Homework 1

September 15, 2022

1 Homework 1: Welcome to Python

1.1 Learning Objectives:

- write Python code using variables and loops to solve simple mathematical problems
- become familiar with Jupyter Notebook and running Python programs
- gain practice printing results to match a specification exactly

You will make use of all of these skills throughout the semester! Python is an easy-to-use but powerful programming language that is particularly well-suited to quickly creating small programs.

Note: you should **NOT** post any of your code on the internet or let other students see your code. If you have questions or problems with your code, try sending an email to staff, or attending office hours.

For problems in this homework, you will edit this notebook only. When you have completed this homework, running the whole notebook should obtain the answers and print the output.

The python interpreter is a piece of software that can evaluate python expressions. One example of this is Python Tutor

2 Materials

2.1 Problem 0: Python for Beginners (20 pts)

You are expected to watch a video Python for Beginners Tutorial before starting Homework 1.

Please type "I have wacthed the video *Python for Beginners*" in the following cell.

Hint: For mastering Python programming, Google's Python Introduction is one of my favorite tutorials.

2.2 Problem 1: Roots (10 pts)

Compute and print both roots of the quadratic equation **with the smaller root first**:

$$3x^2 - 5.86x + 2.5408$$

. You can assume neither answer will be a complex number.

Hint: recall that the roots of a quadratic equation

$$ax^2 + bx + c$$

are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Hint: use the math.sqrt function to compute the square root.

Hint: If you are using a particular value repeatedly, be sure to store it in a variable.

[1]:
```
import math

a = 3
b = -5.86
c = 2.5408

root1, root2 = 0, 0

# YOUR CODE HERE
d = (b**2)-(4*a*c)
root1 = (-b+math.sqrt(d))/2*a
root2 = (-b-math.sqrt(d))/2*a
#raise NotImplementedError()

print(f'Root 1 is {root1}, and root 2 is {root2}')
```

```
Root 1 is 11.73321253055229, and root 2 is 5.846787469447711
```

2.3 Problem 2: Reciprocals (10 pts)

Use a **for** loop to print the decimal representations of 1/2, 1/3, ..., 1/10, one on each line.

Hint:What do you notice about how the loop is counting, and the denominator of the fractions?

[2]:
```
# YOUR CODE HERE
for i in range(1,11):
    print(1/i)
#raise NotImplementedError()
# When we use range(x,y), x is the starting number and last number is (y-1).
```

```
1.0
0.5
0.3333333333333333
0.25
0.2
0.16666666666666666
0.14285714285714285
0.125
0.1111111111111111
0.1
```

2.4 Problem 3: Triangular numbers (10 pts)

Use a **for** loop to compute the 10th triangular number. The nth triangular number is defined as

$$1 + 2 + 3 + ... + n$$

.

[3]:
```
n = 10
triangular = 0
# YOUR CODE HERE
for i in range(1,(n+1)):
    triangular += i
#raise NotImplementedError()
print("Triangular number", n, "via loop:", triangular)
print("Triangular number", n, "via formula:", n * (n + 1) / 2)
```

```
Triangular number 10 via loop: 55
Triangular number 10 via formula: 55.0
```

2.5 Problem 4: Factorial (10 pts)

Use a for loop to compute 10!, the factorial of 10. Recall that the factorial of n is $1 \times 2 \times 3 \times ... \times n$. **Note: you may not use the math.factorial() function for this problem or any other pr**
You should also NOT use recursion. (If you do not know what that means, do not worry because you should not use it :-)]

Hint: Your answer will be similar to your answer to "Problem 3: Triangular numbers".

As in Problem 3, your code should be able to calculate 11!, 12!, or any other number's factorial just by changing the first line to set n to 11, 12, or any other number greater than 0. Be sure the code you submit calculates the factorial for n = 10.

[4]:
```
n = 10
factorial = 0

# YOUR CODE HERE
for i in range(1,(n+1)):
    if factorial == 0:
        factorial = i
        factorial = factorial*i
    else:
        factorial = factorial*i

#raise NotImplementedError()
print(f'{n}!: {factorial}')
```

```
10!: 3628800
```

2.6 Problem 5: Multiple factorials (10 pts)

Write code to print the first 10 factorials, in reverse order. In other words, write code that prints 10!, then prints 9!, then prints 8!, ..., then prints 1!. Its literal output will be:

[5]:
```
# YOUR CODE HERE

## Using a While loop and a for loop ##

# n = int(input("Please enter a number :  \n")) # We can also take a user input␣
 ↪instead of hardcoding.
n = 10
# factorial = 1
# while (n>0):
#     factorial = 1
#     for i in range(1,n+1):
#         factorial = factorial * i
#     print(f'{n}!: {factorial}')
#     n = n-1

## Using two for loops

for i in range(n,0,-1):
    factorial = 1
    for j in range(1,i+1):
        factorial = factorial * j
    print(f'{i}!: {factorial}')
    # print(i)
#raise NotImplementedError()
```

```
10!: 3628800
9!: 362880
8!: 40320
7!: 5040
6!: 720
5!: 120
4!: 24
3!: 6
2!: 2
1!: 1
```

The first line of your solution should assign a variable **num_lines** to 10. Then, as in Problems 3 and 4, the rest of the code should print the correct number of lines and correct factorial on each line for values other than 10 just by setting **num_lines** to a different value.

Hint: Use two nested for loops. The outer loop sets the value of n to the values **num_lines, num_lines-1, num_lines-2, ..., 1**, in succession. Then, the body of that loop is itself a loop — exactly your solution to "Problem 4: Factorial", without the first line **n = 10** that hard-codes the value of n.

Hint: All calculated factorials should be integers

[7]:
```
# YOUR CODE HERE
# raise NotImplementedError()
```

2.7 Problem 6: Sums of reciprocals of factorials (10 pts)

Compute the following value: 1 + 1/1! + 1/2! + 1/3! + 1/4! + … + 1/10! The value should be close to e (2.71828), the base of the natural logarithms.

Like with the previous few problems, you should use a variable n = 10 to determine how many fractions to add. In theory, increasing this number should get you closer to the true value of e.

Hint: The easiest way to solve this is with two nested for loops. It is possible, but tricky, to compute this using only one for loop. That is not necessary for this assignment.

Hint: Copy your solution to "Problem 5: Multiple factorials", then modify it. Rather than printing the factorials, you will add their reciprocals to a running total, then print that total at the end.

Hint: don't try to work the very first "1 +" into your loop; do it outside the loops (either at the very beginning or the very end of the outer loop).

[8]:
```
sum = 0
# YOUR CODE HERE
n = 10
for i in range(0,n+1):
    fact = 1
    for j in range(1,i+1):
        fact = fact * j
    sum = sum + (1/fact)
# raise NotImplementedError()
print(f'Sum of factorials is {sum}')
```

```
Sum of factorials is 2.7182818011463845
```

2.8 Submit your work and answer a brief survey (20 pts)

You are almost done!

- Double-check that running your entire file produces the correct output for each problem in the assignment.
- Submit your **ipynb** file at Canvas.
- Answer a brief REQUIRED survey asking how much time you spent and other reflections on this assignment.

Now you're done!