LAB 2: Introduction to Object Oriented Programming in Python

AZKA AMER - 368792

Task#1:

CODE:

```
list1 = [[1, 2, 3], [2, 3, 3], [1, 3, 3]]
def find_max(list1):
    sum1=[]
    sum=0
    for i in list1:
        for j in i:
            sum+=j
        sum1+=[sum]
        sum=0
    return sum1.index(max(sum1))
```

RESULT:

```
In [2]: find_max(list1)
Out[2]: 1
```

Task#2:

CODE:

```
class Flight:
    def __init__(self):
        self.distance=0
        self.number=0
        self.destination=0
        self.fuel=0
```

```
    def calfuel(self):
        if self.distance<=1000:
            fuel=500
        elif self.distance<=2000:
            fuel=1100
        else:
            fuel=2200
        return fuel

    def feedinfo(self, distance, number, destination):
        self.distance=distance
        self.number=number
        self.destination=destination
        self.fuel=self.calfuel()

    def showinfo(self):
        print('Flight number:',self.number)
        print('Destination:',self.destination)
        print('Distance Trravelled:',self.distance)
        print('Fuel Used:',self.fuel)

def main():
    obj=Flight()
    obj.feedinfo(3469,'EK008','Bali')
    obj.showinfo()

if __name__=='__main__':
    main()
```

RESULT:

```
Flight number: EK008
Destination: Bali
Distance Trravelled: 3469
Fuel Used: 2200
```

Task#3:

CODE:

```
class Batsman:
    def _init_(self):
        self.bcode=0
        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):
        dt = ""
        dt += f"The bcode is {self.bcode}.\n"
        dt += f"The name of the batsman is {self.bname}.\n"
        dt += f"The innings is {self.innings}.\n"
        dt += f"There are {self.notout} notouts.\n"
        dt += f"Total runs are {self.runs}.\n"
        dt += f"The batting average is {self.batavg}."
        return dt

def main():
    obj = Batsman()
```

```
    obj.readdata(2020, "Joe Root", 2, 1, 198)
    print(obj)

if __name__ == '__main__':
    main()
```

RESULT:

```
The bcode is 2020.
The name of the batsman is Joe Root.
The innings is 2.
There are 1 notouts.
Total runs are 198.
The batting average is 198.0.
```

Task#4:

CODE:

```
class Person:
    def __init__(self, name):
        self.name = name
        self.current_statement = None

    def say(self, stuff):
        self.current_statement = 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.current_statement is None:
```

```
            self.say("I squirreled it away before it could catch on fire.")
        else:
            self.say(self.current_statement)


def main():
    steven = Person("Han Soo Jun")
    steven.repeat()
    steven.say("Hello!")
    steven.repeat()
    steven.greet()
    steven.repeat()
    steven.ask("preserve abstraction barriers")
    steven.repeat()


if __name__ == '__main__':
    main()
```

RESULT:

```
I squirreled it away before it could catch on fire.
Hello!
Hello!
Hello, my name is Han Soo Jun.
Hello, my name is Han Soo Jun.
Would you please preserve abstraction barriers?
Would you please preserve abstraction barriers?
```