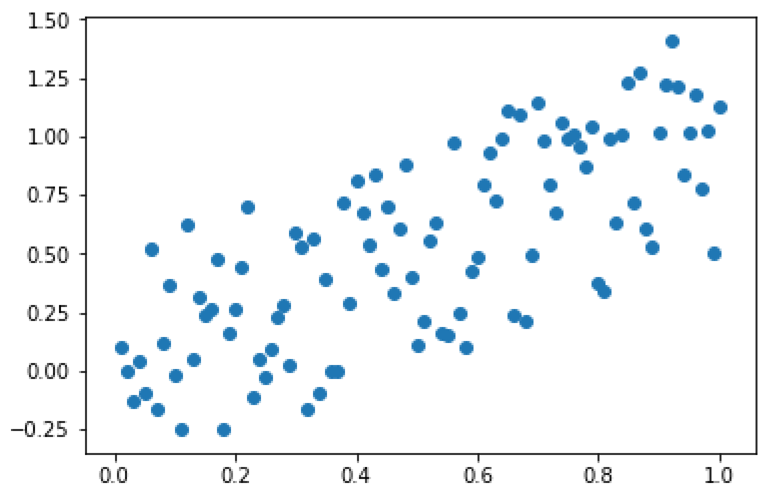
In [1]:
```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
from sklearn import linear_model
```

In [2]:
```
##I reduced the n to 100, it was at 10000..
n = 100
x = np.linspace(0.01, 1, n).reshape(-1, 1)
y = np.linspace(0.01, 1, n)  + np.random.rand(n) - .5


plt.scatter(x,y)
```

Out[2]: <matplotlib.collections.PathCollection at 0x2026ef89f60>



Assignment 5

1. Create and fit a Linear Regression Model

Calculate the Training error and Testing error using sklearn with a .50 split

For error, use `mean_squared` , but if you want to experiment with other mean errors, please do!

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

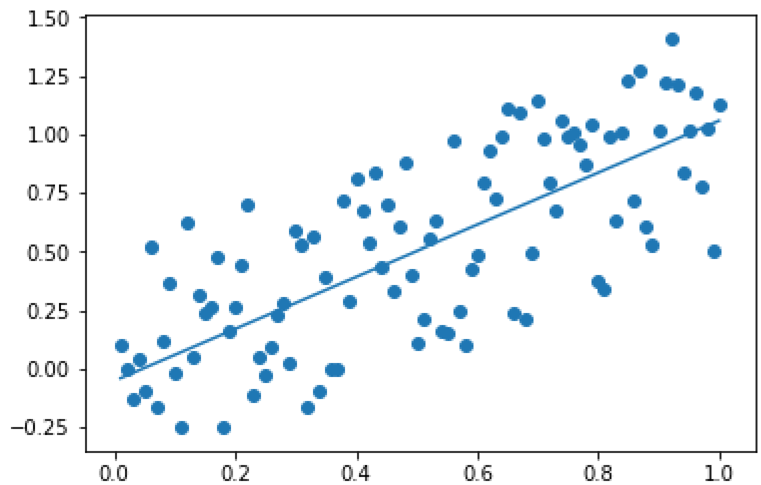
In [4]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.5)
```

In [5]:
```
model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_
```

Out[5]:
```
(array([1.10880687]), -0.05212779300958259)
```

In [6]:
```
plt.scatter(x,y)
plt.plot(x, np.dot(x, model.coef_) + model.intercept_)
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x2026efd2828>]
```



In [7]:
```
mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept_)
```

Out[7]:
```
0.09657465499780817
```

In [8]:
```
mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.intercept_)
```

Out[8]:
```
0.06369368500846505
```

2. Repeat #1 for a Ridge Regression

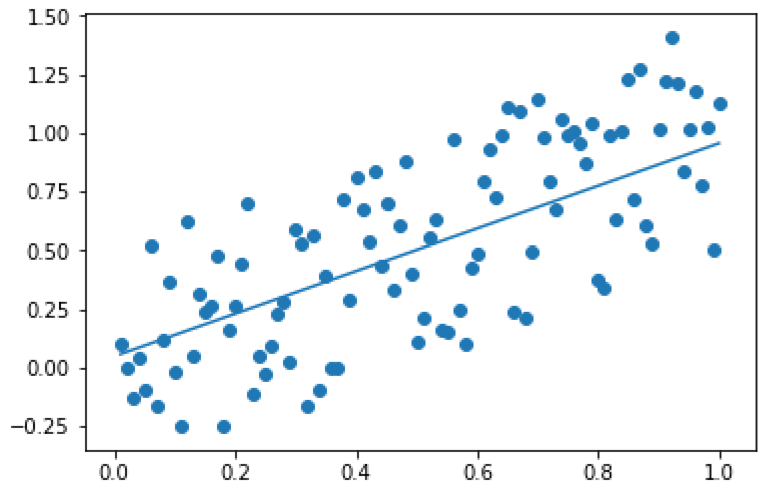
In [9]:
```
from sklearn.linear_model import Ridge
```

In [10]:
```
model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_
```

Out[10]:
```
(array([0.90663284]), 0.048110089318745664)
```

In [11]:
```
plt.scatter(x,y)
plt.plot(x, np.dot(x, model.coef_) + model.intercept_)
```

Out[11]: [<matplotlib.lines.Line2D at 0x2026f071da0>]



In [12]:
```
mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept_)
```

Out[12]: 0.09599631805549995

In [13]:
```
mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.intercept_)
```

Out[13]: 0.06735963725853651

3. Vary the `split` size from .01 to .99 with at least 10 values (the more the merrier!). Plot the resulting Training error and Testing error vs. `split` size. Create separate plots for Linear and Ridge

In [14]:
```
#Linear 0.1
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.1)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.1 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.1 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_1 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_1 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_1 = Test_1 - Train_1
print('Test_Size_1', Test_Size_1)
```

```
Test_0.1 = 0.030789342029746293
Train_0.1 = 0.08503332214122246
Test_Size_1 -0.0542439801114783
```

In [15]:
```
#Linear 0.2
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.2 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.2 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_2 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_2 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_2 = Test_2 - Train_2
print('Test_Size_2', Test_Size_2)
```

```
Test_0.2 = 0.09575687996205615
Train_0.2 = 0.07562833654073225
Test_Size_2 0.02012854342132389
```

In [16]:
```
#Linear 0.3
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.3 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.3 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_3 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_3 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_3 = Test_3 - Train_3
print('Test_Size_3', Test_Size_3)
```

```
Test_0.3 = 0.06921922633980455
Train_0.3 = 0.08477958252803139
Test_Size_3 -0.015560356188226834
```

In [17]:
```
#Linear 0.4
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.4)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.4 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.4 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_4 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_4 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_4 = Test_4 - Train_4
print('Test_Size_4', Test_Size_4)
```

```
Test_0.4 = 0.08474927061595613
Train_0.4 = 0.07691377913683214
Test_Size_4 0.00783549147912399
```

In [18]:
```
#Linear 0.5
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.5)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.5 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.5 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_5 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_5 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_5 = Test_5 - Train_5
print('Test_Size_5', Test_Size_5)
```

```
Test_0.5 = 0.07621652594736558
Train_0.5 = 0.08452108109340735
Test_Size_5 -0.008304555146041767
```

In [19]:
```
#Linear 0.6
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.6)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.6 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.6 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_6 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_6 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_6 = Test_6 - Train_6
print('Test_Size_6', Test_Size_6)
```

```
Test_0.6 = 0.08351697615522947
Train_0.6 = 0.07559412569796028
Test_Size_6 0.00792285045726919
```

In [20]:
```
#Linear 0.7
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.7)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.7 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.7 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_7 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_7 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_7 = Test_7 - Train_7
print('Test_Size_7', Test_Size_7)
```

```
Test_0.7 = 0.07928576910484776
Train_0.7 = 0.08122505505880089
Test_Size_7 -0.0019392859539531238
```

In [21]:
```
#Linear 0.8
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.8)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.8 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.8 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_8 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_8 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_8 = Test_8 - Train_8
print('Test_Size_8', Test_Size_8)
```

```
Test_0.8 = 0.07797733125503074
Train_0.8 = 0.09730161810230858
Test_Size_8 -0.01932428684727784
```

In [22]:
```
#Linear 0.9
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.7)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.9 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.9 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

Test_9 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercept
Train_9 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interce

Test_Size_9 = Test_9 - Train_9
print('Test_Size_9', Test_Size_9)
```

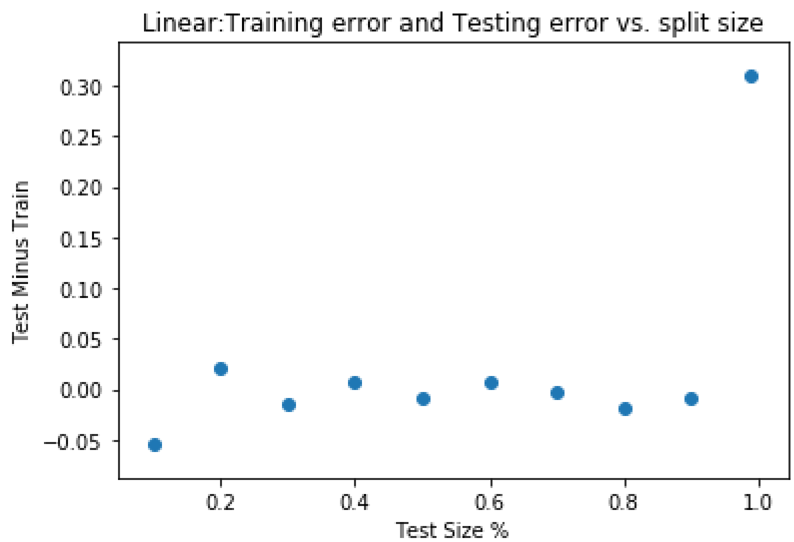
```
Test_0.9 = 0.07865698013102729
Train_0.9 = 0.08667527803337932
Test_Size_9 -0.008018297902352034
```

In [23]:
```
#Linear 0.99
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.99)

model = LinearRegression()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.99 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + mode

print('Train_0.99 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + m

Test_99 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
Train_99 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

Test_Size_99 = Test_99 - Train_99
print('Test_Size_99', Test_Size_99)
```

```
Test_0.99 = 0.3098290785076458
Train_0.99 = 0.0
Test_Size_99 0.3098290785076458
```

In [24]:
```
## Plot of Linear
x1 = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.99]
y1 = [Test_Size_1,Test_Size_2,Test_Size_3,Test_Size_4,Test_Size_5,Test_Size_6,Test

plt.scatter(x1, y1, label='Test Minus Train')
plt.xlabel('Test Size %')
plt.ylabel('Test Minus Train')
plt.title('Linear:Training error and Testing error vs. split size')
```

Out[24]:
```
Text(0.5,1,'Linear:Training error and Testing error vs. split size')
```



In [25]:
```
#Ridge 0.1
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.1)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.1 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.1 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_1 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_1 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_1 = RTest_1 - RTrain_1
print('RTest_Size_1', RTest_Size_1)
```

```
Test_0.1 = 0.07465465661078137
Train_0.1 = 0.0818021804472812
RTest_Size_1 -0.007147523836499831
```

In [26]:
```
#Ridge 0.2
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.2 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.2 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_2 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_2 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_2 = RTest_2 - RTrain_2
print('RTest_Size_2', RTest_Size_2)
```

```
Test_0.2 = 0.0924151000879339
Train_0.2 = 0.08159764228417891
RTest_Size_2 0.010817457803754982
```

In [27]:
```
#Ridge 0.3
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.3 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.3 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_3 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_3 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_3 = RTest_3 - RTrain_3
print('RTest_Size_3', RTest_Size_3)
```

```
Test_0.3 = 0.07904326971692757
Train_0.3 = 0.08209751918796543
RTest_Size_3 -0.0030542494710378565
```

In [28]:
```
#Ridge 0.4
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.4)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.4 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.4 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_4 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_4 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_4 = RTest_4 - RTrain_4
print('RTest_Size_4', RTest_Size_4)
```

```
Test_0.4 = 0.08244762028068427
Train_0.4 = 0.08004733631852812
RTest_Size_4 0.0024002839621561495
```

In [29]:
```
#Ridge 0.5
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.5)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.5 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.5 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_5 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_5 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_5 = RTest_5 - RTrain_5
print('RTest_Size_5', RTest_Size_5)
```

```
Test_0.5 = 0.08368808725593059
Train_0.5 = 0.07946011831081175
RTest_Size_5 0.004227968945118843
```

In [30]:
```
#Ridge 0.6
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.6)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.6 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.6 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_6 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_6 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_6 = RTest_6 - RTrain_6
print('RTest_Size_6', RTest_Size_6)
```

```
Test_0.6 = 0.08346571875804559
Train_0.6 = 0.07899290453105975
RTest_Size_6 0.004472814226985838
```

In [31]:
```
#Ridge 0.7
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.7)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.7 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.7 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_7 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_7 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_7 = RTest_7 - RTrain_7
print('RTest_Size_7', RTest_Size_7)
```

```
Test_0.7 = 0.099887436712691
Train_0.7 = 0.08130209765184913
RTest_Size_7 0.018585339060841866
```

In [32]:
```
#Ridge 0.8
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.8)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.8 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.8 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_8 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_8 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_8 = RTest_8 - RTrain_8
print('RTest_Size_8', RTest_Size_8)
```

```
Test_0.8 = 0.10612010358801602
Train_0.8 = 0.07891059020466709
RTest_Size_8 0.027209513383348927
```

In [33]:
```
#Ridge 0.9
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.9)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.9 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + model

print('Train_0.9 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + mo

RTest_9 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.intercep
RTrain_9 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.interc

RTest_Size_9 = RTest_9 - RTrain_9
print('RTest_Size_9', RTest_Size_9)
```

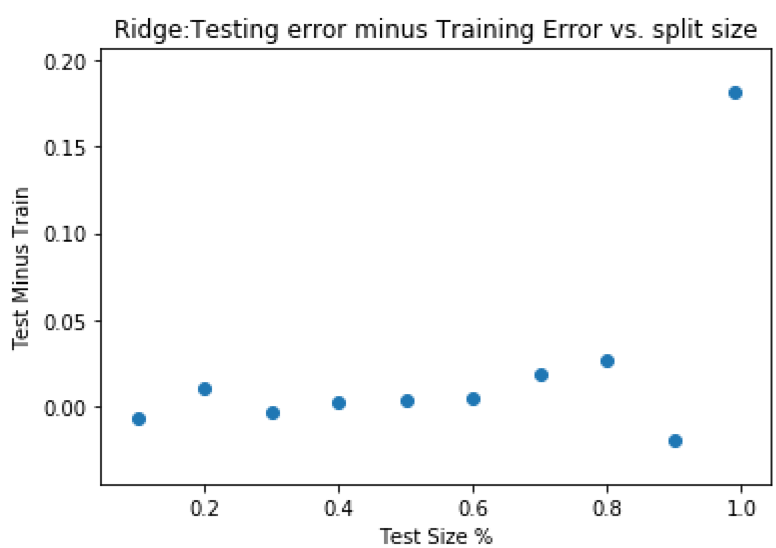
```
Test_0.9 = 0.08614390645554684
Train_0.9 = 0.10548714880918356
RTest_Size_9 -0.019343242353636714
```

In [34]:
```
#Ridge 0.99
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.99)

model = Ridge()
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.99 =', mean_squared_error(y_test, np.dot(x_test, model.coef_) + mode

print('Train_0.99 =', mean_squared_error(y_train, np.dot(x_train, model.coef_) + m

RTest_99 =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.interce
RTrain_99 = mean_squared_error(y_train, np.dot(x_train, model.coef_) + model.inter

RTest_Size_99 = RTest_99 - RTrain_99
print('RTest_Size_99', RTest_Size_99)
```

```
Test_0.99 = 0.18113646424064161
Train_0.99 = 0.0
RTest_Size_99 0.18113646424064161
```

In [35]:
```
## Plot of Ridge
x2 = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.99]
y2 = [RTest_Size_1,RTest_Size_2,RTest_Size_3,RTest_Size_4,RTest_Size_5,RTest_Size_

plt.scatter(x2, y2, label='Test Minus Train')
plt.xlabel('Test Size %')
plt.ylabel('Test Minus Train')
plt.title('Ridge:Testing error minus Training Error vs. split size')
```

Out[35]:
```
Text(0.5,1,'Ridge:Testing error minus Training Error vs. split size')
```



4. Chose an ideal split size based on the previous plot for Ridge.

Vary the Ridge parameter `alpha` from 0 to any value you'd like above 1. Plot the Train and Test error. Describe what you see based on the `alpha` parameter's stiffness.

In [36]:
```
#Ideal split 0.5, Small Alpha
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.5)

model = Ridge(alpha = 0.0005)
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.5_A_Small =', mean_squared_error(y_test, np.dot(x_test, model.coef_)

print('Train_0.5_A_Small =', mean_squared_error(y_train, np.dot(x_train, model.coe

RTest_5_A_Small =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.
RTrain_5_A_Small = mean_squared_error(y_train, np.dot(x_train, model.coef_) + mode

RTest_Size_5_A_Small = RTest_5_A_Small - RTrain_5_A_Small
print('RTest_Size_5_A_Small', RTest_Size_5_A_Small)
```

```
Test_0.5_A_Small = 0.09953342263275707
Train_0.5_A_Small = 0.061490621860249915
RTest_Size_5_A_Small 0.03804280077250715
```

In [37]:
```
#Ideal split 0.5, Medium Alpha
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.5)

model = Ridge(alpha = 50)
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.5_A_Medium =', mean_squared_error(y_test, np.dot(x_test, model.coef_

print('Train_0.5_A_Medium =', mean_squared_error(y_train, np.dot(x_train, model.co

RTest_5_A_Medium =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model
RTrain_5_A_Medium = mean_squared_error(y_train, np.dot(x_train, model.coef_) + mod

RTest_Size_5_A_Medium = RTest_5_A_Medium - RTrain_5_A_Medium
print('RTest_Size_5_A_Medium', RTest_Size_5_A_Medium)
```

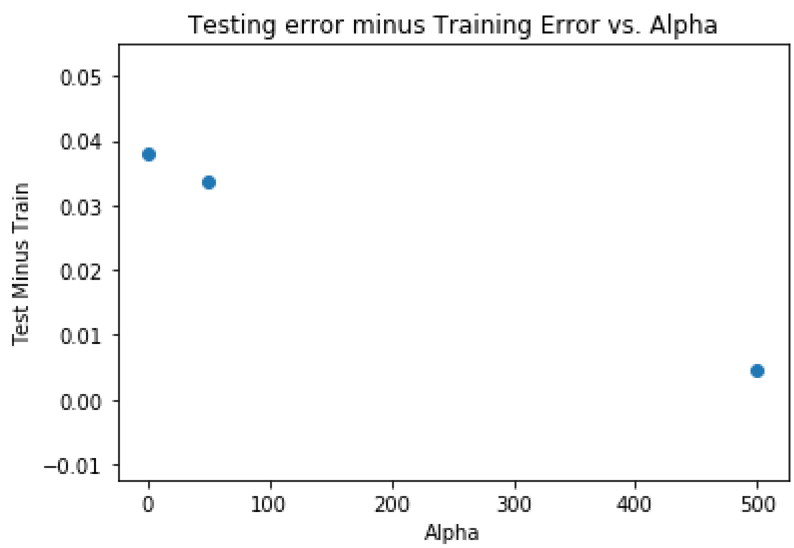
```
Test_0.5_A_Medium = 0.17896598983129294
Train_0.5_A_Medium = 0.14536963487544993
RTest_Size_5_A_Medium 0.03359635495584301
```

In [38]:
```
#Ideal split 0.5, Large Alpha
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.5)

model = Ridge(alpha = 500)
model.fit(x_train, y_train)
model.coef_, model.intercept_

print('Test_0.5_A_Large =', mean_squared_error(y_test, np.dot(x_test, model.coef_)

print('Train_0.5_A_Large =', mean_squared_error(y_train, np.dot(x_train, model.coe

RTest_5_A_Large =  mean_squared_error(y_test, np.dot(x_test, model.coef_) + model.
RTrain_5_A_Large = mean_squared_error(y_train, np.dot(x_train, model.coef_) + mode

RTest_Size_5_A_Large = RTest_5_A_Large - RTrain_5_A_Large
print('RTest_Size_5_A_Large', RTest_Size_5_A_Large)
```

```
Test_0.5_A_Large = 0.17669502479561738
Train_0.5_A_Large = 0.17221568080098332
RTest_Size_5_A_Large 0.004479343994634066
```

In [48]:
```
#Observations Regarding Alpha Stiffness.
#The larger the Alpha, the greather the difference between Test and Train.
x3 = [0.0005,50,500]
y3 = [RTest_Size_5_A_Small,RTest_Size_5_A_Medium,RTest_Size_5_A_Large]

plt.scatter(x3, y3, label='Test Minus Train')
plt.xlabel('Alpha')
plt.ylabel('Test Minus Train')
plt.title('Testing error minus Training Error vs. Alpha')
```

Out[48]:
```
Text(0.5,1,'Testing error minus Training Error vs. Alpha')
```



Bonus. Either: Generate data with a polynomial shape or use real data that you find on your own. Choose whatever regression model and process you'd like (Ridge,

polynomial, etc.) and plot the Train-Test errors vs. any parameter your Model depends on (e.g. `alpha`, `degree`, etc.)

In [49]:
```
n4 = 50
x4 = np.linspace(0.01, 1, n4).reshape(-1, 1)
y4 = np.linspace(0.01, 3, n4) * np.linspace(0.01, 3, n4) + np.random.rand(n4) - .5
```

In [50]:
```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=15)

x_15 = poly.fit_transform(x4.reshape(-1, 1))
```

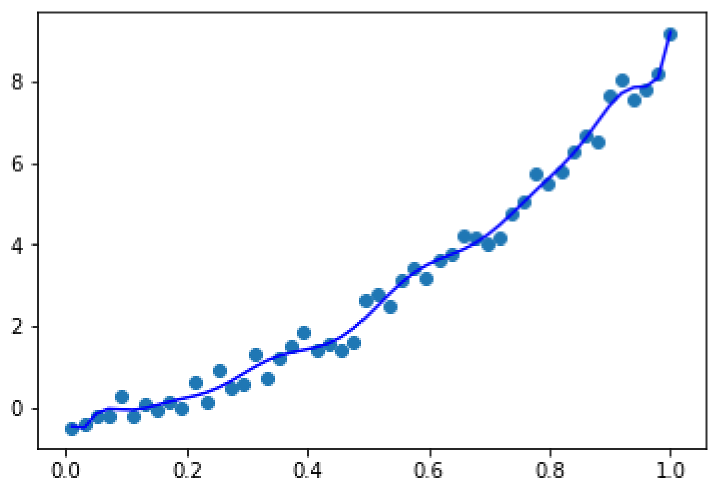
In [51]:
```
linear = linear_model.LinearRegression()

linear.fit(x_15, y4)

(linear.coef_, linear.intercept_)
```

Out[51]:
```
(array([ 0.00000000e+00, -1.51036418e+02,  7.50510842e+03, -1.68929088e+05,
         2.17404851e+06, -1.77854519e+07,  9.82424231e+07, -3.80072175e+08,
         1.05267820e+09, -2.11021288e+09,  3.06397180e+09, -3.18816861e+09,
         2.31577074e+09, -1.11442175e+09,  3.19143882e+08, -4.11586468e+07]),
 0.4387249810743872)
```

In [54]:
```
#Linear 0.1
x15_train, x15_test, y15_train, y15_test = train_test_split(x4,y4, test_size=0.5)

model = LinearRegression()
model.fit(x15_train, y15_train)
model.coef_, model.intercept_

print('Test_Poly =', mean_squared_error(y15_test, np.dot(x15_test, model.coef_) +

print('Train_Poly =', mean_squared_error(y15_train, np.dot(x15_train, model.coef_)

Test_Poly =  mean_squared_error(y15_test, np.dot(x15_test, model.coef_) + model.in
Train_Poly = mean_squared_error(y15_train, np.dot(x15_train, model.coef_) + model.

Test_Size_Poly = Test_Poly - Train_Poly
print('Test_Size_Poly', Test_Size_Poly)

plt.scatter(x4,y4)
plt.plot(x4, np.dot(x_15, linear.coef_) + linear.intercept_, c='b')
```

```
Test_Poly = 0.5474669705328349
Train_Poly = 0.3026120130959152
Test_Size_Poly 0.2448549574369197
```

Out[54]: [<matplotlib.lines.Line2D at 0x202703b9cc0>]



In [ ]: