Developed by Jonathan Scheffe
7/7/2020, University of Florida

Extraterrestrial Solar Radiation and Introduction to Python Data Analysis and Visualization

Extraterrestrial Solar Radiation

Solar radiation flux, or irradiance (G) in W/m^2, outside the earth's atmosphere and normal to earth's surface (G_{on}, where o stands for outside and n for normal direction) may be calculated as a function of the day in the year according to the following equation below.

$$G_{on} = G_{sc}(1 + 0.033\cos\frac{360n}{365})$$

G_{sc} is the solar constant and equal to 1367 W/m^2 and n is the day in the year. The term contained within cosine is in units degrees.

Brief Python Introduction

Packages and Calling Functions: In Python, there are several powerful computing packages commonly used that should be declared at the top of your .py, or python file. For example, some of the more popular computational packages useful for engineering include *numpy*, *scipy*, *matplotlib* and *pandas*, among others. These must be installed on your computer, either manually using a package installer (e.g. pip installer https://pip.pypa.io/en/stable/), or are installed by default with many Python distributions, including the Anaconda Distribution that we will use in this exercise (see link in Question 1). For example, if we want to utilize the *numpy* package we can use the *import* function and say

import numpy as np

which allows us to call the *numpy* package simply by typing *np*. If we wish to use numpy package to calculate the cosine of an angle, we would first need to call it by typing *np* followed by a period and then the function contained within the package we wish to utilize, in this case *cos*, and the value within it (in this case the numerical value pi which is also called using the numpy package as *np.pi*) contained within parenthesis. For example,

np.cos(np.pi)

The input contained within parenthesis should have units radians and the output will be in units radians. For example, to correctly use the equation for G_{on}, the cosine term will look like the following

*np.cos(360*n/365*np.pi/180)*

because we need to convert from degrees to radians by multiplying everything by np.pi/180.

Declaring Functions: Functions are declared in Python by defining them using *def* followed by the function name, parenthesis that include any variables the function is dependent on, followed by a colin. The body of the function is then started on a new, indented line and the value that is returned by the

function must be declared at the end using *return*. For example, if we wish to define a function y that depends on x and is a linear line with slope m and intercept y we would write

```
def y(x):
    return m*x+b
```

and to evaluate the function we would simply call it by typing

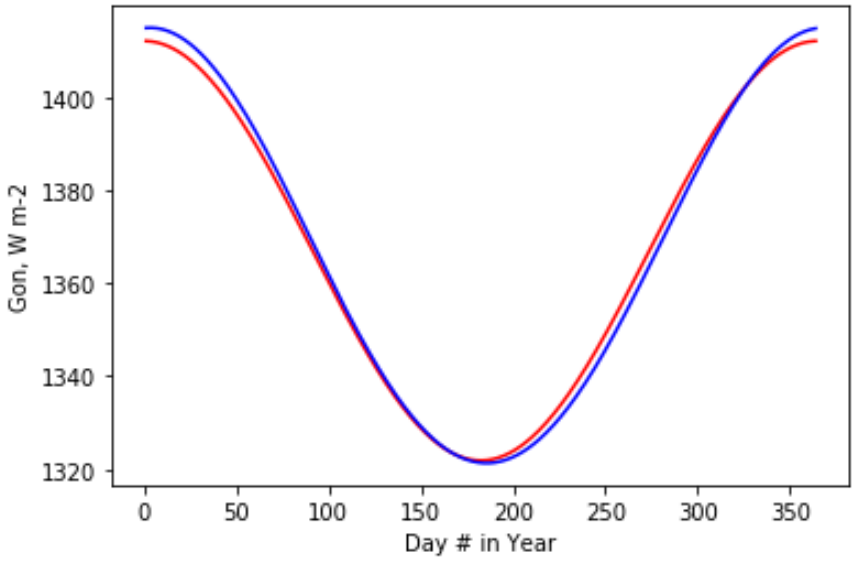
```
y(x)
```

Exercise

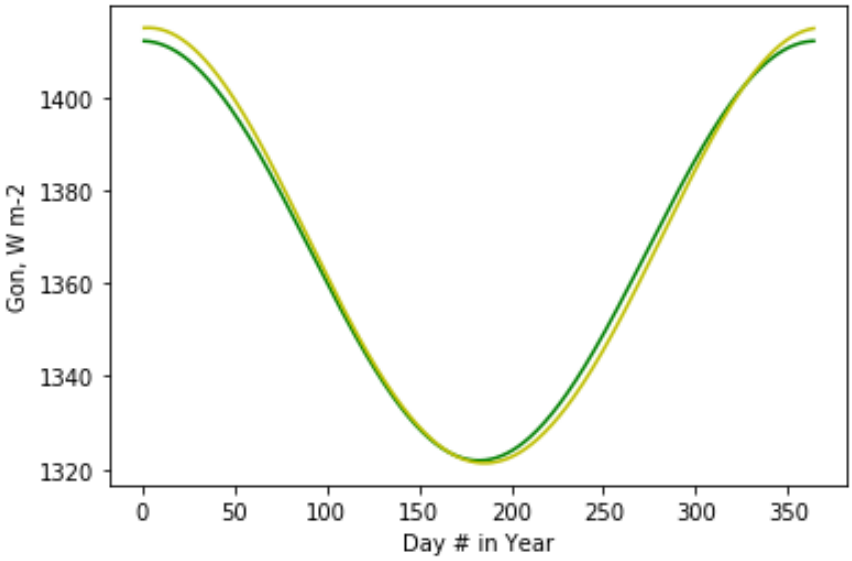
Complete the following questions that deal with determining G_{on} using the above equation and a more accurate form of the equation (not shown, *Gon_accurate* function), as a way to get comfortable computing in Python using some basic techniques, learning to call and declare functions, plotting data using the package *matplotlib*, and using two different methods to evaluate python code, namely a traditional editor named Spyder and a web based notebook style editor called Jupyter.

1) Download Python from https://www.anaconda.com/ ("Anaconda Distribution"). This includes many useful packages but most commonly we will use Spyder and Jupyter, two different ways to evaluate Python code.

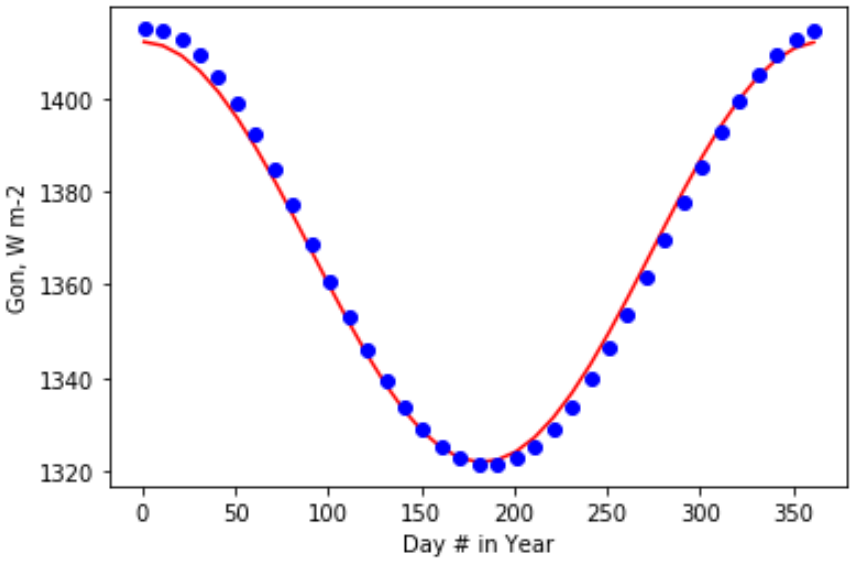
2) Open and run "Extraterrestrial Radiation Calculator.py" using Spyder. This can be downloaded as a pdf file from the website https://faculty.eng.ufl.edu/jonathan-scheffe/solar-energy-educational-resources/, then copied and pasted into the Spyder editor and saved to your computer as a .py file. Alternatively, it can be copied and pasted from the appendix of this document and saved to your computer as a .py file. After running, you should see a plot with two curves.



3) Replot the data using the same code but change the plotted lines from red ‘r’ and blue ‘b’ in the *plt.plot* lines of code, to other colors of your choice. Here are examples and more documentation- https://matplotlib.org/2.0.2/api/colors_api.html https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html



4) We have plotted data every day using the function np.arrange with the line *days=np.arange(1,365,1).* This creates an array from 1 to 365 with a spacing of 1. However, lets imagine we want to see individual data points plotted – it would be difficult to distinguish between them with so many. Therefore, lets plot a point every 10 days by changing the spacing from 1 to 10. Show the results using blue circles for the blue curve by inserting ‘bo’ in place of ‘b’.



5) Use Python Jupyter Notebook to the plot the extraterrestrial solar radiation (G_{on}) between March 1st and July 1st. Copy and paste lines from the prior "Extraterrestrial Radiation Calculator.py.py file". You can group common lines of code in their own cells that can be evaluated independently from the remaining cells. At the end of that file are a few lines of code that you can use to determine the day in the year knowing a date. Adding a # sign at the beginning of each line is used to comment and is not evaluated. Copy and paste these lines into a new Jupyter cell to determine what days the required dates correspond to.

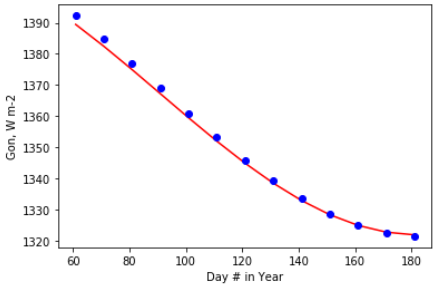
For example, to determine the day in the year on March 1st see below. Thus the day on March 1st is 61 and July 1st is 183.

```
In [5]: #To get day in a year
today = datetime.datetime.now()
day_of_year = (datetime.datetime(today.year, 3, 1) - datetime.datetime(today.year, 1, 1)).days + 1 #Enter month (1 through 12) ar
print(day_of_year)

61
```

Now we will plot between only these days by changing days to equal *days=np.arange(61,183,1).* Below is a screenshot of the cells and output.

```
In [8]: Gsc = 1367 #W m-2
def Gon(n):
    return Gsc*(1+0.033*np.cos(360*n/365*np.pi/180))

#print(Gon(10))

days=np.arange(61,183,10)
#print(days)
#print(Gon(days))

def B(n):
    return (n-1)*360/365*np.pi/180

def Gon_accurate(n):
    return Gsc*(1.000110+0.034221*np.cos(B(n))+.001280*np.sin(B(n))+.000719*np.cos(2*B(n))+.000077*np.sin(2*B(n)))
```

```
In [9]: plt.plot(days, Gon(days),'r')
plt.plot(days, Gon_accurate(days),'bo')
plt.xlabel('Day # in Year')
plt.ylabel('Gon, W m-2')
plt.show()#needed to display plot
```



Appendix

```
# Jonathan Scheffe - Extraterrestrial Solar Radiation Calculator 20200103

#from datetime import datetime,date,time,timedelta
import datetime
# import time
import math
#from geopy.geocoders import Nominatim
import matplotlib.pyplot as plt
import numpy as np



Gsc = 1367 #W m-2
def Gon(n):
   return Gsc*(1+0.033*np.cos(360*n/365*np.pi/180))

#print(Gon(10))

days=np.arange(1,365,1)
#print(days)
#print(Gon(days))

def B(n):
   return (n-1)*360/365*np.pi/180

def Gon_accurate(n):
   return
Gsc*(1.000110+0.034221*np.cos(B(n))+.001280*np.sin(B(n))+.000719*np.cos(2*B(n))+.000077*np.sin(
2*B(n)))


plt.plot(days, Gon(days),'r')
plt.plot(days, Gon_accurate(days),'b')
plt.xlabel('Day # in Year')
plt.ylabel('Gon, W m-2')
plt.show()#needed to display plot


#To get day in a year
today = datetime.datetime.now()
day_of_year = (datetime.datetime(today.year, 8, 6) - datetime.datetime(today.year, 1, 1)).days + 1
#Enter month (1 through 12) and day in month on LHS
#print(day_of_year)
```