

# CSC 401 : Introduction to Programming

Professor: Dan Walker

Midterm Exam

Summer 2021

## General Instructions

1. Download the following files from Session5/Midterm
  - csc401\_sum2021\_midterm.py
  - csc401\_sum2021\_midterm\_test.txt
  - work.txt

Update csc401-sum2021-midterm.py to contain your solutions to the problems described below.

1. No collaboration is permitted during this test. You may not consult with the tutors on this test. The only person you may interact with on this test is me. I will answer questions related to the interpretation of the instructions only.
2. No imports may be added to the source file. All 5 problems are solvable with plain vanilla python.
3. In the problems below, assume that I will test with additional data that is not shown. This additional data will comply with any assumptions given in the problem description but could vary in length and magnitude compared to the provided examples.
4. The doctest results for all tests passing will have this summary:

```
29 tests in csc401_midterm_test.txt
29 tests in 1 items.
29 passed and 0 failed.
Test passed.
```

5. As usual, your deliverables are:
  - a. your updated .py source file
  - b. any additional data files you created for testing
6. When you are done:
  - a. upload your individual files to D2L, into the midterm submission folder. No zip/rar, etc. files will be accepted
  - b. double check that you have uploaded all the right files

## Tips

1. Add a description for each function in the location indicated in the doc string. This is part of your grade.
2. Enter your code AFTER the doc string, replacing the word "pass".
3. Pay close attention to whether the function is supposed to print, return or something else.
4. To get full credit for a problem, your solution must pass the provided tests.
5. Passing the tests does not guarantee full credit. If your solution arrives at the correct result but contains irrelevant code, or contains hard coded values that the function should have calculated, points will be deducted.
6. Develop your solutions incrementally. I recommend you add error handling last, since you will pass most tests without error handling. You will not get full credit without the error handling but it is better to get something working than to have a function that does have error handling but doesn't actually work in any test cases.
7. If you print or add other statements for testing purposes, be sure to remove them before delivering your code to d2l.
8. Make sure the last thing you do before upload to D2L, is test. Too many students have lost points because they made 1 more "insignificant" change before upload, that actually caused their code to fail.

## **Problem 1: calculate\_fine(driver\_id, speed\_limit, actual\_speed) - returns the fine amount**

In this problem you are calculating the fine to be given a driver based on their recent driving history and the number of MPH over the limit they were going. The inputs to the function are:

**driver\_id** : the id of the driver in question.

**speed\_limit** : the legal limit for the area where the driver was stopped

**actual\_speed** : the actual speed at the time of the incident

### **Process**

1. figure out how many miles per hour over the limit the driver was going (called mph\_over below)
2. if mph\_over is less than or equal to 5 and the driver's id does not appear in **recent\_tickets** list, they get off with a warning (fine = 0.0). **recent\_tickets** is provided in the template file.
3. if mph\_over is less than or equal to 5 and the driver's id **does** appear in recent\_tickets list. the fine is \$50
4. if mph\_over is between 6 and 10 (inclusive at both ends), the fine is \$75
5. if mph\_over is greater than 10, the fine is \$100

Review the test file for additional information.

## **Problem 2: average\_work(file\_name) - returns average hours worked**

The single parameter to this function is the name of a file that contains the hours worked for several workers. You must read the hours and calculate the average of the total hours and **return** this value.

The file contains a name and a number of hours on each line. There are 1 or more spaces between the name and the hours.

If there is any problem opening the file, print "Error encountered handling file" and return nothing.

If the file is empty, return 0.0

There are no invalid values in the file.

### **Problem 3: sum\_threes(start, finish) - prints the average of a subset in the given range**

Write a function that is given as input 2 values representing the beginning and ending values of a range. the function sums up only those numbers that are evenly divisible by 3 in that range and prints a message about the result:

**Total: <nnn> Count: <nnn> Average: <nnn>**

where the <nnn> slots are replaced by the appropriate calculation. For example,

`sum_threes(1,6)`

would produce

**Total: 9 Count: 2 Average: 4.5**

This is because the range from 1 up to and including 6 is 1, 2, 3, 4, 5, 6 and the numbers evenly divisible by 3 in that range are 3 and 6, which totals 9. 2 numbers were found so the average is  $9/2$  or 4.5.

If non integers are given as parameters, the function should print:

**start and finish values must be integers**

If no numbers divisible by three are found, the function should print

**No multiples of 3 were found**

## Problem 4: switch(file\_name) - writes results to the given file

For this problem, you will repeatedly:

1. prompt the user for a string
2. process the string using the instructions given below
3. write the result to the given file
4. stop when the user enters a blank line (just presses return)

### Instructions

The function is called “switch” because the process is to chop the user’s string in half and write out the 2nd half and then the first half. The 2 halves are written to the same line and they are separated by a vertical bar. Here are some examples:

User string: “1234” write “34 | 12”

User string: “12345” write “345 | 12”

User string: “This is my bicycle” write “y bicycle | This is m”

User string: “a” write “a | ”

### Tips

1. Remember to write a new line character.
2. The function does not print or return anything, It just writes to the given file
3. This function does not have an automated test due to the use of the input() function.

Here is a sample session:

```
>>> switch('test_0627a.txt')
Enter text: 1234
Enter text: 12345
Enter text: This is my bicycle
Enter text: a
Enter text:
>>>
```

The contents of test\_0627a.txt is now:

```
34 | 12
345 | 12
y bicycle | This is m
a |
```

## Problem 5: find and record defects in the provided code

The solution to this problem is not code. Review the code below and document any flaws you can find. The flaws can be syntax errors, logic errors or just bad practices.

### The code

```
def f(x):  
    '''return the count of all of the even numbers from 0 upto and  
    including x  
    '''  
    cnt = 0  
    for i in x  
        if i % 2 == 0:  
            cnt += 1  
        return cnt  
if cnt > 10:  
    print(f"more than {7.2x} were found".format(cnt))
```

Add your findings to the multi-line string at the bottom of the template file. Find the text “Enter your solution to problem 5 between these 2 lines” and add your findings there. Make sure your text does not expand beyond the boundaries of the multi line string. Make sure that nothing you enter there causes a syntax error.