IT1007 Introduction to Programming with Python and C — Mock PE

Submission instructions [Coursemology]:

1. Duration: 1 hour
2. Complete your code using the skeleton files provided, then **test your code on your computer first** before submitting to Coursemology.
3. To submit your code on Coursemology, click on "Mock PE" in the sidebar followed by the appropriate "Attempt" button.
4. **Copy ONLY the required function** from your completed skeleton file into the Coursemology code window.
5. Click "Run Code" to test that your function works on Coursemology.
6. Click "Finalise Submission" to submit your code for the **ENTIRE Part A/B**. **You will not be able to amend your code after you have finalised your submission.**
7. You must name your functions exactly as the questions state.

Failure to follow each of the instruction will result in 10% deduction of your marks.

Important Note: In this test, you should manipulate the images with `Numpy` and `Scipy` packages ONLY, but **not** other library such as PIL. And the package `matplotlib` should be used for showing the image only, namely only `imshow()` and `show()`.

You should **not** import any other functions other than those that are already imported in the given code.

You should not import any other functions other than those that are already imported in the given code.

Question 1 [10 marks]: Computing the Natural Number *e*

The mathematical constant *e* is the unique number whose natural logarithm is equal to one. And you can compute the number with the following sum the infinite series,

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1 \times 2} + \frac{1}{1 \times 2 \times 3} + \frac{1}{1 \times 2 \times 3 \times 4} + \cdots$$

Of course, we cannot compute the perfect *e* until n equals to infinity. However, we can compute n to a certain steps. E.g.

$$\sum_{n=0}^{1} \frac{1}{n!} = 2$$

$$\sum_{n=0}^{2} \frac{1}{n!} = 2.5$$

$$\sum_{n=0}^{100} \frac{1}{n!} = 2.7182818284590455$$

Write a function '`compute_e(n)`' such that it compute *e* from the terms 0 to n. Here are some sample outputs:

```
>>> compute_e(0)
1.0
>>> compute_e(1)
2.0
>>> compute_e(2)
2.5
>>> compute_e(100)
2.7182818284590455
>>> compute_e(5000)
2.7182818284590455
```

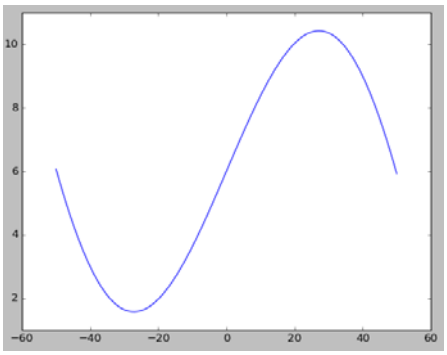
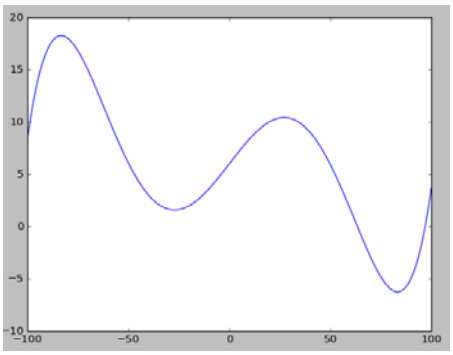
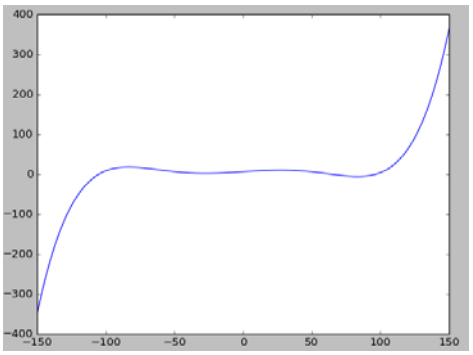
Question 2[30 marks]: Plotting graphs

There are two parts in this question, you are recommended to read both parts first before proceeding.

Question 2a

You are given a function f(x). Write a function '`plot(x1,x2)`' to plot a graph of f(x) from x = x1 to x = x2. You can assume that you only have to plot f(x) with every integer values between x1 and x2, and assuming they are integers.

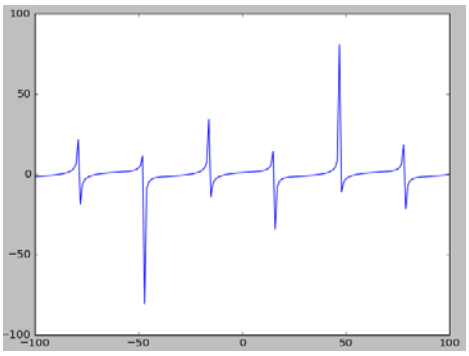
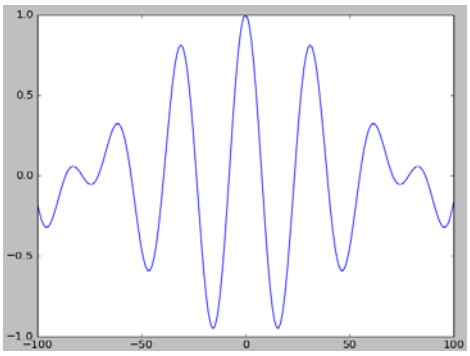
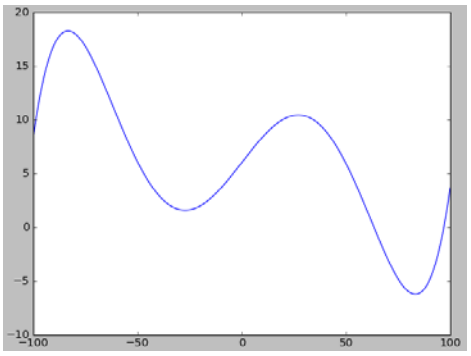
Sample Output: Here are the three graphs of 'plot(-50,50)' , 'plot(-100,100)' and 'plot(-150,150)' respectively.

Question 2b

You are given three functions 'f1', 'f2' and 'f3' in the file '`MPEQ2b.py`'. Rewrite your function '`plot(x1, x2, f)`' such that it does the same function as in Q2a but it can plot an arbitrary function 'f' as an input

Sample Output: Here are the graphs when you call '`plot(-100,100,f1)`', '`plot(-100,100,f2)`' and '`plot(-100,100,f3)`' respectively

Question 3[30 marks]: Dyeing Hair

You are given an image with green hair. Write a function '`dye_hair(filename)`' to read in a file and change the green color hair into pinkish purple. You function should do two tasks. First, change the hair color to pinkish purple and display the original and dyed pictures side by side in the same window WITHOUT the axes. Second, save the new image into a file named '`dye_hair_output.jpg`' (only the dyed one, no need to save the original one). Your output should look like this:

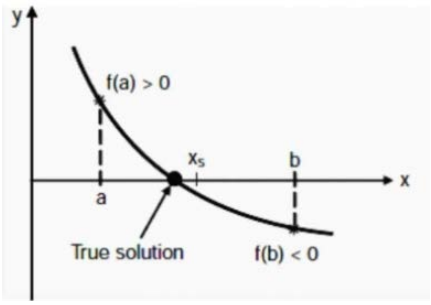
How to change the hair color? First, you have to check which pixel is green. To determine if a pixel is green, you simply check if the green (G) value is greater than the red (R) and the blue (B) values. Once you figured out a pixel is green with its color [R, G, B], you replace the color by [R x 2, G x 0.2, and B x 0.8].

Question 4[30 Marks]: Bisection Method

Note that in this question, you cannot use the 'Newton Method' in the lecture slides. You have to follow the bisection method.

The ***bisection method*** in mathematics is a root-finding method that repeatedly bisects an interval and then selects a subinterval in which a root must lie for further processing. Given a function f(x), you want to solve for x when f(x) = 0.

In this question, we assume that the function f is continuous. And you are given two numbers a and b such that f(a) > 0 and f(b) < 0. So you repeat the following



- Compute x_s = (a + b) /2
- If f(x_s) < 0 then the solution lies on the left of x_s. So, b = x_s.
    - Otherwise, a = x_s.
- Repeat these until the absolute value of f(x_s) is smaller than a constant 'error'

After the program exits the loop, the value of x_s should be a very close answer for f(x) = 0.

You are given a function f(x) to solve by bisection method. (The same function in Q2a) Write a function '`bisection(start,end)`' to solve x for the equation f(x) = 0 with bisection method. The value of the '`error`' is given to you.

Sample Output: (Your numbers could be a bit off. However, `abs(f(x))` should be less than '`error`'

```
Final answer = 63.39649252593517
And f(x) is 8.877840684817784e-09
>>>
```