Homework 11

As with prior homeworks, the lesson notes on Learning Suite contain lots of details that we went over quickly. Go back and study the examples (this is your free text book for this course!) in order to learn what is needed to complete the homework.

Part of programming is following recipes, but not all. You must be able to *synthesize* information, meaning take different parts, make modifications, and create a new thing. Another skill in programming is to compare existing code with its output and deduce how it works and how to modify it for your purposes.

In [1]:
```
# put needed modules here (see what is in the notes)
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Problem 1 (2 points)

Follow the procedure in the notes to solve the matrix equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ where

$$\mathbf{A} = \begin{bmatrix} 0.25 & 0 & 0.5 \\ -0.25 & 0.5 & 0 \\ 0.25 & 0.75 & 0.25 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

First you need to properly enter $\mathbf{A}$ and $\mathbf{b}$ into Python, then solve for $\mathbf{x}$. Pay close attention to how to enter rows and columns of the matrix. Then check your solution by matrix-multiplying $\mathbf{A}$ and $\mathbf{x}$ to prove that it is equal to $\mathbf{b}$. Another check: when you print out $\mathbf{x}$, the zero element should be -1.

In [2]:
```
A = np.array([[0.25, 0, 0.5],[-0.25, 0.5, 0],[0.25, 0.75, 0.25]])     # Define the arra
b = np.array([1, 0, 0])                                                # Insert the valu
x = np.linalg.solve(A,b)                                               # Use numpy to so
print("A  =")                                                           # Print out the
print(A)
print("b  =", b)
print("x  =", x)
print("Ax =", np.dot(A,x))
```

```
A  =
[[ 0.25  0.    0.5 ]
 [-0.25  0.5   0.  ]
 [ 0.25  0.75  0.25]]
b  = [1 0 0]
x  = [-1.  -0.5  2.5]
Ax = [1. 0. 0.]
```

Problem 2 (1 point)

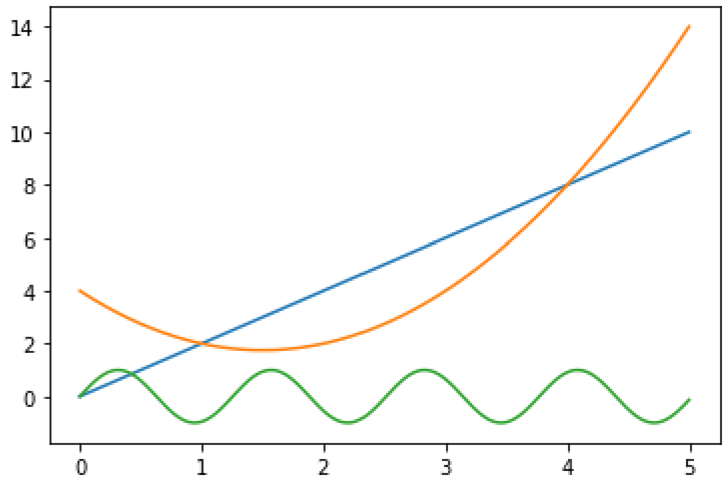
- Make an array of x from 0 to 5 with 1000 points, evenly spaced.
- create the following y-type arrays
  - $2x$
  - $x^2 - 3x + 4$
  - $\sin(5x)$
- Plot each y array versus x on a single plot. The plot doesn't yet have to be "pretty" (we'll learn to do that soon).

In [3]:

```
x = np.linspace(0,5,1000)
y1 = 2*x
y2 = x**2 - 3*x + 4
y3 = np.sin(5*x)


print("Plot of y = 2x, y = x^2 - 3x + 4, and y = sin(5x)")
plt.plot(x, y1)
plt.plot(x, y2)
plt.plot(x, y3)
```

Out[3]:

```
Plot of y = 2x, y = x^2 - 3x + 4, and y = sin(5x)
[<matplotlib.lines.Line2D at 0x7f77dfe7ab80>]
```



Problem 3 (2 points)

The ideal gas law can be expressed as $\rho = \frac{MP}{RT}$ where in this case

- $M = 29\ \mathrm{kg/kmol}$
- $R = 8314\ \mathrm{J/(kmol \cdot K)}$
- $P = 101325\ \mathrm{Pa}$

Find ρ for $T = 300, 350, \ldots 1000\ \mathrm{K}$

- Store ρ and T as arrays
- Plot ρ versus T

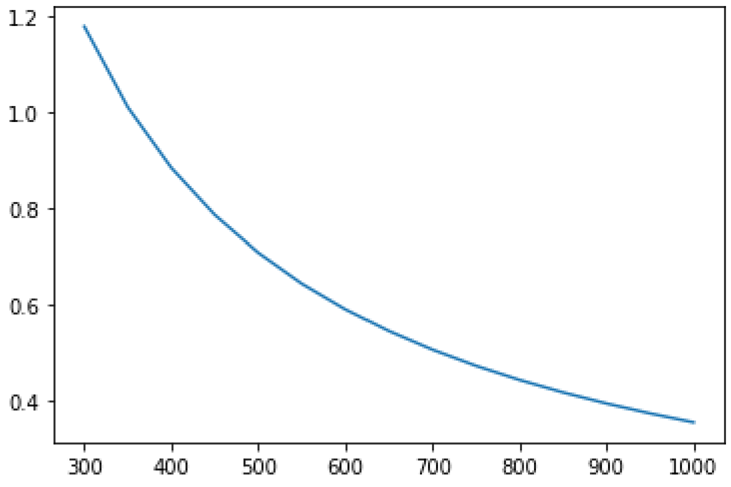
In [4]:

```
M = 29                          # kg/kmol
R = 8314                        # J/(kmol*K)
P = 101325                      # Pa
T = np.linspace(300,1000,15)    # K              | 300 K, 350 K, ... 1000 K

ρ = (M*P)/(R*T)                 # kg/m^3

print("Plot of the density according to the ideal gas law for the following constants:"
print("M =", M, "kg/kmol")
print("R =", R, "J/(kmol*K)")
print("P =", P, "Pa")
print("T = 300 K, 350 K, 400 K ... 1000 K")
plt.plot(T,ρ)
```

```
Plot of the density according to the ideal gas law for the following constants:
M = 29 kg/kmol
R = 8314 J/(kmol*K)
P = 101325 Pa
T = 300 K, 350 K, 400 K ... 1000 K
```

Out[4]:

```
[<matplotlib.lines.Line2D at 0x7f77dfd826d0>]
```



Problem 4 (3 points)

We have an array t that is storing a series of times (see below), and we want to compute the array of differences δt between adjacent times, i.e. $\delta t_i = t_{i+1} - t_i$

1. What is the length of array t?
2. What will be the length of array δt?
3. Create δt using a `for` loop or a nested array (choose one). Print your resulting array. (There is an example of a nested array in the notes.)
4. Create a different version of δt using slice-indexing and not a loop or nested array. Prove that you get the same (correct) result as in 3. above.

In [5]:

```
t = np.array([
    0.01136646,  0.01655847,  0.02548247,  0.07698612,  0.17964401,
    0.19813924,  0.22419994,  0.28622096,  0.28884997,  0.3627257 ,
    0.42555072,  0.44965767,  0.47257237,  0.59335946,  0.73842352,
    0.85900369,  0.88851226,  0.89153229,  0.89355469,  0.97705246,
    ])
```

```
#------------------------------------your code below here

# Print out the required lengths
print("Array t has", len(t), "elements")
print("δt will have", len(t)-1, "elements")

# initialise an empty vector and fill it
δt_looped = np.empty(len(t)-1)
for i in range(len(t)-1):
    δt_looped[i] = t[i+1] - t[i]
print("δt (for loop) =", δt_looped)

# initialise δt directly using two subsets of t.
δt_sliced = t[1:] - t[:-1]
print("δt (sliced) =", δt_sliced)
```

```
Array t has 20 elements
δt will have 19 elements
δt (for loop) = [0.00519201 0.008924   0.05150365 0.10265789 0.01849523 0.0260607
 0.06202102 0.00262901 0.07387573 0.06282502 0.02410695 0.0229147
 0.12078709 0.14506406 0.12058017 0.02950857 0.00302003 0.0020224
 0.08349777]
δt (sliced) = [0.00519201 0.008924   0.05150365 0.10265789 0.01849523 0.0260607
 0.06202102 0.00262901 0.07387573 0.06282502 0.02410695 0.0229147
 0.12078709 0.14506406 0.12058017 0.02950857 0.00302003 0.0020224
 0.08349777]
```

In [ ]: