Developers Institute

Python Course

Week 2

Day 1

Exercises

Exercise 1 – Temperature Advice

1. Create a function called **get_random_temp()**.
    1. This function should return an integer between -10 and 40 degrees (Celsius), selected at random.
    2. Test your function to make sure it generates expected results.
2. Create a function called **main()**.
    1. Inside this function, call **get_random_temp()** to get a temperature, and store its value in a variable.
    2. Inform the user of the temperature, together with a friendly message, eg. “The temperature right now is 32 degrees Celsius.”
3. Add more functionality to the main() function, writing some friendly advice relating to the temperature, if it is:
    1. below zero (eg. “Brrr, that’s freezing! Wear some extra layers today”)
    2. between zero and 16 (eg. “Quite chilly! Don’t forget your coat”)
    3. between 16 and 23
    4. between 24 and 32
    5. between 32 and 40
4. Change the **get_random_temp()** function:
    1. Add a parameter to the function, named ‘season’.
    2. Inside the function, instead of simply generating a random number between -10 and 40, set lower and upper limits based on the season, eg. if season is ‘winter’, temperatures should only fall between -10 and 16.
    3. Now that we’ve changed **get_random_temp()**, let’s change the **main()** function:
        1. Before calling **get_random_temp()**, we will need to decide on a season, so that we can call the function correctly. Ask the user to type in a season - ‘summer’, ‘autumn’ (you can use ‘fall’ if you prefer), ‘winter’, or ‘spring’. Make sure to display a meaningful prompt.
        2. Use the season as an argument when calling **get_random_temp()**.
5. (Bonus: give the temperature as a floating-point number instead of an integer)
6. (Bonus: Instead of asking for the season, ask the user for the number of the month (1 = January, 12 = December). Determine the season according to the month (this page may help you).)

Exercise 2 – Double Dice

1. Create a function that will simulate the rolling of a die. Call it **throw_dice**. It should return an integer between 1 and 6.
2. Create a function called **throw_until_doubles**.
    1. It should **keep throwing** 2 dice (using your **throw_dice** function) until they both land on the same number, ie. until we reach **doubles**. For example: (1, 2), (3, 1), (5,5) → then stop throwing, because doubles were reached.
    2. This function should **return** the number of times it threw the dice in total. In the example above, it should return 3.
3. Create a **main** function. It should throw doubles 100 times (ie. call your **throw_until_doubles** function 100 times), and store the results of those function calls (in other words, how many throws it took until doubles were thrown, each time) in a collection. (What kind of collection? Read below to understand what we will need the data for, and this should help you to decide on what data structure to use).
4. After the 100 doubles are thrown, print out a message telling the user how many throws it took **in total** to reach 100 doubles.
5. Also print out a message telling the user the **average** amount of throws it took to reach doubles. **Round this off** to 2 decimal places.
6. For example:
    1. If the results of the throws were as follows (your code would do 100 doubles, not just 3):
        1. (1, 2), (3, 1), (5, 5)
        2. (3, 3)
        3. (2, 4), (1, 2), (3, 4), (2, 2)
    2. Then my output would show something like this:
        1. Total throws: 8
        2. Average throws to reach doubles: 2.67.