

Assignment 1

ENSF 300 Fall 2023

Department of Electrical and Software
EngineeringSchulich School of Engineering

Due: Monday October 2, 2023 at 11:59 PM

The purpose of these assignments is to build your skills using and utilizing the day-to-day tools you will be needing as a software developer, and learning about Database management Systems and how to use them to effectively store and access data from any program you develop. This assignment is one of five and is worth 10% of your final grade (the worst assignment score out of the five will be dropped).

1. Assignment #1 – Learning Outcomes


- Review python
- Working in a team setup
- Effectively use git to collaborate and maintain version history/control of your project
- Test and debug your code using the methods and tools discussed in lecture

2. Program Specifications

This is adapted from the ENDG233 Fall 2021 portfolio assignment 1 with some changes. The program specifications are discussed below.

Design a terminal-based calculator application that meets the following design specifications:

- Prompt the user to enter a mathematical expression that includes three integer values and two operators. You may prompt for the input to be entered in any form or order you like, but be sure to give the user clear instructions.
- Your program must correctly calculate and print the answer to the terminal.
- The final calculation must show the original expression and the final answer. You must use the output formatting shown in the attached output example.
- Your application must handle the order of operations correctly! (i.e. * and / should be calculated before + and -)
- Your answer should remain in integer format. Any division operations should use floor division.
- You must validate user input (only accepting integer inputs for values and the listed operators above for the operations prompting). In case wrong entry is given you must prompt the user repeatedly until the correct format is provided. Exception handling is allowed (if you know how to use it)
- Your code must include the following:
 - A separate function for each of the operations (myAdd, mySub, myMul, myDiv)
 - A function for validating integer input
 - A function for validating operation input
 - A function to evaluate the expression that will take a list containing the two operations and another list that has the three values (operands) to evaluate – this function has the control flow “if” statements.
 - A display function that takes the list of operations – with an appended “=” at the end – and the list of values(operands) – with the result appended at the end – and displays the full expression to the user.

- 
- A main function that executes the program flow
 - Use the python special variable “__name__” to run the program
 - You may not use any built-in Python functions or external modules.
 - Your code will be run by the TAs as your end user using VS Code and a minimum of Python 3.9.
 - FAQs about the assignment will be answered on the D2L discussion boards. Please check the boards for any clarifications before submitting.
 - The grading rubric will be released the day of Github assignment release.
 - This document is released ahead of your assignment

3. Assignment Tasks

- Start your assignment by using the link posted on D2L. this will redirect you to Github Classroom where you will accept your assignment and join your group. First member of a group to accept the assignment will create a group with any name you want (get creative but make it SFW) and coordinate with the other members to not create redundant groups. Note that group members on the GitHub repo should be the same as the group members on the D2L created groups
- This will generate a repo that is shared between the members of the group
- The initial repo is a clone from the git tutorial library that has information about github and some terms and guidelines, read them carefully as your first task of the assignment
- Before coding:
 - Do your design phase. Show your design in the form of a flowchart or pseudocode. This will be part of the final submission
 - Do your planning phase. Split the code requirements into 2 or more flows that you can easily work on without interfering with other team members to avoid collisions in the repo. Distribute the tasks – does not have to be by branch, can be by tasks within a branch and you can work on a branch at the same time but avoid collisions – which includes development tasks, review tasks, and testing tasks. This will also be part of your final submission
- While coding:
 - Commit often – not after every line of code, more like after each block completion – with useful descriptive commits.
 - Test your code. White box testing will be essential at the point of development, so do unit testing – use pyunit tests if you know how or simply have some testing code written that does some function calls and compares output – and functional testing, some elaborative examples will be covered to give you some guidelines in the lecture or lab
 - When code is ready, push it to the repo and make a pullrequest so that the member tasked with reviewing your task can check it out and discuss – other members can also provide their opinion.
 - Document your findings for all these points as you will need to reflect on them in your final submission
- After completion:
 - Do more testing. This is where each member will test other modules developed by other members without needing to know the code. The person in charge for development will provide the behavior description and another member will do unit testing for the module
 - Work together on integration testing the program by testing incrementally until the full program is put together
 - Document your findings for all these points as you will need to reflect on them in your final submission
- Don't forget to merge branches to main when done with a branch

4. What to submit and where

Your submission will be on two ends: D2L and Github

1- D2L:

- a. Create a report with the following items and submit it to D2L
 - i. Cover page with group information:
 1. Group name on GitHub and group name/number on D2L
 2. Group members information: name, ID, email
 3. Task distribution table: development, review, testing
 4. Link to repo
 - ii. Item 1 proof of design phase
 - iii. Item 2 planning phase including code modularization (code split reasoning) and task assignment
 - iv. Item 3
 1. Proof of individual commits, initial testing, push, and pullrequests (screenshots) – make a sub section per member
 2. Proof of review and testing
 - v. Item 4, reflect on your experience attempting this assignment again as a team using the tools provided – do not write an essay, few paragraphs are enough.
- b. A link to the repo in the comments of D2L submission – help your TA find your work faster.

2- GitHub

- a. Your project code
 - i. History must show consistent contribution by all members – make your commits often and do not forget to pullrequest.

5. Assignment 1 Example Output

You may choose how to prompt for input as long as it is clear to the user.

You must display the expression and final answer using the following format and spacing:

Entered expression: $10 + 2 / 5$

Your final answer = 10

```
(venv) C:\Users\eamarasc\Dropbox\Teaching\ENDG 233\Fall 2021\Portfolio\Assignment 1>python calculator.py
Enter the first value: 10
Enter the first operator: +
Enter the second value: 2
Enter the second operator: /
Enter the third value: 5
Entered expression: 10 + 2 / 5
Your final answer = 10

(venv) C:\Users\eamarasc\Dropbox\Teaching\ENDG 233\Fall 2021\Portfolio\Assignment 1>python calculator.py
Enter the first value: 6
Enter the first operator: *
Enter the second value: 9
Enter the second operator: -
Enter the third value: 2
Entered expression: 6 * 9 - 2
Your final answer = 52

(venv) C:\Users\eamarasc\Dropbox\Teaching\ENDG 233\Fall 2021\Portfolio\Assignment 1>python calculator.py
Enter the first value: 4
Enter the first operator: +
Enter the second value: 15
Enter the second operator: *
Enter the third value: 2
Entered expression: 4 + 15 * 2
Your final answer = 34
```

6. Assignment 1 Grading Rubric

Design and planning process 20% (evaluated from the report)

1. Proof of program design, can be a pseudo code, flowchart, or even sequential bullets/tasks representing the sequential operation of the program (10%)
2. In few paragraphs, explain the reasons and decisions made for your program design – this includes bundling of certain functions in a separate file, reasoning behind the way tasks are split and assigned, predetermined branches that will be created, and so on (10%)

Program requirements 25%

1. Program has the required four functions to carry the operations stated in the handout and functions as desired(10%)
2. Main program is written in a function and is called through checking the name variable `__name__` (5%)
3. Integer validating function (this function can use predefined functions for the validity check) (2%)
4. Operation validating function (2%)
 - Points 3 and 4 can be combined in one function
5. Program defines an evaluate function for evaluating the full expression using the other defined functions (4%)
6. Program has the display final output function that prints out the full equation and its result (2%)

Github hygiene 30%

1. Use of frequent commits with update description in the commit (10%)
2. Use of pullrequests after pushing to seek review by other members (10%)
3. Branching for different modules and merging them back to main (10%)

Testing and reviewing 20%

1. Whitebox test your developed task, you can do that by doing some function calls for the functions you developed to test the output vs the expected output, or do a path discovery (path discovery is recommended for whitebox testing more complex functions or the entirety of the code once everything is integrated) (5%)
2. Gray and Black box testing other modules developed by other members. Best way to do that is by unit testing (5%)
 - You can use PyUnit if you know how to or want to learn it
 - Or you can execute some function calls to compare the output vs the expected output
3. For proof of reviewing provide snapshots of the responses to the pull requests (comments left by reviewers on the pull request) (5%)
4. Incremental testing – similar to what you did in unit testing but show tests along the way as you start building the program up – for example do a test for combined addition and subtraction once they are done, then once another operation is done and tested by itself, test it with the other already implemented operations and so on. (5%)

Report – combined points from above + 5%

1. The 5% are for the inclusion of the table of assigned tasks and the cover page information along with the github repo link as a hyperlink

There will be a -5% penalty for programs that do not use descriptive prompts to the user. Assume the TA knows nothing about your program and walk them through what to exactly provide as input at each step. Whenever you output something to the user such as the equation, make sure to provide a user friendly statement describing the output to come

Note: a lot of these concepts are new to you and we understand the stress and load of combining all these concepts together so do not worry we will be lenient. The purpose of this assignment is to familiarize you with these concepts and tools so that you can add them to your arsenal as a developer. Do not worry and over think about written parts of the report and write an essay, we only want short description to show that you followed the guidelines and understand the process of developing a software in a team set up.