David LaCharite

Assignment 1

In [1]:
```
import numpy as np
import pandas as pd
import math
```

Part 1

In [2]:
```
# Read Elements CSV into a pandas data frame
df_elements = pd.read_csv('elements.csv')
df_elements
```

Out[2]:

	name	symbol	atomic_number
0	Hydrogen	H	1
1	Helium	He	2
2	Lithium	Li	3
3	Beryllium	Be	4
4	Boron	B	5
5	Carbon	C	6
6	Nitrogen	N	7
7	Oxygen	O	8

In [3]:
```
# add ninth and tenth elements to dataframe
df_elements.loc[len(df_elements)]=['Fluorine','F', 9]
df_elements.loc[len(df_elements)]=['Neon','Ne', 10]
df_elements
```

Out[3]:

	name	symbol	atomic_number
0	Hydrogen	H	1
1	Helium	He	2
2	Lithium	Li	3
3	Beryllium	Be	4
4	Boron	B	5
5	Carbon	C	6
6	Nitrogen	N	7
7	Oxygen	O	8
8	Fluorine	F	9
9	Neon	Ne	10

In [4]:
```
# add a column with the atomic weights rounded to the nearest inetger
df_elements['atomic_weight'] = ['1','4','7','9','11','12','14','16','19','2(
df_elements
```

Out[4]:

	name	symbol	atomic_number	atomic_weight
0	Hydrogen	H	1	1
1	Helium	He	2	4
2	Lithium	Li	3	7
3	Beryllium	Be	4	9
4	Boron	B	5	11
5	Carbon	C	6	12
6	Nitrogen	N	7	14
7	Oxygen	O	8	16
8	Fluorine	F	9	19
9	Neon	Ne	10	20

Part 2

In [5]:
```
# Make a list of strings for nine Greek letters, ‘alpha’, for example.
# Make that list such that they are not in alphabetic order
greekLetters = ['delta','alpha','phi','iota','lambda',
                'gamma','eta','tau','epsilon']
```

In [6]:
```
# Make two 9-element numpy arrays of random floating-point numbers with the
# estimated mean 10 and standard deviation 1.5
mu = 10
sigma = 1.5
NPTS = 9
random1 = [sigma * x + mu for x in np.random.randn(NPTS)]
random2 = [sigma * x + mu for x in np.random.randn(NPTS)]
```

In [7]:
```
# Make an array of nine elements ranging from zero to two times pi
range_low = 0
range_high = 2*math.pi
angle = np.random.uniform(range_low, range_high, NPTS)
```

In [8]:
```
# Make another array holding the cosine of that ‘angle’ array.
cosine = [math.cos(x) for x in angle]
```

In [9]:
```
# Construct a dictionary from all of the above
d = {'Letter':greekLetters,
     'Random_1':random1,
     'Random_2':random2,
     'Angle' : angle,
     'Cosine' : cosine}
```

In [10]:
```
# Form a DataFrame from that dictionary and print it out
df_letters = pd.DataFrame(d)
df_letters
```

Out[10]:

	Letter	Random_1	Random_2	Angle	Cosine
0	delta	7.273537	9.888887	0.203054	0.979455
1	alpha	8.691574	10.471234	0.192436	0.981541
2	phi	11.185946	8.073075	1.800864	-0.228043
3	iota	11.417086	11.611333	6.126052	0.987680
4	lambda	8.779779	8.247957	3.577808	-0.906357
5	gamma	13.067288	8.629810	4.557872	-0.153903
6	eta	10.174758	12.722474	1.956316	-0.376041
7	tau	9.479260	8.299020	3.497064	-0.937483
8	epsilon	8.697934	7.975780	3.869171	-0.746787

In [11]:
```
# Sort the DataFrame ascending on the Greek letters,
trimmed_df = df_letters.sort_values(by=['Letter'])
```

In [12]:
```
# drop two columns of yourchoice
trimmed_df.drop(['Random_2', 'Cosine'], axis=1, inplace=True)
```

In [13]:
```
# drop one of the rows
trimmed_df.drop(trimmed_df[trimmed_df['Letter'] == 'eta'].index, inplace=Tru
```

In [14]:
```
# and print that out
trimmed_df
```

Out[14]:

	Letter	Random_1	Angle
1	alpha	8.691574	0.192436
0	delta	7.273537	0.203054
8	epsilon	8.697934	3.869171
5	gamma	13.067288	4.557872
3	iota	11.417086	6.126052
4	lambda	8.779779	3.577808
2	phi	11.185946	1.800864
7	tau	9.479260	3.497064

Part 3

In [15]:
```
# Write a program in Python to create and print out the first twelve Fibonac
numFibs = 12
fibsList = [0, 1]
while len(fibsList) < numFibs:
    fibsList.append(sum(fibsList[-2:]))
fibsList
```

Out[15]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

In [16]:
```
# iterate over the last five numbers to build another list
# with the ratio of each number to its predecessor
numRatios = 5
ratioList = []
for i in range(numFibs - numRatios, numFibs):
    ratioList.append(fibsList[i] / fibsList[i-1])
ratioList
```

Out[16]:
```
[1.625,
 1.6153846153846154,
 1.619047619047619,
 1.6176470588235294,
 1.6181818181818182]
```

What do you observe about this latter list?

The latter list is a convergent sequence with a limit of 1.618 which is ≈ the Golden Ratio.

Part 4

In [17]:
```
# Provide a function that converts temperature in Kelvin to Rankine
def kelvin_to_rankin(K):
    return K * 1.8
```

In [18]:
```
# Make a list of five Kelvin temperatures and print out their values in Rank
kelvinTemps = [0, 223, 283, 333, 373]
print(kelvinTemps)
rankinTemps1 = [kelvin_to_rankin(K) for K in kelvinTemps]
print(rankinTemps1)
```

```
[0, 223, 283, 333, 373]
[0.0, 401.40000000000003, 509.40000000000003, 599.4, 671.4]
```

In [19]:
```
# Repeat using a lambda function.
rankinTemps2 = list(map(lambda x: x * 1.8, kelvinTemps))
print(rankinTemps2)
```

```
[0.0, 401.40000000000003, 509.40000000000003, 599.4, 671.4]
```

In [ ]: