```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr  3 21:18:23 2019
HW 7
@author: Ryan Dalby
"""

import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
from pyfinance.ols import OLS
import seaborn as sb

print("B1:")

data = pd.read_csv("HW7Data.csv")
yds = data['B1 Yds per Attempt'].dropna() #x like
rating = data['B1 Rating'].dropna() #y like
b1slope, b1intercept, _, b3Pval, b3SlopeSE = stats.linregress(yds, rating)
b1LSRL = lambda x: b1slope * x + b1intercept
plt.scatter(yds, rating)
ydsVals = np.linspace(np.min(yds), np.max(yds), 100)
plt.plot(ydsVals, b1LSRL(ydsVals))
plt.title("QB rating vs yds per attempt")
plt.show()
print("a) Slope = {:.5f} Intercept = {:.5f}".format(b1slope, b1intercept))

print("b) There is a predicted 10.1 increase in QB rating given a unit increase in yards per attemp

print("c) To increase the mean rating by 10 points yards per pass attempt should be increased by {:

print("d) With x = 7.21 yds/attempt a QB rating of {:.5f} is predicted".format(b1LSRL(7.21)))


print("\n\n")
print("B2:")

annualTaxes = data['B2 Taxes'].dropna() #x like
salePrice = data['B2 Sale Price'].dropna() #y like
b2slope, b2intercept, _, _, _ = stats.linregress(annualTaxes, salePrice)
b2LSRL = lambda x: b2slope * x + b2intercept

print("a) Slope = {:.5f} Intercept = {:.5f}".format(b2slope, b2intercept))

print("b) The predicted selling price given that the taxes paid are x = 7.50 is predicted to be {:.

b2PredictedValAtPoint = b2LSRL(5.898)
b2ActualValAtPoint = salePrice[np.where(annualTaxes == 5.898)[0][0]]
b2Residual = b2ActualValAtPoint- b2PredictedValAtPoint
print("c) When taxes paid is x = 5.898 the actual value is {:.5f} and the predicted value is {:.5f}

plt.scatter(annualTaxes, salePrice)
annualTaxesVals = np.linspace(np.min(annualTaxes), np.max(annualTaxes), 100)
plt.plot(annualTaxesVals, b2LSRL(annualTaxesVals))
```

```
plt.title("sale price vs annual taxes")
plt.show()
print("d) Yes the plot indicates that taxes paid is a relatively effective regressor variable in pr


print("\n\n")
print("B3:")

print("a) The two-sided p-value for for a hypothesis test with a null hypothesis of slope esitimate

b1model = OLS(rating, yds)
b3InterceptSE = b1model.se_alpha
print("b) The estimated standard error of the slope is {:.5f} and the estimated standard error of t


print("\n\n")
print("B4:")

density = data['B4 Density'].dropna() #x like
compStr = data['B4 Compressive Str.'].dropna() #y like
b4slope, b4intercept, b4CorrCoeff, b4Pval, _ = stats.linregress(density, compStr)
b4LSRL = lambda x: b4slope * x + b4intercept
print("a) Slope = {:.5f} Intercept = {:.5f}".format(b4slope, b4intercept))

print("b) The two-sided p-value for for a hypothesis test with a null hypothesis of slope esitimate

print("c) R^2 = {:.5f} and thus {:.3f}% of the variance of compressive strength is explained by the

b4fig, (b4ax1, b4ax2) = plt.subplots(ncols=2, figsize = (10,5))
sb.residplot(density, compStr, ax = b4ax1)
predictedCompStr = b4LSRL(density)
b4ResidualData = compStr - predictedCompStr
stats.probplot(b4ResidualData, plot = b4ax2)
b4ax2.set_xbound(-5, 5)
print("d) From the plots we can see that our data is approximately normal(residual plot data is app
```