i) It takes command line arguments and puts it in a list. Then multiplies each value in the list by 2. Then reverses the list.

ii) There is no call to the function do\_stuff(). Therefore, the print statement never executes

import sys

def do\_stuff():

if (len(sys.argv) < 2 or str.isdigit(sys.argv[1]) == False):

print("Usage: python q1.py <list of numbers>")

return

int\_list = [int(x) for x in sys.argv[1:]]

int\_list = [x\*2 for x in int\_list]

int\_list = int\_list[::-1]

print(int\_list)

if \_\_name\_\_ == '\_\_main\_\_':

do\_stuff()

[

{

"name" : "Lorenzo",

"role" : "instructor",

"favourite\_number" : "8"

},

{

"name" : "Arthur",

"role" : "adventurer",

"favourite\_number" : "42"

}

]

There was a missing comma after the role “instructor”

[

{

"name" : "Lorenzo",

"role" : "instructor",

"favourite\_number" : "8"

},

{

"name" : "Arthur",

"role" : "adventurer",

"favourite\_number" : "42"

},

{

"name" : "kung fu",

"role" : "Panda",

"favourite\_number" : "69"

},

{

"name" : "jack",

"role" : "Black",

"favourite\_number" : "420"

}

]

import json

with open("data.json", "r") as inF:

content = json.load(inF)

data = content[::-1]

with open("samplefile.txt", "w") as fifi:

json.dump(data, fifi)

3)

import matplotlib.pyplot as plt

f = open("moby.txt", "r")

text = f.read()

theDict ={chr(y):((text.count((chr(y)).lower()))+(text.count((chr(y)).upper()))) for y in range(65,91)}

print(theDict)

freqDict = {chr(y):((text.count((chr(y)).lower()))+(text.count((chr(y)).upper())))/(sum(theDict.values())) for y in range(65,91)}

print(freqDict)

plt.bar(freqDict.keys(), freqDict.values(), color='g')

plt.show()

'A': 0.08163145461307607, 'B': 0.017728454230855218, 'C': 0.02402840179138983,

'D': 0.04002192361146168, 'E': 0.12297518243061559, 'F': 0.021900946073891224,

'G': 0.021935974400474737, 'H': 0.0657038684665732, 'I': 0.06870188112416202,

'J': 0.00121259824672923, 'K': 0.008476855033209944, 'L': 0.04467141884297316,

'M': 0.024403410934813314, 'N': 0.0688100568386111, 'O': 0.0729609135387573,

'P': 0.01841665782373128, 'Q': 0.0016288171861333156, 'R': 0.05522112661400742,

'S': 0.06711839471360737, 'T': 0.09266331699708337, 'U': 0.028035024205603915,

'V': 0.00899094723806796, 'W': 0.023243355177959352, 'X': 0.0010972108179835428,

'Y': 0.01776554304723776, 'Z': 0.0006562660009910956

Chart, histogram

Description automatically generated

import json

with open("q4.json", "r") as inF:

content = json.load(inF)

for i in range(len(content)):

data = content

with open("q4.txt", "w") as fifi:

json.dump(data, fifi)

Q4

import json

import json

with open("buildings.json", "r") as inF:

content = json.load(inF)

for i in range(len(content)):

content[i]["key"] = i

with open("q4\_new.json", "w") as fifi:

json.dump(content, fifi)

import json

with open("buildings.json", "r") as inF:

content = json.load(inF)

count = 0

summ = 0

for i in range(len(content)):

if content[i]["height"] > 5:

count += 1

summ += content[i]["height"]

average = summ / count

print("The average height of these buildings is", round(average, 2), "meters.")

The average height of these buildings is 12.64 meters.

Q1 numbers that start from 0 , 1 and each successive number is the sum of twice the previous number and the number two times before it

Q2 the first two valus of the pell sequnce is 0 and 1 and the program does corectly return 0 and 1 respectively by the following code.

if n == 0:

return 0

if n == 1:

return 1

Q3 if n = 2, the values of n = 0 and n = 1 is called. we know 0 and 1 is correct by q2. Hence, if a n+2 is enterd it will call n+1 and n which we know to be true, making the fucntion correct

import json

with open("songdata.json", "r") as inF:

content = json.load(inF)

below = []

above = []

for i in range(len(content)):

if content[i]["danceability"] > .5:

above.append(content[i]["energy"])

if content[i]["danceability"] < .5:

below.append(content[i]["energy"])

print(below)

print("")

print(above)

import json

import matplotlib.pyplot as plt

with open("songdata.json", "r") as inF:

content = json.load(inF)

below = []

above = []

for i in range(len(content)):

if content[i]["danceability"] > .5:

above.append(content[i]["energy"])

if content[i]["danceability"] < .5:

below.append(content[i]["energy"])

plt.hist(below, color='r', alpha=.5, label="Below .5")

plt.show()

import json

import matplotlib.pyplot as plt

with open("songdata.json", "r") as inF:

content = json.load(inF)

below = []

above = []

for i in range(len(content)):

if content[i]["danceability"] > .5:

above.append(content[i]["energy"])

if content[i]["danceability"] < .5:

below.append(content[i]["energy"])

plt.hist(above, color='r', alpha=.5, label="Below .5")

plt.show()

def fib(n):

if n < 0:

print("Incorrect input")

elif n == 0:

return 0

elif n == 1 or n == 2:

return 1

else:

return fib(n-1) + fib(n-2)

print(fib(10))

import timeit

def fib(n):

if n < 0:

print("Incorrect input")

elif n == 0:

return 0

elif n == 1 or n == 2:

return 1

else:

return fib(n-1) + fib(n-2)

val = timeit.timeit(lambda: fib(10), number=10000)

print("Program Executed in ",val)

**Program Executed in 0.09799754199999999**

import timeit

def fib(n):

if n < 0:

print("Incorrect input")

elif n == 0:

return 0

elif n == 1 or n == 2:

return 1

else:

return fib(n-1) + fib(n-2)

def fib\_for\_loop():

for i in range(30):

fib(i)

def list\_comp():

[fib(i) for i in range(30)]

for\_loop\_val = timeit.timeit(lambda: fib\_for\_loop(), number=100)

print("for loop Executed in ",for\_loop\_val)

list\_comp\_val = timeit.timeit(lambda: list\_comp(), number=100)

print("list comprehension Executed in ",list\_comp\_val)

for loop Executed in 12.19803879095707

list comprehension Executed in 12.191618041950278

with open('q8.txt', 'r') as inp:

y = inp.read().upper()

with open('q8\_out.txt', 'a') as out:

out.write(y)

from time import perf\_counter

t1\_start = perf\_counter()

with open('q8.txt', 'r') as inp:

y = inp.read().upper()

with open('q8\_out.txt', 'a') as out:

out.write(y)

t1\_stop = perf\_counter()

print("Elapsed time:", t1\_stop-t1\_start)

Elapsed time: 0.007799874991178513

This assignment was surprisingly tedious but not particularly difficult. It was not what I was expecting at all. However, I believe as it was simply a python revision, I was able to complete it without any issues due to my previous experience with ENDG 233.

the initial question caught me off guard as there was nothing particularly wrong with it but upon further experimentation, I was able to figure it out. The other questions were fairly trivial with the content being relatively easy.