YSC2221 Intro to Python PE

Important Note:

- You should **NOT** import any packages at all for Questions 1 and 2.
- For Question 3, 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()`.

Question 1 [30 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 (and no one can). Let's say we compute the first $i+1$ terms of e such that

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

So we can actually stop computing e up to the $(i+1)^{th}$ term if $e_{i+1} - e_i$ is smaller than a certain error constant. We assumed that the error we can tolerate must be less than 1.0 .

Write a function '`compute_e_within_error(err)`' such that

- It will compute and return e_i when $e_{i+1} - e_i$ is smaller than the number 'err'.
- It will print a message of how many terms it has computed (namely, the number i)
- And it should NOT waste time on computing extra j^{th} term for $j > i+2$.

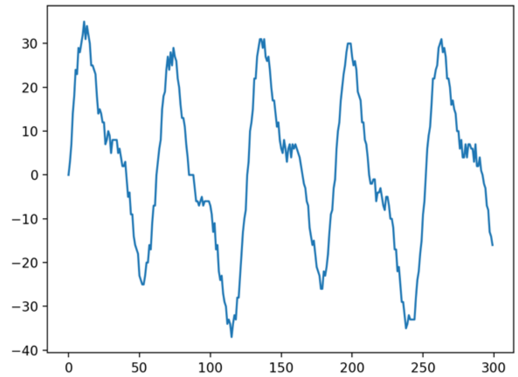
Sample outputs:

```
>>> e = compute_e_within_error(0.99)
No. of terms = 2
>>> print(e)
2.0
>>> e = compute_e_within_error(0.01)
No. of terms = 5
>>> print(e)
2.708333333333333
>>> e = compute_e_within_error(0.0000000000000000001)
No. of terms = 21
>>> print(e)
2.7182818284590455
```

Note that the "No. of terms" is equal to $i+1$. We allow an error of ±1.

Question 2 [45 marks]: Filtering a wave

You are given a wave like your lab exercise before. This time, your job is to smooth the wave out by a very simple filter. You are given a wave in a list named '`original_wave`' as follows:



Part 2a [35 marks]

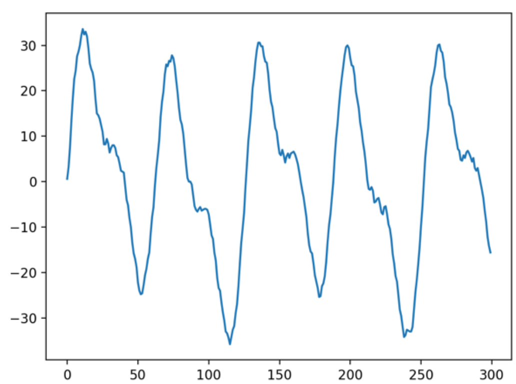
Write a function '`filter_wave(wave)`' and this will produce a new wave which is a smoothed version of the input 'wave'. Following the rules below

- In the new wave, for every position i, it is the weighted sum of three positions of the original wave. More preciously, the new wave value in position i will be equal to

```
new_wave[i] = wave[i-1] * 0.2 + wave[i]*0.6 + wave[i+1]*0.2
```

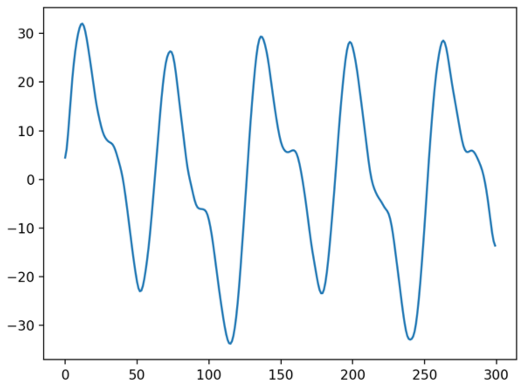
- Let `len(wave)` be L. The formula may require you to compute two out of bounds values, namely wave[-1] and wave[L]. You may substitute these values respectively with wave[0] and wave[L-1].
- As in the lab before, you should **NOT** modify the original wave input

Here is the expected shape of the wave after `filter_wave(original_wave)`



Part 2b [10 marks]

Write a function ‘`filter_wave_n(wave,n)`’ for **n** ≥ 0. And this will repeat the function `filter_wave()` in Part 2a **n** times to the wave accumulatively and produce an even smoother wave. Here is the expected wave for `filter_wave_n(wave,10)`



Again, you should not modify the original wave. **And note that you will gain marks for Part 2b ONLY IF your Part 2a also works correctly**.

Question 3[25 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 first in a window, then the dyed pictures in another window.

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].