Design of Lab7– Jacob Bollinger

Problem:

Design software to illustrate Benford's law.

Plan:

I will start by importing urlopen from urllib.request. Then I’ll create the variable WEB\_FILE and set it equal to urlopen() with the link to the text document inside the parenthesis.

from urllib.request import urlopen

WEB\_FILE = urlopen(‘link’)

I will next define a function named processFile(). It will take the parameter fIn. It will start by setting firstDigitList equal to ten zeros, first2DigitsList equal to one hundred zeros, lastDigitList equal to ten zeros, and last2DigitsList equal to one hundred zeros. Then I will initialize firstDigitListPercent, first2DigitsListPercent, lastDigitListPercent, and last2DigitsListPercent. I then will set a variable named count equal to 0.

Def processFile()

firstDigitList = [0] \* 10

first2DigitsList = [0] \* 100

lastDigitList = [0] \* 10

last2DigitsList = [0] \* 100

firstDigitListPercent = []

first2DigitsListPercent = []

lastDigitListPercent = []

last2DigitsListPercent = []

count = 0

for line in fIn :

line = line.decode('utf-8').strip()

count += 1

Next I will create a for loop in the function that will run for line in fIn. Inside the for loop set line equal to line.decode(‘utf-8’).strip(). Increase count by one. Next I will create an if statement to test if the variable count is greater than two. Inside the if statement I’ll set a variable firstDigit equal to the integer of line[0]. Then I will set a variable first2Digits equal to the integer of line[:2]. Next I’ll set the variable lastDigit equal to the integer of line[-1]. Then I will set a variable last2Digits equal to the integer of line[-2:]. I’ll finish the if statement by adding one to firstDigitList[firstDigit], first2DigitsList[first2Digits], lastDigitList[lastDigit], and last2DigitsList[last2Digits].

if count > 2 :

firstDigit = int(line[0])

first2Digits = int(line[:2])

lastDigit = int(line[-1])

last2Digits = int(line[-2:])

firstDigitList[firstDigit] += 1

first2DigitsList[first2Digits] += 1

lastDigitList[lastDigit] += 1

last2DigitsList[last2Digits] += 1

Back in the for loop I will set firstDigitListPercent equal to getPercent(firstDigitList, count), first2DigitsListPercent equal to getPercent(first2DigitsList, count), lastDigitListPercent equal to getPercent(lastDigitList, count), and last2DigitsListPercent equal to getPercent(last2DigitsList). Next I will set variables firstDigitData equal to firstDigitList and firstDigitListPercent, first2DigitsData equal to first2DigitsList and first2DigitsListPercent, lastDigitData equal to lastDigitList and lastDigitListPercent, and last2DigitsData equal to last2DigitsList and last2DigitsListPercent. I’ll finish the function by returning firstDigitData, first2DigitsData, lastDigitData, and last2DigitsData.

firstDigitListPercent = getPercent(firstDigitList, count)

first2DigitsListPercent = getPercent(first2DigitsList, count)

lastDigitListPercent = getPercent(lastDigitList, count)

last2DigitsListPercent = getPercent(last2DigitsList, count)

firstDigitData = firstDigitList, firstDigitListPercent

first2DigitsData = first2DigitsList, first2DigitsListPercent

lastDigitData = lastDigitList, lastDigitListPercent

last2DigitsData = last2DigitsList, last2DigitsListPercent

return firstDigitData, first2DigitsData, lastDigitData, last2DigitsData

Next I will define a function getPercent(). It will take the inputs digitList and count. In the function initialize the list called percent. I’ll create a for loop to run for n in digitList. In the for loop I’ll append round(n / count \* 100, 2) to percent. Finish the function by returning percent.

def getPercent(digitList, count) :

percent = []

for n in digitList :

percent.append(round(n / count \* 100, 2))

return percent

I will finish this off by printing processFile with the input of WEB\_FILE.

print(processFile(WEB\_FILE))