Homework 11 - Debugging

Instructions: Fix the errors in the following problems. Some of the problems are with the code syntax, causing an error message. Other errors are logical mistakes in the programming that don't give a syntax error but do give an answer that is wrong.

Problem 1 -- Conditionals

(a) x is given below. Write an if statement that will print whether x is negative, postive, or zero.

In [1]:
```
x = 2.2
```

In [2]:
```
if x<0
    print('negative')
elif x>0
    print('positive')
else
    print('zero')
```

```
  File "<ipython-input-2-4579130a5f89>", line 1
    if x<0
         ^
SyntaxError: invalid syntax
```

(b) x and y are given below. Write an if statement that will print whether x and y are not negative, or both negative, or if one or both are zero.

In [ ]:
```
x = 2.2        # or 0.0
y = -6.6       # or +6.6, or 0.0, etc.
```

In [ ]:
```
negative = -1
if x>0 and y>0:
    print(not negative)
elif x<0 and y<0:
    print('both negative')
elif x==0 or y==0:
    print('one is zero')
elif x==0 and y==0:
    print('both are zero')
else:
    print('one if positive, one is negative')
```

(c) Write a single line (condensed) if statement that will evaluate $y = 1/x$ if x is not zero. If x is 0, then y=0.

In [ ]:
```
y = 1.0/x if x!=0: else: 0
print (x)
print (y)
```

Problem 2 -- Loops

(a) Write a for loop that iterates 5 times and prints the iteration value to the screen (that is, on the first pass it should print 0, the second pass should print 1, etc.

In [ ]:
```
for i in range(4):
    print i
```

(b) Write a for loop that will compute the sum of the first 10 numbers in the sequence: 0.1, 0.2, 0.3, 0.4...

In [ ]:
```
import numpy as np

# method 1 - linspace
x = np.linspace(0.0,1.0,10)
sum = 0
for i in x:
    sum = sum + i
print(sum)

# method 2 - arange
x = np.arange(0.1,1.1,0.1)
for i in x:
    sum += i # shortened sum=sum+i
print(sum)

# method 3 - np.sum instead of loop
x = np.linspace(0.0,1.0,10)
print(np.sum(x))
```

(c) Repeat part (b), but exclude value 0.5 from the sum. Use a "continue" statement.

In [ ]:
```
x = np.linspace(0.1,1.0,10)
sum = 0
for i in x:
    if i != 0.5:
        sum = sum + i
print(sum)
```

Problem 3 -- Functions

(a) Write a function called "sum" that takes two numbers and returns the sum of them.

In [ ]:
```
def mySum(a,b):
    c = a + b
    return c

print(mySum([2,3.75]))
```

(b) Write a function called "quad" that takes three numbers a, b, and c as function arguments that are the coefficients of the quadratic formula $ax^2 + bx + c = 0$, and returns the two roots. Test your function on $a = 1$, $b = -1$, $c = -2$. You should get roots of 2 and -1.

In [ ]:
```
def quad(a,b,c):
    x1 = (-b+pow(b**2-4.0*a*c,0.5))/(2.0*a)
    x2 = (-b-pow(b**2-4.0*a*c,0.5))/(2.0*a)
    return x1, x2

x1, x2 = quad(1,-1,-2)
print('Root 1: ' x1)
print('Root 2: ' x2)
```

Problem 4 -- Arrays

(a) Type "import numpy as np" to get the array functionality. Memorize this line.

In [ ]:
```
import numpie as np
```

(b) Use np.linspace(start, end, npoints) to create a list of 20 uniformly spaced points between 0 and 3.14. Call the array "x".

In [ ]:
```
x = np.linearspace(0,3.14,20)
print(x)
```

(c) Use a for loop to add up all of the elements of the the array x in the last part. You will need to initialize a variable to hold the sum, like "s=0". Then add up all the entries. Use the "np.size(x)" function in the for loop when determining how many elements there are in the array. Something like: for i in range(np.size(x)) :

In [ ]:
```
# method 1
s = 0
for i in range(np.size(x)):
    s = x[i]
print(s)

# method 2
s = 0
for i in x
    s = s + i
print(s)
```

(d) Repeat part (c), but this time use numpy's "sum" function: np.sum(x).

In [ ]:
```
# method 3
s = np.mean(x)
print(s)
```

(e) Find the square of all elements of x.

In [ ]:
```
print(x^2)
```

(f) For the array y below, find the array dy, whose elements are the difference in neighboring y elements. That is, dy[0] is y[1]-y[0]; dy[1] is y[2]-y[1], etc. Use array indexing, not a loop. Recall that y[1:] will give an array that has the second to the last element of y. y[0:-1] will have elements from the beginning to the second to the last element of y.

In [ ]:
```
y = np.array([0, 1, 1, 2, 3, 5, 8, 12])
print('y[1:]')
print(y[1,:])
print('y[0:-1]')
print(y[0:-1])
print('dy')
dy = y[1:] - y[0:-1]
print(dy)
```

(g) For the arrays y and x below, find the dot product by multiplying the two arrays and using the "np.sum" function. Do this in one combined command on a single line.

In [ ]:
```
x = np.array([0, 1, 2])
y = np.array([55, -20, 1.1])
print(x * np.sum(y))
```

(h) Create an array of 20 zeros using the "np.zeros" function. Also, create and array of 20 values where each value is 298.15.

In [ ]:
```
x0 = np.zeros(20)
x1 = np.zeros(20)*298.15
print(x1)
```

Problem 5 -- Plotting

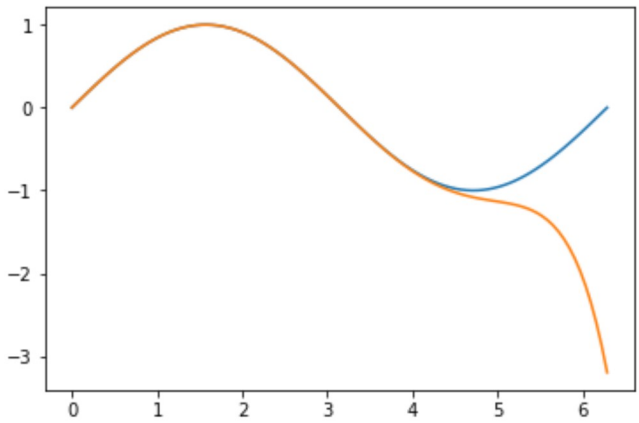
In [3]:
```
# import plotting module
import matplotlib.pyplot as plt
import numpy as np
from math import factorial
%matplotlib inline

x = np.linspace(0,2*np.pi,100)

# exact
y1 = np.sin(x)
# approximate
y2 = x - x**3/factorial(3) \
       + x**5/factorial(5) \
       - x**7/factorial(7) \
       + x**9/factorial(9) \
       - x**11/factorial(11)

plt.plot(x,y1)
plt.plot(x,y2)
```

Out[3]: [<matplotlib.lines.Line2D at 0x1720771e048>]



There are no bugs here, just enhance the plot according to the instructions below.

(a) Add x and y labels to the plot given as a sample above. Use "sin(x)" to label the y-axis and "x" for the x-axis.

(b) Change the symbols to be squares, and the line color to be red.

(c) Add a legend to the plot. Label the curves 'approximate' and 'exact' by including ", label='exact'" in the plot command. Then call plt.legend() to show the legend.

(d) Change the y axis limits to -1.2 to 1.2 by doing:

```
plt.ylim([-1.2,1.2])
```

(e) For the x, y1, y2 data below. Create a 2x1 subplot that plots x,y1 in the top panel, and x,y2 in the lower panel. Use "subplot(nrows, ncols, which_plot)"

In [ ]:
```
x = np.linspace(0,20,100)
y1 = np.sin(x)*np.exp(-0.2*x)
y2 = np.cos(x)*np.exp(-0.2*x)
```

Problem 6 -- Debugging

Debug the following code with print statements or with a debugger that steps through the code one line at a time.

This script provides a problem statement and associated python script. However, various bugs have been introduced into the python script. Find and correct the bugs. There are at least 8 errors. In the end, the graphs should match the paper.

You are investigating a solar-powered high-altitude long-endurance aircraft. You need to know the solar energy flux, mean flux and total energy received over a 24 hour period.

Use the model in the attached paper (Equations 5-13, variable descriptions in Table 2). tao = .85 # the transmittance factor ISC = 1367 # W/m^2, intensity of the extraterrestrial normal solar radiation r0 = 149597890. # km, mean sun-earth distance epsilon = .0167 # eccentricity ratio of the earth dn = 196. # day number T = 24. # hr phi = .492 # radians, latitude of location

Plot Ps, Es and Pms and report the total amount of power received in 24 hours.

Model from: Xian-Zhong Gao, Zhong-Xi Hou, Zheng Guo, Jian-Xia Liu, Xiao-Qian Chen, Energy management strategy for solar-powered high-altitude long-endurance aircraft, Energy Conversion and Management, Volume 70, June 2013, Pages 20-30, ISSN 0196-8904 (http://www.sciencedirect.com/science/article/pii/S0196890413000277 (http://www.sciencedirect.com/science/article/pii/S0196890413000277))

In [ ]:
```
import numpy as np
import scipy.integrate as integrate
import matplotlib.pyplot as plot

def Ps(t):  #Power as a function of time
    tao = 0.85 # the transmittance factor
    ISC = 1.367 # kW/m^2, intensity of the extraterrestrial normal solar radiation
    r0 = 149597890. # km, mean sun-earth distance
    epsilon = 0.0167 # eccentricity ratio of the earth
    dn = 196. # day number
    phi = .492 # radians, latitude of location

    #formulas
    w = np.pi - (np.pi*t/12.0)
    delta = ((23.45*np.pi*np.sin(360.0*(284.+ dn)/365.0))/180.0)
    theta = (np.pi/2.) - (np.arccos(np.sin(phi)*np.sin(delta)) + np.cos(phi)*np.cos
(delta)*np.cos(w))
    alpha = 2.*np.pi*(dn-4.0)/365.0
    r = r0* (1.-(epsilon**2.))/(1.+(epsilon*np.cos(alpha)))
    I0n = ISC*((r0/r)**2)

    if (I0n * tao * np.sin(theta)) < 0
        solar_power = 0
    else
        solar_power = I0n * tao * np.sin(theta)

    return solar_power


N = 50 #Number of points to calculate for the graph
time = np.linspace(0,24, N)
#initialize arrays for graphs
P_s = np.zeros(N)
Es = np.zeros(N)
P_ms = np.ones(N)


for i in np.arange(np.size(time)):
    P_s[i] = Ps(i)

for i in range(N):
    Es[i] = integrate.quad(Ps,0,time[i])[0]

Pms = (integrate.quad(Ps,0,24))/24.
P_ms = Pms * P_ms

fig = plt.figure(1)
plt.plot(time,P_s,label='Solar power')
plt.plot(time,P_ms,label='Mean power')
plt.legend()

fig = plt.figure(2)
plt.plot(time,Es,lable='Total power received')
plt.legend(loc='lower right')
plt.show()

print('Total energy in a day: ', Es[-0], 'Wh/m^2')
```