CSCI2291 Homework 1

Jack Moffatt

January 27, 2022

Problem 1

(a) Our code reads as follows

```
min([x ** 2 for x in L])
```

Our list in this case is constructed as a for statement that creates a list of the squared elements in L. Then, after this list has been constructed, the *min* function takes the smallest element of this list of squares. Using the sample list

```
L = [4, -4, 6, 8, -2, 7]
```

our code gives us the answer, 4, which is the square of -2.

(b) Our code reads as follows

```
D[-3]
```

This code uses negative indexes. $D[-1]$ would return this last element in the list. Likewise, $D[-2]$ would return the firs-from-last element, so it follows that $D[-3]$ returns the second-from-last element. Using the sample NumPy array:

```
D = np.array([4, -4, 6, 8, -2, 7])
```

our code gives us the answer 8, which is the second to last element in the NumPy array.

(c) Our code reads as follows

```
sum([n**3 for n in range(-50, (10**4 + 1))])
```

First we construct a list of all the terms in the series with a for statement. The lower bound in the *range* function is inclusive, so we use -50. But the upper bound is exclusive, so we must use `(10 ** 4 + 1)`, so that the last term iterated over is `10 ** 4`. Then for each term in this range we take its cube and put it into a list, then we take the sum of the elements in the list with the `sum()` function. When executed our code returns the answer:

2498500323344376

(d) Our code reads as follows

```
[n for n in range(100) if n**4 > 500 * n]
```

First, we iterate over the range of $[0, 99]$, by using `range(100)`. Then for each element we iterate over, we apend it to our list if it satisfies the condition `(n ** 4 > 500 * n)`. When executed our code gives us a list

```
[8, 9, 10, ...  99]
```

(e) Our code reads as follows

```
len([ D[key] for key in D if D[key] != 'red'])
```

In this expression, we create a list of dictionary values for each value in our dictionary which is not `'red'`. Then once we have a list of every value in our dictionary that is not `'red'`, we use the `len()` function to get the number of elements in our list. Using the example dictionary

```
D = { 1 : 'red',
      2 : 'black',
      3 : 'red',
      4 : 'black',
      5 : 'red',
      6 : 'black',
      7 : 'red'}
```

Executing our code, we get the answer 3.

Problem 2

First let us take the integral of $f'(x)$:

$$f(x) = \int f'(x)dx$$
$$f(x) = \int 1 - sin(x)dx$$
$$f(x) = x + cos(x) + C$$

Now, let us solve for C. We know that $f(\pi) = 1$, so we compute

$$f(\pi) = \pi + cos(\pi) + C$$
$$1 = \pi - 1 + C$$
$$C = 2 - \pi$$

So we find that $f(x) = x + cos(x) + 2 - \pi$. Finally, we can solve for $f(0)$. We compute

$$f(0) = 0 + cos(0) + 2 - \pi$$
$$f(0) = 1 + 2 - \pi$$
$$f(0) = 3 - \pi$$

Problem 3

(a) We will define a Python function that generates the largest integer, m satisfying the given inequality. To do this, we can initialize $m = 1$ and then create a while loop which increments by 1 with each iteration. Our code reads as follows:

```python
def generate_largest_a():
    m = 1
    while 10 ** m < (1000 * (m ** 6)):
        m += 1
    m -= 1
    print(f"m = {m}")
```

When this function is called, the while loop will begin by checking the statement in the header with $m = 1$. After finding it is true, the loop will increment m by 1 and then jump back to the header, where the process will repeat with $m = 2$. It will continue to repeat the process until $m = 8$.

At this point the header will be satisfied, but then when m increments to 9, the while loop header will be checked again and found to be false, which terminates the loop. Since at this point $m = 9$ but 9 does not satisfy the equality we must decrement m back to 8. And then we may print $m = 8$ as our final answer.

This answer can also be checked by hand if we compute

$$\begin{aligned}
10^8 &= 100000000 \\
1000 \cdot 8^6 &= 262144000 \\
\implies 10^8 &< 1000 \cdot 8^6
\end{aligned}$$

while,

$$\begin{aligned}
10^9 &= 1000000000 \\
1000 \cdot 9^6 &= 531441000 \\
\implies 10^9 &> 1000 \cdot 9^6
\end{aligned}$$

So we confirm that $m = 9$ does not satisfy the inequality, thus our Python code correctly tells us that the largest integer satisfying the inequality is

$$m = 8$$

(b) We know that for $m = 100$, the cubed term will be growing far faster than our squared and linear terms, so by the time we reach $m = 100$, our function will be negative and will coninute to decrease as $m \to \infty$. This is due to the relative growth rate of the functions x, x^2, x^3.
As a result, if we begin iteration from $m = 100$ and decrement m until our condition is satisfied, we can quickly find the largest such m that satisfies the inequality. Our code reads as follows:

```python
def generate_largest_b():
m = 100
while 1000 - (100 * m) + (5 * m ** 2) - ((1 / 15) * (m ** 3)) < 500:
    m -= 1
print(m)
```

When running this code, we see that we are given the answer $m = 45$. We can verify this:

$$1000 - 100(45) + 5(45^2) - frac(45^3)15 = 550$$
$$1000 - 100(46) + 5(46^2) - frac(46^3)15 = 490.933$$

We see that with $m = 45$ the function is greater than 500, while if we set m to 46, then the function is less than 500. So our python code helps us find the answer of

$$m = 45$$