INFO 5502 Mid-term Exam (03/10/2022, 80 points in total)

Question 1 (5 pt). Make an array called `weird_numbers` containing the following numbers (in the given order):

1. -3
2. the sine of 2.1
3. 5
4. 4 to the power of the cosine of 4.2

Hint: `sin` and `cos` are functions in the `math` module.

```
import warnings
warnings.simplefilter('ignore', FutureWarning)
from datascience import make_array
import numpy as np
import math as mt

weird_numbers=make_array(int(-3),mt.sin(2.1),int(5),pow(4,mt.cos(4.2)))
print(weird_numbers)
```

```
[-3.          0.86320937  5.          0.50679646]
```

Question 2 (5 pt). Write a simple function that takes in a number (weight in pounds) and returns a number which is the coreesponding conversion to kg.

Test it by calling the function on 15 and 27. E.g., `convert_pounds_to_kg(15)`, `convert_pounds_to_kg(27)`. Print both to screen.

Hint: 1 pound = 0.453592 kg

```
def convert_pounds_to_kg(number):
    KG=0.453592*number
    return KG

print(convert_pounds_to_kg(15))
print(convert_pounds_to_kg(27))
```

```
6.8038799999999995
12.246984
```

Question 3 (5 pt). We've loaded an array of temperatures in the next cell. Each number is the highest temperature observed on a day at a climate observation station, mostly from the US. Since

they're from the US government agency NOAA, all the temperatures are in Fahrenheit. Convert them all to Celsius by first subtracting 32 from them, then multiplying the results by $\frac{5}{9}$. Make sure to **ROUND** the final result after converting to Celsius to the nearest integer using the `np.round` function. Download the data from here: https://github.com/unt-iialab/info5502-spring2022/blob/main/datasets/temperatures.csv

```
temperatures = Table.read_table("https://raw.githubusercontent.com/unt-iialab/info5502-spring
celsius_temperatures=list(map(lambda x : np.round(((x-32)*(5/9))), temperatures))
print(celsius_temperatures)
```

```
[-4.0, 31.0, 32.0, 36.0, 26.0, 33.0, 24.0, 29.0, 21.0, -14.0, 33.0, 9.0, -14.0, 24.0, 1
```

Question 4 (5 pt). Suppose you have 4 apples, 3 oranges, and 3 pineapples. (Perhaps you're using Python to solve a high school Algebra problem.) Create a table that contains this information. It should have two columns: `fruit name` and `count`. Assign the new table to the variable `fruits`.

Note: Use lower-case and singular words for the name of each fruit, like `"apple"`.

```
counts=make_array(4, 3, 3)
fruitTypes=make_array('apples', 'oranges', 'pineapples')
fruits = Table().with_columns('fruit name',fruitTypes,'count',counts)
fruits
```

Question 5 (10 pt). Below we load a table containing 200,000 weekday Uber rides in the Boston, Massachusetts metropolitan area from the Uber Movement project. The `sourceid` and `dstid` columns contain codes corresponding to start and end locations of each ride. The `hod` column contains codes corresponding to the hour of the day the ride took place. The `ride time` column contains the length of the ride, in minutes. Produce a histogram of all ride times in Boston using the given bins. Download the data from here: https://github.com/unt-iialab/info5502-spring2022/blob/main/datasets/boston.csv

```
bostonDataset = Table.read_table("https://raw.githubusercontent.com/unt-iialab/info5502-sprir
bostonDataset.show(10)
```

```
ax=bostonDataset.hist("ride time",bins=np.arange(0, 120, 4))
```

Question 6 (20 pt). Below is a dataset we collected from this website: https://ddr.densho.org/narrators/?page=1. Narrators are the interview subjects of oral histories contained in the Densho Digital Repository. The interviewees, or narrators, share their life histories to preserve history, educate the public, and promote tolerance. We urge our users to approach these materials in the same spirit. You are required to conduct the exploratory data analysis on the location, year of born, generation, and gender. Please select the best visualiztions to present your results. Download the data from here: https://github.com/unt-iialab/info5502-spring2022/blob/main/datasets/Combined-data.xlsx

```
#Questions
#Visualize the count of narrators w.r.t Gender
#Find which Generation have more narrators w.r.t Gender.
#Visualize the correlation of the Dataset



import pandas as pd

#Reading the csv file from github
df = pd.read_csv('https://raw.githubusercontent.com/gantaphani/Phanesh_INFO5502_Spring2022/ma
print(df.head(5).to_string(index=False))

               ï»¿Narrator   Location   Year Generation Gender
              Kay Aiko Abe Washington 1927.0      Nisei female
                   Art Abe Washington 1921.0      Nisei   male
     Sharon Tanagi Aburano Washington 1925.0      Nisei female
           Toshiko Aiboshi California 1928.0      Nisei female
                Yae Aihara Washington 1925.0      Nisei female


#Dataset Information
df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 904 entries, 0 to 903
     Data columns (total 5 columns):
      #   Column       Non-Null Count  Dtype
     ---  ------       --------------  -----
      0   ï»¿Narrator  904 non-null    object
      1   Location     832 non-null    object
      2   Year         710 non-null    float64
      3   Generation   850 non-null    object
      4   Gender       850 non-null    object
     dtypes: float64(1), object(4)
     memory usage: 35.4+ KB


#Renaming column name 'Narrator'
df.rename(columns={'ï»¿Narrator': 'Narrator'}, inplace=True)
print('After renaming columns\n')
print(df.columns)

     After renaming columns

     Index(['Narrator', 'Location', 'Year', 'Generation', 'Gender'], dtype='object')


#Size of the DataFrame
rows,columns=df.shape
print("Number of Rows are:",rows)
print("Number of Columns are:",columns)

     Number of Rows are: 904
     Number of Columns are: 5
```

```
#summary statastics for a DataFrame
print(df.describe().to_string())
```

```
                Year
count   710.000000
mean   1924.997183
std      35.017545
min    1031.000000
25%    1920.000000
50%    1925.000000
75%    1930.000000
max    1991.000000
```

```
#Finding missing values and dropping them
print(df.isnull().any())
print("All columns has missing values")
df.dropna(inplace=True)
print("\nAfter Dropping Missing Values")
print(df.isnull().any())
```

```
Narrator      False
Location       True
Year           True
Generation     True
Gender         True
dtype: bool
All columns has missing values

After Dropping Missing Values
Narrator      False
Location      False
Year          False
Generation    False
Gender        False
dtype: bool
```

```
#checking duplicates in the dataset
duplicate = df[df.duplicated()]

print("Counts Of Duplicate Rows \n",duplicate.count())
print('\nDataset does not have Duplicate values')
```

```
Counts Of Duplicate Rows
 Narrator      0
Location      0
Year          0
Generation    0
Gender        0
dtype: int64

Dataset does not have Duplicate values
```

```
from importlib import reload
import matplotlib.pyplot as plt
import seaborn as s
plt=reload(plt)

s.set_style('whitegrid')
ax=s.countplot(x='Gender',data=df)
for patch in ax.patches:
    height = patch.get_height()
    width = patch.get_width()
    new_width = width * 1
    patch.set_width(new_width)
    x = patch.get_x()
    patch.set_x(x + (width - new_width) / 2)
    ax.text(x=x + width/2, y=height, s=height, ha='center', va='bottom',fontsize=12)
plt.xlabel('Gender',fontsize=20)
plt.ylabel('Count Of Narrators',fontsize=20)
plt.show()
```

```
from importlib import reload
plt=reload(plt)

fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(10, 4))

plot1=df[df['Gender']=='male'].groupby(['Generation'])['Generation'].count().plot(kind='bar',
plot2=df[df['Gender']=='female'].groupby(['Generation'])['Generation'].count().plot(kind='bar

for plot in plot1,plot2:
  for patch in plot.patches:
      height = patch.get_height()
      width = patch.get_width()
      new_width = width * 1
      patch.set_width(new_width)
```

```
        x = patch.get_x()
        patch.set_x(x + (width - new_width) / 2)
        plot.text(x=x + width/2, y=height, s=height, ha='center', va='bottom')

plot1.set_title("Male",fontsize=14)
plot2.set_title("Female",fontsize=14)
plt.show()
```

```
from importlib import reload
plt=reload(plt)

correlationDataset=df.copy()

correlation = correlationDataset.apply(lambda x: x.factorize()[0]).corr()
plt.figure(figsize=(7,7))
s.heatmap(correlation, annot=True, cmap="Reds")
plt.title('Correlation Heatmap', fontsize=20)
plt.show()
```

Question 7. Monkeys Typing Shakespeare A monkey is banging repeatedly on the keys of a typewriter. Each time, the monkey is equally likely to hit any of the 26 lowercase letters of the English alphabet, 26 uppercase letters of the English alphabet, and any number between 0-9 (inclusive), regardless of what it has hit before. There are no other keys on the keyboard.

This question is inspired by a mathematical theorem called the Infinite monkey theorem (https://en.wikipedia.org/wiki/Infinite_monkey_theorem), which postulates that if you put a monkey in the situation described above for an infinite time, they will eventually type out all of Shakespeare's works.

Question 7-1 (10 pt). Suppose the monkey hits the keyboard 5 times. Compute the chance that the monkey types the sequence `Data8`. (Call this `data_chance`.) Use algebra and type in an arithmetic equation that Python can evalute.

```
data_chance = (1/62)**5
data_chance
```

```
1.0915447684774164e-09
```

Question 7-2 (10 pt). Write a function called `simulate_key_strike`. It should take **no arguments**, and it should return a random one-character string that is equally likely to be any of the 26 lower-case English letters, 26 upper-case English letters, or any number between 0-9 (inclusive).

```
from random import choice as ch
import string as st

KEYS = list(st.digits+st.ascii_lowercase+st.ascii_uppercase)

def simulate_key_strike():
    return ch(KEYS)
```

```
simulate_key_strike()
```

Question 7-3 (10 pt). Write a function called `simulate_several_key_strikes`. It should take one argument: an integer specifying the number of key strikes to simulate. It should return a string containing that many characters, each one obtained from simulating a key strike by the monkey.

Hint: If you make a list or array of the simulated key strikes called `key_strikes_array`, you can convert that to a string by calling `"".join(key_strikes_array)`

```
def simulate_several_key_strikes(strikes):
    return "".join(ch(KEYS) for strike in range(strikes))

simulate_several_key_strikes(9)
```

```
'1KNJsNQQP'
```

✓ 0s completed at 5:09 PM