Assignment 1

September 23, 2022

Exercise 1:

```
[ ]: def temp_tester(temp):

      def range_temp_function(x):
         if x<= temp + 1 and x >= temp - 1:
             return True
         else:
             return False

      return range_temp_function
```

```
[ ]: human_tester = temp_tester(37)
     chicken_tester = temp_tester(41.1)
```

```
[ ]: # Example to test plot function
     print(chicken_tester(42))
     print(human_tester(42))
     print(chicken_tester(43))
     print(human_tester(35))
     print(human_tester(98.6))
```

Exercise 2:

```
[ ]: import pandas as pd
     import sqlite3
     with sqlite3.connect("/Users/polina/Desktop/hw1-population.db") as db:
         data = pd.read_sql_query("SELECT * FROM population", db)
         data.head
```

```
[ ]: data.head
```

```
[ ]: #Age
     data.describe()[["age"]]
```

```
[ ]: ! conda install -c conda-forge plotnine -y
```

```
[ ]: from plotnine import *
[ ]: #Find bin width
     import math
     log2 = math.log2(152361)+1
     log2
```

```
[ ]: data.hist(column = 'age', bins = 18)
```

```
[ ]: #Weight
     data.describe()[["weight"]]
```

```
[ ]: data.hist(column = 'weight', bins = 18)
```

```
[ ]: import plotly.express as px
     fig = px.scatter(data, x="age", y="weight")
     fig.show()
```

```
[ ]: # Filter to detect outlier
     data[(data['age'] > 20) & ((data['weight'] < 40)|(data['weight'] > 100))]
```

```
[ ]: # Filter for Anthony Freeman
     data[(data['age'] == 41.3) | (data['weight'] == 21.7)]
```

Exercise 3:

```
[ ]: import numpy as np
     from matplotlib import pyplot as plt
     import pandas as pd
     url = 'https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.
      ↪csv'
     df = pd.read_csv(url)
     df.head()
```

```
[ ]: # convert the 'Date' column to datetime format
     df['date']= pd.to_datetime(df['date'])
     # Check the format of 'Date' column
     df.info()
```

```
[ ]: #look for new cases
     (df
      .groupby(['state'])
      .cases
      .diff()
     )
```

```
[ ]: def plot(state_list):
         plt.figure(figsize=(30,20))
         for state in state_list:
             df_new = df[df['state']==state]
```

```
        df_new['cases'] = df_new['cases'].diff()
        plt.plot(df_new['date'], df_new['cases'], label=state)
    plt.legend()
    plt.title('New Cases vs Date')
    plt.xlabel('Date')
    plt.ylabel('New Cases')
    plt.show()
```

[ ]:
```
# Example to test plot function
state_list = ['Wyoming', 'Northern Mariana Islands', 'Massachusetts',␣
 ↪'Connecticut']
plot(state_list)
```

[ ]:
```
def highest_case(state):
    df_new_date = df[df["state"] == state]
    df_new_date['cases'] = df_new_date['cases'].diff()
    date = df_new_date[df_new_date['cases'] == np.
 ↪max(df_new_date['cases'])]['date']
    return date.iloc[0]
```

[ ]:
```
# Example to test plot function
print(highest_case('Washington'))
print(highest_case('Illinois'))
print(highest_case('Massachusetts'))
```

[ ]:
```
def peak(state1, state2):
    df_peak_1 = (highest_case(state1))
    df_peak_2 = (highest_case(state2))

    if df_peak_1 >  df_peak_2:
        print(state2, 'had its highest number of daily new cases by',␣
 ↪abs((df_peak_1 - df_peak_2).days, 'days'))
    elif df_peak_1 < df_peak_2:
        print(state1, 'had its highest number of daily new cases by',␣
 ↪abs((df_peak_1 - df_peak_2).days), 'days')
    else:
        print(state1, 'and', state2, 'had its highest number of daily new cases␣
 ↪on', df_peak_1)
```

[ ]:
```
# Example to test plot function
peak('Massachusetts', 'Connecticut')
peak('Florida', 'Washington')
peak('Ohio','Montana')
```

Exercise 4:

[ ]:
```python
import xml.etree.ElementTree as ET
from pprint import pprint as pp
tree = ET.parse('/Users/polina/Desktop/desc2022.xml')
root = tree.getroot()
```

[ ]:
```python
pp(root)
```

[ ]:
```python
desk1 = root[0]
desk1_xml = ET.tostring(desk1)
print(desk1_xml)
```

[ ]:
```python
ET.indent(desk1)
print(ET.tostring(desk1).decode('utf-8'))
```

[ ]:
```python
#Select the child by using its tag name
Descriptor1 = desk1.find("DescriptorUI")
pp(Descriptor1)
```

[ ]:
```python
ui_1 = Descriptor1.text
pp(ui_1)
```

[ ]:
```python
def find_ui(ui):
    for child in root:
        if child[0].text == ui:
            return child[1][0].text
```

[ ]:
```python
print(find_ui('D007154'))
```

[ ]:
```python
def find_name(name):
    for child in root:
        if child[1][0].text == name:
            return child[0].text
```

[ ]:
```python
print(find_name('Nervous System Diseases'))
```

[ ]:
```python
def treeNumber_find(name):
    if not 'D0' in name:
        name = find_name(name)
    for child in root:
        if child[0].text == name:
            for record in child.iter('TreeNumberList'):
                return record[0].text
```

[ ]:
```python
print(treeNumber_find('Nervous System Diseases'))
print(treeNumber_find('D007154'))
```

[ ]:
```
def descendents_common(descendents1, descendents2):


    descendentsTree1 = treeNumber_find(descendents1) + '.'
    descendentsTree2 = treeNumber_find (descendents2) + '.'

    answer = set()  ## empty set

    for child in root:
        for record in child.iter('TreeNumberList'):
            for treeNumber in record:
                if descendentsTree1 in treeNumber.text:
                    for treeNumber in record:
                        if descendentsTree2 in treeNumber.text:
                            answer.add(child[1][0].text)
    return answer
```

[ ]:
```
print(descendents_common('Nervous System Diseases', 'D007154'))
```