In [1]:
```
#### RUN ME PLEASE ####
"""Use 'Shift + Enter' when focused on this cell."""
import numpy as np
```

Python for DS Practice

Data Science for Kaggle Decal Spring 2017

Welcome to the first pset. This should be a really simple intro to numpy. We are structuring it to get you familiar with the tools we will be using for this class. Make sure that you run the first cell before any others otherwise you'll get errors for not including the numpy package.

If you have trouble with any question, first consult any old notebooks. Especially those from class today. Then, ask Piazza (https://piazza.com/class/iy117z6nv30626?cid=9)! We'll try to get back to you asap through there.

Problem 1

Fill `fun_array` with values 1-100 using a numpy method. Use google to find the answer.

In [5]:
```
fun_array = np.linspace(1, 100, 100)
fun_array
```

Out[5]:
```
array([   1.,    2.,    3.,    4.,    5.,    6.,    7.,    8.,    9.,
         10.,   11.,   12.,   13.,   14.,   15.,   16.,   17.,   18.,
         19.,   20.,   21.,   22.,   23.,   24.,   25.,   26.,   27.,
         28.,   29.,   30.,   31.,   32.,   33.,   34.,   35.,   36.,
         37.,   38.,   39.,   40.,   41.,   42.,   43.,   44.,   45.,
         46.,   47.,   48.,   49.,   50.,   51.,   52.,   53.,   54.,
         55.,   56.,   57.,   58.,   59.,   60.,   61.,   62.,   63.,
         64.,   65.,   66.,   67.,   68.,   69.,   70.,   71.,   72.,
         73.,   74.,   75.,   76.,   77.,   78.,   79.,   80.,   81.,
         82.,   83.,   84.,   85.,   86.,   87.,   88.,   89.,   90.,
         91.,   92.,   93.,   94.,   95.,   96.,   97.,   98.,   99.,
        100.])
```

Problem 2

Take the square root of each element in fun_array using numpy commands. Don't use a `for` loop

In [6]:
```
### YOUR CODE HERE
fun_array_sqrt = np.sqrt(fun_array)
fun_array_sqrt
```

Out[6]:
```
array([  1.        ,   1.41421356,   1.73205081,   2.        ,
         2.23606798,   2.44948974,   2.64575131,   2.82842712,
         3.        ,   3.16227766,   3.31662479,   3.46410162,
         3.60555128,   3.74165739,   3.87298335,   4.        ,
         4.12310563,   4.24264069,   4.35889894,   4.47213595,
         4.58257569,   4.69041576,   4.79583152,   4.89897949,
         5.        ,   5.09901951,   5.19615242,   5.29150262,
         5.38516481,   5.47722558,   5.56776436,   5.65685425,
         5.74456265,   5.83095189,   5.91607978,   6.        ,
         6.08276253,   6.164414  ,   6.244998  ,   6.32455532,
         6.40312424,   6.4807407 ,   6.55743852,   6.63324958,
         6.70820393,   6.78232998,   6.8556546 ,   6.92820323,
         7.        ,   7.07106781,   7.14142843,   7.21110255,
         7.28010989,   7.34846923,   7.41619849,   7.48331477,
         7.54983444,   7.61577311,   7.68114575,   7.74596669,
         7.81024968,   7.87400787,   7.93725393,   8.        ,
         8.06225775,   8.1240384 ,   8.18535277,   8.24621125,
         8.30662386,   8.36660027,   8.42614977,   8.48528137,
         8.54400375,   8.60232527,   8.66025404,   8.71779789,
         8.77496439,   8.83176087,   8.88819442,   8.94427191,
         9.        ,   9.05538514,   9.11043358,   9.16515139,
         9.21954446,   9.2736185 ,   9.32737905,   9.38083152,
         9.43398113,   9.48683298,   9.53939201,   9.59166305,
         9.64365076,   9.69535971,   9.74679434,   9.79795897,
         9.8488578 ,   9.89949494,   9.94987437,  10.        ])
```

Problem 3

Fill this array with 100 random values. Your answer should use only a single method call like `np.<method>()`

In [18]:
```
random_array = np.random.choice(100, 100)
random_array
```

Out[18]:
```
array([78, 53, 65, 62, 22,  2, 25, 65, 50, 18, 41,  7, 71,  3, 27, 71,
       84,
       13, 84, 22, 21, 79, 43, 57, 27, 65, 52, 63, 22, 63, 69, 33, 81,
       48,
       74,  7, 25, 72, 58, 58, 75, 36, 68, 67, 80, 40, 58, 31, 77, 35,
       9,
       20, 65, 79, 26, 93,  2, 14, 85, 50, 73, 45, 43, 69, 40, 56, 22,
       63,
       78, 17, 36, 38, 49, 85, 11, 44, 26, 29, 39, 58, 31, 73, 52, 13,
       58,
       74, 49, 27, 90, 42, 85, 63, 95, 33, 31, 98, 68, 23, 98, 54])
```

Problem 4

Multiply each element in `random_array` by `5`. Do not use a `for` loop.

In [19]:
```
random_arrayx5 = random_arrayx5 * 5
random_arrayx5
```

Out[19]:
```
array([ 490000, 1102500,  225625, 2402500,  140625, 5522500, 5522500,
        855625, 4622500, 3610000,  722500, 2030625, 4515625,  140625,
           625,  490000,   90000,       0,  160000, 1210000, 4000000,
       2030625, 2175625,  160000,  765625,  490000, 1000000,  160000,
        302500, 3240000, 1050625, 1690000, 6125625,  160000, 3422500,
       4410000,  902500, 2402500, 3240000, 5290000, 1380625,  330625,
       3150625,  600625,  360000, 6125625, 1822500,  950625, 2890000,
       4100625,  490000, 3062500, 2975625, 1380625,  202500, 4622500,
       5760000,  422500, 1380625,  360000, 1822500, 5640625, 4305625,
       3150625,  275625,  422500, 4950625, 1265625, 2030625, 6125625,
       5522500, 4950625,  490000, 3062500, 1562500, 4410000, 1050625,
        902500,  490000, 3802500,  302500, 2325625, 2325625, 1000000,
       1690000,  202500, 2030625, 1822500, 4202500,  422500,  390625,
        390625,  140625, 1155625, 1625625, 4410000, 1155625, 5405625,
        680625,   30625])
```

Problem 5

Multiply matrix X and multiply with $\vec{y}$ using numpy commands. Should be only 1 line.

In [43]:
```
X = np.array([[1,2,3],[4,5,6],[7,8,9]])
y = np.array([1,2,3])
## YOUR CODE HERE
np.dot(X, y)
```

Out[43]:
```
array([14, 32, 50])
```

Problem 6

Get the shape of `X` and output it. Do it with and without a print() statement for full credit.

In [32]:
```
print(X.shape)
X.shape
```

```
(3, 3)
```

Out[32]:
```
(3, 3)
```

Problem 7

Fill in the missing code so that you can plot

This is a very simple 2 line problem. You should read the `matplotlib` documentation (http://matplotlib.org/users/pyplot_tutorial.html) regardless of how familiar you are with matplotlib.

Alternatively, google how to plot a graph using matplotlib. You'll probably find an answer on stack overflow

I almost always copy and past matplotlib code whenever I need to use it.

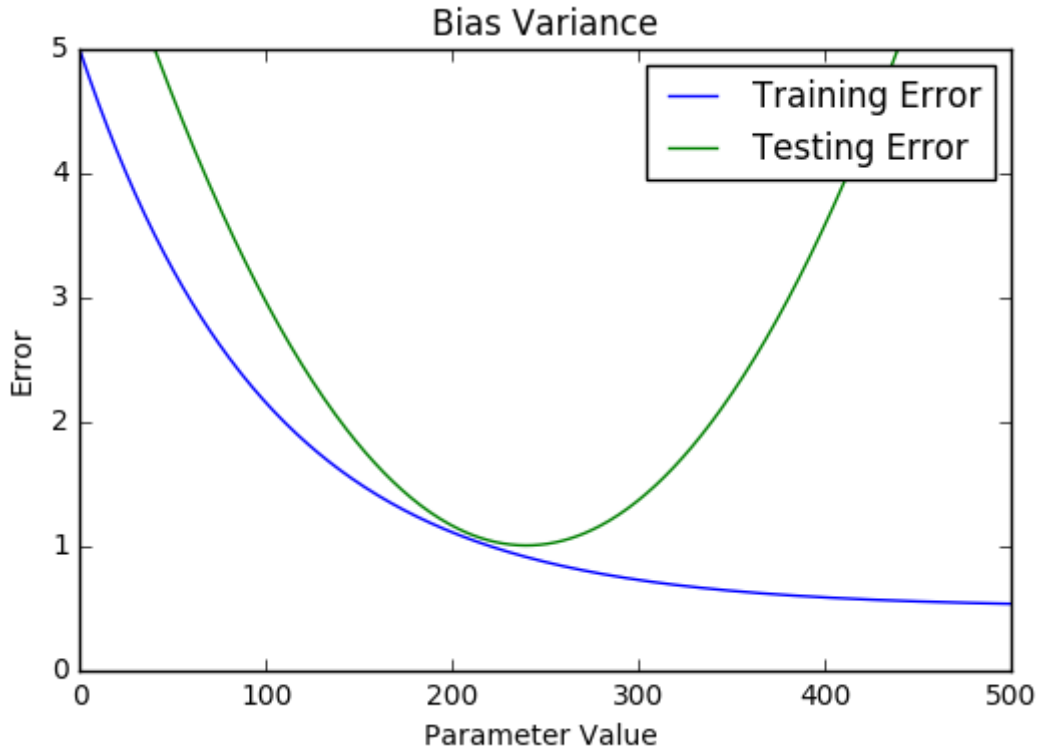
In [41]:
```
%matplotlib inline
import matplotlib.pyplot as plt
X = np.linspace(0, 5, 500) #np.arange(500)

# These are just to represent actual training and testing
training_accuracy = np.exp(-(X)) * np.exp(1.5) + 0.5
testing_accuracy = (X - 2.4)**2 + 1

### YOUR CODE BELOW ###
plt.plot(training_accuracy)
plt.plot(testing_accuracy)
### END YOUR CODE ###

plt.title("Bias Variance")
plt.xlabel("Parameter Value ")
plt.ylabel("Error")
plt.legend(labels=['Training Error','Testing Error'])
axes = plt.gca()
axes.set_ylim([0,5])
```

Out[41]: (0, 5)



HOMEWORK SUBMISSION

We will be submitting work through gradescope. You must make your submission in pdf format so that we can easily process them using gradescope question matching.

Saving the file as a pdf

In the menu bar, you simply follow `File > Print Preview` Then print the webpage and Save as PDF. This differs per browser/OS so you may have to spend some time looking for this option to save.

Submitting

Go to gradescope.com (https://gradescope.com/), and click on the box that says `Add a course`. Type in `9KPPBM` to the course enrollment spot and you should be good to go. If you are not enrolled in the course, we will remove you from gradescope so please only enroll if we have sent you a permission numer.

When you are in the course, navigate to `Problem Set 1`. On the submission page, select upload and upload the pdf you just downloaded. You will be required to label your questions appropriately.

In [ ]: