Feedback comments

–

Hello Ben,

Input (30/ 30)

Good user information.

Limits are set correctly and allows user to enter in reverse order - well done! Line 132 (maxi + step) causes the code to go beyond the maximum value. Try a = -30 and b = 700. The printout includes 720 nm.

Well done on how you dealt with the possibility of unphysical input (negative wavelengths), and string character input by user?

Calculation (30/ 40)

Values are correct except when you enter the wavelength in **nm**, you have to convert to **m** before performing the conversions. That means your frequencies and energies are out by a factor of 10^9. Normally I would penalise such an error heavily, but everything else in your submission is so good, that I gave you only a minor penalty.

Excellent use of user-defined functions

Output (30 / 30)

Correct fill character and field separator used.

Precision, field width and justification is as according to the specifications.

Units included in header - well done!

You forgot to comment out lines 60 and 62 (testing I assume).

Overall:

Well commented throughout, well indented and no lines exceed 80 characters.

Very clear and tidy code. Only error is nm –> m.

Total: 90%

SHOW MENU

Feedback comments

–

Hello Ben,

Really nice work.

Excellent use of functions and comments throughout.

Really nice to include the two plots. Your work deserves a mark of 100%, but there are a few minor points I want to you to be aware of:

line 81 should be outside the **for** loop. It performs the calculation for no purpose. The calculation of c_error is only needed after the **for** loop terminates. The thing to realise is that as n becomes very large, this is a huge waste of computing resource and introduces unecessary delay in running the code. When starting to write highly complex programs (say a global weather prediction program), this type of error has to be avoided at all costs.

You could improve the information to the user telling the condition under which the program terminates, i.e., the precision threshold.

Alternatively, you could ask the user to specify the precision after the initial prompt. If the problem were very complex it might take a week to reach a convergent solution!

Formatting of the output is excellent.

Lines 88-90:

Try setting line 103 to

main(n,0,9) or

main(n,0,13)

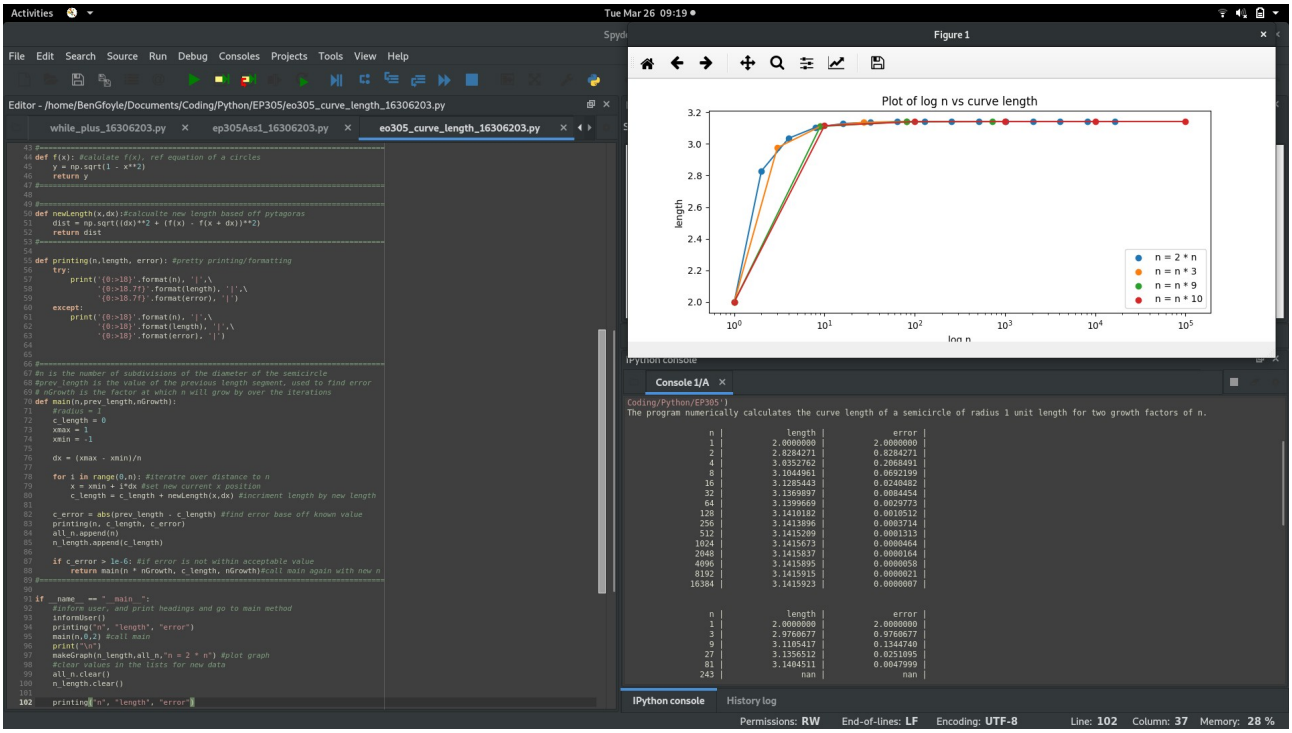
What happens? Why does it happen. Why does the program behaviour depend on n like this?

You can code very well. The above are intended to refine your coding and make it even better.

Frank

```
ggggggggggggggggggggg-60.0 \ gggggggggggggggggUNDEFINED \ ggggggggggggggggggUNDEFINED
ggggggggggggggggggggg-30.0 \ gggggggggggggggggUNDEFINED \ ggggggggggggggggggUNDEFINED
ggggggggggggggggggggg0.0 \ gggggggggggggggggUNDEFINED \ ggggggggggggggggggUNDEFINED
ggggggggggggggggg3.000e+01 \ ggggggggggggggggg4.133e-08 \ ggggggggggggggggg9.993e+06
ggggggggggggggggg6.000e+01 \ ggggggggggggggggg2.066e-08 \ ggggggggggggggggg4.997e+06
ggggggggggggggggg9.000e+01 \ ggggggggggggggggg1.378e-08 \ ggggggggggggggggg3.331e+06
ggggggggggggggggg1.200e+02 \ ggggggggggggggggg1.033e-08 \ ggggggggggggggggg2.498e+06
ggggggggggggggggg1.500e+02 \ ggggggggggggggggg8.266e-09 \ ggggggggggggggggg1.999e+06
ggggggggggggggggg1.800e+02 \ ggggggggggggggggg6.888e-09 \ ggggggggggggggggg1.666e+06
ggggggggggggggggg2.100e+02 \ ggggggggggggggggg5.904e-09 \ ggggggggggggggggg1.428e+06
ggggggggggggggggg2.400e+02 \ ggggggggggggggggg5.166e-09 \ ggggggggggggggggg1.249e+06
ggggggggggggggggg2.700e+02 \ ggggggggggggggggg4.592e-09 \ ggggggggggggggggg1.110e+06
ggggggggggggggggg3.000e+02 \ ggggggggggggggggg4.133e-09 \ ggggggggggggggggg9.993e+05
ggggggggggggggggg3.300e+02 \ ggggggggggggggggg3.757e-09 \ ggggggggggggggggg9.085e+05
ggggggggggggggggg3.600e+02 \ ggggggggggggggggg3.444e-09 \ ggggggggggggggggg8.328e+05
ggggggggggggggggg3.900e+02 \ ggggggggggggggggg3.179e-09 \ ggggggggggggggggg7.687e+05
ggggggggggggggggg4.200e+02 \ ggggggggggggggggg2.952e-09 \ ggggggggggggggggg7.138e+05

In [3]:
```



```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 11 11:19:21 2019

@author: BenGfoyle - 16306203 - github.com/bengfoyle/ep305

Overview: This program will take in 2 values which correspond to the wavelength
of a photon of light in nanometers. The corresponding frequency, and energy in
electron volts will be calculated. The output follows a predefined format.
"""
#import scipy to retieve constants
from scipy import constants as con
from numpy import arange

#define constants
PI = con.pi
PLANCK = con.Planck
LIGHT = con.c
EV = con.e

#==============================================================================
def informUser(): #tell user whats happenings
    print("This program will take in 2 values a and b which correspond to",\
          "the wavelength of a photon of light in nanometers. The ",\
          "corresponding frequency, and energy in electron volts will be",\
          "calculated between the values of a and b. The output follows",\
          "a predefined format.")
#==============================================================================

#==============================================================================
def newInput(): #user input with cluases for bad input
    prompt = "Enter a value for wavelength in nanometers:\n"
    wave_l =input(prompt)
    try: #attempt to parse values to a float
        wave_l = float(wave_l)
        return wave_l

    except ValueError: #bad values or an inforseen error occured
        print("\aYou may have entered an invalid value please try again!")
        return newInput()
#==============================================================================

#==============================================================================
def getFreq(energy): #get frequecny form energy
    if energy == "UNDEFINED": #undefined for zero wavelength
        freq = "UNDEFINED"
    else:
        energy = energy * EV
        freq = energy / PLANCK
    return freq
#==============================================================================

#==============================================================================
def getEnergy(wave_l): #calculate energy based off wavelength
    if wave_l <= 0: #check if undefinedwavelength
        energy = "UNDEFINED"
    else:
        energy = (LIGHT * PLANCK) / wave_l
        energy = energy / EV
    return energy
#==============================================================================

#==============================================================================
def printing(head1,head2,head3): #print formatting

    try:
        print("{0:g>27.3e}".format((head1)),'\\',\
              "{0:g>27.3e}".format((head2)),'\\',\
              "{0:g>27.3e}".format((head3)))
    except:
        print("{0:g>27}".format((head1)),'\\',\
              "{0:g>27}".format((head2)),'\\',\
              "{0:g>27}".format((head3)))

#==============================================================================

#==============================================================================
def findMin(a,b):
    minimum = 0
    difference = a - b #posative if a > b, negative if b > a

    if difference > 0:
        bMin = b - minimum #posative if b > minimum negative if minimum > b
        if 0 > bMin:
            minimum = b

    else:
        aMin = a - minimum
        if 0 > aMin:
            minimum = a

    return minimum

#==============================================================================

#==============================================================================
def findMax(a,b):
    maximum = 360
    difference = a - b #posative if a > b, negative if b > a

    if difference < 0:
        bMax = b - maximum #posative if b > minimum negative if minimum > b
        if 0 < bMax:
            maximum = b

    else:
        aMax = a - maximum
        if 0 < aMax:
            maximum = a

    return maximum

#==============================================================================

#==============================================================================
def main():
    energy = []
    wave = []
    freq = []
    a = newInput()
    b = newInput()
    #get mac and min values
    mini = findMin(a,b)
    maxi = findMax(a,b)
    step = 30

    printing("Wavelength (nm)","Energy eV","Frequency(Hz)")

    for i in arange(mini,maxi + step,step):
        #get energy and frequency
        currentEnergy = getEnergy(i)
        currentFrequency = getFreq(currentEnergy)

        #append calcualted values to relivent lists
        wave.append(i)
        energy.append(currentEnergy)
        freq.append(currentFrequency)

    #print all data
    for i in range(0,len(energy)):
        printing(wave[i],energy[i],freq[i])

#==============================================================================

if __name__ == "__main__":
    #inform user and go to main method
    informUser()
    main()
```