Learning Outcomes: By the end of this lab, you will:

- Successfully write two types of functions in Python:
  1. Non-fruitful with parameters
  2. Fruitful with parameters.
- Implement professional practices in coding such as good function names, variable naming, type hints, and comments, and program layout.
- Build on previous work:
  - Perform and then display the results of calculations using contents of variables.
  - Work with different Python data types, variables, expressions, and built-in function calls.

Assignment: Write your program in Google Colab. Make sure to use good variable names, and include comments to describe what you are doing.

You will write two different versions of a function that will compute and either print or return the predicted number of new infected animals. Your code will then use the function(s) to explore how levels of vaccination affect the number of new infected cases, and how this varies depending on the infection rate.

Begin by opening a new Colab notebook named `lab2` and then:

1. Write a non-fruitful function named `reportNewInfecteds` having four parameters named `infectionRate`, `susceptible`, `vaccinated` and `infected`. Your function must:
   - Keep track of the number of new infected animals using the following formula:
     ```
     newInfecteds = round((susceptible-vaccinated)*infected*infectionRate)
     ```
   - Keep track of the number of new susceptible animals using the following formula:
     ```
     newSusceptible = round((susceptible-vaccinated-newInfecteds)
     ```
   - Assuming the animals do not lose their immunity, keep track of the number of vaccinated animals using the following formula: newVaccinated = vaccinated.
     ```
     newVaccinated = vaccinated
     ```
   - Print the number of newInfecteds, newSusceptible, and newVaccinated including text describing what was run and what is being printed.
   - Your function should not:
     - Use the `input` function.
     - Overwrite or modify the values given in the four parameters.
   - At the bottom of your program (below all function definitions), include multiple (e.g. 2-3) calls to your function to test your function passing in different values for the arguments to your function.
2. Modify your function from above to make it a fruitful function named `returnNewInfecteds`
3. To return multiple variables you can do this by separating them with a comma, e.g.
   ```
   return variableOne, variableTwo, variableThree
   ```
4. Your function should not:
   - Use the `input` function.
   - Overwrite or modify the values given in the three parameters.
5. **What happens to the number of new infected individuals over multiple time steps?**
   At the bottom of your program store and print three different calls to `returnNewInfecteds` using:

- an infection rate of 0.001,
- a susceptible population of 5000,
- a starting number of infecteds of 10,
- a vaccinated population of 1500.

An example of one such call is shown below.

```
# call fruitful functions -- store then print the value
timeOne = returnNewInfecteds(infectionRate = 0.001,
      susceptible = 6000, vaccinated = 1000, infected = 10)
print("The number of new infected animals is " + str(timeOne[0]) + ".")
```

How many animals need to be vaccinated before the number of cases predicted for next week decreases? Don't worry about getting the exact number, but include at least one call to the function where the number of predicted cases for next week decreases.

6. **How does the number of animals vaccinated required change as the disease becomes more infectious?**: At the bottom of your program, store an additional three different calls to `returnNewInfecteds` this time using:

   - an infection rate of **0.002**,
   - a susceptible population of 5000,
   - a infected population of 10
   - different numbers of vaccinated animals.

   Don't worry about getting the exact number, but roughly how many additional animals need to be vaccinated before the number of cases predicted for the next week decreased?

7. At the bottom of your program (below all function definitions), include a call to your function, using an assignment statement to store the result of calling your function. Make sure to test before moving on. Below is an example of a such a call to a fruitful function:

```
# call fruitful function -- either store or use print
newInfections = returnNewInfecteds()
print("The number of new infected animals is " + str(newInfections))
```

Reflection: At the end of your lab, provide a paragraph response comparing the benefit of fruitful vs. non-fruitful?

In a second paragraph, discuss your findings on vaccination. How many animals needed to be vaccinated before the number of predicted infected animals decreased? Roughly how many more vaccinated animals were required as the disease became more infectious (a higher infectionRate)?

Finally, discuss the limitations of the current epidemiological model. What assumptions does it make about equal mixing that are likely not to be true in the wild? How do current measures for Covid-19, such as masking and isolation change the number of susceptible and infecteds respectively? Check out the article by fivethirtyeight for ideas.

Submitting: When finished, download your program as `lab2.py`. (Do not download as a Python notebook having extension `.ipynb`.) Upload your `lab2.py` file to Google Classroom. Also remember your reflection.

Grading Rubric: Below is what I will be looking for in this assignment:

- Did you thoughtfully explore and reflect on the epidemiological questions in the reflection?
- Did you give your functions clear names?
- Do the non-fruitful functions contain no return statement? Do the fruitful functions return the variables to be used in the next time step?

- Are all of your functions defined at the top of your program, and all calls to the functions occur together at the bottom (below your function definitions)?
- Did you use good variables names that describe what is being stored in the variable, particularly avoiding single-letter or unnecessarily abbreviated names?
- Did you avoid using existing Python function names as your variable names (e.g., no use of `sum`, `int`)?
- Do your variables begin with a lowercase letter, and then consistently use either snake case or camel case for any subsequent words in the variable names?

Next Week: We will revisit the assignment next week to add the following:

- Docstring comments for each function.
- Type hints for each parameter and each function's return type.
- Thorough testing to ensure the program works correctly.