Session 4: Pandas, data import/export, descriptive statistics

Module BUSN9690

Business Statistics with Python

Questions from last week

- Difference between random.randint and random.randrange
  - random.randrange(0, 1) will not consider the last item 1, while random.randint(0, 1) returns a choice inclusive of the last item 1
    - random.randrange(0,2) works (only returns 0, or 1), but random.randint(0,2) returns 0, 1, or 2
  - random.randrange([start], stop[, step]), but not random.randint([start], stop[, step])
    - random.randrange(0,3,2) works (only returns 0, or 2), but random.randint(0,3,2) does not work
- Recordings of the lecture sessions
- Assessments
- Attendance
  - Group 1: Click here to join the meeting, 11:00—12:00 am, Thursday, London time; PC ROOM 2, Pears Building
  - Group 2: Click here to join the meeting 9:00—10:00 am, Thursday, London time; PC Room 1, Sibson Building

Other useful methods in numpy: Arithmetic operation

- import numpy as np #np is just an alias, an acronym of numpy

- arr1 = np.array([10, 20, 30, 40, 50, 60])
- arr2 = np.array([3, 5, 6, 8, 2, 1])

- arrAdd = np.add(arr1, arr2)#elements in arr1 add those in arra2. output: [13, 25, 36, 48, 52, 61])
- arrSub = np.subtract(arr1, arr2) #output: [7, 15, 24, 32, 48, 59]
- arrTimes = np.multiply(arr1, arr2) # output: [30, 100, 180, 320, 100, 60]
- arrDiv = np.divide(arr1, arr2) # output: ([3.33333333, 4., 5., 5., 25.,60.])
- arrPow = np.power(arr1, arr2) #output: ([1000,3200000,729000000,-520093696,2500,60],dtype=int32)
- arrMod = np.mod(arr1, arr2) # mode of output: [1, 0, 0, 0, 0, 0]
- arrLog = np.log2(arr1)#log at base 2 of all elements of arr1. output:

Other useful methods: basic statistics

- Basic statistics
  - import numpy as np

  - arr1 = np.array([10, 20, 30, 40, 50, 60])
  - arr2 = np.array([3, 5, 6, 8, 2, 1])

  - arrSum = np.sum([arr1,arr2])#sum of all elements of arr1 and arr2. output: 235
  - arrMin = np.amin(arr1) #the minimum value of the elements in arr1. output: 10
  - arrMax = np.amax(arr1) #the maximum value of the elements in arr1. output: 60
  - arrMean = np.mean(arr1) #the mean value of the elements in arr1. output: 35

Other useful methods: frequency & indices

- find the unique elements in a numpy array. Syntax:

numpy.unique(arr, return_index=False, return_inverse=False, return_counts=False, axis=None)

- arr : Numpy array in which we want to find the unique values.
- return_index : optional bool flag. If True returns an array of indices of first occurrence of each unique value.
- return_counts : optional bool flag. If True returns an array of occurrence count of each unique value.
- axis : If not provided then will act on flattened array. If 0 or 1 then acts on row or column wise.

  - import numpy as np
  - arr1 = np.array([11, 11, 12, 13, 14, 15, 16, 17, 12, 13, 11, 14, 18])
  - arrUnique = np.unique(arr1) #arrUnique==[11, 12, 13, 14, 15, 16, 17, 18]
  - uniqueValues, indicesList = np.unique(arr1, return_index=True) #to find a tuple of unique values & their first index location from a numpy array. Unique Values : [11,12,13,14,15,16,17,18]. Indices of Unique Values : [0 2 3 4 5 6 7 12]
  - uniqueValues, occurCount= np.unique(arr1, return_counts =True) #to find a tuple of unique values & their first index location from a numpy array. Unique Values : [11 12 13 14 15 16 17 18]. Counts of Unique Values : [3 2 2 2 1 1 1 1]

NumPy Random

Shuffling, random number generating

- Shuffling: Shuffling aims to change the arrangement of elements in-place. i.e. in the array itself.
  - from numpy import random
  - import numpy as np
  - arr = np.array([1, 2, 3, 4, 5])
  - random.shuffle(arr)
  - print(arr)

- To generate a random normal distribution of size
  - from numpy import random
  - x = random.normal(size=(2, 30)) #generate two series of random numbers, each with 30 values
  - print(x)

  - How to generate a random distribution of other type?

Useful tutorials

- Numpy
  - https://thispointer.com/numpy-array-tutorials/
  - https://www.w3schools.com/python/numpy/default.asp

Pandas

- pandas: performs five significant steps required for data analysis, i.e.,
  - load,
  - manipulate,
  - prepare,
  - model, and
  - analyse
- pandas
  - https://www.w3schools.com/python/pandas/default.asp
- matplotlib
  - https://www.w3schools.com/python/matplotlib_intro.asp

Pandas DataFrame

- Pandas DataFrame is two-dimensional, size-mutable, potentially heterogeneous tabular data structure with labelled axes (rows and columns)

Name	Business statistics	Simulation	Machine Learning	Big Data	Dissertation
John	65	60	70	80	65
Anna	72	65	71	65	62
Emma	56	64	67	63	65
Nigel	78	70	76	80	75
Ben	72	68	70	76	72

Create a DataFrame from a list or a dictionary

Create a DataFrame from a list

```
# import pandas as pd
    import pandas as pd

# list of strings
    listName =
    ['John','Anna','Emma','Nigel','Ben']

# Calling DataFrame constructor on list
    dfName1= pd.DataFrame(listName)
    print(dfName1)
```

Create a DataFrame from a dictionary

```
# import pandas as pd
    import pandas as pd

# dictionary of students
    dictName = {'Module':
    ['BusinessStats','MachineLearning','Simulation','
    BigData', 'MachineLearning',
    'MachineLearning'], 'StudentLogin':['A00010',
    'C00030', 'E00120','F00130', 'E00120',
    'G00120'],'Marks':[70,65,58,81,73,65]
    }

# Calling DataFrame constructor on dictionary
    dfName2 = pd.DataFrame(dictName)
    print(dfName2)
```

Creating a DataFrame directly

```
# import pandas as pd
    import pandas as pd

# dictionary of students
    marks=pd.DataFrame({'Module': ['BusinessStats','MachineLearning','Simulation','BigData',
    'MachineLearning', 'MachineLearning'], 'StudentLogin':['A00010', 'C00030', 'E00120','F00130',
    'E00120', 'G00120'],'Marks':[70,65,58,81,73,65]})

    print(marks)
```

```
In [13]: marks
Out[13]:
             Module StudentLogin  Marks
0    BusinessStats        A00010     70
1  MachineLearning        C00030     65
2       Simulation        E00120     58
3          BigData        F00130     81
4  MachineLearning        E00120     73
5  MachineLearning        G00120     65
```

Dealing with Rows and Columns: Selecting, deleting, adding, and renaming

```
In [13]: marks
Out[13]:
             Module StudentLogin  Marks
0    BusinessStats        A00010     70
1  MachineLearning        C00030     65
2       Simulation        E00120     58
3          BigData        F00130     81
4  MachineLearning        E00120     73
5  MachineLearning        G00120     65
```

- Column Selection

```
# select two columns
marks[['Module', 'Marks']]
```

```
In [14]: marks[['Module', 'Marks']]
Out[14]:
             Module  Marks
0    BusinessStats     70
1  MachineLearning     65
2       Simulation     58
3          BigData     81
4  MachineLearning     73
5  MachineLearning     65
```

- Row Selection: DataFrame.loc[] method is used to retrieve rows from Pandas DataFrame

```
# select the second row
marks.loc[1]
```

```
In [15]: marks.loc[1]
Out[15]:
Module          MachineLearning
StudentLogin             C00030
Marks                        65
Name: 1, dtype: object
```

Dealing with Rows and Columns: Selecting, deleting, adding, and renaming

- DataFrame.drop() method is used to delete rows and columns by position from Pandas DataFrame.

```
In [13]: marks
Out[13]:
             Module StudentLogin  Marks
0     BusinessStats        A00010     70
1   MachineLearning        C00030     65
2        Simulation        E00120     58
3           BigData        F00130     81
4   MachineLearning        E00120     73
5   MachineLearning        G00120     65
```

```
# delete two columns named Module and Marks
marks.drop(['Module','Marks'], axis=1)
```

#axis=0 denotes row, and axis=1 denotes column

```
In [17]: marks.drop(['Module','Marks'], axis=1)
Out[17]:
  StudentLogin
0        A00010
1        C00030
2        E00120
3        F00130
4        E00120
5        G00120
```

```
# delete the 1st two rows
marks.drop([0,1])
```

```
In [20]: marks.drop([0,1])
Out[20]:
             Module StudentLogin  Marks
2        Simulation        E00120     58
3           BigData        F00130     81
4   MachineLearning        E00120     73
5   MachineLearning        G00120     65
```

Dealing with Rows and Columns: Selecting, deleting, adding, and renaming

#add a new column called 'Grades' to marks.

marks['Grade'] = ['Dis', 'Merit', 'Pass', 'Dis', 'Dis','Merit']

```
In [23]: marks['Grade'] = ['Dis', 'Merit', 'Pass', 'Dis', 'Dis','Merit']

In [24]: marks
Out[24]:
            Module StudentLogin  Marks  Grade
0    BusinessStats       A00010     70    Dis
1  MachineLearning       C00030     65  Merit
2       Simulation       E00120     58   Pass
3          BigData       F00130     81    Dis
4  MachineLearning       E00120     73    Dis
5  MachineLearning       G00120     65  Merit
```

add a row onto the dataframe

new_row={'Module':'BigData','StudentLogin':'H00120','Marks':56,'Grade':'Pass'}

marks = marks.append(new_row, ignore_index=True)

#without ignore_index=True, an error will be thrown

```
In [13]: marks
Out[13]:
            Module StudentLogin  Marks
0    BusinessStats       A00010     70
1  MachineLearning       C00030     65
2       Simulation       E00120     58
3          BigData       F00130     81
4  MachineLearning       E00120     73
5  MachineLearning       G00120     65
```

```
Out[9]:
            Module StudentLogin  Marks  Grade
0    BusinessStats       A00010     70    Dis
1  MachineLearning       C00030     65  Merit
2       Simulation       E00120     58   Pass
3          BigData       F00130     81    Dis
4  MachineLearning       E00120     73    Dis
5  MachineLearning       G00120     65  Merit
6          BigData       H00120     56   Pass
```

Dealing with Rows and Columns: Selecting, deleting, adding, and renaming

#rename the column names.

>>>marks.columns=['Module1', 'StudentLogin1', 'Marks1', 'Grade1']

```
In [11]: marks.columns=['Module1', 'StudentLogin1', 'Marks1', 'Grade1']

In [12]: marks
Out[12]:
           Module1 StudentLogin1  Marks1 Grade1
0    BusinessStats        A00010      70    Dis
1  MachineLearning        C00030      65  Merit
2       Simulation        E00120      58   Pass
3          BigData        F00130      81    Dis
4  MachineLearning        E00120      73    Dis
5  MachineLearning        G00120      65  Merit
6          BigData        H00120      56   Pass
```

```
Out[9]:
           Module StudentLogin  Marks  Grade
0    BusinessStats        A00010     70    Dis
1  MachineLearning        C00030     65  Merit
2       Simulation        E00120     58   Pass
3          BigData        F00130     81    Dis
4  MachineLearning        E00120     73    Dis
5  MachineLearning        G00120     65  Merit
6          BigData        H00120     56   Pass
```

>>>marks1=marks.rename(columns={'Marks1':'Marks2'}) #rename mark1 to mark2: only rename one of the column names

What is the result from the above command? Try it

Dot product between matrices using pandas DataFrames

- Multiplying two matrices of same dimensions

```
import pandas as pd
#to define two matrices
matrix1 = [(1, 1, 2),
        (0, 2, 1),
        (2, 0, 1)];

matrix2 = [(2, 0, 2),
        (1, 1, 1),
        (2, 2, 2)];

# Data loaded into pandas DataFrames
dataFrame1 = pd.DataFrame(data=matrix1);
dataFrame2 = pd.DataFrame(data=matrix2);
```

```
#to find the dimensions of dataFrame1, 2
Dim1 = dataFrame1.shape;
Dim2 = dataFrame2.shape;

# to find the dot product between Matrix1 and Matrix2
dotResult = dataFrame1.dot(dataFrame2)

#to find the addition and substraction netween dataFrame 1 and dataFrame2
addResult = dataFrame1+dataFrame2
subResult = dataFrame1-dataFrame2


#to find the transpose of matrix2
dataFrame2.transpose()
```

SciPy

SciPy

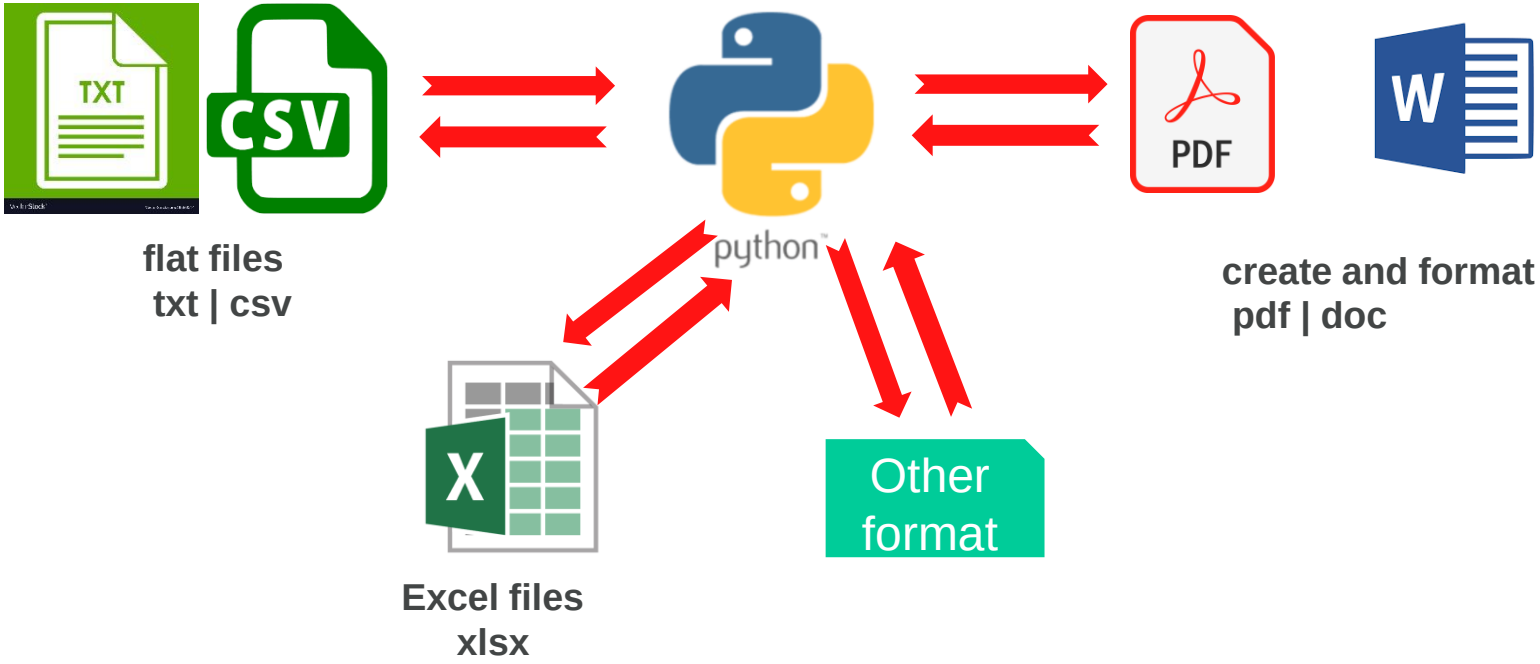
- To find the solution to minimise a function
- An example: **#to find the value that can minimise $x^2 + x + 2$**

```
from scipy import optimize
def eqn(x):                          #to define a function eqn()
  return x**2 + x + 2

mymin = optimize.minimize_scalar(eqn) #to minimize function eqn()
x_min=mymin.x #to assign the value that minimizes eqn() to x_min
print(x_min)
```

Data Import and Export

Import data into Python



A CSV file

#What is a csv file: A Comma Separated Values (CSV) file is a plain text file that contains a list of data.

- adult.csv

age,workclass,education,marital_status,occupation,gender,hours_per_week,income

39, State-gov, Bachelors, Never-married, Adm-clerical, Male,40, <=50K

50, Self-emp-not-inc, Bachelors, Married-civ-spouse, Exec-managerial, Male,13, <=50K

38, Private, HS-grad, Divorced, Handlers-cleaners, Male,40, <=50K

53, Private, 11th, Married-civ-spouse, Handlers-cleaners, Male,40, <=50K

28, Private, Bachelors, Married-civ-spouse, Prof-specialty, Female,40, <=50K

37, Private, Masters, Married-civ-spouse, Exec-managerial, Female,40, <=50K

49, Private, 9th, Married-spouse-absent, Other-service, Female,16, <=50K

52, Self-emp-not-inc, HS-grad, Married-civ-spouse, Exec-managerial, Male,45, >50K

31, Private, Masters, Never-married, Prof-specialty, Female,50, >50K

Read a cvs file into Python

- If you are going to read an existing file into Python, you first need to
  - Know the folder the file is located in;
  - Know the name of the file;
  - Know whether you will need to read the variable names, normally from the first row of the existing file; and
  - Give a name of the data frame to which the dataset will be assigned.

- **#Basic syntax: use** read_csv() to read a comma-separated values (csv) file into DataFrame.

- Dataframe0=pandas.read_csv(file, header =) #you have read the csv file into Dataframe0
  - file: the path to the file to read
  - header: a logical value. If header=0, the first row is used as the names of the variables, header =1, the first row is not treated as the names of the variables

Examples

- Example 1: when header=**0**

```
>>>import pandas as pd #import library pandas
>>>Adult_df = pd.read_csv("adult.csv",header=0) #read the csv file into data frame Adult_df, the 1st row being read
>>>Adult_df.head() #print the first 5 observations onto the console (screen)
```

```
In [8]: Adult_df.head()
Out[8]:
   age          workclass   education  ...  gender hours per week  income
0   39          State-gov   Bachelors  ...    Male             40   <=50K
1   50   Self-emp-not-inc   Bachelors  ...    Male             13   <=50K
2   38            Private     HS-grad  ...    Male             40   <=50K
3   53            Private        11th  ...    Male             40   <=50K
4   28            Private   Bachelors  ...  Female             40   <=50K

[5 rows x 8 columns]
```

- Example 2: when header=**1**

```
>>>Adult_df = pd.read_csv("Adult.csv",header=1) #read the csv file into data frame Adult_df
>>>Adult_df.head() #print the first 5 observations onto the console (screen)
```

```
In [11]: Adult_df.head()
Out[11]:
   39          State-gov   Bachelors  ...    Male  40   <=50K
0  50   Self-emp-not-inc   Bachelors  ...    Male  13   <=50K
1  38            Private     HS-grad  ...    Male  40   <=50K
2  53            Private        11th  ...    Male  40   <=50K
3  28            Private   Bachelors  ...  Female  40   <=50K
4  37            Private     Masters  ...  Female  40   <=50K

[5 rows x 8 columns]
```

```
>>>Adult_df.tail() #print the last 5 observations
```

Write a dataframe to a cvs file

- If you are going to write a dataframe to a csv file, you first need to
  - Know the folder the file is located in;
  - Know the name of the dataframe;
  - Give a name of the csv file to which the dataframe will be written to.

- **#Basic syntax: use** to_csv() to write a dataframe to a comma-separated values (csv) file.
- df.to_csv(file) #you have written a dataframe called df to a csv file
  - file: the path to the file to read
- Example:
  - import pandas as pd #import library pandas
  - marks=pd.DataFrame({'Module': ['BusinessStats','MachineLearning','Simulation','BigData', 'MachineLearning', 'MachineLearning'], 'StudentLogin':['A00010', 'C00030', 'E00120','F00130', 'E00120', 'G00120'],'Marks':[70,65,58,81,73,65]})
  - marks.to_csv("C:\Wutemp\Python\example.csv")

Examples

1. You are asked to insert Nigel's and Ben's marks, and also student marks of another module: dissertation (see the following cells in green)

Name	Business statistics	Simulation	Machine Learning	Big Data	Dissertation
John	65	60	70	80	65
Anna	72	65	55	65	62
Emma	56	64	67	73	65
Nigel	78	70	76	80	75
Ben	72	68	70	76	72

2. We want to analyse the dataset.
   - The first step is to store the data. This can be done by creating a dictionary;
   - Then create a dataframe
   - Then analyse the data in the dataframe
   - Then output the dataframe to a csv file

mark_dict

```
import pandas as pd
mark_dict={
   "Name":["John","Anna","Emma","Nigel","Ben"],
   "Business_statistics":[65,72,56,78,72],
   "Simulation":[60,65,64,70,68],
   "Machine_Learning":[70,55,67,76,70],
   "Big_Data":[80,65,73,80,76],
   "Dissertation":[65,62,65,75,72]
}
df = pd.DataFrame(mark_dict)
```

EDA: Exploratory Data Analysis

- df.info() #the structure of the dataframe

```
In [32]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Name                 5 non-null      object
 1   Business_statistics  5 non-null      int64
 2   Simulation           5 non-null      int64
 3   Machine_Learning     5 non-null      int64
 4   Big_Data             5 non-null      int64
 5   Dissertation         5 non-null      int64
dtypes: int64(5), object(1)
memory usage: 368.0+ bytes
```

- df.describe() #basic statistics of the dataframe

```
In [31]: df.describe()
Out[31]:
       Business_statistics  Simulation  ...   Big_Data  Dissertation
count             5.000000    5.000000  ...   5.000000      5.000000
mean             68.600000   65.400000  ...  74.800000     67.800000
std               8.414274    3.847077  ...   6.220932      5.449771
min              56.000000   60.000000  ...  65.000000     62.000000
25%              65.000000   64.000000  ...  73.000000     65.000000
50%              72.000000   65.000000  ...  76.000000     65.000000
75%              72.000000   68.000000  ...  80.000000     72.000000
max              78.000000   70.000000  ...  80.000000     75.000000
```

Head() and **Tail()**

- df.head() #show the first 5 rows
- df.tail() #show the last 5 rows

- Question: Can you try
  - df.Simulation.head()
  - df.Simulation.tail()
  - df.Simulation.describe()

```
In [33]: df.head()
Out[33]:
    Name  Business_statistics  ...  Big_Data  Dissertation
0   John                   65  ...        80            65
1   Anna                   72  ...        65            62
2   Emma                   56  ...        73            65
3  Nigel                   78  ...        80            75
4    Ben                   72  ...        76            72

[5 rows x 6 columns]

In [34]: df.tail()
Out[34]:
    Name  Business_statistics  ...  Big_Data  Dissertation
0   John                   65  ...        80            65
1   Anna                   72  ...        65            62
2   Emma                   56  ...        73            65
3  Nigel                   78  ...        80            75
4    Ben                   72  ...        76            72

[5 rows x 6 columns]
```

Difference between numpy array and pandas dataframe

Characteristics	NumPy Array	Pandas Dataframe
Homogeneity	Arrays consist of only homogeneous elements (elements of same data type)	Dataframes have heterogeneous elements.
Mutability	Arrays are mutable	Dataframes are mutable
Access	Array elements can be accessed using integer positions.	Dataframes can be accessed using both integer position as well as index.
Flexibility	Arrays do not have flexibility to deal with dynamic data sequence and mixed data types.	Dataframes have that flexibility.
Data type	Array deals with numerical data.	Dataframes deal with tabular data.