps”.

- We can do this by logging into gmail account and then visiting the following page

<https://www.google.com/settings/security/lesssecureapps>.

- In this link we should click on “Turn on” option against “Access for less secure apps”

- Now execute program again it will work.

**Two communication between server and client:**

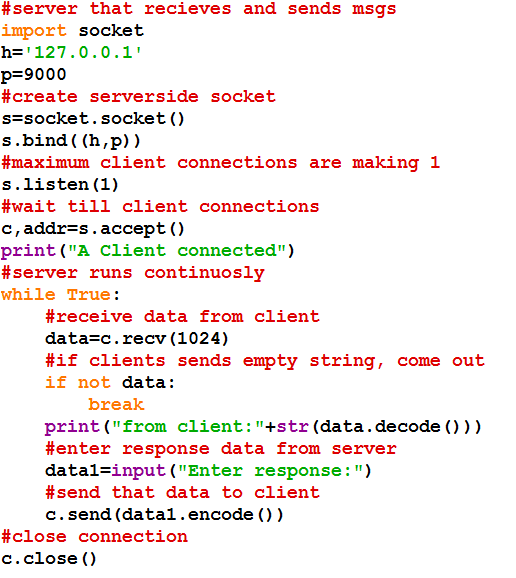
- It is possible to send data from client to server and from server to client using socket module properties

- Here we write 2 programs (i)chatserver.py and (ii)chatclient.py

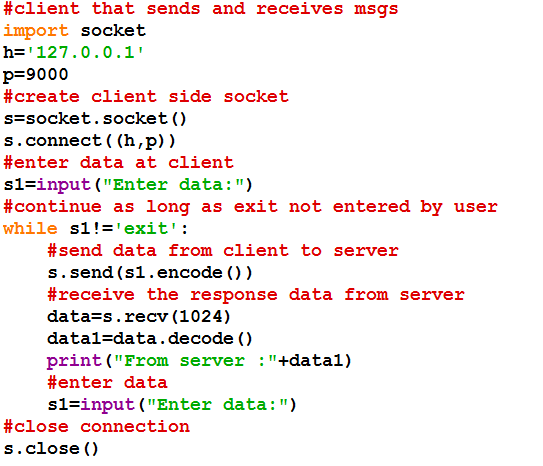
- After completion of writing both programs then we execute them in separate command prompts

- First execute chatserver.py in 1 cmd and come to another cmd and execute chatclient,py like below

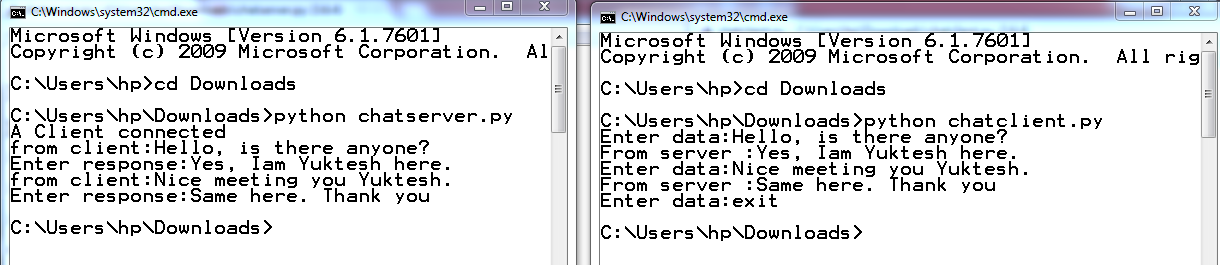
Program4: chatserver.py



Program5: chatclient.py



Op:



**Data Science using Python**

Data Life Cycle:

- Data(s) –> Data Warehousing –>Data Analysis –> Data Visualization

- DBA at Data warehousing helps in managing the data

- Difference between Data Scientist and Data Analyst

|  |  |
| --- | --- |
| Data Scientist | Data Analyst |
| 1)Data Scientist formulates the questions that will help a business and then proceed in solving them | 1) Data Analyst receives questions by the business team and provides answers to them |
| 2) A data scientist builds statistical models or uses machine learning | 2) A Data Analyst works on structured SQL or similar databases |
| 3) Data Scientist will have strong data visualization skills and the ability to convert data into a business story | 3) Data Analyst Simply analyze the data and provides information to the team |

**Data Analysis:**

* Pandas is a package useful for data analysis and manipulation
* Dataframe is the main object in pandas package
* Dataframe is an object that is useful to represent data in the form of rows and columns (tabular data or excel spreadsheet data)
* Dataframe is created from csv files, excel files, python dictionary, list of tuples, list of dictionaries

**Data Visualization:**

* matplotlib is a package used for 2D graphics
* Data Visualization is the presentation of data in pictorial or graphical format
* Data Visualization allows us to quickly interpret the data and adjust different variables to see their effect.

**Install pandas,xlrd and matplotlib:**

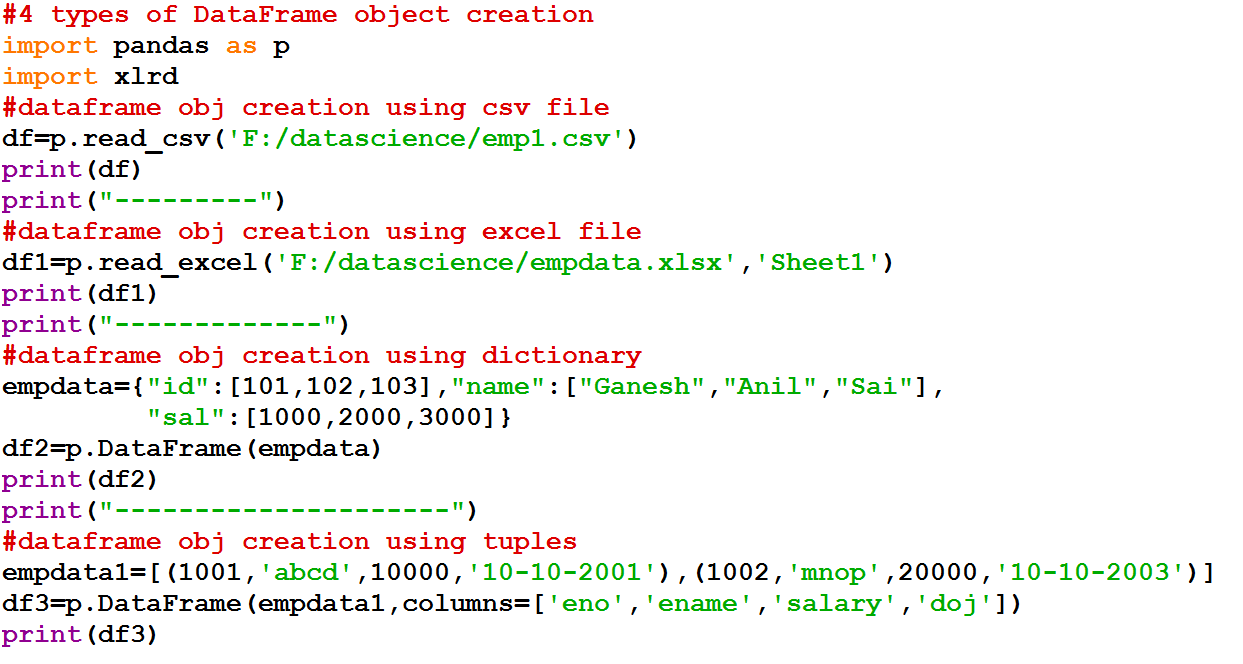
c:\python36>pip install pandas

c:\python36>pip install xlrd # to handle xls sheets this is helpful

c:\python36>pip install matplotlib

**Working on data analytics using pandas**

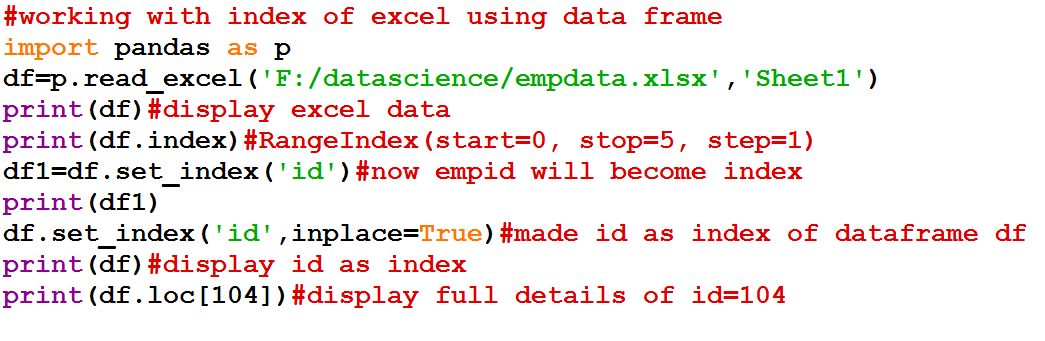
Ex1: Program to creating DataFrame object in 4 different types.



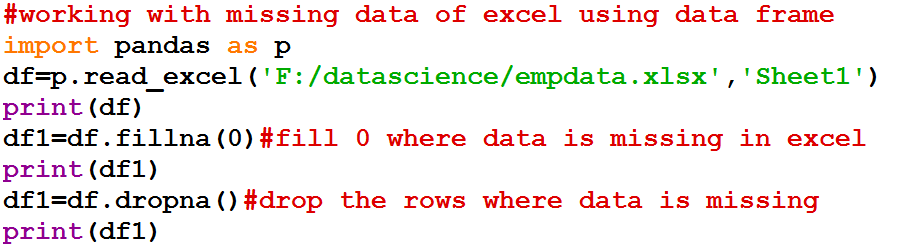
Ex2: Program to work with excel using data frame object.



Ex3: Program to work with index of excel using data frame object

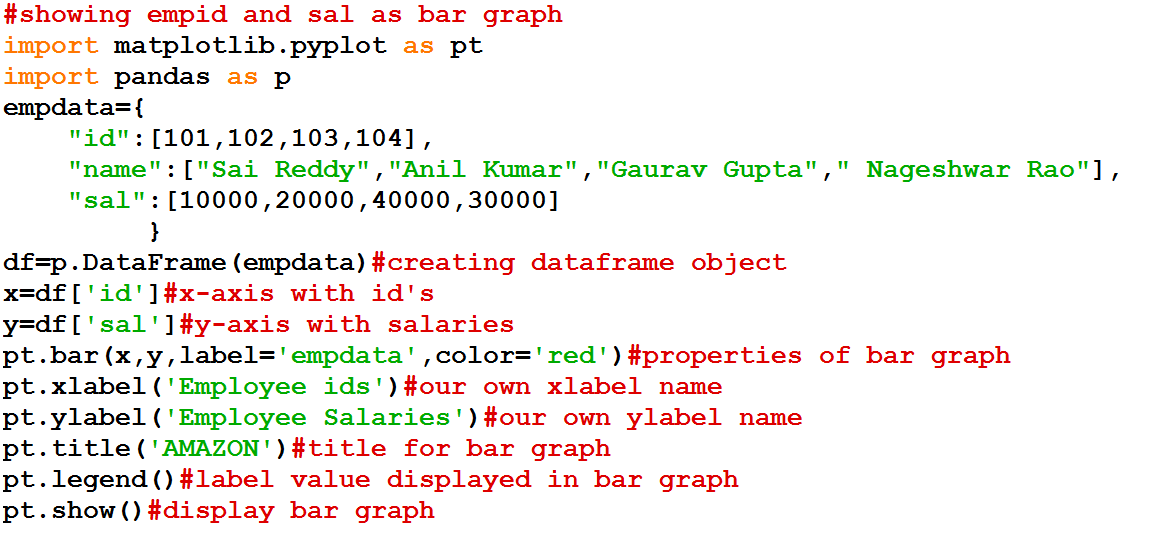


Ex4: Working with missing data in excel.

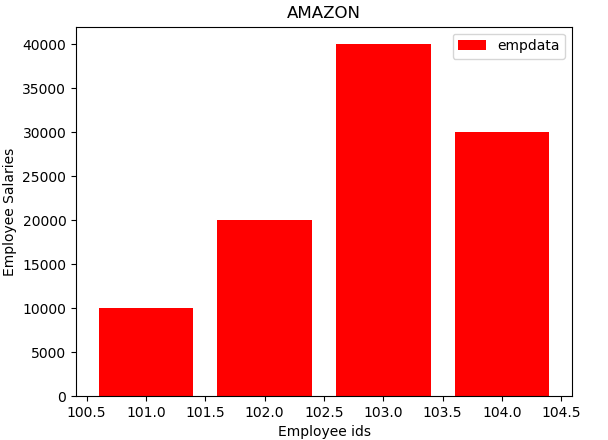


**Data Visualization:**

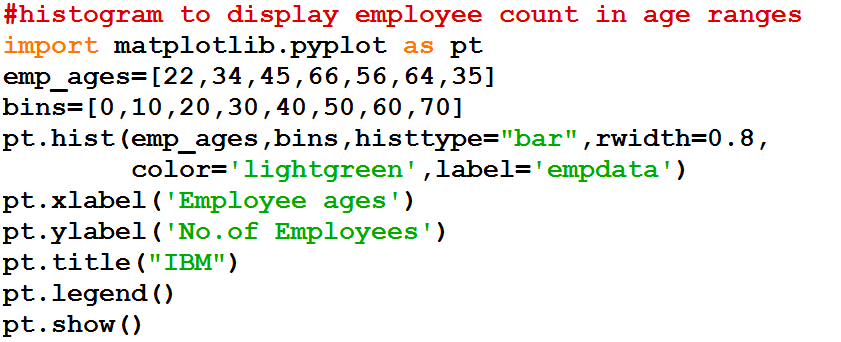
Ex1: Visualizing specific emp data in bar graph



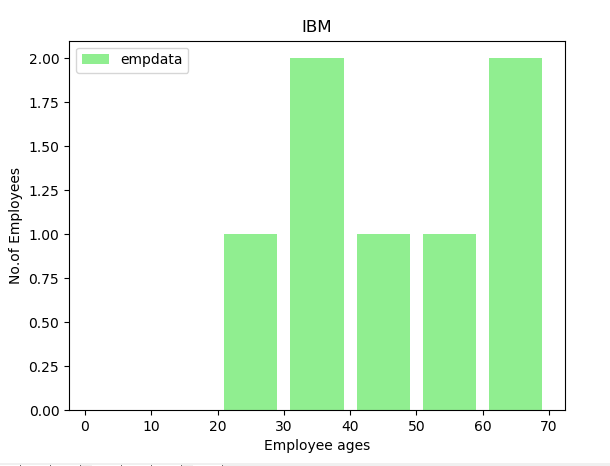
op:



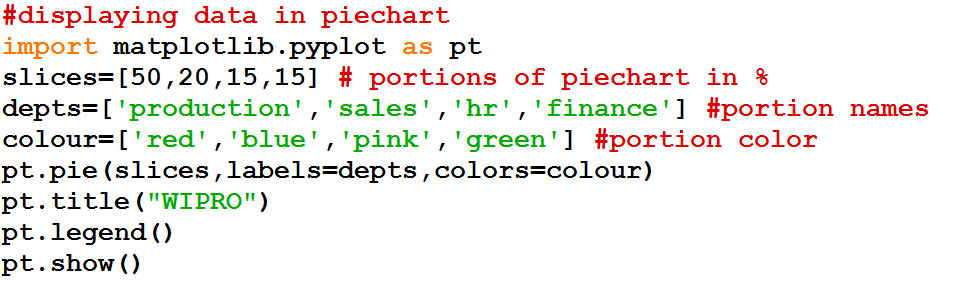
Ex2: Visualizing specific emp data in histogram.



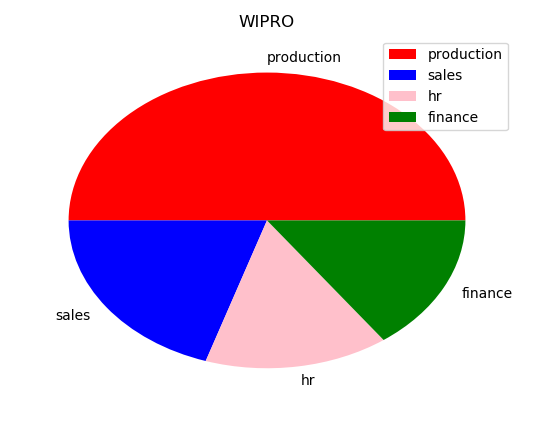
Op:



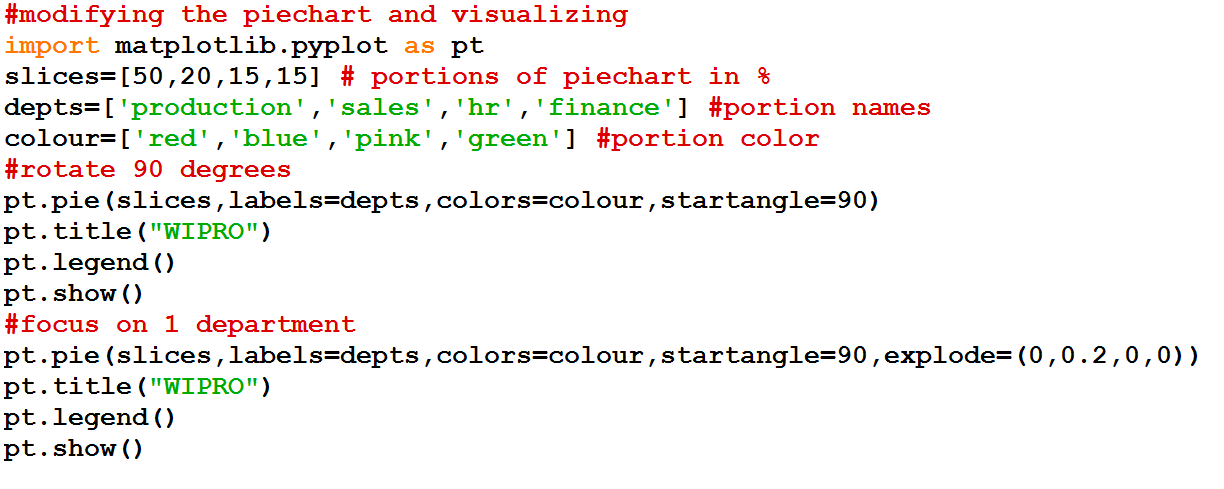
Ex3: Visualizing specific emp data in pie chart.



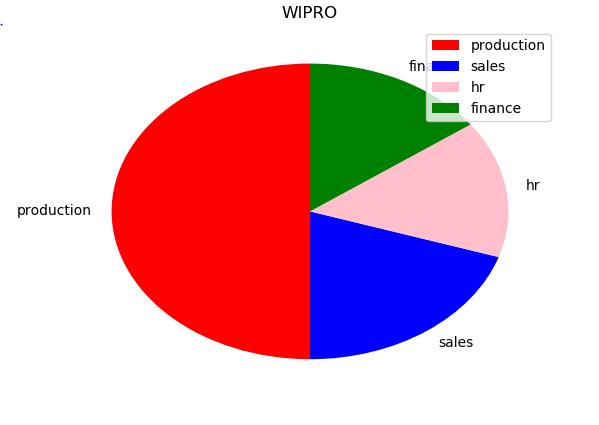
Op:



Ex4: Modifying the above pie chart program



Op:



And

