```
# You are expected to familiarize yourself with the most useful packages in Python.
# Pandas library offers a set of power datastructures and supported operations to manipulate data.
# You should use Google Colaboratory (https://colab.research.google.com/notebooks/intro.ipynb) for executing your code.
# Familiarize yourself with the DataFrame object.
# Explore how to
# 1) Use Python lists and dictionaries. How to do slicing of lists using the colon (:) specifications.
# 2) Create a DataFrame object from the data loaded from a dataset.
# 3) Print various rows and columns of the dataset.
# 4) Print a subset of rows and a subset of columns
# 5) How to group (aggregate) specific columns that correspond to certain specific values of rows. (use DataFrame.groupby())
```

```
[ ] # You are expected to familiarize yourself with the most useful packages in Python.


    # On searching over internet I came across these 3 most useful packages widely used packages in the domain of Machine Learning and Analytics
    import pandas as pd #importing pandas package as a "pd" variable
    import numpy as np #importing numpy package as a "np" variable
    import matplotlib.pyplot as plt

    # Dataset in use - Titanic Database
    titanic = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv') # reads csv file from the URL and importing to titanic Data
    #titanic.head # printing DataFrame to check if the dataset is still available to the endpoint
    # Till this step we have imported libraries, imported data from Github repository (Just for the example, Titanic data is imported)
    titanic
```

```
[ ] # Familiarizing with DataFrame object
    # DataFrame is a multi-dimensional data which is similar to a csv file or an excel table, majorly used for exploratory analysis
    # There are numerous in-built functions available for DataFrame object https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html
    # Some basic functions
    print(
    titanic.columns, # Column names of the DataFrame
    '\n',
    '\n',
    '\n',
    titanic.dtypes, # DataType of the column data
    '\n',
    '\n',
    '\n',
    titanic.info, # Short description of the DataFrame
    '\n',
    '\n',
    '\n',
    titanic.size, # Number of elements i.e, rows*columns
    '\n',
    '\n',
    '\n',
    titanic.shape, # Dimensionality
    '\n',
    '\n',
    '\n',
    titanic.memory_usage # Bytes of memory used
    )
```

Task 1

```
[ ] # 1) Use Python lists and dictionaries. How to do slicing of lists using the colon (:) specifications.
    # Difference between list and arrays is that arrays have to declared where as lists are native to python
    # Dictionaries are key value pairs of information
    list_columns = titanic.columns.values.tolist() # learned from: https://datatofish.com/convert-pandas-dataframe-to-list/
    list_columns
```

```
['survived',
 'pclass',
 'sex',
 'age',
 'sibsp',
 'parch',
 'fare',
 'embarked',
 'class',
 'who',
 'adult_male',
 'deck',
 'embark_town',
 'alive',
 'alone']
```

```
[ ] # We can make dictionary manually with elements jus to show as:
    dictionary_manual = {1: 'This',2:'is',3:'Assignment', 4:'First'}
    dictionary_manual
    # we can basic operations to access the elements from the dictionary using dictionary_manual[i] or using dot operator and so on.
```

```
{1: 'This', 2: 'is', 3: 'Assignment', 4: 'First'}
```

```
[ ] # Let us alsoe make a dictionary of top 5 elements of the titanic DataFrame
    import json
    head_elements = titanic.head()
    dictionary = head_elements.to_dict()
    dictionary
```

```
[ ] # Elements can also be accessed using slicing (:) the list, How to do slicing of lists using the colon (:) specifications.

    # using list_columns
    # list_columns
    list_columns[3:5] # prints the elements with index 3 and 4
```

```
['age', 'sibsp']
```

Task 2

```
[ ] # 2) Create a DataFrame object from the data loaded from a dataset.

    people = pd.DataFrame(titanic.survived.value_counts()) # Created a DataFrame for survived variable in the dataset and assigned with the sum/count value of ea
    people_survived = people['survived'][1] # gives the number of people survived
    people_not_survived = people['survived'][0] # gives the number of people not survived
    print ("people_survived = "+ str(people_survived) + " and people not survived = "+str(people_not_survived)) # printing the results
```

```
people_survived = 342 and people not survived = 549
```

Task 3

```
[ ]  # 3) Print various rows and columns of the dataset.
    # Again to simplify we will just print the rows and columns of the top head elements of the dataset
    # Also there is a problem with pandas that it skips the middle rows and columns of the dataset. in order to print all elements, we can use other operations ht
    head_elements_top = titanic.head()
    # head_elements_top['class','fare','who','alive']
    het = head_elements_top
    # het
    print (het[het.columns[1:4]]) # Prints columns ranging from 1 to 3
    # to print rows we can iterate through each tuple and print it using for loop
    print('\n')
    print('\n')
    print('\n')
    print('\n')
    for i in range(len(het)):
      print(het['pclass'][i],het['age'][i],het['fare'][i],het['alive'][i])
```

```
   pclass     sex   age
0       3    male  22.0
1       1  female  38.0
2       3  female  26.0
3       1  female  35.0
4       3    male  35.0
```

```
3 22.0 7.25 no
1 38.0 71.2833 yes
3 26.0 7.925 yes
1 35.0 53.1 yes
3 35.0 8.05 no
```

Task 4

```
[ ] # 4) Print a subset of rows and a subset of columns

    # titanic

    # het
    # subset of rows
    het[1:3] # using slicing operator we can print subset of rows,
    #we can also print using for loop if pandas skips the middle columns in the print outpur as seen in previous snippet
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True

```
[ ] # subset of columns

    het[['pclass','age','fare','alive']] # we have used the square brackets twice, it is because, single square bracket gives series and using double, it gives da
```

	pclass	age	fare	alive
0	3	22.0	7.2500	no
1	1	38.0	71.2833	yes
2	3	26.0	7.9250	yes
3	1	35.0	53.1000	yes
4	3	35.0	8.0500	no

Task 5

```
[ ] # 5) How to group (aggregate) specific columns that correspond to certain specific values of rows. (use DataFrame.groupby())

    grouped = titanic.groupby('pclass').count()
    grouped

    # grouped by specific column "pclass" which is class of the passenger and the related data.
```

pclass	survived	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
1	216	216	186	216	216	216	214	216	216	216	175	214	216	216
2	184	184	173	184	184	184	184	184	184	184	16	184	184	184
3	491	491	355	491	491	491	491	491	491	491	12	491	491	491

```
[ ] grouped = titanic.groupby('age').agg('sum')
    number_of_survived = grouped['survived']
    number_of_survived
    # Prints the total number of person died grouped by age
```

```
age
0.42     1
0.67     1
0.75     2
0.83     2
0.92     1
       ..
70.00    0
70.50    0
71.00    0
74.00    0
80.00    1
Name: survived, Length: 88, dtype: int64
```

```
[ ] # we can also group by "pclass" and get the number of people died per class
    grouped = titanic.groupby('pclass').agg('sum')
    number_of_survived = grouped['survived']
    number_of_survived
```

```
pclass
1    136
2     87
3    119
Name: survived, dtype: int64
```

```
[ ]

    class_one_not_sur =titanic[(titanic['survived']==0) & (titanic['pclass']==1)] # dataframe of class one passengers not survived
    class_two_not_sur =titanic[(titanic['survived']==0) & (titanic['pclass']==2)] # dataframe of class two passengers not survived
    class_three_not_sur =titanic[(titanic['survived']==0) & (titanic['pclass']==3)] # dataframe of class three passengers not survived
```

```
[ ] class_one_not_sur
    #Class one passengers that not survived
```

```
[ ] class_two_not_sur
    #Class two passengers that not survived
```

```
[ ] class_three_not_sur
    #Class three passengers that not survived
```

```
[ ] print("There are " + str(len(class_one_not_sur)) + ", "+ str(len(class_two_not_sur))+", "+str(len(class_three_not_sur))+ " number of passengers that did not s
```

```
There are 80, 97, 372 number of passengers that did not survive in class 1,2 and 3 respectively
```

```
[ ] # Also available as a private repo on https://github.com/samy280497/MLClass2020
```