q1

February 3, 2021

[1]:
```
import numpy as np
```

Part A: The largest among the absolute values of the elements of the list L

[2]:
```
L = [1, 2, 3, -1, -2, -3, 4, -5]
max_abs_val = max([abs(item) for item in L])
max_abs_val
```

[2]: 5

Part B: The next-to-last element (second highest index position) of the numpy array a

[3]:
```
numpy_array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
numpy_array[numpy_array.size - 2]
```

[3]: 9

Part C: The sum of n starting at n-25 to 10^6

[4]:
```
sum = sum([i for i in range(25, 10**6 + 1)])
sum
```

[4]: 500000499700

Part D: List of all non-negative ints n < 100 such that n^3 > 1000n

[5]:
```
valid = [n for n in range(0, 100) if n**3 > 1000*n]
```

Part E: Number of entries in the dictionary D that do not have the value 'approved'

[6]:
```
D = {'A': 'approved', 'B': 'not approved', 'C': 'approved', 'D': 'not␣
↪approved', 'E':
    'approved', 'F': 'approved', 'G': 'approved'}

num_elements_not_approved = len([key for key in D.keys() if D[key] !=␣
↪'approved'])
num_elements_not_approved
```

[6]: 2

q2

February 3, 2021

We first need to find the definition of the function f(x). To do so, we integrate the first derivate of f(x) as follows:

$$f'(x) = 1 - x^2 \int f'(x)dx = x - \frac{1}{3}x^3 + C \implies f(x) = x - \frac{1}{3}x^3 + C$$

We know that f(0) = 2, so we can use this to solve for C:

$$2 = 0 - \frac{1}{3}(0) + C \implies C = 2 \implies f(x) = x - \frac{1}{3}x^3 + 2 \implies f(3) = 3 - \frac{1}{3}(27) + 2 = -4$$

q3

February 3, 2021

Part A: The largest positive integer, m, for which 2^m < 100m

[1]:
```
def find_largest_m(current):

    if (2**current) >= (100 * current):
        return current - 1
    else:
        return find_largest_m(current+1)
```

[2]:
```
find_largest_m(1)
```

[2]: 9

Explanation: The term 2^m grows exponentially, whereas the term 100m grows linearly. Thus, there is a point at which the term 2^m becomes larger than 100m and continues to grow even greater, never again being less than 100m. For smaller values of m, 100m is larger, however, after m=9, 2^m significantly "overtakes" 100m and continues to grow much faster and larger than 100m as m increases.

Part B: Largest positive integer m for which 800 - 100m + 5m^2 - (1/15)m^3 >= 500

[3]:
```
def find_largest_m_partb(current):

    val = 800 - (100*current) + (5*(current**2)) - ((1/15)*(current**3))

    if val < 500:
        return current - 1
    else:
        return find_largest_m_partb(current+1)
```

[4]:
```
find_largest_m_partb(1)
```

[4]: 3

Explanation: When m=0, we are left with the inequality 800 >= 500. As m increases, however, the left-hand side of the inequality decreases. This is because the term m^3, which is being subtracted from 800, becomes very large for larger values of m. Although we are also adding the term m^2 multiplied by a constant, the subtraction of the term m^3 and the term 100m "outweigh" the addition of 5m^2, so the value on the left gets smaller and smaller as m increases.