MTH 3300
Spring 2024

Programming Problem 1: solar.py

Directions: Download the template files I have provided on Blackboard. Then open Spyder, load these template files, and write the following programs. Submit your source code via Gradescope in `.py` format. READ THE INSTRUCTIONS on submitting your work in the Course Documents section of Blackboard.

Specify collaborators/resources or ***explicitly specify*** *that* ***none*** *were used* in the comments at the top of your `.py` file. You need not list me or our class notes as collaborators/resources. **Failure to include this will result in a zero on the assignment.**

Be sure to read the SPECIFICATIONS carefully! And write comments!

Assignment goals: Get practice with various numerical operations in Python, getting user input, and appropriately formatting output.

You are working for a solar energy company. You will write a program that asks the user for information about their house — namely, the length and width of the house (both in feet) — and then compute the amount of energy that can be generated with rooftop solar for their house.

Here is a sample interaction of the way that your program will work (the numbers 30 and 40 are entered by the user; the program should produce the rest):

```
Enter width (in ft): 30
Enter length (in ft): 40
You have 1200.0 square feet of roof area and 450.0 square feet usable for panels.
You can fit 21 solar panels on your roof.
You can install a system rated for 8400 watts.
You can generate about 8215.2 kilowatt-hours in a year.
That's equivalent to about $1971.65 of electricity.
```

To perform the calculations shown in this sample, we will make some highly simplifying assumptions about the energy production capacity of a roof and solar panels — but they are not too far off (see `https://solarpowerrocks.com/square-feet-solar-roof/`).

Here are the assumptions used to calculate the quantities found in the sample output above:

- To calculate the usable area for panels, we will assume exactly **half** of the area of the roof of a house faces south (towards the sun); only the south-facing roof is suitable for solar panels. Of the south-facing roof, only **75%** can be used for panels because there must be a setback (to allow walking space for firefighters if they need to access the roof).
- A single solar panel takes up **21.2 ft^2** of the roof. You cannot cut solar panels in half. So the number of solar panels that you can fit on a roof that has, for example, 100 ft^2 of usable area is 100/21.2 = 4.71... rounded down, in other words, 4 panels (to round down, you can use the `int()` function). We'll assume that the shape of the roof is such that it's possible to tile the usable part entirely with solar panels without any wasted space.
- A single solar panel is rated for **400 watts**. In New York, you can expect to generate approximately **0.978 kilowatt-hours per year for each watt** of your system, and electricity costs about **$0.24 per kilowatt-hour**. The dollar amount should be printed with exactly two digits after the decimal place. (So it's "$1.00" and not "$1.0", for example.)

Fill in `solar.py` with a program that performs as specified here.

Hint: make sure you try calculating my sample values by hand before programming to ensure you understand the task!

The output should be *exactly* as specified in the sample above. When I say exact, I mean identical, down to the spacing. In this class, I have written autograders for most assignments, which run on your code as soon as you submit it and let you know if you pass the tests I've written. Those tests check if your output matches what I've specified down to the spaces, so you must follow my specifications exactly. (This may seem annoying, but it is a valuable skill to master. It is very common in computer science to have precise instructions for something to work or for a client to be happy.) The benefit of autograders is that you can immediately know if your code isn't doing exactly the right thing and fix it before the deadline!

Specifications: your program must

- ask the user to enter the width and length of their house in feet (as formatted in the sample run above)
- calculate and output the following quantities, as formatted in the sample run above:
  - the total area (in square feet) of the roof and the area (in square feet) usable for solar panels
  - the number of panels that can fit on the roof (on the usable area)
  - the number of watts of the system
  - the number of kilowatt-hours the system can generate in a year
  - the cost in a year of the electricity that is generated by the system

Finally, any program with more than a few lines of code (so nearly all of them) should have comments describing what blocks of the code do. We'll learn more about how to organize code better in the future, but for now, you should put in a couple of comments describing what blocks of code are doing so that someone reading your program can easily see what is going on. You should also ensure any variables you use are named well, and your header describes what the program does. Finally, be mindful of the spacing in your program: how can you use an extra line here or there to make it easier to read, but not too many? There are several correct answers to these style questions, so go with what you think makes it easiest to read and stay consistent. I will give feedback on your style to help you improve it.

Note: This problem has two visible test cases on Gradescope. You will see if your code passes these two test cases. In addition, it has two invisible test cases, which you will not be able to see until after grades are released (you will not see if your code passed or did not pass these). So, run additional tests to cover other possible inputs! (You must work out the correct output in your test cases. You may get inaccurate answers in some cases due to "float weirdness" – don't worry about trying to fix that.)

Again, your input prompts must match the ones given above *exactly* (including spaces) to pass the test cases on Gradescope.

You can submit your code as many times as you wish before the deadline to fix any issues. If you have questions about what the autograder tells you is wrong, please ask!