Exp-1: Accessing and Understanding the data in Python using Pandas.

Roll NO :- 22354 Name:- Malhar Choure

Performance Date: 02/02/2022

Submission Date: 09/02/2022

Objectives:

1. Loading a simple delimited data file.
- Data Source: see attached csv file named as: `gap-every-five-years.csv`
1. Counting how many rows and columns were loaded.
2. Determining which types of data was loaded.
3. Looking at different parts of the data by subsetting rows and columns.

Important Shortcut Keys

- A -> To **create** cell **above**
- B -> To **create** Cell **below**
- D D -> For **deleting** the cell
- M -> To **markdown** the Cell
- Y -> For **code** the cell
- Z -> To **undo** the deleted cell

1. Import Python Libraries: pandas

- Pandas is an open source Python library for data analysis. It gives Python the ability to work with spreadsheet-like data for fast data loading, manipulating, aligning, and merging, among other functions.
- To give Python these enhanced features, Pandas introduces two new data types to Python: Series and DataFrame. The DataFrame represents your entire spreadsheet or rectangular data, whereas the Series is a single column of the DataFrame. A Pandas DataFrame can also be thought of as a dictionary or collection of Series objects.
- When given a data set, we first load it and begin looking at its structure and contents. The simplest way of looking at a data set is to examine and subset specific rows and columns. We can see which type of information is stored in each column, and can start looking for patterns by aggregating descriptive statistics.
- Since Pandas is not part of the Python standard library, we have to first tell Python to load (import) the library.

```
# Import the libraries pandas and numpy
import pandas as pd
import numpy as np
```

2. Read and then Print the Data File in Python

```
# Load (Read) the csv data file (Caution: Use backslash in writing
location of the file) & store it in variable df
df= pd.read_csv('C:/Users/malhar/Documents/DA pyhton programs/expt
1/gap-every-five-years.csv')

# Show the Loaded CSV data
print(df)
```

```
          country continent  year  lifeExp       pop   gdpPercap
0     Afghanistan      Asia  1952   28.801   8425333  779.445314
1     Afghanistan      Asia  1957   30.332   9240934  820.853030
2     Afghanistan      Asia  1962   31.997  10267083  853.100710
3     Afghanistan      Asia  1967   34.020  11537966  836.197138
4     Afghanistan      Asia  1972   36.088  13079460  739.981106
...           ...       ...   ...      ...       ...         ...
1699     Zimbabwe    Africa  1987   62.351   9216418  706.157306
1700     Zimbabwe    Africa  1992   60.377  10704340  693.420786
1701     Zimbabwe    Africa  1997   46.809  11404948  792.449960
1702     Zimbabwe    Africa  2002   39.989  11926563  672.038623
1703     Zimbabwe    Africa  2007   43.487  12311143  469.709298

[1704 rows x 6 columns]
```

Note:

The above data shows:

For various countries: **life expectancy** (lifeExp), **population** (pop) and **GDP per Capita** (gdpPercap) in every 5 years.

3. Get the Data Frame Information

- `shape`: Get the number of rows and columns of the data frame
- `columns`: Get the Columns names
- `dtypes`: Get the data type of each column
- `info`: Get the more information about the data types and missing values information
- `head()`, `tail()`: Get the first and last five obseravtions of the data frame, respectively.

```
# Find the number of rows and columns in the data frame
print(df.shape)
```

```
(1704, 6)
```

```
# Print (Get) the columns labels (names) (heading of each column)
print(df.columns)
```

```
Index(['country', 'continent', 'year', 'lifeExp', 'pop', 'gdpPercap'],
dtype='object')
```

```
# Print (Get) the data types of each columns of the data frame

print(df.dtypes)
```

```
country       object
continent     object
year           int64
lifeExp      float64
pop            int64
gdpPercap    float64
dtype: object
```

```
# Show the first 5 observations of the data frame

print(df.head())
```

```
       country continent  year  lifeExp       pop   gdpPercap
0  Afghanistan      Asia  1952   28.801   8425333  779.445314
1  Afghanistan      Asia  1957   30.332   9240934  820.853030
2  Afghanistan      Asia  1962   31.997  10267083  853.100710
3  Afghanistan      Asia  1967   34.020  11537966  836.197138
4  Afghanistan      Asia  1972   36.088  13079460  739.981106
```

```
# Show th last 5 observations of the data frame

print(df.tail())
```

```
       country continent  year  lifeExp       pop   gdpPercap
1699  Zimbabwe    Africa  1987   62.351   9216418  706.157306
1700  Zimbabwe    Africa  1992   60.377  10704340  693.420786
1701  Zimbabwe    Africa  1997   46.809  11404948  792.449960
1702  Zimbabwe    Africa  2002   39.989  11926563  672.038623
1703  Zimbabwe    Africa  2007   43.487  12311143  469.709298
```

Read the following table to know more: Pandas data types Vs Python data types

pandas Type	Python Type	Description
object	string	most common data type
int64	int	whole number
float64	float	numbers with decimal
datetime64	datetime	datetime is found in the Python standard library which is not loaded by default

```
# Print (Get) the more information of the data types of each columns
```

Observe non-null in the above output

non-null in any particular column means there is no missing data in that particular cloumn.

4. Looking into the Rows, Columns and Cells

- Different Methods of Indexing Rows (or Columns)

Subset Method
loc
iloc

4.1 Subsetting the Rows

- by `loc`
- by `iloc`
- by `head(n)` and `tail(n)`: shows the first n rows data and last n rows data, respectively. For example, If we give n=1 then head(n=1) will show first row and tail(n=1) will show last row.

```
# Single row data
# Show the first row of the data frame [Caution: Python counts from 0]
# using loc command

print(df.loc[[0],:])
```

```
       country continent  year  lifeExp      pop   gdpPercap
0  Afghanistan      Asia  1952   28.801  8425333  779.445314
```

```
# Single row data
# Show the first row of the data frame [Caution: Python counts from 0]
# using iloc command
print(df.iloc[[0],:])
```

```
       country continent  year  lifeExp      pop   gdpPercap
0  Afghanistan      Asia  1952   28.801  8425333  779.445314
```

```
# Single row data
# Show the first row of the data frame [Caution: Python counts from 0]
# using head function
print(df.head(n= 1))
```

```
       country continent  year  lifeExp      pop   gdpPercap
0  Afghanistan      Asia  1952   28.801  8425333  779.445314
```

```
# Single row data
# Show the 15th row of the data frame [Caution: Python counts from 0]
# using loc command

print(df.loc[[14],:])
```

```
    country continent  year  lifeExp      pop    gdpPercap
14  Albania    Europe  1962    64.82  1728137  2312.888958
```

```
# Single row data
# Save the 15th row data into its own variable row15_df & also show
the 15th row of the data frame [Python counts from 0]
# using iloc command

print(df.iloc[[14],:])
```

```
    country continent  year  lifeExp      pop    gdpPercap
14  Albania    Europe  1962    64.82  1728137  2312.888958
```

```
# Single row data
# Show the last row of the data frame
# using tail function

print(df.tail(1))
```

```
       country continent  year  lifeExp       pop   gdpPercap
1703  Zimbabwe    Africa  2007   43.487  12311143  469.709298
```

```
# Single row data
# Show the last row of the data frame
# using iloc command
print(df.iloc[[-1],:])
    # Compare with previous code using tail fuction
```

```
       country continent  year  lifeExp       pop   gdpPercap
1703  Zimbabwe    Africa  2007   43.487  12311143  469.709298
```

Note: difference between iloc and loc

With iloc, we can pass -1 to get the last row --- something we could not do with loc.

That is in previous code if you write `print(df.loc[-1])`, it will show error. Try and understand difference between label vs index.

Exercise 1: Show the last row of the data frame (using loc command).

Hint: You need to write some extra lines of code to do the task.

Your Solution Code (write in the cell given below):

```
# Solution for Exercise 1:

# Multiple rows data (all at once)
# Show the 1st, 100th and 1000th rows data (all at once)
# using loc function
print(df.loc[[0,99,999],:])
    # Note the two square brackets
```

```
         country continent  year  lifeExp       pop    gdpPercap
0    Afghanistan      Asia  1952   28.801   8425333   779.445314
99    Bangladesh      Asia  1967   43.453  62821884   721.186086
999     Mongolia      Asia  1967   51.253   1149500  1226.041130
```

```
# Multiple rows data (all at once)
# Show the 1st, 100th and 1000th rows data (all at once)
# using iloc function
print(df.iloc[[0,99,999],:])
     # Note the two square brackets
```

```
         country continent  year  lifeExp       pop    gdpPercap
0    Afghanistan      Asia  1952   28.801   8425333   779.445314
99    Bangladesh      Asia  1967   43.453  62821884   721.186086
999     Mongolia      Asia  1967   51.253   1149500  1226.041130
```

4.2 Subsetting the Columns

by Name

- `dataframevariable['column_name']`: Get only one column data.
- `dataframevariable[['ith_column_name', 'jth_column_name', ..., 'kth_column_name']]`: Get multiple columns data.

by loc and iloc command

- by `loc`
- by `iloc`

by Range

- You can use the built-in `range(start, stop, step)` function to create a range of values in Python. This way you can specify beginning and end values, and Python will automatically create a range of values in between. By default, every value between the beginning and the end (inclusive left, exclusive right) will be created, unless you specify a step.
- In Python 3, the range function returns a generator. For example, when range(5) is called, five integers are returned: 0 – 4.
- We will see that we subset columns using a list of integers (in iloc method). Since `range` returns a generator, we have to convert the generator to a list first.
- We use range method for multilple columns data.

by Slicing

- Python's slicing syntax, :, is similar to the range syntax. Instead of a function that specifies start, stop, and step values delimited by a comma, we separate the values with the colon.
- Slicing can be seen as a shorthand means to the same thing as range.
- The colon syntax for slicing only has meaning when slicing and subsetting values, and has no inherent meaning on its own.

```
# Single column data
# Get first column (namely country) data and save it to its own
variable (country_df)
# using by name
column_data=df['country']
print(column_data)
```

```
0       Afghanistan
1       Afghanistan
2       Afghanistan
3       Afghanistan
4       Afghanistan
           ...
1699       Zimbabwe
1700       Zimbabwe
1701       Zimbabwe
1702       Zimbabwe
1703       Zimbabwe
Name: country, Length: 1704, dtype: object
```

```
# Single column data
# Show the first 5 observations of country column
print(column_data.head())
```

```
0    Afghanistan
1    Afghanistan
2    Afghanistan
3    Afghanistan
4    Afghanistan
Name: country, dtype: object
```

```
# Single column data
# Show the last 5 observations of country column
print(column_data.tail())
```

```
1699    Zimbabwe
1700    Zimbabwe
1701    Zimbabwe
1702    Zimbabwe
1703    Zimbabwe
Name: country, dtype: object
```

```
# Multiple columns data

# Question: Show the last 5 observations of first ('country') column,
third ('year') column and fifth ('pop') column.
# using by name

# Answer:
# first save the given three coulmns data in a new variable
  # Note the two square braces
```

```
three_column_data= df[['country','year','pop']]

# Show the last 5 observation data of the variable subset1
print(three_column_data.tail())

#print certain rows
print(three_column_data[0:3])

#printing alternate rows till 10
print(three_column_data[0:10:2])

#printing all using slicing
print(three_column_data[:])
```

```
       country  year       pop
1699  Zimbabwe  1987   9216418
1700  Zimbabwe  1992  10704340
1701  Zimbabwe  1997  11404948
1702  Zimbabwe  2002  11926563
1703  Zimbabwe  2007  12311143
       country  year       pop
0  Afghanistan  1952   8425333
1  Afghanistan  1957   9240934
2  Afghanistan  1962  10267083
       country  year       pop
0  Afghanistan  1952   8425333
2  Afghanistan  1962  10267083
4  Afghanistan  1972  13079460
6  Afghanistan  1982  12881816
8  Afghanistan  1992  16317921
          country  year       pop
0     Afghanistan  1952   8425333
1     Afghanistan  1957   9240934
2     Afghanistan  1962  10267083
3     Afghanistan  1967  11537966
4     Afghanistan  1972  13079460
...           ...   ...       ...
1699     Zimbabwe  1987   9216418
1700     Zimbabwe  1992  10704340
1701     Zimbabwe  1997  11404948
1702     Zimbabwe  2002  11926563
1703     Zimbabwe  2007  12311143

[1704 rows x 3 columns]
```

```
# Single column data
# Show the first column of the data frame
# using loc cammand
print(df.loc[:,'country'])
```

```
0       Afghanistan
1       Afghanistan
2       Afghanistan
3       Afghanistan
4       Afghanistan
           ...
1699       Zimbabwe
1700       Zimbabwe
1701       Zimbabwe
1702       Zimbabwe
1703       Zimbabwe
Name: country, Length: 1704, dtype: object
```

```
# Single column data
# Show the first column of the data frame [Caution: Python counts from
0]
# using iloc command
print(df.iloc[:,[0]])
```

```
          country
0     Afghanistan
1     Afghanistan
2     Afghanistan
3     Afghanistan
4     Afghanistan
...           ...
1699     Zimbabwe
1700     Zimbabwe
1701     Zimbabwe
1702     Zimbabwe
1703     Zimbabwe

[1704 rows x 1 columns]
```

```
# Multiple columns data
# Show the first ('country') column, third ('year') column and fifth
('pop') column data.
# using loc command
print(df.loc[:,['country','year','pop']])
```

```
          country  year       pop
0     Afghanistan  1952   8425333
1     Afghanistan  1957   9240934
2     Afghanistan  1962  10267083
3     Afghanistan  1967  11537966
4     Afghanistan  1972  13079460
...           ...   ...       ...
1699     Zimbabwe  1987   9216418
1700     Zimbabwe  1992  10704340
1701     Zimbabwe  1997  11404948
1702     Zimbabwe  2002  11926563
```

```
1703     Zimbabwe  2007  12311143

[1704 rows x 3 columns]
```

```
# Multiple columns data
# Question: save first ('country'), third ('year') and fifth ('pop')
columns data in a variable and print its first 6 data.
# using iloc function
print(df.iloc[:,[0,2,4]])

#print only 0 to three rows of three columns

print(df.iloc[0:6,[0,2,4]])
# Answer:
# first save the given three coulmns data in a new variable (subset2)
  # Note the two square braces

#range creation

# Show the first 6 data of the variable (subset2)
```

```
          country  year       pop
0     Afghanistan  1952   8425333
1     Afghanistan  1957   9240934
2     Afghanistan  1962  10267083
3     Afghanistan  1967  11537966
4     Afghanistan  1972  13079460
...           ...   ...       ...
1699     Zimbabwe  1987   9216418
1700     Zimbabwe  1992  10704340
1701     Zimbabwe  1997  11404948
1702     Zimbabwe  2002  11926563
1703     Zimbabwe  2007  12311143

[1704 rows x 3 columns]
       country  year       pop
0  Afghanistan  1952   8425333
1  Afghanistan  1957   9240934
2  Afghanistan  1962  10267083
3  Afghanistan  1967  11537966
4  Afghanistan  1972  13079460
5  Afghanistan  1977  14880372
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
# Multiple columns data
# Question: get the first 4 columns data
# using Range method
range_def=list(range(4))
print(range_def)
# Answer:
```

```
# create a range of integers from 0 to 3 inclusive
print(df.iloc[:,range_def])

# subset the dataframe with the range
```

```
[0, 1, 2, 3]
          country continent  year  lifeExp
0     Afghanistan      Asia  1952   28.801
1     Afghanistan      Asia  1957   30.332
2     Afghanistan      Asia  1962   31.997
3     Afghanistan      Asia  1967   34.020
4     Afghanistan      Asia  1972   36.088
...           ...       ...   ...      ...
1699     Zimbabwe    Africa  1987   62.351
1700     Zimbabwe    Africa  1992   60.377
1701     Zimbabwe    Africa  1997   46.809
1702     Zimbabwe    Africa  2002   39.989
1703     Zimbabwe    Africa  2007   43.487

[1704 rows x 4 columns]
```

```
# Multiple columns data
# Question: get the last 3 columns data
# using Range method

# Answer:
# create a range of integers from 3 to 5 inclusive
range_def=list(range(3,6))
print(df.iloc[:,range_def])

# subset the dataframe with the range
```

```
      lifeExp       pop   gdpPercap
0      28.801   8425333  779.445314
1      30.332   9240934  820.853030
2      31.997  10267083  853.100710
3      34.020  11537966  836.197138
4      36.088  13079460  739.981106
...       ...       ...         ...
1699   62.351   9216418  706.157306
1700   60.377  10704340  693.420786
1701   46.809  11404948  792.449960
1702   39.989  11926563  672.038623
1703   43.487  12311143  469.709298

[1704 rows x 3 columns]
```

```
# Multiple columns data
# Question: get the data of first, third and fifth column data
# using Range method
# Hint: (first column - python index 0, third column - python index 2,
```

```
fifth column - python index 4)

# Answer:
# create a range of integers from 0 to 5 exclusive, every other data
range_def=list(range(0,6,2))
print(df.iloc[:,range_def])
# subset the dataframe with the range
```

```
          country  year       pop
0     Afghanistan  1952   8425333
1     Afghanistan  1957   9240934
2     Afghanistan  1962  10267083
3     Afghanistan  1967  11537966
4     Afghanistan  1972  13079460
...           ...   ...       ...
1699     Zimbabwe  1987   9216418
1700     Zimbabwe  1992  10704340
1701     Zimbabwe  1997  11404948
1702     Zimbabwe  2002  11926563
1703     Zimbabwe  2007  12311143

[1704 rows x 3 columns]
```

```
# Multiple columns data
# Question: get the last 3 columns data
# using Slicing method

# Answer:
# subset the dataframe with the slicing the last 3 columns (3 to 5
inclusive)
print(df.iloc[:,3:6])
```

```
      lifeExp       pop   gdpPercap
0      28.801   8425333  779.445314
1      30.332   9240934  820.853030
2      31.997  10267083  853.100710
3      34.020  11537966  836.197138
4      36.088  13079460  739.981106
...       ...       ...         ...
1699   62.351   9216418  706.157306
1700   60.377  10704340  693.420786
1701   46.809  11404948  792.449960
1702   39.989  11926563  672.038623
1703   43.487  12311143  469.709298

[1704 rows x 3 columns]
```

```
# Multiple columns data
# Question: get the first 3 columns data
# using Slicing method
```

```
# Answer:
# subset the dataframe with the slicing the first 3 columns (0 to 2
inclusive)
  # or print(df.iloc[:, :3])
print(df.iloc[:,:3])
```

```
          country continent  year
0     Afghanistan      Asia  1952
1     Afghanistan      Asia  1957
2     Afghanistan      Asia  1962
3     Afghanistan      Asia  1967
4     Afghanistan      Asia  1972
...           ...       ...   ...
1699     Zimbabwe    Africa  1987
1700     Zimbabwe    Africa  1992
1701     Zimbabwe    Africa  1997
1702     Zimbabwe    Africa  2002
1703     Zimbabwe    Africa  2007

[1704 rows x 3 columns]
```

```
# Multiple columns data
# Question: get the data of third, fourth and fifth column data
# using Slicing method

# Answer:
# subset the dataframe with the slicing the columns 2 to 4 inclusive
print(df.iloc[:,3:6])
```

```
      lifeExp       pop   gdpPercap
0      28.801   8425333  779.445314
1      30.332   9240934  820.853030
2      31.997  10267083  853.100710
3      34.020  11537966  836.197138
4      36.088  13079460  739.981106
...       ...       ...         ...
1699   62.351   9216418  706.157306
1700   60.377  10704340  693.420786
1701   46.809  11404948  792.449960
1702   39.989  11926563  672.038623
1703   43.487  12311143  469.709298

[1704 rows x 3 columns]
```

```
#40th row of country column
print(df.loc[47,'country'])
#using iloc
print(df.iloc[47,0])
#using country dusing pop
print(df.loc[47,['country','pop']])
print(df.iloc[47,[0,4]])
```

```
#data for two years
print(df.iloc[:2,[0,2,4]])
```

```
Angola
Angola
country      Angola
pop        12420476
Name: 47, dtype: object
country      Angola
pop        12420476
Name: 47, dtype: object
       country  year      pop
0  Afghanistan  1952  8425333
1  Afghanistan  1957  9240934
```

```
# Multiple columns data
# Question: get the every other first 5 columns
# using Slicing method

# Answer: every other first 5 columns are first, third and fifth
columns
# subset the dataframe with the slicing the columns 0 to 5 inclusive
with step 2

df.iloc[:,0:6:2]
```

```
          country  year       pop
0     Afghanistan  1952   8425333
1     Afghanistan  1957   9240934
2     Afghanistan  1962  10267083
3     Afghanistan  1967  11537966
4     Afghanistan  1972  13079460
...           ...   ...       ...
1699     Zimbabwe  1987   9216418
1700     Zimbabwe  1992  10704340
1701     Zimbabwe  1997  11404948
1702     Zimbabwe  2002  11926563
1703     Zimbabwe  2007  12311143

[1704 rows x 3 columns]
```

Exercise 2: What is the result in each of the following cases? Verify and Justify.

- `df.iloc[:, 0:6:]`
- `df.iloc[:, 0::2]`
- `df.iloc[:, :6:2]`
- `df.iloc[:, ::2]`
- `df.iloc[:, ::]`

4.3 Subsetting the Cell (Rows and Columns both)

- by `loc`
- by `iloc`

```
# Specific row and specific column data
# Get the 43rd country name in our data frame (df)
# using loc command

print(df.loc[[42],["country"]])
```

```
   country
42  Angola
```

```
# Specific row and specific column data
# Get the 43rd country name in our data frame (df)
# using iloc command

print(df.iloc[42,[0]])
```

```
country    Angola
Name: 42, dtype: object
```

```
# Specific row and multiple columns data
# Get the 43rd country name and its population in our data frame (df)
# using loc command

print(df.loc[42:42, ["country", "pop"]])
```

```
   country      pop
42  Angola  7016384
```

```
# Specific row and multiple columns data
# Get the 43rd country name and its population in our data frame (df)
# using iloc command

  # country is 1st column and population is 5th column
print(df.iloc[42:43, [0, 4]])
```

```
   country      pop
42  Angola  7016384
```

```
# Multiple rows and specific column data
# Get the 43rd and 54th country names in our data frame (df)
# using loc command

print(df.loc[[42, 53], ["country"]])
```

```
      country
42     Angola
53  Argentina
```

```
# Multiple rows and specific column data
# Get the 43rd and 54th country names in our data frame (df)
# using iloc command
print(df.iloc[[42, 53], [0]])
```

```
      country
42     Angola
53  Argentina
```

```
# Multiple rows and multiple columns data
# Get the 1st, 100th and 1000th rows data
# Get the corresponding data of columns 'country', 'lifeExp' and
'gdpPercap'
# using loc command

print(df.loc[[0, 99, 999], ['country', 'lifeExp', 'gdpPercap']])
```

```
         country  lifeExp    gdpPercap
0    Afghanistan   28.801   779.445314
99    Bangladesh   43.453   721.186086
999     Mongolia   51.253  1226.041130
```

```
# Multiple rows and multiple columns data
# Get the 1st, 100th and 1000th rows data
# Get the corresponding data of columns 'country', 'lifeExp' and
'gdpPercap'
# using iloc command

print(df.iloc[[0,99,999],[0,3,5]])
```

```
         country  lifeExp    gdpPercap
0    Afghanistan   28.801   779.445314
99    Bangladesh   43.453   721.186086
999     Mongolia   51.253  1226.041130
```