# DATA STRUCTURES AND ALGORITHMS

# LAB 2

## Zeenia Asr - 382732

**Task 1**

**Code:**

class Flight():

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

self.\_\_flight\_number = 0

self.\_\_destination = 0

self.\_\_distance = 0

self.\_\_fuel = 0

def calfuel(self):

if self.\_\_distance <= 1000:

self.\_\_fuel = 500

elif self.\_\_distance <= 2000:

self.\_\_fuel = 1100

else:

self.\_\_fuel = 2200

return self.\_\_fuel

def FEEDINFO(self, num, des, dist):

self.\_\_flight\_number = num

self.\_\_destination = des

self.\_\_distance = dist

self.\_\_fuel = self.calfuel()

def SHOWINFO(self):

print(f"Flight number: {self.\_\_flight\_number}")

print(f"Destination: '{self.\_\_destination}'")

print(f"Distance: {self.\_\_distance}")

print(f"Fuel: {self.\_\_fuel}")

def main():

obj = Flight()

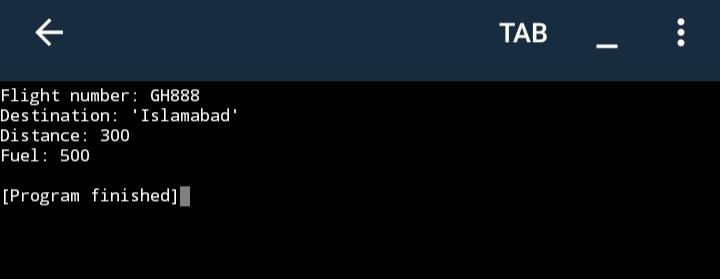
obj.FEEDINFO("GH888", "Islamabad", 300)

obj.SHOWINFO()

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Output:**



**Task 2**

**Code:**

class Batsman():

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

self.\_\_bcode = "0000"

self.\_\_bname = ""

self.\_\_innings = 0

self.\_\_notout = 0

self.\_\_runs = 0

self.\_\_batavg = 0

def calcavg(self):

return self.\_\_runs / (self.\_\_innings - self.\_\_notout)

def readdata(self, bcode, bname, innings, notout, runs):

self.\_\_bcode = str(bcode)

self.\_\_bname = str(bname)

self.\_\_innings = innings

self.\_\_notout = notout

self.\_\_runs = runs

self.\_\_batavg = self.calcavg()

def \_\_repr\_\_(self):

i = ""

i += f"Bcode: {self.\_\_bcode}\n"

i += f"Batsman: {self.\_\_bname}\n"

i += f"Innings: {self.\_\_innings}\n"

i += f"Notouts: {self.\_\_notout}\n"

i += f"Total runs: {self.\_\_runs}\n"

i += f"Batting average: {self.\_\_batavg}."

return i

def main():

obj = Batsman()

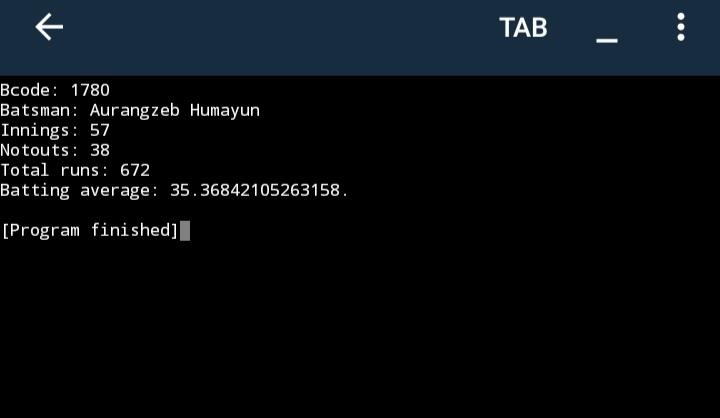
obj.readdata("1780", "Aurangzeb Humayun", 57, 38, 672)

print(obj)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Output:**

****

**Task 3:**

**Code:**

class Person():

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

self.name = name

self.\_\_state = None

def say(self, stuff):

self.\_\_state = stuff

print(stuff)

return stuff

def ask(self, stuff):

self.say(f"Would you please {stuff}?")

def greet(self):

self.say(f"Hello, my name is {self.name}.")

def repeat(self):

if self.\_\_state is None:

self.say("I squirreled it away before it could catch on fire.")

else:

self.say(self.\_\_state)

def main():

steven = Person("Steven")

steven.repeat()

steven.say("Hello")

steven.repeat()

steven.greet()

steven.repeat()

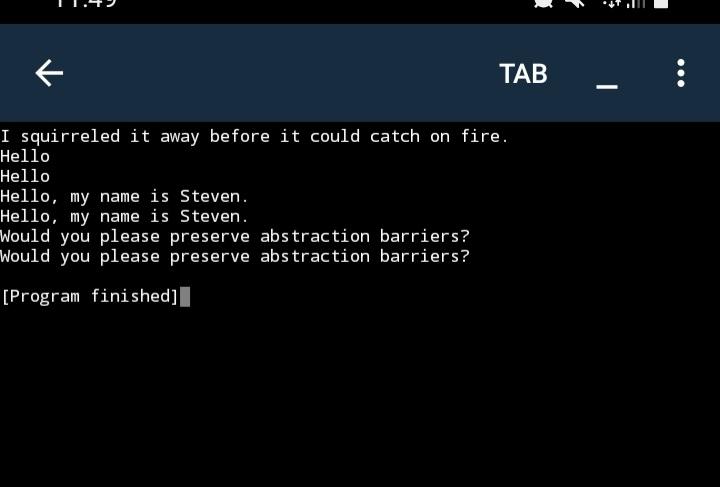
steven.ask("preserve abstraction barriers")

steven.repeat()

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Output:**

****

# 