

Read chapters 4-5 in [Fundamentals of Python Programming](#) and complete the following exercises.

Complete computer setup by following the directions here:

<http://cs.appstate.edu/~rmp/cs5245/setup.pdf>

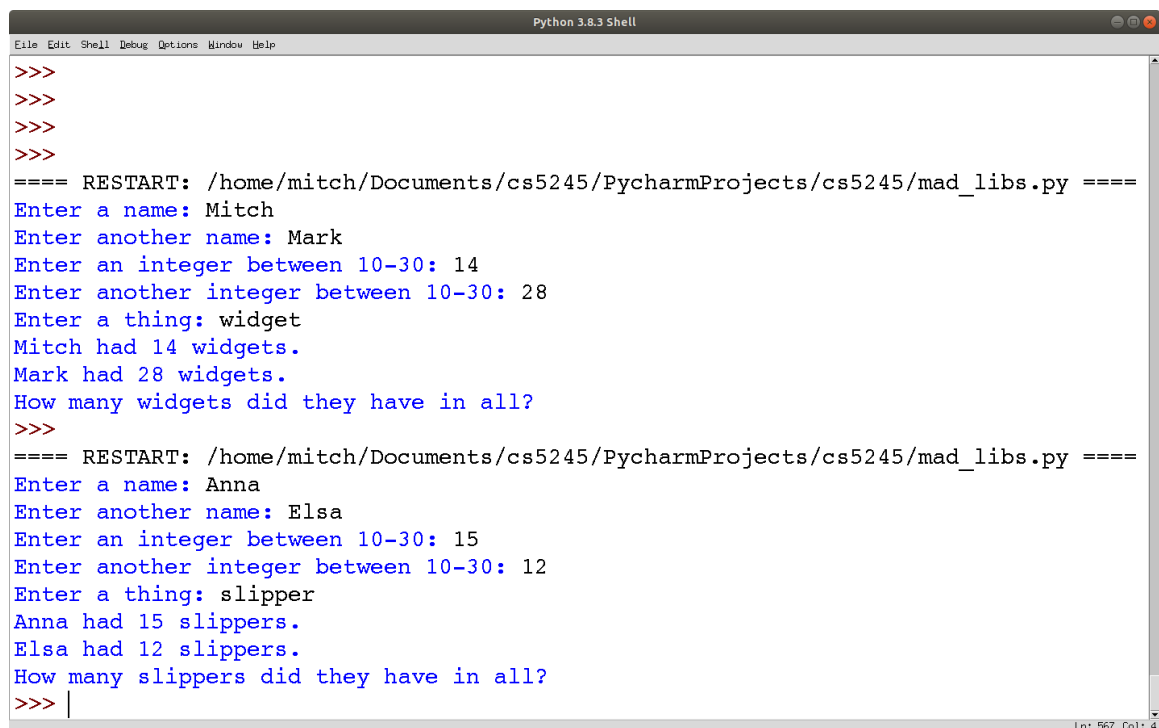
Create a folder called `cs02` to store your files for this lab.

Complete the following activities using the **exact variable names** and save the files using the **exact file names** as specified.

To test your programs, run them from IDLE and check their output.

1 mad_libs

Mad Libs math problem game. In a file called `mad_libs.py`, prompt the user for two names (name a, and name b), two integers (number a and number b), and a thing, and print the following math problem. For the given input it should look like this:



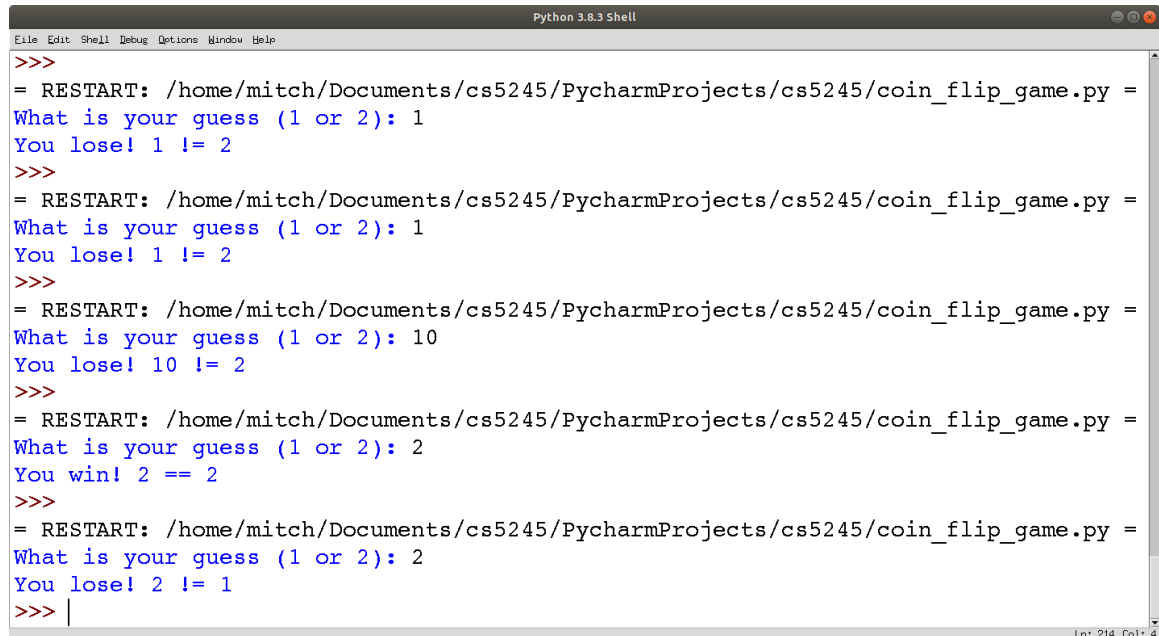
```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>>
==== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/mad_libs.py ====
Enter a name: Mitch
Enter another name: Mark
Enter an integer between 10-30: 14
Enter another integer between 10-30: 28
Enter a thing: widget
Mitch had 14 widgets.
Mark had 28 widgets.
How many widgets did they have in all?
>>>
==== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/mad_libs.py ====
Enter a name: Anna
Enter another name: Elsa
Enter an integer between 10-30: 15
Enter another integer between 10-30: 12
Enter a thing: slipper
Anna had 15 slippers.
Elsa had 12 slippers.
How many slippers did they have in all?
>>> |
```

2 coin_flip_game

Coin flip game. Use the `random.randint` function to select a random number either 1 or 2. Prompt the user to guess a number (1 or 2). And print whether they guessed correctly. You must import the `randint` function at the top of your Python file like this:

```
from random import randint
```

Save your program in a file called `coin_flip_game.py`. For example, running it might look like this:



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/coin_flip_game.py =
What is your guess (1 or 2): 1
You lose! 1 != 2
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/coin_flip_game.py =
What is your guess (1 or 2): 1
You lose! 1 != 2
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/coin_flip_game.py =
What is your guess (1 or 2): 10
You lose! 10 != 2
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/coin_flip_game.py =
What is your guess (1 or 2): 2
You win! 2 == 2
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/coin_flip_game.py =
What is your guess (1 or 2): 2
You lose! 2 != 1
>>> |
```

3 grade_converter

In a file called `grade_converter.py`, write a program that prompts the user for a numerical grade (between 0 and 100) and prints the letter grade (A, B, C, D, or F). If the score is less than zero or greater than 100, the letter grade is 'invalid'. Store the numerical grade in a variable called `score` and store the letter grade in a variable called `grade`. Finally, print the original score and letter grade that matches the format below. For example, when your program works it should run like this (notice the program is run more than once here with multiple RESTART lines):

Range	Grade
> 100	Invalid
90 up to 100	A
80 up to 90	B
70 up to 80	C
60 up to 70	D
0 up to 60	F
< 0	Invalid

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/grade_converter.py
Enter the score: 85
The score is 85.0 and the letter grade is B.
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/grade_converter.py
Enter the score: 70
The score is 70.0 and the letter grade is C.
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/grade_converter.py
Enter the score: 45
The score is 45.0 and the letter grade is F.
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/grade_converter.py
Enter the score: 60
The score is 60.0 and the letter grade is D.
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/grade_converter.py
Enter the score: 101
The score is 101.0 and the letter grade is invalid.
>>>
= RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/grade_converter.py
Enter the score: -55
The score is -55.0 and the letter grade is invalid.
>>> |
```

4 bmi

In a file called `bmi.py`, write a program that prompts the user for a **height** and **weight**, computes the body mass index (BMI). Then, categorize it as underweight, healthy, overweight, or obese, according to the table below. The formula for BMI is the following:

$$\text{BMI} = 703 \times \frac{\text{weight}}{\text{height}^2}$$

where the weight is in pounds and the height is in inches. The BMI is categorized as follows.

BMI Min	Category
less than 18.5	underweight
at least 18.5 and less than 25	healthy
at least 25 and less than 30	overweight
at least 30	obese

The category is determined by the exact floating point BMI value but the number rounded to the nearest tenth place is printed. When you run your program it should look like this:

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/bmi.py =====
Enter height in inches: 72
Enter weight in pounds: 125
The BMI is 17.0 which is considered to be underweight.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/bmi.py =====
Enter height in inches: 72
Enter weight in pounds: 165
The BMI is 22.4 which is considered to be healthy.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/bmi.py =====
Enter height in inches: 72
Enter weight in pounds: 200
The BMI is 27.1 which is considered to be overweight.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/bmi.py =====
Enter height in inches: 72
Enter weight in pounds: 225
The BMI is 30.5 which is considered to be obese.
>>> |
```

5 body_fat

Body fat percentage chart. According to the [American Council on Exercise](#) body fat percentage falls into five classifications for men and women:

Classification	Women(% fat)	Men (% fat)
Essential fat	10-13	2-5
Athletes	14-20	6-13
Fitness	21-24	14-17
Average	25-31	18-24
Obese	≥ 32	≥ 25

In a file called `body_fat.py`, write a program that prompts the user for their sex (F or M) and their body fat percentage, then prints the correct classification. You can expect the percentage to be entered as an integer. If the percentage is less than the “Essential fat” category, you can label it “Deficient”. Your program should look like this:

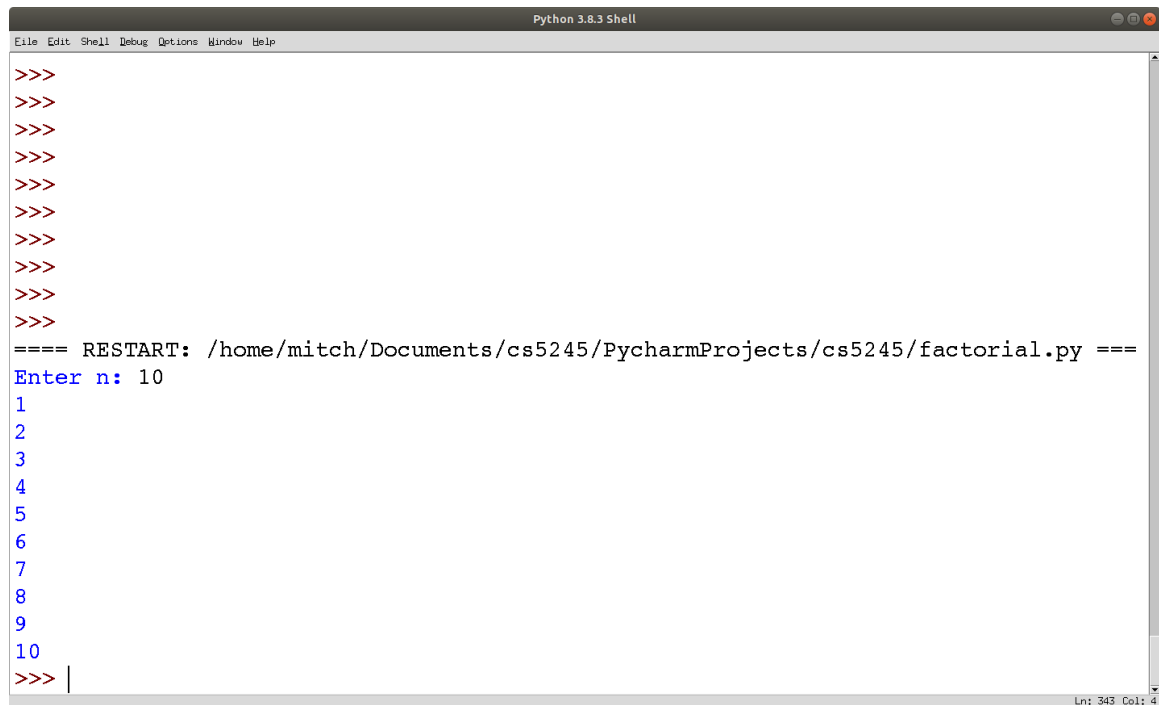
7 factorial

The **factorial** of a positive integer n , denoted by $n!$ is the following:

$$n! = 1 \times 2 \times 3 \times \cdots \times (n-3) \times (n-2) \times (n-1) \times n.$$

In a file called `factorial.py`, write a program that prompts the user for a positive integer n and computes the factorial. Start by printing the numbers that need to be multiplied together (1, 2, ..., n).

For example, you might have output like the following:



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
==== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/factorial.py ====
Enter n: 10
1
2
3
4
5
6
7
8
9
10
>>> |
```

Then, you will need to keep track of the product. Doing so will allow you to print the partial results as you cycle through the integers. Finally, you can print the result of computing the factorial. When you are done it should look like this:

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
==== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/factorial.py ====
Enter n: 10
1 x 1 = 1
2 x 1 = 2
3 x 2 = 6
4 x 6 = 24
5 x 24 = 120
6 x 120 = 720
7 x 720 = 5040
8 x 5040 = 40320
9 x 40320 = 362880
10 x 362880 = 3628800
10! = 3628800
>>>
==== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/factorial.py ====
Enter n: 5
1 x 1 = 1
2 x 1 = 2
3 x 2 = 6
4 x 6 = 24
5 x 24 = 120
5! = 120
>>> |
```

8 coin_flips

In a “coin flip” game, your program simulates a coin flip (1 or 2) and prompts the user to guess 1 or 2. If the user guesses correctly, they “win” that round. [Section 6.6](#) of the textbook describes functions that return a pseudorandom sequence of numbers. For example, you can use the `randint` function to return 1 or 2 with equal probability and without being easily predictable. This can be used to simulate a coin flip.

You can import the `randint` function as follows:

```
from random import randint
```

For example, you can print a random number between 1 and 10 like this:

```
print(randint(1, 10))
```

You can read about the `randint` function here:

<https://docs.python.org/3.9/library/random.html#random.randint>

In a file called `coin_flips.py`, write a program that plays the coin flip game 10 times and reports how many times the user guessed correctly. You can use a `while` loop to do this as described in [Chapter 5](#).

Running your program might look like this:

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
=== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/coin_flips.py ===
Enter your guess (1 or 2): 1
Sorry, 1 != 2.
Enter your guess (1 or 2): 2
Sorry, 2 != 1.
Enter your guess (1 or 2): 2
You are correct 2 == 2.
Enter your guess (1 or 2): 1
You are correct 1 == 1.
Enter your guess (1 or 2): 1
You are correct 1 == 1.
Enter your guess (1 or 2): 2
You are correct 2 == 2.
Enter your guess (1 or 2): 1
Sorry, 1 != 2.
Enter your guess (1 or 2): 1
Sorry, 1 != 2.
Enter your guess (1 or 2): 2
Sorry, 2 != 1.
Enter your guess (1 or 2): 3
Sorry, 3 != 1.
You were correct 4 times.
>>> |
```

9 mean

The `random.randint` function returns integers. Another function that generates a sequence of pseudorandom numbers is the `random.random` function. It returns a real number between 0 and 1 like this:

```
from random import random
x = random()
```

In this example, $0 \leq x < 1$.

Section 6.6 of the textbook describes the `random` function in more detail and other real-valued random number generators can be found here:

<https://docs.python.org/3.9/library/random.html#real-valued-distributions>

In a file called `mean.py`, write a program that prompts the user for a number of trials n and then computes the average (mean) of n values drawn from the `random` function. If every number between 0 and 1 is equally likely, what should the mean of those numbers be close to? What should happen as n increases?

Your program should look something like this:

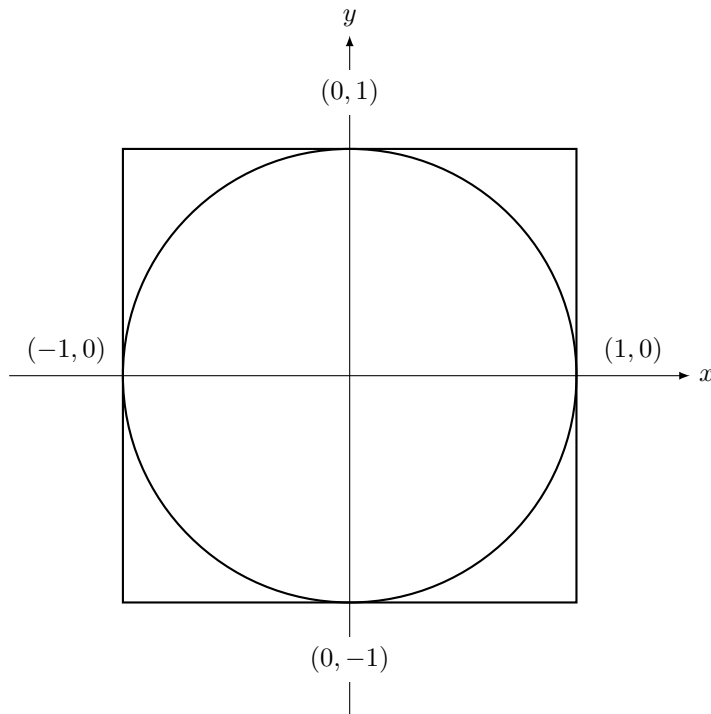

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/mean.py =====
Enter n: 1
The average of 1 trials is 0.029.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/mean.py =====
Enter n: 10
The average of 10 trials is 0.589.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/mean.py =====
Enter n: 100
The average of 100 trials is 0.501.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/mean.py =====
Enter n: 1000
The average of 1000 trials is 0.502.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/mean.py =====
Enter n: 10000
The average of 10000 trials is 0.499.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/mean.py =====
Enter n: 100000
The average of 100000 trials is 0.500.
>>> |
Ln: 613 Col: 4
```

10 pi

The ratio between the area of a circle and the area of a square that circumscribes it (fits perfectly around it) is the following:

$$\frac{\pi r^2}{(2r)^2} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

For example, the unit circle ($r = 1$, centered at the origin $(0, 0)$) is circumscribed by a 2×2 square:



If we randomly sample (select) points within the square, we expect $\frac{\pi}{4}$ of them to fall within the circle. This provides a way to numerically approximate the value of π using sampling. We can randomly sample n points within the 2×2 square and count the number of points that also fall within the unit circle, c . The ratio between c and n should be approximately:

$$\frac{c}{n} \approx \frac{\pi}{4}$$

Therefore, we can approximate π as follows:

$$\pi \approx 4 \frac{c}{n}$$

To estimate π you can randomly sample n points (x, y) that fall within the square and count the ones that also fall within the circle. To generate points within the square, you can use the `random.uniform` function which generates a random value a , where $-1 \leq a \leq 1$:

<https://docs.python.org/3.9/library/random.html#random.uniform>

After sampling an x and y value, determine if it falls within the unit circle using the [Pythagorean theorem](#). Do this n times and keep track of how many fall within the circle.

Finally print a message that shows the estimated value for π .

In a file called `pi.py`, write a program that prompts the user for the number of trials, n , and estimates π using this approach. It should look something like this:

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/pi.py =====
Enter n: 1
After 1 trials, the approximation is pi = 0.000000.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/pi.py =====
Enter n: 10
After 10 trials, the approximation is pi = 4.000000.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/pi.py =====
Enter n: 100
After 100 trials, the approximation is pi = 2.920000.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/pi.py =====
Enter n: 1000
After 1000 trials, the approximation is pi = 3.088000.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/pi.py =====
Enter n: 10000
After 10000 trials, the approximation is pi = 3.129200.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/pi.py =====
Enter n: 100000
After 100000 trials, the approximation is pi = 3.145600.
>>>
Ln: 745 Col: 0
```

Submit to Web-CAT to compute your score!

1. Create a ZIP file for your `cs02` folder by right-clicking the folder and selecting:

- **Send to → Compressed (zipped) folder** on Windows
- **Compress Items** on MacOS
- **Compress** on Linux

You should find the new ZIP file in the same directory where `cs02` resides.

2. Login to <http://webcatvm.cs.appstate.edu:8080/Web-CAT> and submit your ZIP file for grading. You may submit as many times as you want before the deadline.