Northeastern University DS 2000 — Programming with Data, Fall 2022, Profs. Muzny and Rachlin
Fall 2022
Homework 3
Released: September 23, 2022
Due: September 30, 2022 @ 9pm Boston time

Homework 3: While Loops

Introduction
The main skills that we're focusing on this week are creating code that has branches (with conditionals) and that can repeat (with loops). This is important to making programs that can do things like filter data or that can do a computation for every row in a data file, even if we don't know how many rows there are!

Expectations
Write a separate program to answer each question. Be sure to document your code with a descriptive header and concise comments. It isn't necessary to comment every line, particularly those lines whose purpose is obvious. Instead, use comments to outline the main steps of your program. Use variable names that are descriptive, concise, and not excessively verbose. Practice good style, for instance: x = 5, not x=5 and separate sections of your program by blank lines.

See the style guide for more information: https://course.ccs.neu.edu/ds2000/ds2000_fall2022_styleguide.pdf

For your homework submission, submit these files:

- `change.py`
- `simulation.py`
- `walk.py + walk.png (image output)`
    - If you do not submit walk.png, you **will receive a very significant penalty** on this problem.
- `walk2d.py + walk2d.png (image output)`
    - If you do not submit walk2d.png, you **will receive a very significant penalty** on this problem.

In summary, you'll be submitting **SIX** files.
- Make sure to watch the demonstration video, which we will post on Piazza!

Each .py file should have one (and only one!) `main()`!

You are welcome to upload your files multiple times, but only your last submission will be graded. Verify that all files have been submitted *at the same time*.

See the grading rubric for more information:
https://course.ccs.neu.edu/ds2000/ds2000_fall2022_gradingpolicy.pdf

Question 1 (`change.py`)

Write a program that takes in the price of one US dollar or less for an item from the user then calculates the appropriate change to give back to the user, assuming that they always pay $1. You may assume that the user always enters an integer, however, if they enter a number that is under 1 or over 100, your program should display a message telling the user that they have entered an incorrect purchase price and exit.

In summary:

- Ask the user to enter a number 1 - 100
- If the user does, calculate the change to give them:
    - Display the change using the fewest number of pennies, nickels, dimes, and quarters. Do not display coin denominations that are not needed.
- If the user does not enter a number 1 - 100, inform them that they have entered a bad input and do not execute any more code.

Example outputs (user input in **bold underlined**)

```
Enter price in cents (1-100): 27
Change to be given: 73
Quarters:  2
Dimes   :  2
Pennies :  3
```

```
Enter price in cents (1-100): -17
I'm sorry, you need to enter a number between 1 and 100!
```

Hints:

- Instead of thinking about this question all at once, it might be helpful to start by taking the sub-question "how many quarters does the user need" and writing a program that only calculates the appropriate number of quarters before adding in dimes, nickels, and pennies.
- We recommend that you either start with ***just*** the input verification OR ***just*** calculating the change, then add in the second part later.

Question 2 (`simulation.py`)

For this program, you'll write a simulation that repeatedly rolls three 6-sided dice. You'll be calculating how often the three dice all have different values. If you've taken a discrete math or probability course (by no means required!), you may have learned that the theoretical probability of obtaining three different values is $\frac{6 \times 5 \times 4}{6^3} \cong 0.5556$. This program will calculate the *experimental* probability of getting three different values for rolling the three dice 10 times, 100 times, 1000 times, and 10000 times.

Consider the case when you roll the dice 10 times. Each time my while loop iterates, you want to roll all three dice once and answer the question "are these all different?".

Iteration number	dice	all different values?
1	⚀ ⚂ ⚀	no
2	⚄ ⚃ ⚀	yes
3	⚁ ⚅ ⚅	no
…	…	…
10	⚅ ⚀ ⚄	yes

Example output (remember that since rolling dice is random, your numbers might be slightly different):

```
Rolling dice 10 times.
Number of mis-matched rolls: 5
Experimental probability: 0.5

Rolling dice 100 times.
Number of mis-matched rolls: 45
Experimental probability: 0.45

Rolling dice 1000 times.
Number of mis-matched rolls: 555
Experimental probability: 0.555

Rolling dice 10000 times.
Number of mis-matched rolls: 5535
Experimental probability: 0.5535
```

For this question you will use the **`random`** library.

To import the package:

```
# import the package and give it the nickname "rnd"
import random as rnd
```

A few useful functions:

- `rnd.randrange(`*`min, max`*`)`- Generate and return a random integer from min to max-1
- `rnd.randint(`*`min, max`*`)`- Generate and return a random integer from min to max

- `rnd.random()` - Generate and return a random float from 0.00 to 1.00

Hints:

- We recommend starting by writing a program that *just* does the simulation for 10 rolls.
- *After* you've done 10 rolls, *then* think about making the program do 100, 1000, etc. You'll need to use a nested while loop to achieve this. The outer loop will be in charge of indicating how many rolls are to be done (10 to 100 to 1000 to 10000). The outer loop would therefore iterate 4 total times. The inner loop will perform a single roll of the set of three dice and will itself iterate 10, 100, 100, 1000, and finally 10000, times.
- Think about what the value of 10 ** 1, 10 ** 2, etc is.

(Question 3 on next page)

Question 3 (`walk.py`)

For this question you will again use the **random** library. A squirrel starts at position 0. Each second, the squirrel moves either left (pos = pos – 1) or right (pos = pos + 1) with equal probability. Plot the position of the squirrel as a function of time over 1000 seconds. (Time is the x-axis, and the squirrel's position is plotted along the y-axis.) Be sure to label your axis and include an appropriate title.

Here are a few plotting functions that you may find helpful:

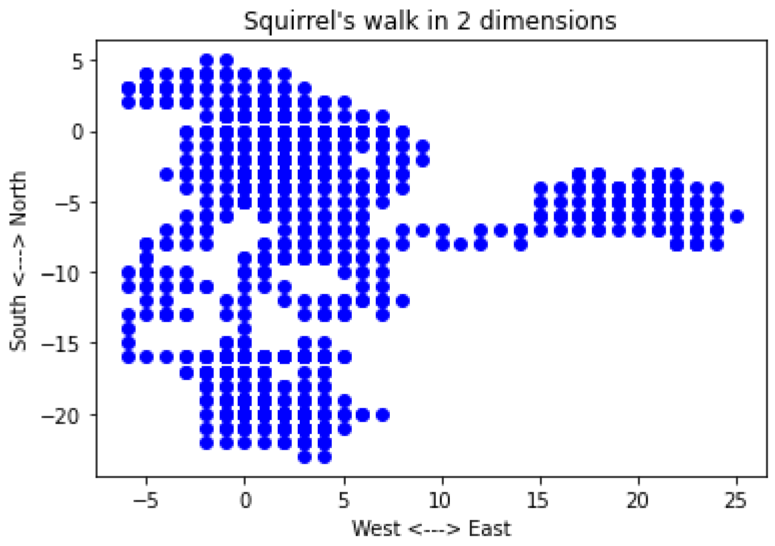
- `import matplotlib.pyplot as plt` - not a plotting function, but you'll need to import the matplotlib library to use its plotting functions!
- `plt.plot(x coordinate, y coordinate, marker type, color="color code")` - add a single point to your graph, optionally pass in a color code.
  - Marker type documentation: https://matplotlib.org/stable/api/markers_api.html
  - Color code documentation: https://matplotlib.org/stable/gallery/color/named_colors.html#css-colors
- `plt.xlabel(string for the label of the x-axis)` - set the x-axis label for your graph
- `plt.ylabel(string for the label of the y-axis)` - set the y -axis label for your graph
- `plt.title(string for the title of the graph)` - set the title for your graph
- `plt.savefig(filename to save to, bbox_inches="tight")` - save your graph as a file on your computer. bbox_inches = "tight" makes it so that there isn't a lot of extra whitespace included around your graph.
- `plt.show()` - a function that tells python "I'm done making the current graph" and displays it.

Make sure to save your output as `walk.png` and include this image with your homework submission!

Question 4 (`walk2d.py`)

For this question, you will create an updated version of your previous program. Now with each second, the squirrel is walking in two dimensions. Starting at the origin (at position (x=0, y=0)), the squirrel now moves one step randomly in any of four directions, NORTH, SOUTH, EAST, or WEST. (They cannot move diagonally.) However, to make it a little more interesting, the squirrel is *twice as likely to move in the EAST-WEST direction* rather than the NORTH-SOUTH direction.

Use the **color = "color-code"** parameter to pick a single color for all of your points. Example output updated 9/26/2020.



Hints:

- Start by copy+pasting your solution from the previous problem.
- Use the random library's `random.randint()`, `random.randrange()`, or `random.random()` functions to decide which direction the squirrel is walking in.

Make sure to save your output as `walk2d.png` and include this image with your homework submission!