CSE 1321L: Programming and Problem Solving I Lab

Assignment 1 Spring 2026

Module 1

What students will learn

- o Problem Solving.
- o Terminology.
- o Basic Program Structure.
- o Input and Output with the user.
- o Basic calculations and calculations that require an intermediate solution.

Content

- o Overview
- o Assignment1A: Event Check-In
- o Assignment1B: Highway Stopping Distance Calculator
- o Assignment1C: Minutes to Days, Hours and Minutes

Overview:

For most of you, this will be the first time you have done any programming, which is exciting! The write-up of this first assignment will be a little longer than the others because we want you to understand how things are going to roll out the rest of the semester. Advice: Start early (certainly not the day the assignment is due), practice, and ask a lot of questions.

Unless calculations are trivial, you will almost always want to use an intermediate variable – where you store part of the solution. For example, you might remember the equation for gravity as:

$$F = G\frac{m_1 \times m_2}{R^2}$$

How would you write this as code? You might solve it in parts instead of one shot because it makes it easier to check. It would look like:

```
Temp1 = m1 * m2
Temp2 = r * r
Temp3 = temp1 / temp2
F1 = g * temp3
```

This is assuming the variables ***m1***, ***m2***, and ***g*** declared previously in the code.

Final note: _Do not cheat_
If your temptation is to look online, don't. Come see us instead and ask questions – we are here to help. Remember, you are going to have to write codes in your future job interviews, so learn it now to secure a high-paying job later.

Assignment1A: Event Check-In

You've been asked to create a simple check-in message generator for a community event. When a participant arrives, the program should display a personalized confirmation message using the information they provide.

For this assignment:

- Prompt the user to enter their **full name**
- Prompt the user to enter the **event name**
- Prompt the user to enter the **number of guests** they are bringing, store it as an **integer.**
- Use **string concatenation or f-strings** to display a confirmation message in the format shown below

Welcome, **[name]!**
You are checked in for **[event name]** with **[number]** guest(s).

Hint:

o You are allowed to use F-strings

Below are two example runs. The user input is shown in **red and bold**.

Sample Output #1:

```
[Event Check-In System]
Enter your full name: Alex Johnson
Enter the event name: Owl Tech Meetup
Enter number of guests: 2

Welcome, Alex Johnson!
You are checked in for Owl Tech Meetup with 2 guest(s).
```

Assignment1B: Highway Stopping Distance Calculator

You will write a Python program that estimates a car's **total stopping distance** on a dry road. The total stopping distance is calculated as the sum of the **reaction distance** and the **braking distance**.

The formula for total stopping distance (in **feet**) is:

$$D = (1.47 \times v \times t) + \frac{v^2}{30 \times f}$$

Where:

- D= total stopping distance (feet)
- v= speed (**miles per hour**, mph)
- t= reaction time (**seconds**)
- f= friction factor (unitless)

The user will **NOT** enter the friction factor f directly. Instead, the user enters the **road friction coefficient** μ.
Convert μ to the friction factor using:

$$f = 0.85 \times \mu$$

This computed value of f must then be substituted into the **original stopping distance equation**.

<u>For this assignment:</u>

- o Define following variables in your code
  - ▪REACTION_CONST = 1.47
  - ▪BRAKE_CONST = 30
  - ▪FRICTION_MULTIPLIER = 0.85
- o In your equations you will use these variables instead of putting hard coded numbers.
- o Prompt the user to enter:
  - ▪Speed (in mph)
  - ▪Reaction time (in seconds)
  - ▪Road friction coefficient μ
- o Read all values as floats
- o Compute the friction factor f by using the equations provided above.
- o Use the full stopping distance equation and output the stopping distance rounded to **2 decimal places**

Example runs are shown below. The user input is shown in **red and bold**.

Sample Output #1:

```
[Highway Stopping Distance Calculator]
Enter the speed (in mph): 60
Enter the reaction time (in seconds): 1.5
Enter the road friction coefficient (mu): 0.70
The estimated stopping distance is 333.98 feet.
```

Sample Output #2:

```
[Highway Stopping Distance Calculator]
Enter the speed (in mph): 80
Enter the reaction time (in seconds): 1.5
Enter the road friction coefficient (mu): 0.7
The estimated stopping distance is 534.94 feet.
```

Assignment1C: Minutes to Days, Hours and Minutes

For this assignment, you will create a Python program that converts a total number of minutes into **days**, **hours**, and **remaining minutes**.

These are the conversions you should use in your solution:

- o 1 day = 24 hours
- o 1 hour = 60 minutes

For this assignment:

- o Prompt the user to enter a **total number of minutes**
- o Read the input as a **float**
- o Convert the total minutes into:
  - ▪ Total days
  - ▪ Remaining hours
  - ▪ Remaining minutes
- o Round the remaining minutes to 2 decimal places.

Notes:

- o Do **not** simply divide and print days, hours, and minutes independently.
- o You are expected to break the total time hierarchically:
  1. Extract full days
  2. Then extract full hours from the remainder
  3. Then compute remaining minutes

Hint:

Following built-in math operations are helpful:

- o int() to truncate decimal values
- o // for floor division
- o % or subtraction to calculate remainders

Example runs are shown below. The user input is shown in **red and bold**.

Sample Output #1:

```
[Time Conversion - Minutes to Days, Hours, and Minutes]
Enter the total minutes: 3500
3500.0 minutes is approximately 2 day(s), 10 hours, and 20.00 minutes.
```

Sample Output 2:

```
[Time Conversion - Minutes to Days, Hours, and Minutes]
Enter the total minutes: 1450
1450.0 minutes is approximately 1 day(s), 0 hours, and 10.00 minutes.
```

Submission Instructions:

o Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
o Programs must be working correctly.
o Programs must be written in Python.
o Programs must be submitted with the correct **.py** format.
o Programs must be saved in files with the correct file name:

- Assignment1A.py
- Assignment1B.py
- Assignment1C.py

o Programs (source code files) must be uploaded to Gradescope by the due date.