Lab Assignment- 1

Source Code

Task1:

def lengthOfLongestSubstring(s):

"""

:type s: str

:rtype: int

"""

res=[]

dict = {}

str = ""

final\_str = ""

i = 0

while i < len(s):

if s[i] in dict:

if len(str) > len(final\_str):

final\_str = str

res = [(final\_str, len(final\_str))]

elif len(str) == len(final\_str):

res.append((str, len(str)))

i = dict[s[i]]

dict.clear()

str = ""

else:

dict[s[i]] = i

str = str + s[i]

i = i + 1

if len(str) > len(final\_str):

final\_str = str

res=[(final\_str,len(final\_str))]

elif len(str)== len(final\_str):

res.append((str,len(str)))

return res

str=input("Enter the string")

print(lengthOfLongestSubstring(str))

Task2:

n=int(input("Enter the number of records"))

list=[]

dict={}

for i in range(n):

print("Record "+str(i+1)+" : ")

name=str(input("Enter name "))

subject = str(input("Enter subject "))

j = 1

while (j):

try:

marks=float(input("Enter marks "))

j = 0

except ValueError:

print('Non-numeric data.')

list.append((name,(subject,marks)))

print("\nEntered input data")

print(list)

for i in list:

if i[0] in dict:

dict[i[0]].append(i[1])

else:

dict[i[0]]=[i[1]]

print("\n----------------------------------------------------------------------\n")

print("Data stored in dictionary\n")

print(dict)

Task3:

class Person:

\_\_per\_count=0

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

self.\_\_name=name

self.\_\_age=age

self.\_\_family=family

Person.\_per\_count=Person.\_per\_count+1

@classmethod

def get\_persons\_count(self):

return Person.\_\_per\_count

def get\_name(self):

return self.\_\_name

def get\_family(self):

return self.\_\_family

def get\_age(self):

return self.\_\_age

class Employee (Person):

\_\_emp\_count = 0

\_\_avg\_sal = 0.0

def \_init\_(self, name, age, family, department,salary,EID):

self.\_\_salary=salary

self.\_\_department=department

self.\_\_EID=EID

Employee.\_emp\_count=Employee.\_emp\_count+1

Employee.\_avg\_sal=Employee.\_avg\_sal+salary

Person.\_init\_(self,name,age,family)

@classmethod

def average\_salary(self):

if Employee.\_\_emp\_count>0:

return Employee.\_avg\_sal / Employee.\_emp\_count

else:

return 0

@classmethod

def get\_employees\_count(self):

return Employee.\_\_emp\_count

def get\_department(self):

return self.\_\_department

def get\_salary(self):

return self.\_\_salary

def get\_EID(self):

return self.\_\_EID

class Passenger (Person):

\_\_pas\_count = 0

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

Passenger.\_pas\_count=Passenger.\_pas\_count+1

Person.\_init\_(self,name,age,family)

self.\_\_PID=PID

def get\_PID(self):

return self.\_\_PID

@classmethod

def get\_passengers\_count(self):

return Passenger.\_\_pas\_count

class Flight :

\_\_flights\_count = 0

def \_init\_(self,name, FID, seats\_count):

Flight.\_flights\_count=Flight.\_flights\_count+1

self.\_\_FID=FID

self.\_\_fname=name

self.\_\_seats\_count=seats\_count

self.\_\_passengers={}

self.\_\_available\_seats\_count=seats\_count

def get\_FID(self):

return self.\_\_FID

def get\_seats\_count(self):

return self.\_\_seats\_count

def get\_available\_seats\_count(self):

return self.\_\_available\_seats\_count

def get\_fname(self):

return self.\_\_fname

@classmethod

def get\_flights\_count(self):

return Flight.\_\_flights\_count

def assign\_passenger(self,p,pid):

if(self.\_\_available\_seats\_count>0):

self.\_\_passengers[pid]=p

self.\_available\_seats\_count=self.\_available\_seats\_count-1

print('Passenger added')

else:

print('No seats available')

def remove\_passengers(self,p,pid):

if(pid in self.\_\_passengers):

#self.\_\_passengers.remove(p)

self.\_\_passengers.pop(pid)

self.\_available\_seats\_count=self.\_available\_seats\_count+1

print('Passenger removed')

else:

print('No Passenger')

def get\_passengers\_list(self):

return self.\_\_passengers

print("Select from below")

print("Case 1 :Create Employee class object\nCase 2 :Create Passenger class object\nCase 3 :Create Flight class object\n 4 :View average salary of Emplyees\n"

+"Case 5 :View Flights count\nCase 6 :View Employees count\nCase 7 :View Passengers count\nCase 8 :View Employee Deails\nCase 9 :View Flight Details\nCase 10 :View Passengers Details\n"

"Case 11 :View Flight Passengers\nCase 12 :Add Passenger\n Case 13 :Remove Passenger\nCase 14 :Exit")

bool=1

employees\_list={}

passengers\_list={}

flights\_list={}

def addEmployee():

name=str(input("Enter name"))

EID = str(input("Enter EID"))

j = 1

while (j):

try:

age = int(input("Enter age"))

j = 0

except ValueError:

print('Non-numeric data.')

j = 1

while(j):

try:

salary=float(input("Enter salary"))

j=0

except ValueError:

print('Non-numeric data.')

department=str(input("Enter department"))

family=str(input("Enter family"))

E1=Employee(name,age,family,department,salary,EID)

employees\_list[EID]=E1

print("Employee added")

def addPassenger():

name=str(input("Enter name"))

j = 1

while(j):

try:

age=int(input("Enter age"))

j=0

except ValueError:

print("Non-numeric data")

pid = str(input("Enter PID"))

family=str(input("Enter family"))

P1=Passenger(name,age,family,pid)

passengers\_list[pid]=P1

print("Passenger added")

def addFlight():

name=str(input("Enter name"))

fid=str(input("Enter fid"))

j = 1

while (j):

try:

seats\_count = int(input("Enter seats count"))

j = 0

except ValueError:

print("Non-numeric data")

F1=Flight(name,fid,seats\_count)

flights\_list[fid]=F1

print("Flight added")

def EmployeeAvgSal():

print("Employee Average Salary: "+str(Employee.average\_salary()))

def EmpCount():

print("Employee Count: " + str(Employee.get\_employees\_count()))

def FlightsCount():

print("Flights Count: " + str(Flight.get\_flights\_count()))

def passengerCount():

print("Passenger Count: " + str(Passenger.get\_passengers\_count()))

def ViewEmployee():

if(len(employees\_list)>0):

for key in employees\_list:

print(str(key))

name=str(input("Select Employee"))

if(name in employees\_list):

print("Name : "+employees\_list[name].get\_name()+"\nSalary :"+str(employees\_list[name].get\_salary())+"\n Department : "+employees\_list[name].get\_department()+"\n Family : "+employees\_list[name].get\_family()+"\n Age : "+str(employees\_list[name].get\_age())+"\n EID : "+name)

else:

print("invalid employee ID")

else:

print("No employees to view")

def ViewPassenger():

if(len(passengers\_list)>0):

for key in passengers\_list:

print(str(key))

name=str(input("Select Passenger"))

if(name in passengers\_list):

print("Name : "+passengers\_list[name].get\_name()+"\n Age : "+str(passengers\_list[name].get\_age())+"\n Family : "+passengers\_list[name].get\_family()+"\n PID : "+name)

else:

print("invalid passenger ID")

else:

print("No passengers to view")

def ViewFlight():

if(len(flights\_list)>0):

for key in flights\_list:

print(str(key))

name=str(input("Select Flight"))

if(name in flights\_list):

print("Name : "+flights\_list[name].get\_fname()+"\n Seats Count : "+str(flights\_list[name].get\_seats\_count())+"\n Available Seats Count : "+str(flights\_list[name].get\_available\_seats\_count())+"\n FID : "+name)

else:

print("invalid flight ID")

else:

print("No flights to view")

def AddPassenger():

if(len(flights\_list)>0):

for key in flights\_list:

print(str(key))

name=str(input("Select Flight"))

if(name in flights\_list):

print('Selected Flight : '+flights\_list[name].get\_fname())

if(len(passengers\_list)>0):

for key in passengers\_list:

print(str(key))

pname = str(input("Select Passenger"))

if (pname in passengers\_list):

print('Selected passenger : ' + passengers\_list[pname].get\_name())

if(pname not in flights\_list[name].get\_passengers\_list()):

flights\_list[name].assign\_passenger(passengers\_list[pname],pname)

else:

print('Passenger already in flight')

else:

print("Invalid Passenger")

else:

print('No Passengers to add')

else:

print("invalid flight ID")

else:

print("No flights available")

def RemovePassenger():

if(len(flights\_list)>0):

for key in flights\_list:

print(str(key))

name=str(input("Select Flight"))

if(name in flights\_list):

print('Selected Flight : '+flights\_list[name].get\_fname())

if(len(flights\_list[name].get\_passengers\_list())>0):

for key in flights\_list[name].get\_passengers\_list():

print(key)

pname = str(input("Select Passenger"))

if (pname in flights\_list[name].get\_passengers\_list()):

flights\_list[name].remove\_passengers(passengers\_list[pname],pname)

else:

print("Invalid Passenger")

else:

print('No Passengers to remove')

else:

print("invalid flight ID")

else:

print("No flights available")

def ViewFlightPassengers():

if (len(flights\_list) > 0):

for key in flights\_list:

print(str(key))

name = str(input("Select Flight"))

if (name in flights\_list):

for i in flights\_list[name].get\_passengers\_list():

print(i)

else:

print("invalid flight ID")

else:

print("No flights available")

def Exit():

global bool

bool=0

def choices(argument):

switcher = {

1: addEmployee,

2: addPassenger,

3: addFlight,

4: EmployeeAvgSal,

5: FlightsCount,

6: EmpCount,

7: passengerCount,

8: ViewEmployee,

9: ViewFlight,

10: ViewPassenger,

11: ViewFlightPassengers,

12: AddPassenger,

13: RemovePassenger,

14: Exit

}

# Get the function from switcher dictionary

func = switcher.get(int(argument), lambda: "Invalid choice")

func()

while(bool!=0):

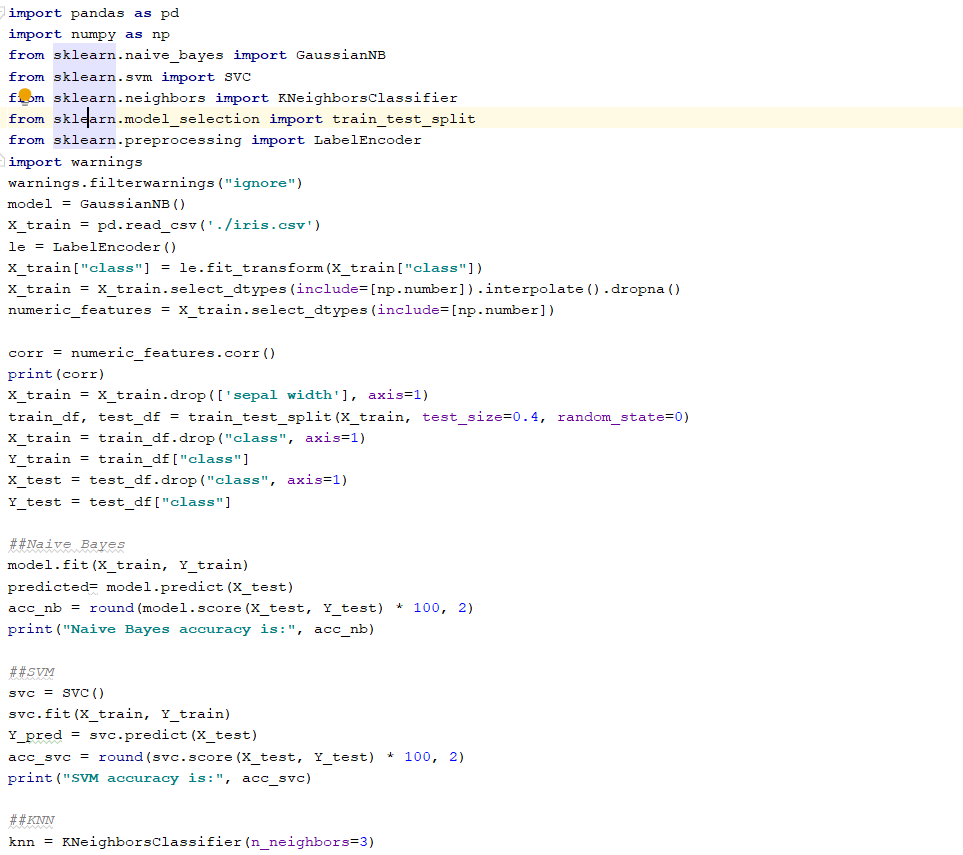
choice =input("Enter your choice")

choices(choice)

Task4:



Task5:



Task6:

