Skill Building 2

Each unit will feature a series of questions designed to give you additional practice on new programming constructs and techniques beyond the ones discussed in class. Working through the exercises will help reinforce the learned material and can help diagnose areas that you may be struggling with. When you find yourself having difficulty with an exercise, this is a great time to reach out to your instructor or other authorized resource to fill in any gaps or misunderstandings in your knowledge.

Likely, all the information needed to complete the exercises won't have been discussed in class by the time this assignment is assigned. The activities follow the flow of the unit, so the first exercises will occur earlier in the class discussion. Work through the exercises already covered in class and complete the later ones after the second day's lecture.

All exercises should be an individual effort. You will have a minimum of 4 days to complete the activities, though it is highly recommended that you start them after the first lecture in a unit and finish them after the second one. To receive credit, push your solution to GitHub before the deadline.

Activities

Setup:
Create a new directory called "`skills`" in your Unit02 GitHub repository. In the skills directory, create a file named "`skills.py`". All of the work should be completed in that file. Please commit and push after each activity to receive full credit.

Each exercise should begin with a triple quote comment that lists the exercise number.
Example:
```
""" Exercise 1 """
```

Exercise 1:
Write a block of code to calculate the final score in a game given a starting score and new points earned and lost.

1. Create a variable named `points` that has a value of `50`.
2. Create a variable named `points_gained` that has a value of `10`.
3. Create a variable named `points_lost` that has a value of `5`.
4. Create a variable named `final_score` that is calculated using the following formula:

   final_score = points + points_gained - points_lost

5. Print the value stored in `final_score`.

Exercise 2:
Write a block of code that calculates the amount of ingredients needed to adjust a recipe for a specific number of guests.

1. Create a variable named `recipe_serves` that has a value of `4`.
2. Create a variable named `num_eggs` that has a value of `2`.
3. Create a variable named `cheese_ounces` that has a value of `4`.
4. Create a variable named `num_guests` that has a value of `6`.
5. The given set of ingredients prepares enough food for the number of people indicated by `recipe_serves` (`4`). Calculate the number of eggs needed to complete the recipe for `num_guests` (`6`) guests and store it in an appropriately named variable.
6. Calculate the number of ounces of cheese needed to complete the recipe for `num_guests` guests and store it in an appropriately named variable.
7. Print the required amount of eggs and cheese on separate lines.

Exercise 3:
Write a block of code that includes a budget and daily cost and computes the amount left after a specified number of days.

1. Create variables for `total_budget` and `daily_cost`. You may choose the values assigned to them.
2. Create a variable named `trip_length` with a value between `3` and `12`.
3. Calculate the amount of money left over in the budget after a trip of `trip_length` days (based on the `daily_cost`) and store the value in an appropriately named variable.
4. Print the calculated remaining value at the end of the trip (using the variable created in the previous step).

Exercise 4:
Write a function that takes the cost of food and the tip rate (e.g., 0.15 for 15%). The function should calculate and return the total cost with the tip.

1. Create a new function named `calculate_price` that has two parameters named `cost` and `tip` (as a percentage).
2. Calculate the total cost of the meal with tip and store it in a variable named `total`.
3. Return the `total` value.
4. Call the function (make sure the function call is left justified) and print the returned value.

Exercise 5:
Write a function that calculates the amount of change received from a purchase.

1. Create a new function named `calculate_change`.
2. The purchase price and amount paid should be provided as parameters to the function.
3. Calculate and return the amount of change that should be given to the payer.

4. Call the function (make sure the function call is left justified) and print the returned value.

Exercise 6:
Write a function that takes the current hour and minute and calculates how many minutes are left until 24:00 (midnight).

1. Create a new function named `minutes_till_midnight` that accepts two arguments: `hour` and `minute`. Note: Hours will use a 24-hour clock, e.g., 3 PM is hour 15.
2. Calculate the number of minutes until midnight and store it in a variable (there are 24 * 60 minutes in a day).
3. Return the calculated value.
4. Call the function (make sure the function call is left justified) and print the number of minutes until midnight.

Exercise 7:
Write a block of code that calculates the cost of an item, taking into account its price, sales tax, and a discount.

1. Implement a function named `apply_discount` that has parameters for `price` and `discount` and returns the total cost of the item after applying the discount. Note: Discounts will be a percentage (i.e., `0.10` for a 10% discount).
2. Implement a function named `apply_tax` that has parameters for `price` and `sales_tax` and returns the total cost of the item after adding in the tax. Note: Tax will be a percentage (i.e., `0.08` for 8% sales tax).
3. Write a function named `final_price` that accepts arguments for `price`, `discount`, and `sales_tax` and returns the final cost of the item. Use the `apply_discount` and `apply_tax` functions created in the previous steps to help calculate the final cost. Note: Discounts are applied *before* tax.
4. Call the `final_price` function with appropriate argument values (make sure the call is left justified) and print the returned value. Example call: `final_price(85, 0.15, 0.08)`

Exercise 8:
Using ChatGPT or similar tools to generate *solutions* to your homework exercises is a violation of the academic integrity policy in this class. However, there are lots of productive ways to use generative AI to help you learn the material in this course. Here is just one example.

1. Download this document as a PDF file.
2. Open a ChatGPT chat and drag the PDF into the chat window.
3. Type a prompt to ask ChatGPT to use the document to generate 5 additional practice problems for you to use to practice.
4. Take a screenshot of your prompt and ChatGPT's response. Save it to your repository.
5. Choose 2 of the 5 problems and implement a solution in your `skills.py` file.

Submission Instructions & Grading

The in-class activities and assignments in Software Development & Problem Solving have been designed to allow you to practice what you have learned in class and to identify potential gaps in your understanding so that you can seek help from your instructor or the Teaching Assistants.

Future activities and assignments build on the learning outcomes from previous assignments. Each one helps you understand key topics that are used in upcoming assignments. It's important to stay on track so that the assignments are not more challenging than they are intended to be.

Submit your solution by pushing it to GitHub before the assignment deadline (see the course schedule). See the course syllabus for the rubric that is used for grading homework assignments.