Week 1 Homework: Exercise 1

Exercise 1

In this five-part exercise, we will count the frequency of each letter in a given string.

Exercise 1a

1/1 point (graded)

Import the `string` library.

Create a variable `alphabet` that consists of the lowercase and uppercase letters in the English alphabet using the `ascii_letters` data attribute of the `string` library.

What line of code did you use to create the `alphabet` variable?

(

import string

```
alphabet = string.ascii_letters)
```

Answer = [alphabet = string.ascii_letters]

Exercise 1b

2/2 points (graded)

The lower and upper case letters of the English alphabet should stored as the string variable `alphabet`.

Consider the sentence 'Jim quickly realized that the beautiful gowns are expensive'. Create a dictionary `count_letters` with keys consisting of each unique letter in the sentence and values consisting of the number of times each letter is used in this sentence. Count upper case and lower case letters separately in the dictionary.

What is the 3rd value of the dictionary?

Answer [1]

What is the 8th value of the dictionary?
Answer = [3]

Code = [

```
for letter in sentence:
    if letter in alphabet:
        if letter in count_letters.keys():
            count_letters[letter] += 1
        else:
            count_letters[letter] = 1

print(list(count_letters.values())[2])
print(list(count_letters.values())[7])
]
```

Exercise 1c

1/1 point (graded)

Rewrite your code from Exercise 1b to make a function called `counter` that takes a string `input_string` and returns a dictionary of letter counts `count_letters`.

Use your function to call `counter(sentence)`.

What is the correct function header for the `counter` function?

```
def counter(count_letters):
```

```
counter <- function(input_string) {
```

```
def counter(input_string):
```

correct

```
def counter(input_string) {
```

Code = [

```
def counter(input_string):
    count_letters = {}
    for letter in input_string:
        if letter in alphabet:
            if letter in count_letters:
                count_letters[letter] += 1
            else:
                count_letters[letter] = 1
    return count_letters
]
```

Exercise 1d

1/1 point (graded)

Abraham Lincoln was a president during the American Civil War. His famous 1863 Gettysburg Address has been stored as `address`. Use the `counter` function from 1c to return a dictionary consisting of the count of each letter in this address and save it as `address_count`.

```
address = """Four score and seven years ago our fathers
brought forth on this continent, a new nation,
conceived in Liberty, and dedicated to the proposition
that all men are created equal. Now we are engaged in a
```

```
great civil war, testing whether that nation, or any
nation so conceived and so dedicated, can long endure.
We are met on a great battle-field of that war. We have
come to dedicate a portion of that field, as a final
resting place for those who here gave their lives that
that nation might live. It is altogether fitting and
proper that we should do this. But, in a larger sense,
we can not dedicate -- we can not consecrate -- we can
not hallow -- this ground. The brave men, living and
dead, who struggled here, have consecrated it, far
above our poor power to add or detract. The world will
little note, nor long remember what we say here, but it
can never forget what they did here. It is for us the
living, rather, to be dedicated here to the unfinished
work which they who fought here have thus far so nobly
advanced. It is rather for us to be here dedicated to
the great task remaining before us -- that from these
honored dead we take increased devotion to that cause
for which they gave the last full measure of devotion
-- that we here highly resolve that these dead shall
not have died in vain -- that this nation, under God,
shall have a new birth of freedom -- and that
government of the people, by the people, for the
people, shall not perish from the earth."""
```

How many times does the letter h appear in the Gettysburg Address?

Answer = [80]

Code = [

```
ddress_count = counter(address)
address_count['h']
]
```

Exercise 1e

1/1 point (graded)

In Exercise 1d, you stored the frequency of each letter in the Gettysburg Address is in `address_count`. Use this dictionary to find the most common letter in the Gettysburg address.

What is the most frequent letter in the Gettysburg Address?
Answer = [e]

Code = [

```
maximum = 0
most_frequent_letter = ""
for letter in address_count:
    if address_count[letter] > maximum:
        maximum = address_count[letter]
        most_frequent_letter = letter
print(most_frequent_letter)
```

]

Week 1 Homework: Exercise 2

Exercise 2

Consider a circle inscribed in a square. The ratio of their areas (the ratio of the area of the circle to the area of the square) is

π
4

. In this six-part exercise, we will find a way to approximate this value.

Exercise 2a

1/1 point (graded)

Using the `math` library, calculate and print the value of $\pi/4$.

What is the value of $\pi/4$?

Answer = [0.785398]

Code =[

```
import math
print(math.pi/4)
]
```

Exercise 2b

1/1 point (graded)

Using `random.uniform()`, create a function `rand()` that generates a single float between –1 and 1.

Call `rand()` once. For us to be able to check your solution, we will use `random.seed()` to fix the seed value of the random number generator.

We include some sample code to get you started:

```
import random

random.seed(1) # Fixes the seed of the random number
generator.

def rand():
    # define `rand` here!
```

`rand()`

What is the value you get from calling `rand()`?

Answer = [-0.7312715117751976]

Code = [

```
import random

random.seed(1) # Fixes the seed of the random number generator.

def rand():
   # define `rand` here!
    return random.uniform(-1,1)

rand()
```

]

Exercise 2d

1/1 point (graded)

Write a function `in_circle(x, origin)` that determines whether a point in a two dimensional plane falls within a unit circle surrounding a given origin.

Your function should return a boolean `True` if the distance between `x` and `origin` is less than 1 and `False` otherwise.

Use `distance(x, y)` as defined in Exercise 2c.

Use your function to determine whether the point (1,1) lies within the unit circle centered at (0,0):

```
def in_circle(x, origin = [0,0]):
   # Define your function here!
```

Does the point (1,1) lie within the unit circle centered at (0,0)?

Yes

No
Correct

Code = [

```
def in_circle(x, origin = [0,0]):
    if len(x) != 2:
        return "x is not two-dimensional!"
    elif distance(x, origin) < 1:
        return True
    else:
        return False

print(in_circle((1,1)))
]
```

Exercise 2e

1/1 point (graded)

Create a list `inside` of `R=10000` booleans that determines whether or not a point falls within the unit circle centered at `(0,0)`.

Set the seed to 1 using `random.seed(1)`.

Use the `rand` function from Exercise 2b to generate `R` randomly located points.

Use the function `in_circle` to test whether or not a given pint falls within the unit circle.

Find the proportion of points that fall within the circle by summing all `True` values in the inside list; then divide the answer by `R` to obtain a proportion.

Print your answer. This proportion is an estimate of the ratio of the two areas!

What is the proportion of points within the unit circle?

Answer = [0.779]

Code = [

```
random.seed(1)

inside = []
R = 10000
for i in range(R):
    point = [rand(), rand()]
    inside.append(in_circle(point))

print(sum(inside) / R)
0.779
]
```

Exercise 2f

1/1 point (graded)

Calculate the difference between your estimate from Exercise 2e and `math.pi / 4`. Note: `inside` and `R` are defined as in Exercise 2e.

What is the difference between our estimate from 2e and the true value of

$\pi/4$?

Answer = [0.0063981633974482 53]

Code = [

```
print(math.pi / 4 - sum(inside) / R)
]
```

Week 1 Homework: Exercise 3

Exercise 3

A list of numbers representing measurements obtained from a system of interest can often be noisy. One way to deal with noise to smooth the values by replacing each value with the average of the value and the values of its neighbors. We will practice data smoothing in this three-part exercise.

Exercise 3b

0/1 point (graded)

Compute and store `R=1000` random values from 0-1 as `x`.

Compute the moving window average for `x` for values of `n_neighbors` ranging from 1 to 9 inclusive.

Store `x` as well as each of these averages as consecutive lists in a list called `Y`.

Use this code to get started:

```
random.seed(1) # This line fixes the value called by your function,
               # and is used for answer-checking.
```

```
# write your code here!
```

What is the moving window average for the 10th entry in `x` for `n_neighbors = 5`?
Answer = [0.45325045824763405]

Code = [
```
random.seed(1)

R = 1000
x = [random.random() for i in range(R)]
Y = [x] + [moving_window_average(x, i) for i in range(1, 10)]

Y[5][9]
0.45325045824763405
]
```

Exercise 3c

1/1 point (graded)

For each list in `Y`, calculate and store the range (the maximum minus the minimum) in a new list `ranges`.

Print your answer. As the window width increases, does the range of each list increase or decrease? Why do you think that is?

As window width increases, does the range of each list increase or decrease?

increase

decrease
Correct

Code [
```
ranges = [max(x) - min(x) for x in Y]
print(ranges)
        ]
```