CSCI 141
Spring 2021
Assignment 1
Topics: Variables, `print`, `input`, operators, conditionals

Introduction

For this first assignment, you will write two small Python programs. The following general guidelines and tips apply to both programs.

Getting Started

Refer to lab 1, as well as the lecture slides, to review what you've learned so far. In this and future assignments, you may not have seen all the topics in lecture before the assignment is released, but they will be covered well before the deadline. As usual, seek help early if you get stuck: come talk to me or the TAs during office hours, or visit the CS tutors for help. Please keep track of how much time you spend on both portions of this assignment. You will be asked to report your hours on Canvas after you submit. This helps us calibrate the workload for the rest of the quarter, and for future quarters. It's also useful information about your own study patterns!

Collaboration and Academic Honesty

The programs you write solution MUST be authored solely by you. You can discuss the problems with your peers, but these discussions should happen without Thonny (or your IDE of choice) open, and you should take a break before returning to write code to help ensure that you truly understand the answers. You may not copy another person's code, or have another person tell you what code to type. If you have any questions, or are unsure about whether a specific sort of collaboration violates academic honesty, please come talk to me.

Coding Style

All your programs must have a comment at the top stating the author, date, and a short description of your program. You should also include comments elsewhere in your code anytime you think it's helpful to outline is happening. Be sure to include a comment when your code does something non-obvious and a reader of your code would benefit from some explanation.

Your code should be written as clearly as possible (i.e., try to avoid the need for explanatory comments). Variable names should follow the guidelines discussed in lecture for sensible naming: neither too verbose nor too terse.

Valid Input and Error Checking

You may assume that the user is well-behaved and enters data as prompted. Your program is *not* required to check the user's input to make sure it's well-formed. Your program is allowed to throw an error if the input is bad (i.e., the user enters a string instead of a number).

Testing

Testing is a major component in the process of writing software. Often, testing (detecting errors) and debugging (locating and fixing errors) takes way more effort than writing the code did in the first place. We'll talk more about testing as the quarter progresses; in the meantime, for each problem below we provide a table with some helpful test cases that you can use to see if your program is working correctly. Try your code out with the given inputs and make sure your output matches the program output specified in the table.

1 Mortgage Calculator

Many online real estate websites have mortgage calculator features[1]. These calculators ask for some information, such as the price of a home, the down payment (amount of the home price you'd pay up front), and the interest rate, then calculate the amount you'd have to pay monthly on a loan for the home.

According to NerdWallet[2], the formula used to calculate the monthly payment based on these inputs is as follows:

$$M = (P - D)\frac{r(1+r)^N}{(1+r)^N - 1}$$

Where:

$M =$ The monthly payment
$P =$ The price of the home
$D =$ The down payment amount
$N =$ The number of months over which the loan will be paid off
$r = R * .01/12$, the monthly interest rate (the yearly percentage converted to a decimal and divided by 12)

Write a program called `mortgage.py` that prompts the user to enter (one at a time, and in exactly the given order order) values for P, D, N, and R, and outputs the monthly payment amount M. Notice that you are asked to prompt the user for R, the annual interest rate as a percentage (e.g., 3.7), but the formula uses r, the monthly interest rate (e.g., 0.037).

A sample invocation of the program is shown in Figure 1.

Test Cases

For brevity, the output is truncated after 3 decimal places in the table below; your program will output more decimal places (as in the example invocation).

[1] See https://www.zillow.com/mortgage-calculator/ for an example

[2] Go to https://www.nerdwallet.com/mortgages/mortgage-calculator/calculate-mortgage-payment and click "How to calculate your mortgage payment" for the source of the formula

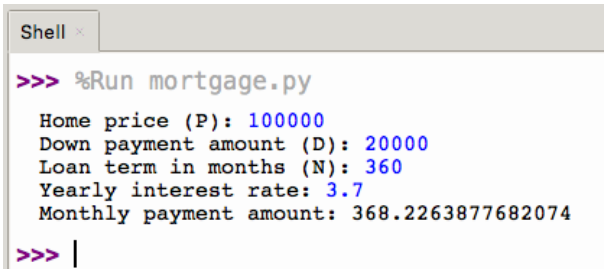

Figure 1: Sample Output

P	D	N	R	Output (M)
100000	20000	360	3.7	368.226
1000000	10	180	3.4	7099.747
549050	103200	800	5.1	1960.773

2 Modulo Helper

Congratulations! You've just been hired as a Python programmer at an education start-up company. Your first task is to develop a prototype of a program that CS students will use to check their understanding of modulo. The program first prompts the user to enter two positive integers (*number 1* and *number2*). The program then asks the user to solve *number 1* mod *number 2* and *number 2* mod *number 1*, checkinging the user's answer for each. In total the program prints 11 lines, two of which are blank, each time the program runs:

1. Ask the user to enter *number 1*
2. Ask the user to enter *number 2*
3. Ask them what is *number 1* modulo *number 2*
4. Tell them if they got it right or wrong
5. Tell the detailed correct answer
6. (blank line)
7. Ask them what is *number 2* modulo *number 1*
8. Tell them if they got it right or wrong
9. Tell the detailed correct answer
10. (blank line)
11. Output the user's score out of 2

All numerical outputs must be integers (whole numbers, without decimals). A sample invocation of the program is shown in Figure 2:

Although this is a simple set of steps, there are many, many different Python programs that can achieve it. The text of your prompts does not need to match the example exactly. However, your solution **must** follow the instructions above exactly as specified.

```
>>> %Run modhelper.py
  Enter the first number: 2
  Enter the second number: 20
  What is 2 mod 20? 2
  Great job!
  2 divided by 20 is 0 remainder 2

  What is 20 mod 2? 5
  Oops!
  20 divided by 2 is 10 remainder 0

  Your score is: 1
>>>
```

Figure 2: Sample Output

Test Cases

First Integer	Second Integer	number 1, number 2	number 2, number 1
7	5	1 remainder 2	0 remainder 5
3	3	1 remainder 0	1 remainder 0
1	678	0 remainder 1	678 remainder 0
8364724	9738	858 remainder 9520	0 remainder 9738

Submission

Double check that your programs work according to the specification and produce the output given in the test cases. Take a look through the rubric below and make sure you won't lose points for reasons that could easily be foreseen and fixed. When you're finished, submit each of your programs as a `.py` file named `modhelper.py` and `mortgage.py`, respectively. Finally, fill out the **A1 Survey** on Canvas.

Rubric

`mortgage.py` **(15 points)**

Author, date, and program description given in a comment at the top of the file	1 point
Code is commented adequately and variables are appropriately named	1 points
Prompts for the correct values	4 points
Prompts for the values in the correct order	4 points
Produces the correct output	5 points

`mod_helper.py` **(20 points)**

Author, date, and program description given in a comment at the top of the file	1 point
Code is commented adequately and variables are appropriately named	1 points
Code gives appropriate feedback to getting an answer right or wrong	1 point
Prompts for number 1 and number 2	2 points
Number 1 modulo number 2 prints on the right line	3 points
Detailed answer to number 1 modulo number 2 prints on the right line	3 points
Number 2 modulo number 1 prints on the right line	3 points
Detailed answer to number 2 modulo number 1 prints on the right line	3 points
Correct score prints on the right line	3 points

3 Optional: Challenge Problem

Some assignments will come with an optional challenge problem. In general, these problems will be worth very small amounts of extra credit: this one is worth up to two points. Though the grade payoff is small, you may find them interesting to work on and test your skills in Python and algorithm development. The skills and knowledge needed to solve these problems are not intended to go beyond those needed for the base assignment, but less guidance is provided and more decisions are left up to you. The A1 challenge problem is as follows:

Write a program that takes command line arguments representing the hours and minutes of two times of day, and prints the number of minutes between them. Your program needs to work for time ranges that cross hour boundaries (e.g., 7:59 to 8:01) and day boundaries (e.g., 6:00AM to 5:00AM). You'll earn 1 point if your program takes the times of day in 24-hour format. For 2 points, the program should handle 12-hour times with AM or PM attached.

Because I haven't specified the details of how the user specifies the inputs, you should be sure to document this clearly: In addition to the Author, Date, and Description comments at the top of your program, include a "Usage" comment that explains how the program should be called, including at least one example run and the corresponding output.

Upload your challenge program to Canvas in a file called `minutes.py`.