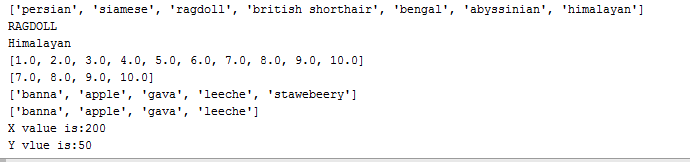
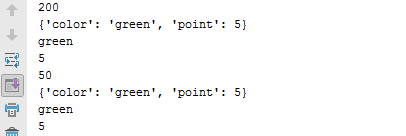
cats = [**'persian'**, **'siamese'**, **'ragdoll'**, **'british shorthair'**, **'bengal'**, **'abyssinian'**, **'himalayan'**]  
print(cats)  
**for** cat **in** cats:  
 **if** cat ==**'ragdoll'**:  
 print(cat.upper())  
**else**:  
 print(cat.title())  
  
squares = []  
**for** value **in** range(1, 11):  
 square = (value \*\* 2)\*\*0.5  
 squares.append(square)  
print(squares)  
print(squares[-4:])  
  
fruit=[**'banna'**,**'apple'**,**'gava'**,**'leeche'**]  
frien\_fr=fruit[:]  
fruit.append(**'stawebeery'**)  
print(fruit)  
print(frien\_fr)  
  
dimension=(200,50)  
print(**"X value is:"**+str(dimension[0]))  
print(**"Y vlue is:"**+str(dimension[1]))  
**for** dimension **in** dimension:  
 print(dimension)  
  
 alien\_o={**'color'**:**'green'**,**'point'**: 5}  
 print(alien\_o)  
 print(alien\_o[**'color'**])  
 print(alien\_o[**'point'**])





FUNCTION IN PHYTON

def sumProblem(x, y):  
 sum = x + y  
 sentence = 'The sum of {} and {} is {}.'.format(x, y, sum)  
 print(sentence)  
  
def main():  
 sumProblem(2, 3)  
 sumProblem(1234567890123, 535790269358)  
 a = int(input("Enter an integer: "))  
 b = int(input("Enter another integer: "))  
 sumProblem(a, b)  
  
main()

def fact(n):  
 if n==0:  
 return 1  
 return n\*fact(n-1)  
def main():  
 a = int(input("Enter the number of factorial:"))  
 print("factorial is:",fact(a))  
main()

def fob(i):  
 if i==0 or i==1 or i==2:  
 return i  
 else :  
 return fob(i-1)+fob(i-2)  
def main():  
 a = int(input("Enter the number:"))  
 print("Fobonicii series is:", fob(a))  
main()

def area(x,y):  
 return x\*x\*y  
def main():  
 c = int(input("Enter the radius of circle:"))  
 print("Area of the circle is:",area(3.142 ,c))  
main()

def cal(x ,y):  
 total = x + y  
 total = x - y  
 total = x \* y  
 total = x / y  
 return total  
def main():  
 a = int(input("Enter the 1st value:"))  
 b = int(input("Enter the 2nd value:"))  
 print("the total calculation is:", cal(a ,b))  
main()

def fact(n,k):  
 if n==0 or k==0:  
 return 1  
 return (n\*fact(n-1, 1))/(k\*fact(1, k-1))  
def main():  
 a = int(input("Enter the total number of combination:"))  
 b = int(input("Enter the number of selection:"))  
 print("combination is:",fact(a ,b))  
main()

import string  
class khan():  
 def setdata(self,m,n):  
 a = string(input("Enter the name of car"))  
 b = int(input("Enter the year of car"))  
 print("Name of car is:", a)  
 print("year of the car:", b)  
 def main(self):  
 k1 = khan()  
 print(k1.setdata())

class Car():  
 *"""A simple attempt to represent a car."""* def \_\_init\_\_(self, make, model, year):  
 *"""Initialize attributes to describe a car."""* self.make = make  
 self.model = model  
 self.year = year  
  
 def get\_descriptive\_name(self):  
 *"""Return a neatly formatted descriptive name."""* long\_name = str(self.year) + ' ' + self.make + ' ' + self.model  
 return long\_name.title()  
  
  
my\_new\_car = Car('audi', 'a4', 2016)  
print(my\_new\_car.get\_descriptive\_name())

from chatterbot import ChatBot  
from chatterbot.trainers import ListTrainer  
import os  
  
bot = ChatBot('Bot')  
bot.set\_trainer(ListTrainer)  
  
for files in os.listdir('C:/Users\Me\Documents\chatterbot-corpus-master\chatterbot\_corpus\data\english')  
  
 data =open('C:/Users\Me\Documents\chatterbot-corpus-master\chatterbot\_corpus\data\english'+ files , ).readline()  
  
bot.train(data)  
  
while True:  
 message = input("you:")  
 if message.strip()!='Bye':  
 reply = bot.get\_response(message)  
 print('ChatBot :', reply)  
 if message.strip() == 'Bye':  
 print ('ChatBot: Bye')  
 break

class Employee() :  
 def \_\_init\_\_(self, name, yrs\_of\_ser):  
 self.name = name  
 self.yrs\_of\_ser = yrs\_of\_ser  
 def sal(self):  
 return self.years\_of\_service \* 1500 + 100  
 class manager():  
 def \_\_init\_\_(self, name, yrs\_of\_ser):  
 self.name = name  
 self.yrs\_of\_ser = yrs\_of\_ser  
  
 def sal(self):  
 return self.years\_of\_service \* 2500 + 100  
  
 Employee = {'jhon': 1, 'mark': 2, 'julie': 6, 'paul': 10}  
 for key, value in Employee.items():  
 print(key, value)

#CLASSES

class Employee(object):  
  
 # \_\_init\_\_ is known as the constructor  
 def \_\_init\_\_(self, name, yrs\_of\_ser,SAL):  
 self.name = name  
 self.yrs\_of\_ser = yrs\_of\_ser  
 self.SAL = SAL  
  
 def display(self):  
 print(self.name)  
 print(self.yrs\_of\_ser)  
 self.SAL = 1500 + (100 \* self.yrs\_of\_ser)  
 print(self.SAL)  
  
 # child class  
  
  
#class Manager(Employee):  
 # invoking the \_\_init\_\_ of the parent class  
 # Employee.\_\_init\_\_(self, name='zunair', yrs\_of\_ser=12)  
  
 # creation of an object variable or an instance  
a = Employee('Rahul', 886012,11)  
  
# calling a function of the class Person using its instance  
a.display()

from \_\_future\_\_ import print\_function

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

columns = ['species', 'sepal\_lenght', 'sepal\_widht', 'petal\_lenght','petal\_widht']

user1 = pd.DataFrame([['alice', 19, "F", "student","yar"],

['john', 26, "M", "student","jaffar"],['khan', 22, "M", "proffessor","saddiq"]],

columns=columns)

user2 = pd.DataFrame([['eric', 22, "M", "student","naqwi"],

['paul', 58, "F", "manager","rizwi"],["zunair",21,"M","lecturer","alvi"]],

columns=columns)

user3 = pd.DataFrame(dict(species=['setosa', 'versicolor','verginicia'],

sepal\_lenght=[5.07, 5.09, 6.58], sepal\_widht=[3.4, 2.7, 2.97],

petal\_lenght=[4.2, 1.4, 5.55], petal\_widht=[0.24, 1.3, 2.026]))

print(user3)

petal\_lenght petal\_widht sepal\_lenght sepal\_widht species

0 4.20 0.240 5.07 3.40 setosa

1 1.40 1.300 5.09 2.70 versicolor

2 5.55 2.026 6.58 2.97 verginicia

class Account:

def \_\_init\_\_(self, number, balance):

self.a = number

self.b = balance

def myfunc(self):

print("The Account is " + str(self.a))

print("The balance is " + str(self.b))

p1 = Account(23, 3600)

p1.myfunc()



class Time:

pass

def \_\_init\_\_(self, hrs, mminu, sec):

self.a = hrs

self.b = mminu

self.c =sec

def myfunc(self):

print(str(self.a) +":"+ str(self.b) +":" + str(self.c))

def add\_time(self):

self.a = self.a + self.a

self.b = self.b + self.b

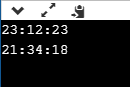
self.c = self.c + self.c

p1 = Time(23, 12,23)

p2 = Time(21,34,18)

p1.myfunc()

p2.myfunc()



class User:

hei = 4

wid = 7

def setName(self,hei):

self.hei = hei

def setJob(self,wid):

self.wid = wid

def printName(self):

print ("Height is = " + str(self.hei))

def printJob(self):

print ("widht is = " + str(self.wid))

class Programmer(User):

style = ""

area =12

def \_\_init\_\_(self, style,):

self.style = style

def getarea(self,area):

self.area = area

def printLanguage(self):

print ("Style is = " + self.style)

guido = Programmer("isocales")

guido.setName(4)

guido.setJob(5)

guido.getarea(34)

gui = Programmer("right")

gui.setName(12)

gui.setJob(13)

gui.getarea(78)

guido.printName()

guido.printJob()

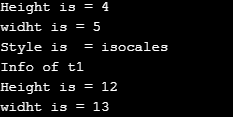
guido.printLanguage()

print("Info of t1")

gui.printName()

gui.printJob()

gui.printLanguage()



class User:

hei = 4

job = 7

def setName(self,hei):

self.hei = hei

def setJob(self,wid):

self.wid = wid

def printName(self):

print ("Height is = " + str(self.hei))

def printJob(self):

print ("widht is = " + str(self.wid))

class Programmer(User):

language = ""

def \_\_init\_\_(self, style):

self.style = style

def printLanguage(self):

print ("Style is = " + self.style)

guido = Programmer("Python")

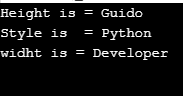
guido.setName("Guido")

guido.setJob("Developer")

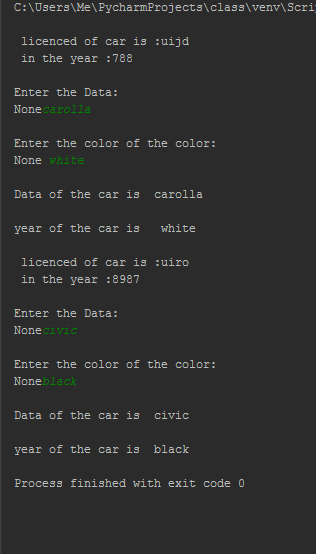
guido.printName()

guido.printLanguage()

guido.printJob()



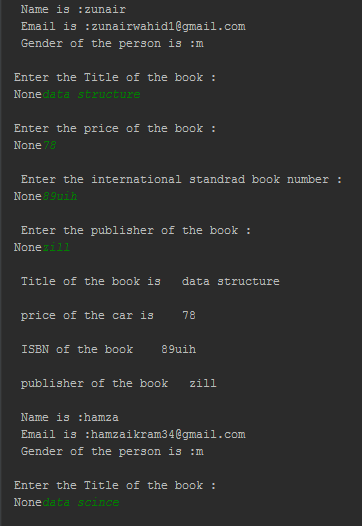
class Vechicle:  
 pass  
  
 def \_\_init\_\_(self, licence, year):  
 self.licence = licence  
 self.year = year  
  
 def showdata(self):  
 print("\n licenced of car is :" + self.licence + "\n in the year :" + str(self.year))  
  
  
class Car(Vechicle):  
  
 def \_\_init\_\_(self, licence, year, data, color):  
 Vechicle.\_\_init\_\_(self, licence, year)  
 self.data = data  
 self.color = color  
  
 def getdata(self):  
 self.data = input(print("\nEnter the Data:"))  
 self.color = input(print("\nEnter the color of the color:"))  
  
 def display(self):  
 print("\nData of the car is\t" + self.data)  
 print("\nyear of the car is\t" + self.color)  
  
  
we = Car("uijd", 788, " ", " ")  
tr = Car("uiro", 8987, " ", " ")  
we.showdata()  
we.getdata()  
we.display()  
tr.showdata()  
tr.getdata()  
tr.display()

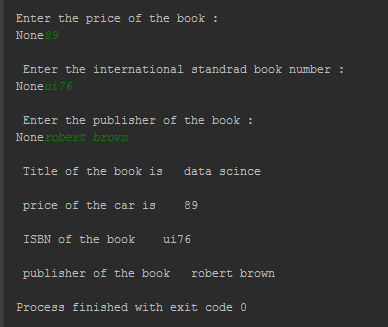


Another way to show the above example

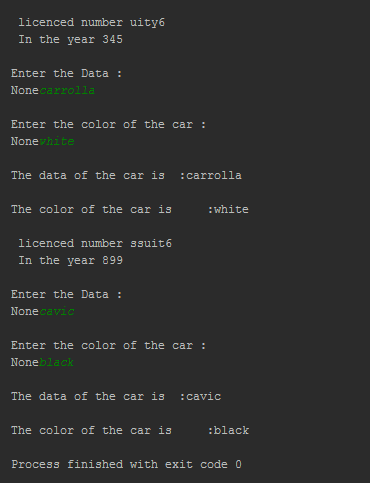
class Vechicle:  
  
  
 def \_\_init\_\_(self, licence, year):  
 self.licence = licence  
 self.year = year  
  
 def showdata(self):  
 print("\n licenced number " + self.licence + "\n In the year " +str(self.year))  
  
  
  
  
class Car(Vechicle):  
  
 def \_\_init\_\_(self, licence, year, data, color):  
 # call parent constructor to set name and color  
 super().\_\_init\_\_(licence, year)  
 self.data = data  
 self.color = color  
  
 def getdata(self):  
 self.data = input(print("\nEnter the Data :"))  
 self.color = input(print("\nEnter the color of the car :"))  
  
 def display(self):  
 print("\nThe data of the car is \t:" + self.data)  
 print("\nThe color of the car is \t:" + self.color)  
  
  
we = Car("uity6", 345, "\_\_", "\_\_")  
tr = Car("ssuit6", 899, "\_\_", "\_\_")  
we.showdata()  
we.getdata()  
we.display()  
tr.showdata()  
tr.getdata()  
tr.display()

class Author:  
 pass  
  
 def \_\_init\_\_(self, name, email, gender):  
 self.name = name  
 self.email = email  
 self.gender = gender  
  
 def showdata(self):  
 print("\n Name is :" + self.name + "\n Email is :" + self.email + "\n Gender of the person is :" + self.gender)  
  
  
class Book(Author):  
  
 def \_\_init\_\_(self, name, email, gender, title, price, isbn, publisher):  
 Author.\_\_init\_\_(self, name, email, gender)  
 self.title = title  
 self.price = price  
 self.ISBN = isbn  
 self.publisher = publisher  
  
 def getdata(self):  
 self.title = input(print("\nEnter the Title of the book :"))  
 self.price = input(print("\nEnter the price of the book :"))  
 self.ISBN = input(print("\n Enter the international standrad book number :"))  
 self.publisher = input(print("\n Enter the publisher of the book :"))  
 def display(self):  
 print("\n Title of the book is\t" + self.title)  
 print("\n price of the car is\t" + str(self.price))  
 print("\n ISBN of the book\t " + self.ISBN)  
 print("\n publisher of the book\t " + self.publisher)  
  
we = Book("zunair","zunairwahid1@gmail.com", "m", " ", " ", " ", " ")  
tr = Book("hamza","hamzaikram34@gmail.com", "m", " ", " ", " ", " ")  
we.showdata()  
we.getdata()  
we.display()  
tr.showdata()  
tr.getdata()  
tr.display()





import math  
class Student:  
 def \_\_init\_\_(self, name, roll\_no):  
 self.name = name  
 self.roll\_no = roll\_no  
  
 def getdata(self):  
 self.name = input("\n Enter the name of student:")  
 self.roll\_no = input("\n Enter the Rollno of Student:")  
  
 def showdata(self):  
 print("\nName of the is :" + self.name + "\n Rollno of the Student is" + self.roll\_no)  
  
  
class Course:  
  
 def \_\_init\_\_(self, coursen, courset):  
 self.coursen = coursen  
 self.courset = courset  
  
 def getdata(self):  
 self.coursen = input("\n Enter the Course number :")  
 self.courset = input("\n Enter the Course title : ")  
  
  
 def showdata(self):  
 print("\n Course number of the course" + self.coursen + "\n Course Tile of course" + self.courset)  
  
  
class Marks(Student, Course):  
  
 def \_\_init\_\_(self, name, roll\_no, coursen, courset, marks, percentage):  
 Student.\_\_init\_\_(self, name, roll\_no)  
 Course.\_\_init\_\_(self, coursen, courset)  
 self.marks = marks  
 self.percentage = percentage  
  
  
 def getdata(self):  
 Student.getdata(self)  
 Course.getdata(self)  
  
  
  
  
 def showdata(self):  
 Student.showdata(self)  
 Course.showdata(self)  
 self.marks = input(" \nEnter the Marks of Student :")  
 self.percentage = (str(self.marks) / 100)  
 print("\n% of the course is the :" + str(self.percentage))  
  
  
  
  
  
m = Marks(" ", " "," "," ", " ", " ")  
m.getdata()  
m.showdata()



class Numbers:  
 MULTIPLIER = 3.5  
 def \_\_init\_\_(self, x, y):  
 self.x = x  
 self.y = y  
 def add(self):  
 return self.x + self.y  
  
 def multiply(cls, a):  
 return cls.MULTIPLIER \* a  
  
 def subtract(b, c):  
 return b - c  
  
 def value(self):  
 return (self.x, self.y)  
  
 def Value(self, xy\_tuple):  
 self.x, self.y = xy\_tuple  
  
 def vAlue(self):  
 del self.x  
 del self.y