CCPS 844 Data Mining Lab 1

Answer the following questions and submit a PDF file on the D2L.

You can consult the following link to revise/improve your Python programming skills

https://docs.python.org/2/tutorial/ (https://docs.python.org/2/tutorial/)

Let's begin by revising

- Basic operations
- Data Structures
- creating/using functions

What is 7 to the power of 4?

```
In [15]: 7 ** 4
```

```
Out[15]: 2401
```

Split this string:

```
s = "Hi there Sam!"
```

into a list.

```
In [16]: s = 'Hi there Sam!'
```

```
In [17]: s.split()
```

```
Out[17]: ['Hi', 'there', 'Sam!']
```

Given the variables:

```
planet = "Earth"
diameter = 12742
```

Use .format() to print the following string:

```
The diameter of Earth is 12742 kilometers.
```

In [18]:
```
planet = "Earth"
diameter = 12742
```

In [19]:
```
'The diameter of {} is {}'.format(planet, diameter)
```

```
The diameter of Earth is 12742 kilometers.
```

Given this nested list, use indexing to grab the word "hello"

In [20]:
```
lst = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]
```

In [21]:
```
lst[3][1][2][0]
```

Out[21]:
```
'hello'
```

Given this nest dictionary grab the word "hello". Be prepared, this will be annoying/tricky

In [22]:
```
d = {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello'
]}]}]}
```

In [23]:
```
d['k1'][3]['tricky'][3]['target'][3]
```

Out[23]:
```
'hello'
```

What is the main difference between a tuple and a list?

In [24]:
```
# Tuple is immutable
# A list is mutable
```

Create a function that grabs the email website domain from a string in the form:

```
user@domain.com
```

So for example, passing "user@domain.com" would return: domain.com

In [25]:
```
def domainGet(email):
    return email.split('@')[1]
```

In [26]:
```
domainGet('user@domain.com')
```

Out[26]:
```
'domain.com'
```

Create a basic function that returns True if the word 'dog' is contained in the input string. Don't worry about edge cases like a punctuation being attached to the word dog, but do account for capitalization.

In [27]:
```
def findDog(inputStr):
    return 'dog' in inputStr.lower()
```

In [28]:
```
findDog('Is there a dog here?')
```

Out[28]: True

Create a function that counts the number of times the word "dog" occurs in a string. Again ignore edge cases.

In [29]:
```
def countDog(inputStr):
    return inputStr.lower().count('dog')
```

In [30]:
```
countDog('This dog runs faster than the other dog dude!')
```

Out[30]: 2

Use lambda expressions and the filter() function to filter out words from a list that start with the letter 's'. For example:

```
seq = ['soup','dog','salad','cat','great']
```

should be filtered down to:

```
['soup','salad']
```

In [37]:
```
seq = ['soup','dog','salad','cat','great']
```

In [38]:
```
list(filter(lambda x: x.startswith('s'), seq))
```

Out[38]: ['soup', 'salad']

Learning Pandas Basics

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

http://pandas.pydata.org/pandas-docs/stable/ (http://pandas.pydata.org/pandas-docs/stable/)

Data Frame

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object. There are several ways to create a DataFrame. One way way is to use a dictionary.

In [42]:
```
#Importing the Pandas library
import pandas as pd
```

Create a dataframe df from the dictionary dict

In [43]:
```
dict = {"country": ["Brazil", "Russia", "Pakistan", "China", "South Africa"],
       "capital": ["Brasilia", "Moscow", "Islamabad", "Beijing", "Pretoria"],
       "area": [8.516, 17.10, 3.286, 9.597, 1.221],
       "population": [200.4, 143.5, 1252, 1357, 52.98] }
```

In [44]:
```
df = pd.DataFrame(dict)
```

Out[44]:

	area	capital	country	population
0	8.516	Brasilia	Brazil	200.40
1	17.100	Moscow	Russia	143.50
2	3.286	Islamabad	Pakistan	1252.00
3	9.597	Beijing	China	1357.00
4	1.221	Pretoria	South Africa	52.98

In [45]:
```
indices = ["BR", "RU", "PK", "CH", "SA"]
```

Set the list indices as index of the dataframe brics

In [46]:
```
df.index = indices
```

Out[46]:

	area	capital	country	population
BR	8.516	Brasilia	Brazil	200.40
RU	17.100	Moscow	Russia	143.50
PK	3.286	Islamabad	Pakistan	1252.00
CH	9.597	Beijing	China	1357.00
SA	1.221	Pretoria	South Africa	52.98

Select records from the dataframe indexed with 'RU' and 'SA'

In [47]:
```
df.loc[['RU', 'SA']]
```

Out[47]:

	area	capital	country	population
RU	17.100	Moscow	Russia	143.50
SA	1.221	Pretoria	South Africa	52.98

In [48]:
```
#Importing the Numpy library
import numpy as np
raw_data = {'first_name': ['Jason', 'Molly', np.nan, np.nan, np.nan],
        'nationality': ['CAN', 'CAN', 'France', 'UK', 'UK'],
        'age': [42, 52, 36, 24, 70]}
```

Create a dataframe from raw_data

In [49]:
```
df = pd.DataFrame(raw_data)
```

Out[49]:

	first_name	nationality	age
0	Jason	CAN	42
1	Molly	CAN	52
2	NaN	France	36
3	NaN	UK	24
4	NaN	UK	70

Select all elderly Canadians from the dataframe (age > 50)

In [51]:
```
df.query('age > 50')
```

Out[51]:

	first_name	nationality	age
1	Molly	CAN	52