Student Names and IDs:

- Divya Chowbey, 0710368
- Feroze Mohideen, 0707410
- Caroline Wang, 0707659

Problem 1.1 ¶

Spell out the system

$$A\mathbf{c} = \mathbf{a}$$

for this special case.

Consider:

$$A = \begin{bmatrix} 1 & x_0 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}, \text{and} \quad \mathbf{a} = \begin{bmatrix} y_0 \end{bmatrix}$$

Rearranging:

$$c_0 + c_1 x_0 = y_0$$

Problem 1.2

Give expressions for two different possible solutions $\mathbf{c}$ to this equation in terms of x_0 and y_0.

Possible solutions include:

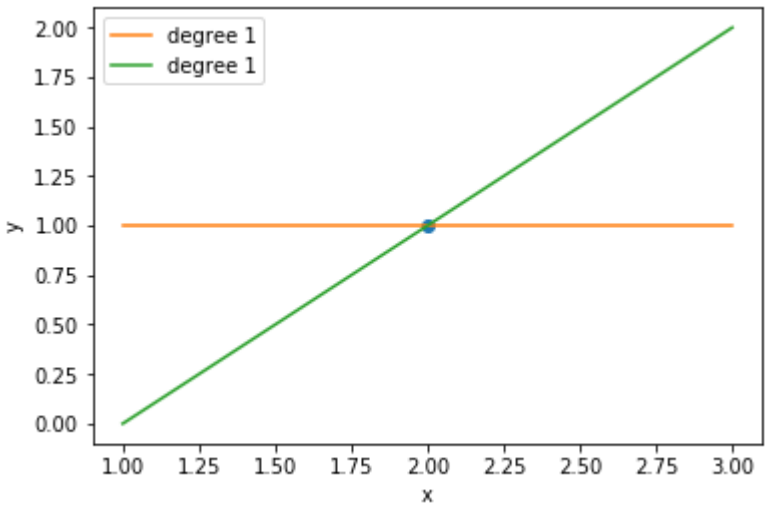
$$\mathbf{c} = \begin{bmatrix} y_0 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} y_0 - x_0 \\ 1 \end{bmatrix}$$

Problem 1.3

Write and run Python code to draw, in a single plot, the two lines corresponding to the two solutions you gave when

$$(x_0, y_0) = (2, 1).$$

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

def show(x, y, cList = []):
    plt.ion()
    plt.plot(x, y, marker='.', markersize=12, ls='')
    npt = 100
    xrange = [x - 1, x + 1] if x.size == 1 else [np.amin(x), np.amax(x)]
    xFine = np.linspace(xrange[0], xrange[1], npt)
    for c in cList:
        nc = c.size
        ycFine = np.zeros(xFine.shape)
        xPow = np.ones(xFine.shape)
        for i in range(nc):
            ycFine += c.item(i) * xPow
            xPow *= xFine
        plt.plot(xFine, ycFine, label = 'degree ' + str(nc-1))
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.show()

# Usage example
p = [2, 1]
show(np.array(p[0]), np.array(p[1]), [np.array([1, 0]), np.array([-1, 1])])
```



Problem 2.1

Derive this last equation from the system

$$A\mathbf{c} = \mathbf{a}$$

spelled out for this special case.

Consider: $A = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$, and $\mathbf{a} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$

Substituting:

$$\begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

Applying Gaussian Elimination:

$$\left[\begin{array}{cc|c} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & x_0 & y_0 \\ 0 & x_1 - x_0 & y_1 - y_0 \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & 0 & y_0 - \frac{x_0}{x_1 - x_0}(y_1 - y_0) \\ 0 & x_1 - x_0 & y_1 - y_0 \end{array}\right]$$

$$\left[\begin{array}{cc|c} 1 & 0 & y_0 - \frac{x_0}{x_1 - x_0}(y_1 - y_0) \\ 0 & 1 & \frac{y_1 - y_0}{x_1 - x_0} \end{array}\right]$$

Therefore,

$$c_0 = y_0 - \frac{x_0}{x_1 - x_0}(y_1 - y_0) \quad \text{and} \quad c_1 = \frac{y_1 - y_0}{x_1 - x_0}$$

Substituting c_0 and c_1 :

$$c_0 + c_1 x = y$$

$$y_0 - \left(\frac{x_0}{x_1 - x_0}\right)(y_1 - y_0) + \left(\frac{y_1 - y_0}{x_1 - x_0}\right) x = y$$

$$(x - x_0)\left(\frac{y_1 - y_0}{x_1 - x_0}\right) = y - y_0$$

Problem 3.1

Write the matrix A^TA and vector $A^T\mathbf{a}$ in terms of N, X, Y, S, P for the special case $k = 1$.

Consider: $A = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ 1 & x_2 \end{bmatrix}$ and $\mathbf{a} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$, where $A^T = \begin{bmatrix} 1 & 1 & 1 \\ x_0 & x_1 & x_2 \end{bmatrix}$

(i)

$$A^TA = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ x_0 & x_1 & x_2 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & x_0 + x_1 + x_2 \\ x_0 + x_1 + x_2 & x_0^2 + x_1^2 + x_2^2 \end{bmatrix}$$

$$= \begin{bmatrix} N & X \\ X & S \end{bmatrix}$$

(ii)

$$A^T\mathbf{a} = \begin{bmatrix} 1 & 1 & 1 \\ x_0 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} y_0 + y_1 + y_2 \\ x_0y_0 + x_1y_1 + x_2y_2 \end{bmatrix}$$

$$= \begin{bmatrix} Y \\ P \end{bmatrix}$$

Problem 3.2

Find expressions for c_0 and c_1 in terms of N, X, Y, S, P by solving the normal equations. Recall that

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$A^T A\mathbf{c} = A^T\mathbf{a}$$

$$\begin{bmatrix} N & X \\ X & S \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} Y \\ P \end{bmatrix}$$

$$\begin{bmatrix} N & X \\ X & S \end{bmatrix}^{-1} \begin{bmatrix} N & X \\ X & S \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} N & X \\ X & S \end{bmatrix}^{-1} \begin{bmatrix} Y \\ P \end{bmatrix}$$

Where:

$$\begin{bmatrix} N & X \\ X & S \end{bmatrix}^{-1} = \frac{1}{NS - X^2} \begin{bmatrix} S & -X \\ -X & N \end{bmatrix}$$

Therefore:

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \frac{1}{NS - X^2} \begin{bmatrix} S & -X \\ -X & N \end{bmatrix} \begin{bmatrix} Y \\ P \end{bmatrix}$$

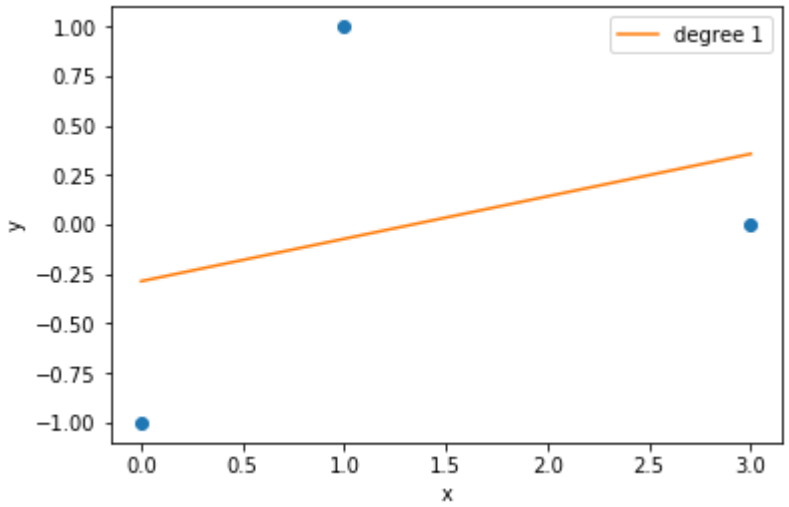
$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \frac{1}{NS - X^2} \begin{bmatrix} SY - PX \\ -XY + NP \end{bmatrix}$$

Problem 3.3

Use the function `show` given earlier to display the three points $(0, -1), (1, 1), (3, 0)$ and the line fit to them with the formulas you just found for c_0 and c_1.

In [2]:
```
x = [0, 1, 3]
y = [-1, 1, 0]

N = len(y)
S = sum([x[i]**2 for i in range(len(x))])
X = sum(x)
Y = sum(y)
P = sum([x[i]*y[i] for i in range(len(x))])

c_0 = (1/(N*S - X**2))*(S*Y- P*X)
c_1 = (1/(N*S - X**2))*(-X*Y+N*P)

print("c0: ", "{0:.3f}".format(c_0))
print("c1: ", "{0:.3f}".format(c_1))



show(np.array(x), np.array(y), [np.array([c_0, c_1])])
```

```
c0:  -0.286
c1:  0.214
```



Problem 4.1

In problem 1.3 you found two different solutions to a data fitting problem. Let us call the corresponding coefficient vectors $\mathbf{c}$ and $\mathbf{d}$. Write the values of $\mathbf{c}$ and $\mathbf{d}$, and report their norms. Which is smaller?

Recall:

$$\mathbf{c} = \begin{bmatrix} y_0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{d} = \begin{bmatrix} y_0 - x_0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

From code included below, the euclidean norm of $\mathbf{d}$ (1.414) was found to be smaller than that of $\mathbf{c}$ (2).

In [3]:
```
from numpy import linalg
c = [2, 0]
d = [-1, 1]

norm_c = linalg.norm(np.array(c))
norm_d = linalg.norm(np.array(d))
```

Problem 4.2

Write a function with header

`def fit(x, y, k):`

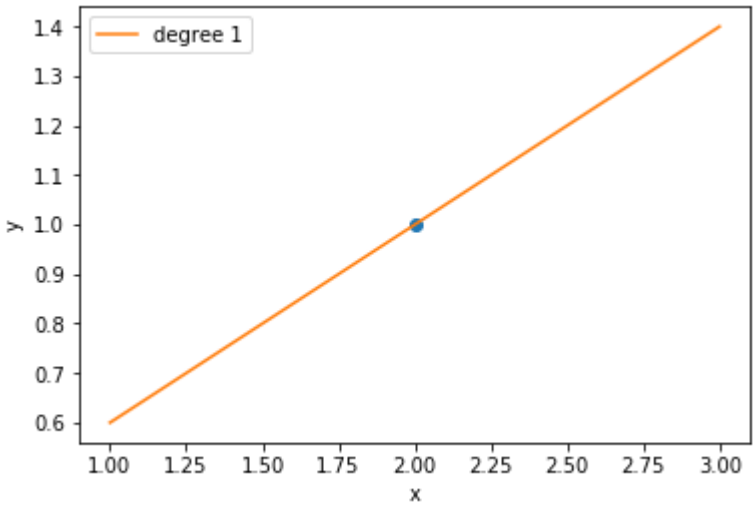
that fits a polynomial of degree `k` to the data in `x` and `y`.

In [4]:
```
import numpy as np
from numpy import linalg

def fit(x, y, k):
    N = y.size
    A = np.zeros((N,k+1))
    for i in range(N):
        for j in range(k+1):
            A[i,j]= x.item(i)**j
    y = np.reshape(np.matrix(y),[y.size,1])
    matA = np.matrix(A)
    c = linalg.lstsq(matA, y, rcond=None)[0]
    return c
```

Problem 4.3

> Show the plot and output obtained with the following code, where `fit` is your function. The file `T.txt` should be in the same directory as your notebook.

In [5]:
```
try:
    p13 = {'x' : np.array(2), 'y' : np.array(1), 'k' : [1]}

    np.set_printoptions(formatter={'float': '{: .3f}'.format})

    def run(p):
        x, y = p['x'], p['y']
        cList = [fit(x, y, k) for k in p['k']]
        show(x, y, cList)
        for c in cList:
            print(c.T)

    run(p13)
except NameError:
    pass
```




```
[[ 0.200  0.400]]
```

Problem 4.4

What is the approximate norm of the solution (to three decimal digits), and is this result consistent with your results in problem 1.3? Why or why not?

The norm of the found c value is is 0.447 (see code below). The norm of this vector is less than the norms of the vectors found earlier. This is expected because for an undetermined system, the `numpy.linalg.lstsq` function finds the coefficient vector with the lowest norm which still satisfies the solution found by the quadratic loss function.

In [6]:
```
c_fit= [0.200, 0.400]

norm_c = linalg.norm(np.array(c_fit))
print("{0:.3f}".format(norm_c))
```

```
0.447
```

Problem 4.5

> a. Show the plots and outputs obtained with the code below, and answer the following questions:
>
> b. Are the plot and value of $\mathbf{c}$ from the data in `p33` consistent with your answer to problem 3.3?

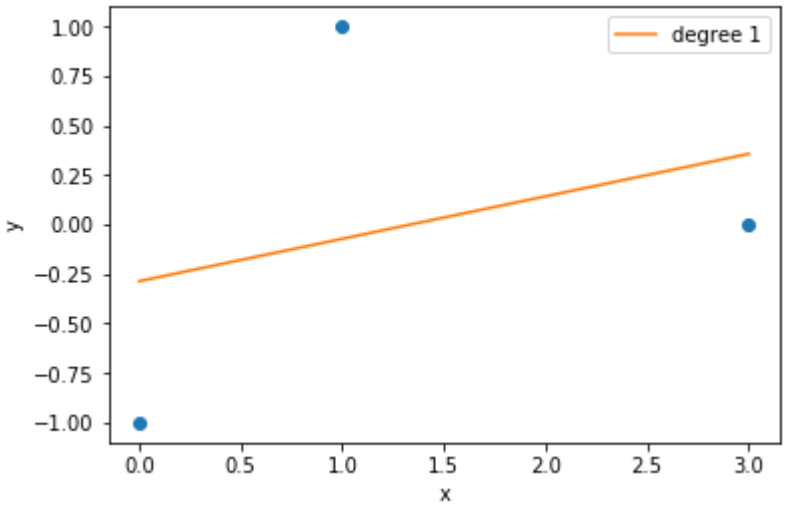
In problem 3.3, we computed $c_0 = -0.286$, $c_1 = 0.214$. From the data in `p33`, we got $[[-0.286, 0.214]]$. These two values for c are the same for two reasons: (1) Problem 3.3 and `p33` are both fitting a degree 1 polynomial to the dataset (0,-1),(1,1),(3,0). (2) The least-squares solution to the normal equations is equivalent to minimizing the quadratic loss function (what the `numpy.linalg.lstsq` function is doing).

>

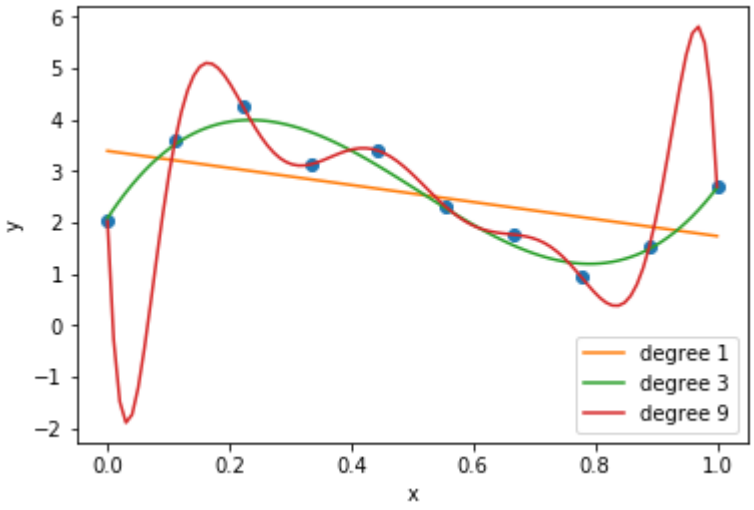
> c. Give a brief qualitative description of the way(s) in which the curves in the plot from the data in `notes` differ from the curves in Figure 1 of the class notes on data fitting (https://www2.cs.duke.edu/courses/fall18/compsci371d/notes/01_DataFitting.pdf). The plots in that Figure were computed from the same ten points as in `T.txt` but with a different numerical package.

Using the human eye, our curves are qualitatively indistinguishable from the curves in the notes.

In [7]:
```
try:
    p33 = {'x' : np.array([0, 1, 3]), 'y' : np.array([-1, 1, 0]), 'k' : [1]}

    x, y = np.loadtxt('T.txt', unpack=True)
    notes = {'x' :x, 'y' : y, 'k' : [1, 3, 9]}

    run(p33)
    run(notes)
except NameError:
    pass
```




```
[[-0.286  0.214]]
```




```
[[ 3.394 -1.655]]
[[ 2.089  18.049 -49.900  32.440]]
[[ 2.030 -292.804  7046.148 -62434.420  285853.947 -759114.988
   1214741.224 -1155128.463  601089.629 -131759.602]]
```

Problem 4.6 (Extra Credit, 5 Points)

> If you noticed any discrepancies between your plot and the Figure in the class notes, first state why the discrepancies should not occur under ideal circumstances. Then suggest some reasons for them. For your information, the coefficients found for $k = 9$ for the Figure in the class notes are as follows:
>
> $$2.030, -295.642, 7107.144, -62956.114, 288218.499, -765400.622, 1224862.633, -1164835.6$$

The reason why there are discrepancies is that `numpy.linalg.lstsq` may be implemented slightly differently in this HW versus in the class notes. Specifically, the implementation differences may lie in the optimization technique used to minimize the quadratic loss function, or any other functions may need to be minimized.