```
#! /usr/bin/python
#Solution to exercise 5.3.4 Bootstrap

import numpy as np
import pandas as pd

import statsmodels.formula.api as formula

def fn(data, X_column, Y_column):
    # data the data set to operate on
    # X_column the column, in data, we want to get the coefficient for
    # Y_column the column ,in data, we want to estimate the relation to X_column.
    Y = data.loc[:, Y_column]
    X = data.loc[:, X_column]
    cov = data.cov().loc[X_column,Y_column]
    return (Y.var() - cov)/(X.var()+Y.var()-2*cov)

def boot(data, X, Y, B):
    # data the data to bootstrap
    # B the number of iterations of bootstrapping
    means = []
    for i in range(0,B):
        v = fn(data.sample(frac=1.0, replace=True), X, Y)
        means.append(v)
    return means

def boot_lm(data, r_formula, B):
    # data the data to bootstrap
    # r_formula the linear regression formula to use on the data
    # B the number of iterations of bootstrapping
    coef = []
    for i in range(0,B):
        # Take least squares regression of each resample using r_formula.
        v = formula.ols(r_formula, data=data.sample(frac=1.0, replace=True)).fit()
        coef.append(v.params)
    return coef

data = pd.read_csv("Portfolio.csv", header=0, index_col=0)

#Original data
print "Original data {}".format(fn(data, "X", "Y"))

#A single resample
print "A single resample {}".format( fn(data.sample(frac=1.0, replace=True,
random_state=0), "X", "Y") )


bootstrap = pd.DataFrame({"values": boot(data, "X", "Y", 1000)})

print "Bootstrap mean: {}".format(bootstrap.mean()["values"])
print "Bootstrap variance: {}".format(bootstrap.var()["values"])

# Estimating the accuracy

print "----------------------------------------------------------------------------"
print "---------------------------  Linear section  -------------------------------"
print "----------------------------------------------------------------------------"

auto_data = pd.read_csv("Auto.csv", header=0, index_col=0)
original_auto = formula.ols("mpg ~ horsepower", data=auto_data).fit()

print original_auto.summary()

print "original params"
print " Intercept:  mean:  {}".format(original_auto.params["Intercept"])
print " Intercept:  error: {}".format(original_auto.bse["Intercept"])
print " horsepower: mean:  {}".format(original_auto.params["horsepower"])
```

```python
print " horsepower: error: {}".format(original_auto.bse["horsepower"])

bootstrap_auto = pd.DataFrame(boot_lm(auto_data, "mpg ~ horsepower", 1000))

print "Bootstrapped params"
print " Intercept:  mean:  {}".format(bootstrap_auto.loc[:, "Intercept"].mean())
print " Intercept:  error: {}".format(bootstrap_auto.loc[:, "Intercept"].sem())
print " horsepower: mean:  {}".format(bootstrap_auto.loc[:, "horsepower"].mean())
print " horsepower: error: {}".format(bootstrap_auto.loc[:, "horsepower"].sem())

# Quadratic estimates

print "----------------------------------------------------------------------------"
print "-----------------------------  Quad section  -------------------------------"
print "----------------------------------------------------------------------------"

quad_auto = formula.ols("mpg ~ horsepower + np.power(horsepower, 2)", data=auto_data).fit()

print quad_auto.summary()

print "original params"
print " Intercept:  mean:               {}".format(quad_auto.params["Intercept"])
print " Intercept:  error:              {}".format(quad_auto.bse["Intercept"])
print " horsepower: mean:               {}".format(quad_auto.params["horsepower"])
print " horsepower: error:              {}".format(quad_auto.bse["horsepower"])
print " np.power(horsepower, 2): mean:  {}".format(quad_auto.params["np.power(horsepower, 2)"])
print " np.power(horsepower, 2): error: {}".format(quad_auto.bse["np.power(horsepower, 2)"])

bootstrap_quad_auto = pd.DataFrame(boot_lm(auto_data, "mpg ~ horsepower + np.power(horsepower, 2)", 1000))

print "Bootstrapped params"
print " Intercept:  mean:               {}".format(bootstrap_quad_auto.loc[:, "Intercept"].mean())
print " Intercept:  error:              {}".format(bootstrap_quad_auto.loc[:, "Intercept"].sem())
print " horsepower: mean:               {}".format(bootstrap_quad_auto.loc[:, "horsepower"].mean())
print " horsepower: error:              {}".format(bootstrap_quad_auto.loc[:, "horsepower"].sem())
print " np.power(horsepower, 2): mean:  {}".format(bootstrap_quad_auto.loc[:, "np.power(horsepower, 2)"].mean())
print " np.power(horsepower, 2): error: {}".format(bootstrap_quad_auto.loc[:, "np.power(horsepower, 2)"].sem())

```