Assignment4

Part 1

Development:

1. Requirements:
   a. Code and run **demo.py. (Note: *be sure* necessary packages are installed!)**
   Note: If needed, see previous assignment for installing Python packages.
   b. Then. use **demo.py** to backward-engineer the screenshots below it.
   c. When displaying the required graph (see code below), answer the following question:
   Why is the graph line split?

2. Be sure to test your program using both **IDLE** and **Visual Studio Code**.

Part 2

README.md file should include the following items:

1. **Assignment requirements, as per A1.**
2. Screenshot as per example below, **including graph**.
3. Upload A4 **.ipynb** file and create link in README.md;
   Note: *Before* uploading **.ipynb** file, ***be sure*** to do the following actions from **Kernal** menu:
   a. **Restart & Clear Output**
   b. **Restart & Run All**

Deliverables:

1. Provide **Bitbucket** read-only access to **lis4369** repo, include links to the repos you created in the above tutorials in **README.md**, using Markdown syntax
   (**README.md** must also include screenshots as per above.)
2. **FSU's Learning Management System**: lis4369 **Bitbucket** repo

Note: combine demo1.py and demo2.py into demo.py

demo1.py

```python
import re  # re module provides regular expression matching operations similar to Perl
# np supports arrays and matrices, along with mathematical functions for scientific computing
import numpy as np  # Numerical Python
np.set_printoptions(threshold=np.inf)  # print full NumPy array, no ellipsis
import pandas as pd

# Read CSV (comma-separated values) file into DataFrame
# Data sets: https://vincentarelbundock.github.io/Rdatasets/
# https://raw.github.com/vincentarelbundock/Rdatasets/master/csv/COUNT/titanic.csv
# https://vincentarelbundock.github.io/Rdatasets/csv/datasets/Titanic.csv
# https://raw.github.com/vincentarelbundock/Rdatasets/master/csv/carData/TitanicSurvival.csv

url = "https://raw.github.com/vincentarelbundock/Rdatasets/master/csv/Stat2Data/Titanic.csv"
df = pd.read_csv(url)

print("***DataFrame composed of three components: index, columns, and data. Data also known as values.***")
# https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c
index = df.index
columns = df.columns
values = df.values

print("\n1. Print indexes:")
print(index)

print("\n2. Print columns:")
print(columns)

# Same as above
print("\n3. Print columns (another way):")
print(df.columns[:])  # using slicing notation

print("\n4. Print (all) values, in array format:")
print(values)

print("\n5. ***Print component data types:***")
print("\na) index type:")
print(type(index))
# pandas.core.indexes.range.RangeIndex

print("\nb) columns type:")
print(type(columns))
# pandas.core.indexes.base.Index

print("\nc) values type:")
print(type(values))
# numpy.ndarray

print("\n6. Print summary of DataFrame (similar to 'describe tablename;' in MySQL):")
print(df.info())

print("\n7. First five lines (all columns):")
print(df.head())

# https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.drop.html
# Note: 'Unnamed: 0' appears to be used just to number rows
df = df.drop('Unnamed: 0', 1)  # drop column 'Unnamed: 0'
print("\n8. Print summary of DataFrame (after dropping column 'Unnamed: 0'):")
print(df.info())

print("\n9. First five lines (after dropping column 'Unnamed: 0'):")
print(df.head())

# Precise data selection (data slicing):
# Questions? Do some research! https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c
# Note: contolled/precise data selection (data slicing)
# 1) DataFrame.loc gets rows (or columns) with particular labels (names) from index
# 2) DataFrame.iloc (stands for integer location) gets rows (or columns) at particular positions in index (i.e., only takes integers)
# .loc/.iloc accepts same slice notation that Python lists do for both row and columns. Slice notation being start:stop:step
# .loc includes last value with slice notation, .iloc does *not*--that is, .iloc slice is ***exclusive*** of last integer!

print("\n***Precise data selection (data slicing):***")
print("\n10. Using iloc, return first 3 rows:")
print(df.iloc[:3])
# print(df.iloc[0:3:1]) # equivalent to above (slice notation = start:stop:step)

print("\n11. Using iloc, return last 3 rows (start on index 1310 to end):")
print(df.iloc[1310:])

# select rows and columns simultaneously
# separate row and column with comma
# example: df.iloc[row_index, column_index]

print("\n12. Select rows 1, 3, and 5; and columns 2, 4, and 6 (includes index column):")
a = df.iloc[[0, 2, 4], [1, 3, 5]]
print(a)

print("\n13. Select all rows; and columns 2, 4, and 6 (includes index column):")
a = df.iloc[:, [1, 3, 5]]
print(a)

print("\n14. Select rows 1, 3, and 5; and all columns (includes index column):")
a = df.iloc[[0, 2, 4], :]
print(a)
# same as above
# a = df.iloc[[0, 2, 4]]  # Note: leaving out colon selects all columns as well
# print(a)

print("\n15. Select all rows, and all columns (includes index column). Note: only first and last 30 records displayed:")
a = df.iloc[:, :]
print(a)

print("\n16. Select all rows, and all columns, starting at column 2 (includes index column). Note: only first and last 30 records displayed:")
a = df.iloc[:, 1:]
print(a)

print("\n17. Select row 1, and column 1, (includes index column):")
# Note: .iloc does *not* contain last index value--here, should have included 1!
a = df.iloc[0:1, 0:1]
print(a)

print("\n18. Select rows 3-5, and columns 3-5, (includes index column):")
# Note: .iloc does *not* contain last index value--here, should have included 5!
a = df.iloc[2:5, 2:5]
```

demo2.py

```
print("\n18. Select rows 3-5, and columns 3-5, (includes index column):")
# Note: .iloc does *not* contain last index value--here, should have included 5!
a = df.iloc[2:5, 2:5]
print(a)

print("\n19. ***Convert pandas DataFrame df to NumPy ndarray, use values command:***")
# Select all rows, and all columns, starting at column 2:
b = df.iloc[:, 1:].values  # ndarray = N-dimensional array (rows and columns)

print("\n20. Print data frame type:")
print(type(df))

print("\n21. Print a type:")
print(type(a))

print("\n22. Print b type:")
print(type(b))

print("\n23. Print number of dimensions and items in array (rows, columns). Remember: starting at column 2:")
print(b.shape)

print("\n24. Print type of items in array. Remember: ndarray is an array of arrays. Each record/item is an arrary.")
print(b.dtype)

print("\n25. Printing a:")
print(a)

print("\n26. Length a:")
print(len(a))

print("\n27. Printing b:")
print(b)

print("\n28. Length b:")
print(len(b))

# Print element of ndarray b in *second* row, *third* column
print("\n29. Print element of (NumPy array) ndarray b in *second* row, *third* column:")
print(b[1, 2])

# Print full NumPy array, no ellipsis: here is why np.set_printoptions(threshold=np.inf) is set at top of file
print("\n30. Print all records for NumPy array column 2:")
print(b[:, 1])

print("\n31. Get passenger names:")
names = df["Name"]
print(names)

print("\n32. Find all passengers with name 'Allison' (using regular expressions):")
# Note: 'r' obviates the need for an escape sequence. For example: \'(Allison)\'
# See: https://docs.python.org/2/library/re.html
for name in names:
    print(re.search(r'(Allison)', name))
# Note: there are various ways of retrieving data

# Note: print full DataFrame, w/no ellipsis
# will automatically return options to their default values
# with pd.option_context('display.max_rows', None):
# print(df) # print entire dataframe

print("\n***33. Statistical Analysis (DataFrame notation):***")
# difference between np.mean and np.average: average takes optional weight parameter. If not supplied they are equivalent.
print("\na) Print mean age:")
avg = df["Age"].mean()  # second column
print(avg)

print("\nb) Print mean age, rounded to two decimal places:")
avg = df["Age"].mean().round(2)  # will *not* display last 0
print(avg)

print("\nc) Print mean of every column in DataFrame (may not be suitable with certain columns):")
avg_all = df.mean(axis=0)  # mean every column
# avg_all = df.mean(axis=1)  # mean every row
print(avg_all)

print("\nd) Print summary statistics (DataFrame notation):")
# returns three quartiles, mean, count, min/max values, and standard deviation
describe = df["Age"].describe()  # second column
# describe = df["Age"].describe(percentiles=[.10, .20, .50, .80])  # choose different percentiles
print(describe)

print("\ne) Print minimum age (DataFrame notation):")
# can also do functions separately
min = df["Age"].min()  # second column
print(min)

print("\nf) Print maximum age (DataFrame notation):")
max = df["Age"].max()  # second column
print(max)

print("\ng) Print median age (DataFrame notation):")
median = df["Age"].median()  # second column
print(median)

print("\nh) Print mode age (DataFrame notation):")
mode = df["Age"].mode()  # second column
print(mode)

print("\ni) Print number of values (DataFrame notation):")
count = df["Age"].count()  # second column
print(count)

print("\n***Graph: Display ages of the first 20 passengers (use code from previous assignment):***")
```

Assignment Requirements

```
Data Analysis 2

Program Requirements:
1. Run demo.py.
2. If errors, more than likely missing installations.
3. Test Python Package Installer: pip freeze
4. Research how to install any missing packages:
5. Create at least three functions that are called by the program:
        a. main(): calls at least two other functions.
        a. get_requirements(): displays the program requirements.
        c. data_analysis_2(): displays results as per demo.py.
6. Display graph as per instructions w/in demo.py.
```

Part 3

Questions (Python: Chs. 9, 10):

1. A Unicode character is represented by a
    2-digit code
    2-character code
    1-byte ASCII code
    1-byte character code

2. Given the following code, what would display?

```
car = "PORSCHE"
color = "red"
my_car2 = car.join(color)
print(my_car)
```

    PredOredRredSredCredHredE
    PORSCHEred
    rPORSCHEePORSCHEd
    redPORSCHE

3. Given the following code, what will be displayed after the code executes?

```
name = "Mervin the Magician"
words = name.split()
print(words[0] + ", you are quite a " + words[2].lower())
```

    Mervin
, you are quite a
magician
    Mervin, you are quite a Magician
    Mervin, you are quite a magician
    Mervin
, you are quite
the magician

4. If word = "a horse", which of the following snippets of Python code will display this result?

a horse! a horse! My kingdom for a horse!

```
print((word * 2) + "! My kingdom for " + word + "!")
print((word + "! " + " My kingdom for " + word + "!") * 2)
print(word * 2 + " My kingdom for " + word + "!")
print((word + "! ") * 2 + " My kingdom for " + word + "!")
```

5. Consider the following code:

```
1. phone_number = input("Enter phone number: ").strip()
2. if len(phone_number) == 10:
3.     phone_number = "(" + phone_number[:3] + ")"
                      + phone_number[3:6]
                      + "-" + phone_number[6:]
4.     print("Phone number: ", phone_number)
5. else:
6.     print("Phone number: ", "Phone number must be 10 digits")
```

If the user enters two extra spaces at the end of a phone number, what will happen?

The length of the number will be greater than 10 so the else clause will execute.
Whitespace at the end of the input is always ignored.
The strip() method on line 1 will strip away the extra whitespace.
The else clause will be executed.

6. Consider the following code:

```
1. phone_number = input("Enter phone number: ").strip()
2. if len(phone_number) == 10:
3.     phone_number = "(" + phone_number[:3] + ")"
                      + phone_number[3:6]
                      + "-" + phone_number[6:]
4.     print("Phone number: ", phone_number)
5. else:
6.     print("Phone number: ", "Phone number must be 10 digits")
```

If the user enters 555-123-4567 at the prompt, what will happen?

The length of the number will be greater than 10 so the else clause will execute.
Non-numeric characters will be ignored.
The strip() method on line 1 will strip away the extra characters..
The hyphens will be ignored.

7. Consider the following code:

```
1. phone_number = input("Enter phone number: ").strip()
2. if len(phone_number) == 10:
3.     phone_number = "(" + phone_number[:3] + ")"
                      + phone_number[3:6]
                      + "-" + phone_number[6:]
4.     print("Phone number: ", phone_number)
5. else:
6.     print("Phone number: ", "Phone number must be 10 digits")
```

If the user enters 5551234567 at the prompt, what will be displayed?

Phone number: 5551234567
Phone number: (555)123-4567
Phone number: (555)123-456
Phone number: (555)1234-567

8. The isdigit() method of a string returns
   the digits that are in the string
   the string if it only contains digits
   true if the string contains only digits
   true if the string contains only digits and a decimal point

9. The join() method of a list can be used to combine
   the items in the list into a string
   the items in the list into a string that's separated by delimiters
   two or more lists
   two or more strings into a list

10. To access the first three characters in a string that's stored in a variable named message, you can use this code:
   first_three = message[0:2]
   first_three = message[1:3]
   first_three = message.slice(0:2)
   first_three = message.split(0:2)

11. To determine the length of a string that's in a variable named city, you can use this code:
   len(city)
   city.len()
   length(city)
   city.length()

12. To retrieve the fourth character in a string that's stored in a variable named city, you can use this code:
   city(3)
   city[3]
   city(4)
   city[4]

13. What is the value of s3 after the code that follows is executed?

```
s1 = "abc def ghi";
s2 = s1[1:5]
s3 = s2.replace('b', 'z')
print(s3)
```

   bc d
   zc d
   abc d
   azc d
   zc de

14. What is the value of s2 after the code that follows is executed?

```
s1 = "118-45-9271"
s2 = ""
for i in s1:
   if i != '-':
      s2 += i
s1.replace("-", ".")
```

   118-45-9271
   118 45 9271
   118459271
   118.45.9271

15. What is the value of the variable named result after this code is executed?

```
email = "marytechknowsolve.com"
result = email.find("@")
```

- true
- false
- 0
- -1

16. What is the value of the variable named result after this code is executed?

```
email = "joel.murach@com"
result = email.find("@") - email.find(".")
print(result)
```

- true
- 7
- 0
- -1

17. What will be displayed after the following code executes?

```
book_name = "a tale for the knight"
book = book_name.title()
print(book)
```

- A Tale For The Knight
- A Tale for the Knight
- A Tale for the knight
- A tale for the knight

18. Which of the following will display this result?

```
B = 66
```

```
print("B = ", char("B"))
print("B = ", ord("B"))
print("B = ", ascii("B"))
print(ord(66))
```

19. Which of the following code snippets will result in this display:

```
Countdown...
5...
4...
3...
2...
1...
Blastoff!
```

```
    counting = "5...4...3...2...1"
print("Countdown...")
for char in counting:
   print(char + "...")
print("Blastoff!")
```

```
    counting = "54321"
for char in counting:
   print("Countdown...")
   print(char + "...")
   print("Blastoff!")
```

```
    counting = "54321"
print("Countdown...")
for char in counting:
   print(char + "...")
print("Blastoff!")
```

```
    counting = 54321
print("Countdown...")
for char in counting:
   print(char + "...")
print("Blastoff!")
```

20. Which of the following statements will modify the string that's stored in a variable named s?
    s.strip()
    s.replace('a', 'A')
    s[1] = "A"
    You can't modify a string.

21. Which of the Python examples that follow cannot be used to create a string named n2?
    n2 = "817542235"

```
    numbers = "817542235"
n2 = numbers.replace("2", "")
```

```
    numbers = [8, 17, 54, 22, 35]
n2 = "".join(numbers)
```

```
    numbers = ["8", "17", "54", "22", "35"]
n2 = "".join(numbers)
```

22. You can use the split() method to split a string into a list of strings based on a specified
    word
    character
    delimiter
    punctuation mark