

**College of Engineering**

## CS 1337/1337L Introduction to Object-Oriented Programming

## Dr. Fatemeh (Azi) Tabei

## Fall 2021

**Lab - 09/14/2021**

Please upload your assignments through your WTClass on time. The assignment will NOT be accepted if the time is passed or not submitted properly through the WTClass. It is your responsibility to make your submissions before the deadline.

1. **Test and debug a Test Scores program**

In this exercise, you will test and debug a variation of the Test Scores program of chapter 3. 1.

In IDLE, open the **test\_scores.py** file that’s in your lab assignment folder: Then, review the code.

1. Create a test plan that thoroughly tests the program with valid data.

Check for any syntax errors. To test the variables, test valid inputs, test invalid inputs. X, 10, 2, and 0 as valid inputs. -10, 120, and X as invalid inputs. By testing the before variables, it is checking if there is any logic errors. Optimize the user interface.

1. Use the following table as a guide to testing the program. Then, note any inaccurate results that you discover during testing.

| input | Expected results | Is it data valid or invalid? | Is the code result correct? |
| --- | --- | --- | --- |
| x | Program to quit, give termination message. | valid | no |
| -10 | Enter a digit >0 and <100 | invalid | yes |
| 0 | Program to accept the input | valid | no |
| 120 | Enter a digit >0 and <100 | invalid | yes |
| 10 | Program to accept the input | valid | yes |
| 2 | Program to accept the input | valid | yes |
| x | Program to quit | valid | yes |

1. Debug any logic errors. Please copy and paste the line that has the logic error and also the version that you debugged. Explain about your answer.

Hint: Add **print("counter =", counter)** before the else statement to find the logic error.

Counter was being added twice, and was skewing the average.

ERROR CODE

while True:

test\_score = input("Enter test score (or 'x' to quit): ")

if test\_score != "x":

test\_score = int(test\_score)

counter+=counter

else:

break

CORRECT CODE

while True:

test\_score = input("Enter test score (or 'x' to quit): ")

if test\_score != "x":

test\_score = int(test\_score)

else:

break

1. Test the program with the same data to be sure it works correctly.

| input | Expected results | Is it data valid or invalid? | Is the code result correct? |
| --- | --- | --- | --- |
| 10 | Program to accept the input | Valid | Correct |
| 2 | Program to accept the input | Valid | Correct |
| x | Program to terminate and give answer | Valid | Correct |

1. **Trace and test the functions of the Future Value program**

In this exercise, you will trace the operation of the calculate\_future\_value() function of a Future Value program. You’ll also use the IDLE shell to test the functions in this program. You will see print() functions to trace the execution of a function 1.

1. **Use print() functions to trace the execution of a function**
2. In IDLE, open the future\_value.py file that’s in your lab assignment folder.
3. Test the program and note that the future value results are obviously inaccurate. In fact, the calculate\_future\_value() function has two logic errors. Please copy and paste the lines that have logic errors.

def calculate\_future\_value(monthly\_investment, yearly\_interest, years):

# convert yearly values to monthly values

monthly\_interest\_rate = yearly\_interest/12

months = years \* 12

# calculate future value

future\_value = 0.0

for i in range(1, months):

future\_value += monthly\_investment

monthly\_interest = future\_value \* monthly\_interest\_rate

future\_value += monthly\_interest

1. Debug the code. Please copy and paste the lines that you fixed the error.

Hint: To debug this problem, scroll down to the calculate\_future\_value() function, and add print() functions that show you how the values of the variables change each time through the for loop.

def calculate\_future\_value(monthly\_investment, yearly\_interest, years):

# convert yearly values to monthly values

monthly\_interest\_rate = yearly\_interest/12/100

months = years \* 12

# calculate future value

future\_value = 0.0

for i in range(1, months+1):

future\_value += monthly\_investment

monthly\_interest = future\_value \* monthly\_interest\_rate

future\_value += monthly\_interest

1. **Use the IDLE shell to test the functions of this program**
2. Run the Future Value program to make sure its functions are loaded into the shell. Then, cancel the execution of the program.
3. Test its three functions as shown in figure 5-6. This shows you how easy it is to test a function without running the entire program.

| Function | Test statement | Results | Explain about the result (Is it valid or not and why) |
| --- | --- | --- | --- |
| get\_number(prompt, low, high) | get\_number("prompt", 0, 10) | Prompt | No, it was not a float on int |
| get\_number("prompt=", 0, 10)  note: set prompt as 12 | Runtime error caused by a value error | Input validation loop |
| get\_number("prompt=", 0, 10)  note: set prompt as 7.7 | 7.7 | I gave an input that was between 0-10 |
| get\_number("prompt=", 0, 10)  note: set prompt as -1 | Runtime error caused by a value error | Input validation loop |
| get\_number (5,0,10) | 5 | No, I changed the string “Prompt=” to an integer that I gave the program. |
| get\_integer(prompt, low, high) | get\_integer("prompt=", 0, 10)  note: set prompt as 1 | 1 | Yes, I gave an input that was between 0-10 |
|  | get\_integer("prompt=", 0, 10)  note: set prompt as 7.7 | Runtime error caused by a value error | No, without a rounding function the program cannot print an integer from a float |
| calculate\_future\_value (monthly\_investment, yearly\_interest, years) | calculate\_future\_value(-1, 15, 50) | -139718.03 | No, as -1 is not within the parameters of the monthly\_investment variable. |