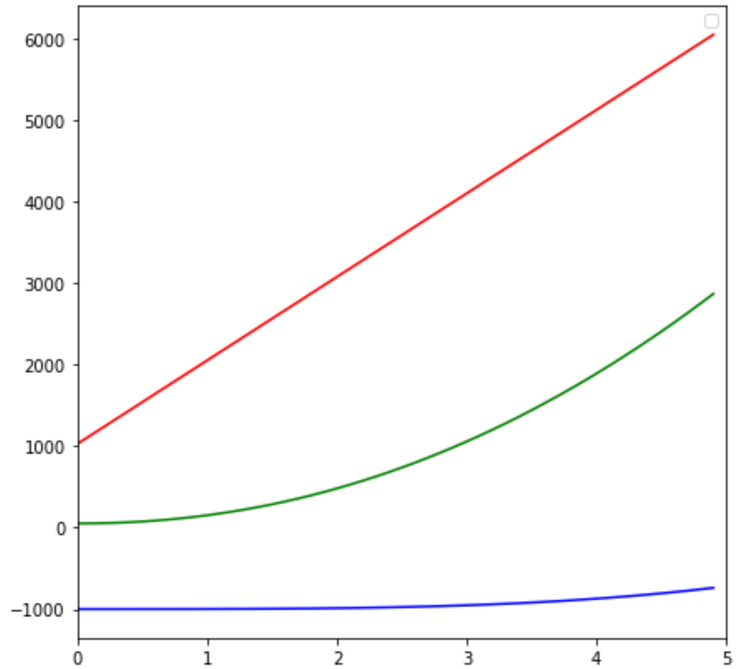
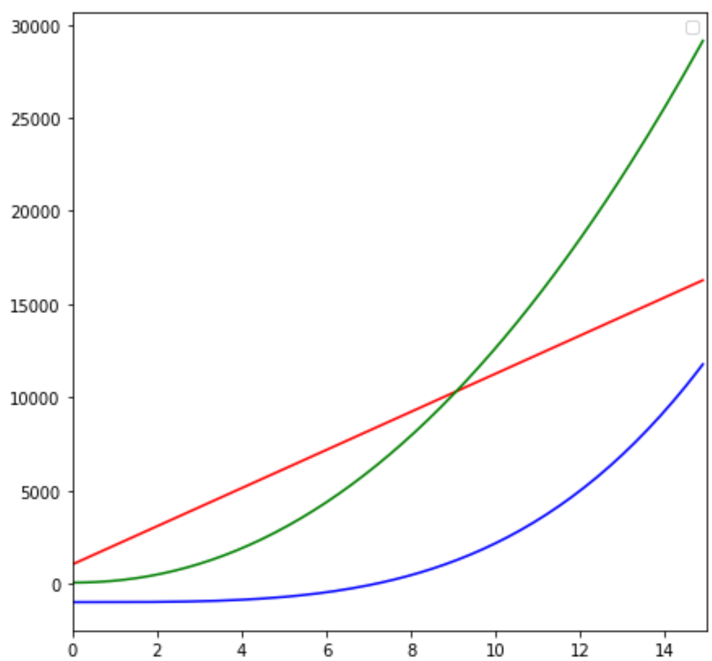
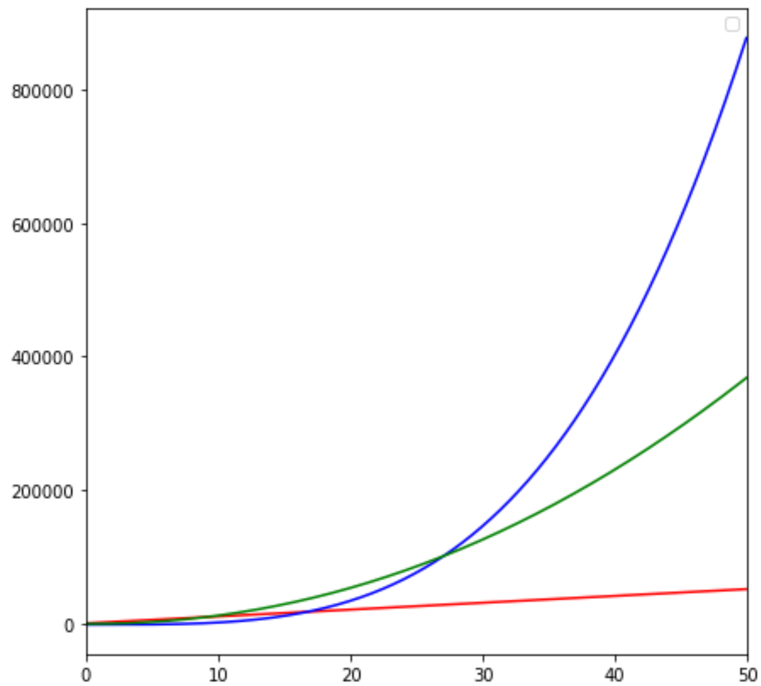
Jacey (Jixing) Man CS 566
Question 1
X-axis =5



x-axis = 15.



x-axis = 50.



Code:

```
"""
Created on Sun Sep 13 14:50:19 2020

@author: jixingman
"""


import math
import numpy as np
import matplotlib.pyplot as plt

msize = 5

t = np.arange(0, msize, 0.1)
plt.plot(t, 2**10*t + 2**10 , 'red', t, t**3.5 - 1000, 'blue', t, 100*t**2.1 + 50, 'green')

plt.xlim(0, msize)
plt.rcParams["figure.figsize"] = (7,7)
plt.legend()
plt.show()

#########
msize = 15
t = np.arange(0, msize, 0.1)
plt.plot(t, 2**10*t + 2**10 , 'red', t, t**3.5 - 1000, 'blue', t, 100*t**2.1 + 50, 'green')

plt.xlim(0, msize)
plt.rcParams["figure.figsize"] = (7,7)
plt.legend()
plt.show()


#####
msize = 50
t = np.arange(0, msize, 0.1)
plt.plot(t, 2**10*t + 2**10 , 'red', t, t**3.5 - 1000, 'blue', t, 100*t**2.1 + 50, 'green')

plt.xlim(0, msize)
plt.rcParams["figure.figsize"] = (7,7)
plt.legend()
plt.show()
```

Description:

F1 is a linear function, F2 and F3 are both exponential growth function, therefore when the x-axis is small, such as 5, it will show F1 as more steep, with the other two line be more flat. However as the x-axis increased, both F2 and F3 will have higher growth and have much higher y-value on the graph compare to linear function F1

Question 2:

Rule: O(g(n)) = {f (n) there exists constants c, $n_0 > 0$ such that $0 \le f(n) \le c \times g(n)$ for all $n \ge n_0$} f grows no faster than g

1. 2^(n+1.3) = O(2^n)
   Answer: Yes

f(n) = 2^1.3 * 2^n = 2.46*2^n

n0 = 1 and c = 2.46

0 <= 2.46 <= 2.46

#####################################

2. 3^(2*n) = O(3^n)
   Anwser: NO

f(n) = (3^2)^n = 9^n, I can not find the constants c to make this possible, and f does grow faster than g

Question 3: Rule: (holds true if there exist positive constants n0,c1 and c2), f grows as the same rate as g

$\Theta(g(n))$ = { f(n): there exist positive constants c_1, c_2 and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$ }

1. f(n) = (4*n)^150+(2*n+1024)^400 = 4^150+n^150+2*n^400+1024^400
   g(n) = 20*n^400+(n+1024)^200 = 20*n^400+n^200+1024^200

Answer: Yes, f(n) does equal to Theta(g(n)) , because f grow rate is same as g

2. f(n) = n^1.4*4^n
   g(n) = n^200*3.99^n

Answer: No, f(n) does not equal Theta(g(n)), because f grows rate is faster than g. The exponent 4^n of f(n) will grow faster than g(n) 3.99^n

3. f(n) = 2^log(n)
   g(n) = n^1024

Answer: No, f(n) does not equal Theta (g(n)), because f grows rate is slower than g. The f(n) formula even as a exponent, its growth is very slow

Question 4: Start with inner while loop at line 8 is with two constant, so it is log(n)+2, line 6's for loop, is the bigger inner loop, so n*(log(n)+2), with line 7 as another constant, so n*(((log(n)+2))+1) , the while loop at line 2, is not part of the for loop in line 6, it is log(n)+2, line1: i= 1 is also constant, so the whole code is:
1+ log(n)+2+ n*((log(n)+2)+1)
=1+log(n)+2+n*log(n)+2n+n*1
=n*log(n)+3n+3
so the tight Theta of this pseudocode is f(n) = Theta(n^2)

Question 5:
Start with inner loop at line 3, 1 for loop with a constant, it is n+1, with the bigger loop at line 2, so it is n(n+1), with line 1 as another constant :
n(n+1)+1
=n^2+n+1
with f(n) = Theta(n^2)
so the big O of this pesudocode is f(n) = O(n^3)