ITSE 1359 – Lab 11 Assignment (Chapters 8, 10, and 11):

NOTE: Lab 11 requires you to work on and submit two solutions (four files total). Read this document carefully and comply with submission and grading criteria.

Your first problem (Drop Lowest Grade with Refactoring) will result in one file and require refactoring of a previous program. The second program (Employee Unit Testing) will result in three files, which should include an employee class, employee client, and employee test.

Pseudocode and screenshots are provided for both programs. Do not forget general and specific comments at top of each file.

Program 1 – Drop Lowest Grade with Refactoring:

Problem-statement: This program will start with your lab 4 program – drop_grade.py. Refer to that problem statement to review logic and design. Save that file as drop_grade_refactored.py and update to new specs. Your task is to transform this old program into one that has been refactored and has a main function. *Your output should be exactly the same as before, but your code must be modular and utilize a main function.*

Coding: In this program you will use functions to make the program more modular and open to code reuse. You should put all your general tasks into functions and then call them when needed from the main function. The names and design of these functions is up to you. I do offer my pseudocode for your comparison.

Output: Your output should look the same as the original program. The only difference is that instead of coding without modularity in mind, you will organize your code so that it will be placed into functions and capable of code reuse.

Resources (Drop Lowest Grade with Refactoring):

- Refer to Chapter 8 for a review of functions (if needed).
- Refer to Chapter10 for discussion of refactoring (p. 206 – 208).
- Refer to Ch 10 - Part 4 YouTube video for discussion on refactoring. GoTo **18:20** on the timeline.
- Refer to Chapter 11 (p. 211 – 229) for a discussion of the main function.
- Refer to Ch. 11 YouTube video and Unit IV Overview in your Canvas class for more information about the main() function (optional – FYI).

Output (drop_grade_refactored.py):

```
DROP LOWEST GRADE PROGRAM:

Grades:
100
80
70
60
90
Number of grades: 5
Average: 80.00
Lowest grade: 60

LOWEST GRADE DROPPED.

Grades:
100
80
70
90
Number of grades: 4
Average: 85.00
Lowest grade: 70

[Finished in 0.1s]
```

Specific Comments / Pseudocode (drop_grade_refactored.py):

```
# function definition for show grades (receives grades list):

    # print “Grades: “
    # use for loop to print grades in list

# function definition for display grade stats (receives grades list):

    # use sum function on grades list to find the total
    # calculate the average by dividing total by len(grades)
    # print number of grades using len function
    # print average using f string - format to 2 decimal places

# function definition for display lowest grade (receive grades list):

    # using min function on grades list to find the lowest grade
    # print lowest grade using f string

# function definition for drop lowest grade (receive grades list):

    # find the lowest grade using min function
    # remove the lowest grade from the list
    # state lowest grade dropped

# function definition for main():

    # print report name
    # create list of 5 grades and initialize (use 100, 80, 70, 60, 90)
    # call function that shows grades (pass grades list)
    # call function that displays grade stats (pass grades list)
    # call function to display lowest grade (pass grades list)
    # call function that drops lowest grade (pass grades list)
    # call function that shows grades (pass grades list)
    # call function that displays grade stats (pass grades list)
    # call function to show lowest grade (pass grades list)
    # print blank line

# main function check and call
```

Coding Examples(drop_grade_refactored.py):

Below is my version of the main function along with the call to the main function. Your task is to write the functions that are called in the main function. The functions that you write must be in the same file as the main function and the call to the main function. So, you submit one file for this program.

```
def main():
    """ main function controls flow of program """

    print("DROP LOWEST GRADE PROGRAM: \n")

    grades = [100, 80, 70, 60, 90]

    show_grades(grades)

    display_grade_stats(grades)

    display_lowest_grade(grades)

    drop_lowest_grade(grades)

    show_grades(grades)

    display_grade_stats(grades)

    display_lowest_grade(grades)

    print()

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

FYI: Why use a main function in Python?

- Refer to chapter 11 in your book.
- Good reference: [Main Function in Python | Edureka]

Program 2 – Employee Unit Testing:

Problem-statement: Create a project that creates an employee class, uses the employee class, and tests the employee class (unit testing). Use separate files for each of these project components.

The first file should contain the Employee class (employee_class.py). This Employee class should contain the init() method and a special method that gives a raise.

The second file should be a client that uses the employee class (employee_client.py). In this client, you should have two instances of the Employee class and use the method that gives a raise on each instance.

The third file should contain the testing of the Employee class (employee_test.py). This test file should contain a setup method, a call to the default raise method and a call to the custom raise method.

Output: Your output should look the same as the screenshots provided.

Employee Class (employee_class.py): In this file, write a class called Employee. The __init__() method should take in a first name, a last name, and an annual salary, and store each of these as attributes. In your Employee class, write a method called give_raise() that adds $5,000 to the annual salary by default but also accepts a different raise amount.

Employee Client (employee_client.py): In this file, write a program that uses the Employee class. In this client of the Employee class, instantiate two employees. Give a default raise of 5K to one employee and then give a custom raise of 10K to the other employee.

Employee Test (employee_test.py): In this file, write a test case for Employee. Write two test methods, test_give_default _raise() and test_give_custom_raise(). In your class, use the setUp() method so you don't have to create a new employee instance in each test method. Make sure you also test for main() function. Run your test case, and make sure both tests pass.

How do I start this project?

1. First, create the Employee class (employee_class.py).
2. Second, create the client program (employee_client.py) and make sure the project is working before you do any testing.
3. Lastly, test your project (employee_test.py).

Resources (Employee Unit Testing):

- This project is a modified version of Try It Yourself 11-3 Employee (page 221). See online solutions.
- This project is discussed in my YouTube video lecture of Part 2 of Chapter 11. GoTo **29.00** on the timeline.
- For a discussion of the main() function see Chapter 11 and "Unit 4 Overview – Read this First!" in your canvas class shell.

Comments for Employee Class (employee_class):

```
# class Employee():
    # init method receives self, first name, last name, and salary)
        # initialize the employee
    # give raise method receives self, and default raise of 5K
        # increment salary
```

Example Code (employee_class):

```
# look at TIY 11.3 online solution
```

Comments for Employee Client (employee_client):

create two instances of Employee
show attributes of employee object 1 and give default raise of 5K
show attributes of employee object 2 and give custom raise of 10K

Example Code (employee_client.py):

```python
from employee_class import Employee

# create instance of Employee
eric = Employee('eric', 'matthes', 65_000)
bob = Employee('bob', 'benavides', 100_000)

# show attributes of object 1 and give default raise
print(eric.first)
print(eric.last)
print(eric.salary)
eric.give_raise()
print(f"Salary after default raise of 5K: {eric.salary}")
print()

# show attributes of object 2 and give custom raise
print(bob.first)
print(bob.last)
print(bob.salary)
bob.give_raise(10_000)
print(f"Salary after custom raise of 10K: {bob.salary}")
```

Output of Employee Client (employee_client.py):

```
Eric
Matthes
65000
Salary after default raise of 5K: 70000

Bob
Benavides
100000
Salary after custom raise of 10K: 110000
[Finished in 0.1s]
```

NOTE: Formatting for currency is optional.

Comments for Employee Test:

```
# import unittest
# import Employee

# class TestEmployee:

    # setup method:

        # instantiate one employee object
        #       - passing first name, last name, and 65000

    # test give default raise method:

        # call give_raise() method of employee object
        # call assertEqual() method – pass object salary and 70000

    # test give custom raise method:

        # call give_raise() method of employee object and pass 10000
        # call assertEequal() method – pass 75000

# main()
```

Example code for Employee Test (employee_test.py):

See TIY 11.3 online solution.

Output of Employee Test (employee_test.py):

```
..
----------------------------------------
Ran 2 tests in 0.000s

OK
[Finished in 0.2s]
```

Submit your lab assignment:

Since this lab assignment involves multiple files, create a folder called "lab11_username" where username is your Collin username. Put your FOUR files in this folder (drop_grade_refactored.py, employee_class.py, employee_client.py, employee_test.py). Zip your folder.

Using the Canvas assignment tool, upload your zipped file to the lab 11 assignment. After you have uploaded your zipped file, submit your lab assignment.

See last page of this document for directions on how to create a folder and zip a folder. If you do not submit right, you may get points off or no grade at all.

All lab assignments must be submitted using the CANVAS assignments tool. Lab Assignments will not be accepted any other way**.** Make sure you submit your work to the right lab assignment number otherwise you will not get credit.

Grading Criteria:

- ✓ Don't forget general comments. Specific comments are optional.
- ✓ Use white space to make your code easy to read.
- ✓ Comply with PEP-8 conventions for variable names, file names, etc.
- ✓ Satisfy the problem definition and other grading standards.
- ✓ Your work should not have syntax errors (unless you want points off).
- ✓ Your work must be your own.
- ✓ Match your output screen to screenshot provided.
- ✓ Any deviations from lab specifications will result in points off.

VIP Videos:

ITSE 1359 Videos: VIEW MY PYTHON PLAYLIST

- Review videos from chapters 8, 10, and 11 to help you understand this lab assignment.

Closing:

If you have questions about this lab send me a message using Canvas inbox or attend zoom office hours. See Unit 0 for link for my office hours. Programming tutors are also available and listed in your Canvas class as announcements.

By-the-way, a great way to get ready for your lab assignments (and exams) is do the Try It Yourself problems in your book. Most of the solutions are on the authors website and I also discuss them in my YouTube video lectures.

Another way to get ready for your lab assignments is to review my lecture notes for each chapter. In Unit 0, find the link for my chapter notes.

Warning: This lab may be complex to some beginners. A beginner may need 5 to 15 hours to complete it. Please start early because there will be no extensions.

Consider doing this lab over a three-day period:

- ✓ Day 1 – Get confused and run out of time.
- ✓ Day 2 – Research and debug errors.
- ✓ Day 3 – Polish, double-check everything, and submit.

Have an exception free day!

Prof. Benavides

Resources – Create a folder, put files in it, and Zipp folder:

Win 10 - How do I create a folder?

1.) Right-mouse click on the Desktop.
2.) Point to “New” and then click on “Folder”.
3.) Type in the name of your folder and press Enter.

Win 10 – How do I put files in a folder?

- Just drag them to your folder if they are not in there to begin with.

Win 10 - How do I zip a folder? (Do it this way please.)

1.) Right-mouse click on the folder.
2.) Point to “Send to” and then click “Compressed (zipped”) folder”.
3.) Now you see zipped file by same name as folder with a zipper on it.
4.) Do not send a .rar file or any other file type other than .zip.

macOS - How do I create a folder?

1.) Right-mouse click on the Desktop.
2.) Select “New Folder”.
3.) Type in the name of your folder and press Enter.

macOS – How do I put files in a folder?

- Just drag them to your folder if they are not in there to begin with.

macOS - How do I zip a folder? (Do it this way please.)

1.) Right-mouse click on the folder.
2.) Select “Compress”. Name of folder appears after word Compress.
3.) Now you see zipped file by same name as folder with a zipper on it.
4.) Do not send a .rar file or any other file type other than .zip.