CSCI 1411: Fundamentals of Computing
Lab 13
Due Date: December 1, 2023

Name: _____

Goals:

- Creating classes
- Instantiating objects for the given class
- PyDoc

Development Environment: IDLE

Deliverables:

1. This completed document with required screen shots.
2. Python file created for the lab. Name the file using the following format:
3. Your Python code for Part I of this lab (Student.py and GPACalculator.py).
4. Your Python code for Part II of this lab (BankAccount.py and Register.py).

How to take a **screen shot**:

- For a Windows 10: Use Snipping Tool to copy and press CTRL + V to paste screen shot.
- For Mac: Press Shift + Command (⌘) + 4 to copy and press Command (⌘) + V to paste screen shot.

Part I – Skill Practice

- Start IDLE.
- Create a new file.
- You will create two py files: Student.py (will contain code for Student class) and GPACcalculator.py (will contain code for main function).
- Type the following code in the file. Do not cut and paste. You will learn more by typing it in.
- Make sure that you read all comments to understand the code.
- We are using PyDoc style comments in Student.py file (Student class)

```
"""
Name:
Class: CSCI 1411-00X
Due Date:
Status:
"""#

class Student:
    """Student class which can keep track of student data
    (name, total credit hours, and total quality points. It
    can also calculate gpa"""


    def __init__(self, name, hours, qpoints):
        """Initialize name, credit hours, and quality points"""
        self.name = name
        self.hours = float(hours)
        self.qpoints = float(qpoints)

    def getName(self):
        """Return student's name"""
        return self.name

    def getHours(self):
        """Return total credit hours"""
        return self.hours

    def getQPoints(self):
        """Return total quality points"""
        return self.qpoints

    def gpa(self):
        """Calculate and return gpa"""
        return self.qpoints/self.hours

    def add_grade (self, grade_point, credits):
        """Add a new course information (Credit hours and Quality
        Points)"""
        self.hours = self.hours + credits
        self.qpoints = self.qpoints + grade_point
```

```
"""
Name:
Class: CSCI 1411-00X
Due Date:
Status:
"""

#Import Student class
from Student import *

def main():
    name = input ('Enter your name: ')

    # Create a student object with 0 credit and 0 quality points
    s1 = Student (name, 0,0)

    # Ask for number of courses and grades
    count = int (input ('Enter number of grades: '))

    # Ask for credit hours and total quality point for each course
    # and add the course using add_grade method in student class.
    for i in range (count):
        credit = float (input('Enter credit hours for course ' + str(i+1) + ': '))
        gp = float (input ('Enter grade points for course ' + str(i+1) + ': '))
        s1.add_grade (gp, credit)

    # Display information include student name, total credit hours, total
    # quality points and gpa
    print ('Student name:', name)
    print ('Total Credit Hours {0:.2f}'.format(s1.getHours()))
    print ('Total Quality Points {0:.2f}'.format(s1.getQPoints()))
    print ('GPA {0:.2f}'.format(s1.gpa()))
```

- Save the files `Student.py` and `GPAalculator.py`
- Click Run -> Run Module
- Type `help(Student)` in shell and it will display PyDoc comments. Take a screen shot and paste it below.

Screen Shot 1

- Type `main()` in shell to run your program
- If there are any syntax errors, then fix those errors and run your program again.
- Use the following input to test your program:

```
Enter your name: David Brown
Enter number of grades: 4
Enter credit hours for course 1: 4
Enter grade points for course 1: 12
Enter credit hours for course 2: 3
Enter grade points for course 2: 9
Enter credit hours for course 3: 4
Enter grade points for course 3: 16
Enter credit hours for course 4: 3
Enter grade points for course 4: 12
```

- You will get following output:

```
Student name: David Brown
Total Credit Hours 14.00
Total Quality Points 49.00
GPA 3.50
```

- If you get the correct result then your program is working as expected.
- Once you are satisfied with your results take a screen shot and past them below.

Screen Shot 2

Part II – Bank Transactions Register

- Implement a class named `BankAccount`. Every bank account has a starting balance of $0.00. The class should implement methods to accept deposits and make withdrawals.
    - `__init__(self)`: Sets the balance to 0.
    - `deposit(self, amount)`: Deposits money. Return `True` if transaction is successful. Return `False` if amount is less than 0 and ignore the transaction.
    - `widthdraw(self, amount)`: Withdraws money. Return `True` if transaction is successful. Return `False` if amount is more than the balance and ignore the transaction.
    - `getBalance(self)`: Returns the amount of money in the account.
- Include PyDoc comments for your classes and methods.
- Write a program with main function which will perform the following tasks:
    - Create a `BankAccount` object.
    - Ask user for the number of transactions.
    - For each transaction ask for type of transaction and amount of transaction.
    - If type is deposit then use `deposit` method to complete the transaction. If return value from the `deposit` method is `False` then display an error message.
    - If type of the transaction is withdarw then use the `withdraw` method to complete the transaction. If return value from the `withdraw` method is `False` then display an error message.
    - After the loop display the number of transactions completed and account balance. If any transaction is rejected then it will not be included in the count of completed transactions.
- Save the files `BankAccount.py` and `Register.py`
- Click Run -> Run Module
- Type `help(BankAccount)` in shell and it will display PyDoc comments. Take a screen shot and paste it below:

Screen Shot 3

- Type `main()` in shell to run your program
- If there are any syntax errors then fix those errors and run your program again.

- Sample I/O is as follows:

```
Enter number of transactions: 5
Enter transaction type: deposit
Enter transaction amount: -45
Deposit amount $-45.00 is less than 0. Transaction rejected
Enter transaction type: deposit
Enter transaction amount: 100.45
Transaction was successful. Your account balance is $100.45
Enter transaction type: withdraw
Enter transaction amount: 19.65
Transaction was successful. Your account balance is $80.80
Enter transaction type: withdraw
Enter transaction amount: 100.99
Withdraw amount $100.99 is higher than balance of $80.80.
Transaction rejected
Enter transaction type: deposit
Enter transaction amount: 99.99
Transaction was successful. Your account balance is $180.79
After 3 transactions, your balance is: $180.79
```

- If you get the correct result then your program is working as expected.
- Once you are satisfied with your results a screen shot and past them below.

Screen Shot 4

Add the following comment block at the top of your file. Also make sure that you have block of comments after start each of the functions.

```
"""
Name:
Class: CSCI 1411-00X
Due Date:
Description:
Status:
"""
```

Rubric for Lab 13:

Criteria	Rating
Part 1: Screen shot 1	Screen shot included – 5 points No screen shot included – 0 points
Part 1: Screen shot 2	Screen shot included – 5 points No screen shot included – 0 points
Part 1: Python Files	Submitted Student.py and GPACalculator.py files – 5 points No Python files submitted – 0 points
Part 2: Screen shot 3	Screen shot included – 5 points No screen shot included – 0 points
Part 2: Screen shot 4	Screen shot included – 5 points No screen shot included – 0 points
Part 2: Constructor	Initialized balance to 0 – 5 points Does not initialize balnace to 0 or is not done – 0 points
Part 2: Deposit method	Updates the balance correctly – 5 points Does not update the balance or is not done – 0 points
Part 2: Deposit method	Return the correct boolean values – 5 points Does not return the correct boolean value – 0 points
Part 2: Withdraw method	Updates the balance correctly – 5 points Does not update the balance or is not done – 0 points
Part 2: Withdraw method	Return the correct boolean values – 5 points Does not return the correct boolean value – 0 points
Part 2: Get Balance method	Returns the correct balance – 5 points Does not return the balance or is not done – 0 points
Part 2: Main function	Crearte the BankAccount object – 5 points Does not create the BankAccount object – 0 points
Part 2: Main function	Displays the appropriate error message (if return value from Deposit or Withdraw method is False) – 5 points Does not display errore messages – 0 points
Part 2: Main Function	Use the appropriate methods (Deposit/Withdraw) to update the balance – 5 points Does not use appropriate methods – 0 points
Part 2: Main Function	Displays the correct summary information after the end of the loop – 5 points Displays the incorrect summary information (also counts invalid transactions) – 2 points Does not display the summary information – 0 points
Part 2: PyDoc	PyDoc is include for all classes and methods – 5 points PyDoc is not included for all classes and methods – 0 points
Total	80