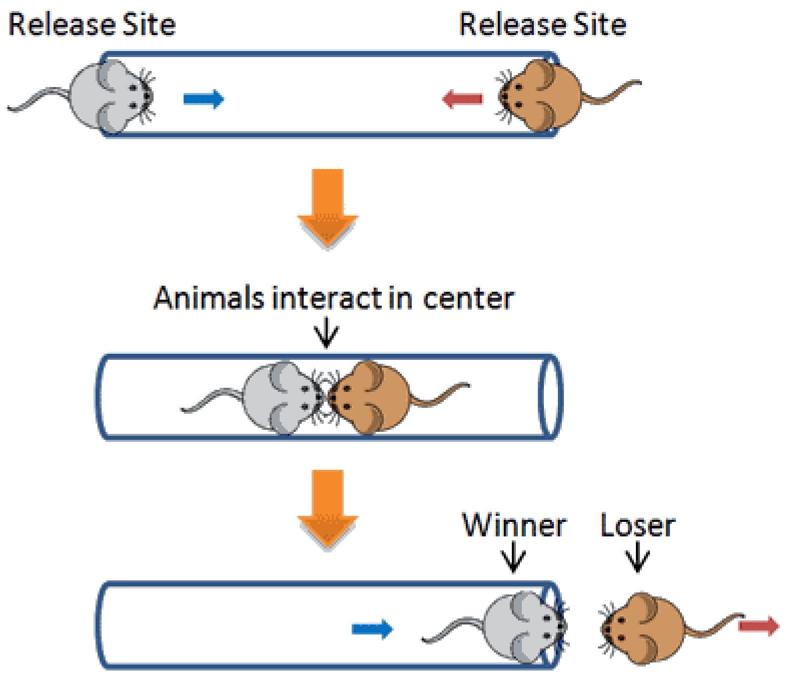
You have taken on a summer job working in a neuroscience lab on a new project that is testing the effectiveness of an up-and-coming psychiatric medication in rescuing social dominance behavior. Your supervisor explains the task as follows:

- There is a cohort of mice that, while on different doses of a new medication being tested, were assessed using the tube dominance test (as shown below). This test put the medicated mice in one side of a tube and a control mouse (unmedicated) on the other side.

- The 'loser' of the tube dominance test is the mouse that emerges from the tube first.



Your supervisor then goes on to explain that they have a datastore (a text file) consisting of data collected from researchers over the past few months. This data is in the form of:

<mouse_id>, <dosage>, <win or loss>

Where <mouse_id> refers to the unique identification label of a particular mouse in the cohort, <dosage> refers to which dosage the mouse was taking at the time of the test (5mg/kg, 10mg/kg, or 15mg/kg), and <win or loss> refers to whether the medicated mouse won or lost the tube dominance test.

Here are some sample rows of data as an example:

mouse_2, 10mg/kg, W
mouse_2, 10mg/kg, W
mouse_3, 10mg/kg, L

The researchers in the lab are now trying to figure out the impacts of each dosage of the medication. They have their textfile full of results in the aforementioned format, and they need you, the resident programmer, to write some Python code that calculates the average win rate of each medication dosage.

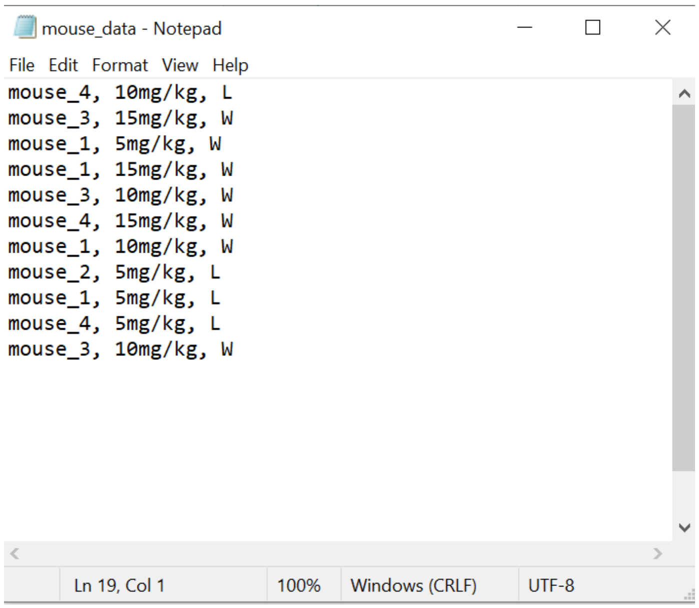
To do this, you should write two functions – generateDictionary, and calculateAverageWinRatios.

The **generateDictionary** function should take one argument, the textfile containing the data, and returns a dictionary in the format of:

```
{<mouse_id> :{'5mg/kg' : [<wins>, <wins + losses>],
              '10mg/kg' : [<wins>, <wins + losses>],
              '15mg/kg': [<wins>, <wins + losses>]},
 <mouse_id> :{'5mg/kg' : [<wins>, <wins + losses>],
              '10mg/kg' : [<wins>, <wins + losses>],
              '15mg/kg': [<wins>, <wins + losses>]},
              ...
}
```

So, as we can see, we should have a dictionary where each key is a unique mouse id. Then, the corresponding value is another list where each key is a dosage. The value for each dosage should be a list with two elements where the first element is the number of wins, and the second element is the number of wins + losses (total fights).

For example, if we called generateDictionary and passed the following textfile as an argument:



```
mouse_4, 10mg/kg, L
mouse_3, 15mg/kg, W
mouse_1, 5mg/kg, W
mouse_1, 15mg/kg, W
mouse_3, 10mg/kg, W
mouse_4, 15mg/kg, W
mouse_1, 10mg/kg, W
mouse_2, 5mg/kg, L
mouse_1, 5mg/kg, L
mouse_4, 5mg/kg, L
mouse_3, 10mg/kg, W
```

Then the returned dictionary should look like:

{'mouse_4': {'10mg/kg': [0, 1], '15mg/kg': [1, 1], '5mg/kg': [0, 1]}, 'mouse_3': {'15mg/kg': [1, 1], '10mg/kg': [2, 2]}, 'mouse_1': {'5mg/kg': [1, 2], '15mg/kg': [1, 1], '10mg/kg': [1, 1]}, 'mouse_2': {'5mg/kg': [0, 1]}}

Note that 'mouse_data.txt' is the text file that should be passed in as an argument to your function in this case (as that is where the data is stored). This textfile can be found on the assignment page and should be stored in the same directory as your Python script. This is the data that you will be working with.

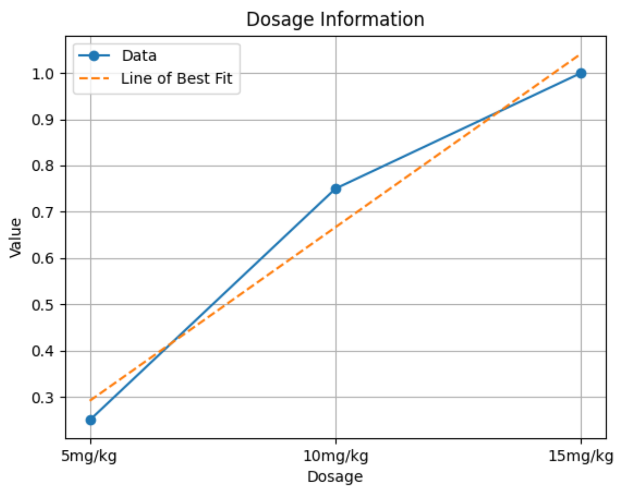
Next, the **calculateAverageWinRatios** function should take a dictionary as an argument (the one you created using generateDictionary) and return a dictionary where the keys are the dosages, and the value is a float representing the win ratio for each dosage. This should be an average for all mice in the cohort. So, if we passed the above dictionary as the argument, then the return should be:

{'10mg/kg': 0.75, '15mg/kg': 1.0, '5mg/kg': 0.25}

Once you have written these two functions, you're nearly done! However, there's one more thing we need to do. Your supervisor tells you that she has some code from a previous employee that might help you plot your findings so a visual can be included in their next academic paper. She shows you the following code:

```python
import matplotlib.pyplot as plt
import numpy as np

def generateDictionary(textfile):

        # TODO: WRITE THIS CODE

def calculateAverageWinRatios(data):

        # TODO: WRITE THIS CODE

def plotDosageInformation(dosage_info):
    # Extract the dosages and their corresponding values
    dosages = list(dosage_info.keys())
    values = [dosage_info[dosage] for dosage in dosages]

    # Sort dosages and values in ascending order of dosages
    dosages, values = zip(*sorted(zip(dosages, values), key=lambda x:
float(x[0].split('mg/kg')[0])))

    # Create an array of x-values for the dosages
    x = np.arange(len(dosages))
    # Create a figure and axis for the plot
    fig, ax = plt.subplots()

    # Plot the values
    plt.plot(x, values, marker='o', linestyle='-', label='Data')

    # Add labels to the x-axis
    ax.set_xticks(x)
    ax.set_xticklabels(dosages)

    # Add a line of best fit
    z = np.polyfit(x, values, 1)
    p = np.poly1d(z)
    plt.plot(x, p(x), linestyle='--', label='Line of Best Fit')

    # Add labels and a legend
    plt.xlabel('Dosage')
    plt.ylabel('Value')
    plt.title('Dosage Information')
    plt.legend()

    # Display the plot
    plt.grid(True)
    plt.show()

data = generateDictionary('mouse_data.txt')
plotDosageInformation(calculateAverageWinRatios(data))
```

If we use the example results discussed previously, we should see a plot just like this:



Your supervisor realizes that you might not have matplotlib or numpy installed on your machine (which need to be imported to successfully create the graph), so she guides you to the website 'Trinket' (https://trinket.io/embed/python3/a5bd54189b) which contains scientific libraries like matplotlib and numpy (so you'll be able to import them). Here, you can plug in your functions to the code above, copy and paste that code into the Trinket workspace, load in the mouse_data textfile provided to you using the middle icon with an arrow to the top right of the Trinket workspace, and then press the 'run' button at the top of the screen. Your graph should appear displaying the visual results of both your hard work and the researchers who collected the data.

Now you're done! Save the figure you've created along with the code that you copy and pasted into Trinket (with your code for the generateDictionary and calculateAverageWinRatios functions), add both the image and the Python script (labeled mouseTests.py) to a folder, zip the folder (so that it has the .zip extension), and submit it under the assignments tab in OWL.

Great work! The neuroscience lab thanks you (and will hopefully put you as an author on the paper).

Non-Functional Requirements

At the top of your code, you should include comments identifying:

- Your name
- Your student number
- Date
- Course name
- Instructor name
- A brief description of your code

Your code should include comments throughout describing chunks of logic.

The code should use informative variable names that follow the conventions discussed in class.

Remember that you should develop your code with Python 3.9 or higher. Failure to do so may result in your code not working when it is tested using Python 3.11 (and thus you will lose marks).

Rules

- Read and follow the instructions carefully.
- You must submit a zip (compressed) file with a folder containing the image of the graph as well as your Python script.
- Submit the assignment on time!
- Late submissions will receive a late penalty of 10% per day up to three days.

- Submissions must be made using OWL. They will not be accepted by email.
- You may re-submit your code to OWL as many times as you like before the deadline.
- Assignments must be done individually and must be your own work. Software will be used to detect academic dishonesty and those found cheating will receive a mark of 0 (and may face additional penalties).

Grading Guidelines

The assignment will be marked as a combination of the output of your script in addition to your code logic, comments, formatting, style, etc. Below is a breakdown of the grading guideline for this assignment.

- [50 marks] Code execution (does it produce the expected output?)
- [10 marks] Provided figure (is it there and is it correct?)
- [20 marks] Code logic and completeness
- [10 marks] Sufficient comments
- [5 marks] Code formatting
- [5 marks] Meaningful variables names and properly formatted variables Total: 100 marks