CHAPTER – 4 - Class Resumes at 02: 05 PM (PKT)

DICTIONARIES & PANDAS

You will learn about the dictionary, an alternative to the Python list, and the pandas DataFrame, the de facto standard to work with tabular data in Python.

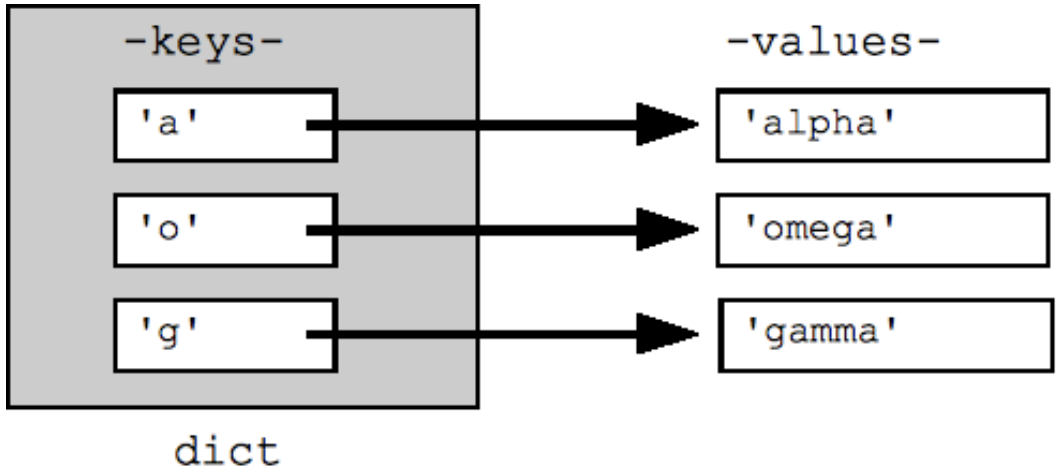
You will get hands-on practice with creating and manipulating datasets, and you'll learn how to access the information you need from these data structures.

DICTIONARIES

python

DICTIONARIES

- Dictionary in Python is an unordered collection of data values.
- They are used to store data values like a map, which unlike other Data Types hold only single value as an element.
- Dictionary holds key:value pair. Key value is provided in the dictionary to make it more optimized.
- Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a 'comma'.
- A Dictionary in Python works similar to the Dictionary in a real world. Keys of a Dictionary must be unique.
- Keys should be immutable data type such as Strings & Integers, but the key-values can be repeated and be of any type.



EUROPE - TASK BACKGROUND

To see why dictionaries are useful, have a look at the two lists defined on the right. countries contains the names of some European countries. capitals lists the corresponding names of their capital.


```
# Definition of countries and capital
countries = ['spain', 'france', 'germany', 'norway']
capitals = ['madrid', 'paris', 'berlin', 'oslo']
```

TASK

Use the index() method on countries to find the index of 'germany'. Store this index as ind_ger.

Use ind_ger to access the capital of Germany from the capitals list. Print it out.

ANSWER

```
#save index of ‘germany’ : ind_ger
ind_ger=countries.index('germany’)
#print out the capital of Germany  using ind_ger
Print(capitals[ind_ger])
```

CREATE DICTIONARY

In Python, a Dictionary can be created by placing sequence of elements within curly {} braces, separated by ‘comma’.

Dictionary holds a pair of values, one being the Key and the other corresponding pair element being its Key:value.

Values in a dictionary can be of any datatype and can be duplicated.

keys can’t be repeated and must be immutable.

```
my_dict = {"key1":"value1","key2":"value2",}
```

DICTIONARY -EXAMPLE

Creating a dictionary Dist from two list keys and values and printing the data type of Dist

```
#creating keys and values
keys = ['name', 'age', 'food']
values = ['Monty', 42, 'hotdog']
#creating a dictionary - Dist
Dist={'name':'Monty', 'age':42,'food':' hotdog'}
print(Dist)
#print data type
print(type(Dist))
Out:
{'name': 'Monty', 'age': 42, 'food': 'hotdog'}
<class 'dict'>
```

TASK

Using countries and capitals lists, create a dictionary called europe with 4 key:value pairs.

Print out europe to see if the result is what you expected.

ANSWER

```
# Definition of countries and capital
countries = ['spain', 'france', 'germany', 'norway']
capitals = ['madrid', 'paris', 'berlin', 'oslo']
# From string in countries and capitals, create dictionary Europe
europe = { 'spain':'madrid', 'france':'paris','germany':'berlin', 'norway':'oslo'}
# Print Europe
print(europe)
```

ACCESSING A DICTIONARY

To access dictionary elements, you can use the familiar square brackets [] along with the key to obtain its value.

Syntax: dictionaryName[‘key’]

Get name from the Dict dictionary

```
#using a dictionary
Dict={'name': 'Monty', 'age': 42, 'food': ‘hotdog'}
#print out name
Print(Dict[‘name’])
```

TASK

Check out which keys are in europe by calling the keys() method on europe. Print out the result.

ANSWER

```
# dictionary Europe
europe = { 'spain':'madrid', 'france':'paris','germany':'berlin', 'norway':'oslo' }
# Print Europe
print(europe)
#print out keys of europe dictionary
print(europe.keys())
```

TASK

Print out the value that belongs to the key 'norway‘ from your dictionary.

ANSWER

```
# dictionary Europe
europe = { 'spain':'madrid', 'france':'paris','germany':'berlin', 'norway':'oslo' }
# print out 'norway' capital from europe dictionary
print(europe['norway'])
```

DICTIONARY MANIPULATION - ADDITION

- If you know how to access a dictionary, you can also add a new value to a dictionary.
- If you want to add a new key to the dictionary, then you can use assignment operator with dictionary key and values.
- Syntax: dictionaryName['key'] = 'value'

Note that if the key already exists, then the value will be overwritten.

DICTIONARY MANIPULATION - EXAMPLE

Adding new value to dictionary Dist (Height: 6.1)

```
#creating a dictionary - Dist
Dist={'name':'Monty', 'age':42,'food':' hotdog'}
Dist['height':]= 6.1
print(Dist)
```

Out:

{'name': 'Monty', 'age': 42, 'food': 'hotdog' , 'height': 6.1}

TASK

Add the key 'italy' with the value 'rome' to europe and print out europe.

ANSWER

```
# dictionary Europe
europe = { 'spain':'madrid', 'france':'paris','germany':'berlin', 'norway':'oslo' }
# Print Europe
print(europe)
#Add the key 'italy' with the value 'rome' to europe.
europe['italy']= 'rome'
# Print Europe
print(europe)
```

TASK

Add these two key:value pairs to europe: 'poland‘ 'warsaw & 'iceland' 'reykjavik’.

Print out europe.

ANSWER

```
# dictionary Europe
europe = { 'spain':'madrid', 'france':'paris','germany':'berlin', 'norway':'oslo', 'italy':'rome' }
# Print Europe
print(europe)
#Adding new keys to europe.
europe['iceland']= 'reykjavik'
europe['poland']= 'warsaw'
# Print Europe
print(europe)
```

TASK

Add another two key:value pairs to europe: 'australia':'vienna' & 'germany' :'bonn'.

ANSWER

```
#Add the key australia with the value vienna' to europe.
europe['australia']= 'vienna'
#update the value of germany
europe['germany']= 'bonn'
# Print Europe
print(europe)
```

DICTIONARY MANIPULATION – UPDATING

This dictionary does not look right, can you tell what's wrong in it?

```
# europe dictionary
europe = {'spain': 'madrid', 'france': 'paris', 'germany': 'bonn', 'norway': 'oslo', 'italy': 'rome', 'iceland': 'reykjavik', 'poland': 'warsaw',
'australia': 'vienna'}
```

Can you clean up?

Remember if a key already exists, then the value will be overwritten.

TASK

You updated the capital of Germany to 'bonn'; it was 'berlin'. Update its value..

ANSWER

```
#update Germany capital bronn : berlin
europe['germany']='berlin'
#print europe
print(europe)
```

TASK

1. Delete the Key Australia and add Austria with the Capital Vienna ?
2. You updated the capital of Germany to 'bonn'; it was 'berlin'. Update its value..

DICTIONARY MANIPULATION - DELETION

- Just like updating you can also delete values to a dictionary.
- To delete a key, value pair in a dictionary, you can use the del method.
- Syntax: del(dictName['key'])

DICTIONARY MANIPULATION - EXAMPLE

Deleting a key from the dictionary Dist (Height: 6.1)

```
#deleting a key from dictionary - Dist
del(Dist['height'])
print(Dist)
```

Out:

```
{'name': 'Monty', 'age': 42, 'food': 'hotdog'}
```

TASK

Australia is not in Europe, Austria is! Remove the key 'australia' from europe.

Print out europe to see if your cleaning work paid off.

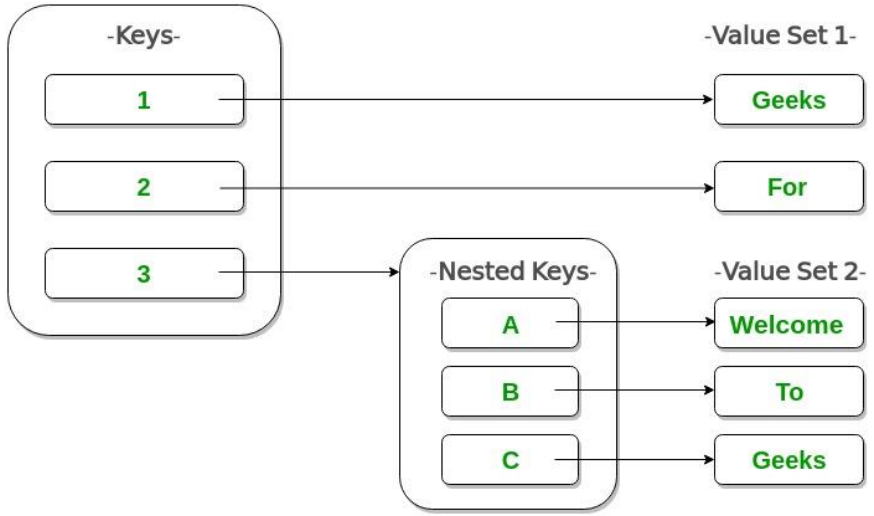
ANSWER

```
# dictionary Europe

europe = {'spain':'madrid', 'france':'paris', 'germany':'bonn', 'norway':'oslo', 'italy':'rome', 'poland':'warsaw',
'australia':'vienna' }

#print europe

print(europe)

#remove 'Australia' form Europe

del(europe['australia'])

#print europe

print(europe)
```

NESTED DICTIONARY

- In Python, a nested dictionary is a dictionary inside a dictionary.
- It's a collection of dictionaries into one single dictionary.
- A Nested dictionary can be created by placing the comma-separated dictionaries enclosed within braces {}.

Syntax: nested_dict = { 'dict1': {'key_A': 'value_A'},
'dict2': {'key_B': 'value_B'}}



NESTED DICTIONARY-EXAMPLE

Creating a dictionary of people within a dictionary

```
#creating an individual dictionary
pep1={'name': 'John', 'age': '27', 'sex': 'Male'}
pep2={'name': 'Marie', 'age': '22', 'sex': 'Female'}
#creating a dictionary of people using pep1 &pep2
people = {‘pep1’: {'name': 'John', 'age': '27', 'sex': 'Male'},
          ‘pep2:’ {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
#print people
print(people)
Out:
{pep1: {'name': 'John', 'age': '27', 'sex': 'Male'},
 pep2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
```

TASK

Create a dictionary of dictionaries europe using the dictionaries of individual countries.

Use the following data for this task:

spain={ 'capital':'madrid', 'population':46.77 }

france= { 'capital':'paris', 'population':66.03 }

germany= { 'capital':'berlin', 'population':80.62 }

norway ={ 'capital':'oslo', 'population':5.084 }

ANSWER

```
#creating countries dictionaries
spain={ 'capital':'madrid', 'population':46.77 }
france= { 'capital':'paris', 'population':66.03 }
germany= { 'capital':'berlin', 'population':80.62 }
norway ={ 'capital':'oslo', 'population':5.084 }

#creating europe a nested dictionary
europe = {
'spain': { 'capital':'madrid', 'population':46.77 },
'france': { 'capital':'paris', 'population':66.03 },
'germany': { 'capital':'berlin', 'population':80.62 },
'norway': { 'capital':'oslo', 'population':5.084 } }

#print out europe
print(europe)
```

ACCESSING A NESTED DICTIONARY

- To access nested dictionary elements, you can use the familiar square brackets along with the key to obtain its value.

Syntax: dictionaryName[‘key’][‘key’]

Get John from the Nested dictionary

```
#using a dictionary
people = {'pep1': {'name': 'John', 'age': '27', 'sex': 'Male'},
          'pep2': {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
#print out name
print(people['pep1']['name'])
```

Out:

John

TASK

Use your nested dictionary europe to fetch the population for Spain.

ANSWER

```
# dictionary Europe
europe = {
'spain': { 'capital':'madrid', 'population':46.77 },
'france': { 'capital':'paris', 'population':66.03 },
'germany': { 'capital':'berlin', 'population':80.62 },
'norway': { 'capital':'oslo', 'population':5.084 } }
#print out europe
print(europe)
#print out capital of France
print(europe[‘spain'][‘population'])
```

TASK

Print out the capital of France.

ANSWER

```
# dictionary Europe
europe = {
'spain': { 'capital':'madrid', 'population':46.77 },
'france': { 'capital':'paris', 'population':66.03 },
'germany': { 'capital':'berlin', 'population':80.62 },
'norway': { 'capital':'oslo', 'population':5.084 } }
#print out europe
print(europe)
#print out capital of France
print(europe['france']['capital'])
```

TASK

Create a dictionary, named data, with the keys 'capital' and 'population'. Set them to 'rome' and 59.83, respectively.

Add a new key-value pair to europe; the key is 'italy' and the value is data, print the dictionary you just built.

ANSWER

```
# dictionary Europe
europe = {
'spain': { 'capital':'madrid', 'population':46.77 },
'france': { 'capital':'paris', 'population':66.03 },
'germany': { 'capital':'berlin', 'population':80.62 },
'norway': { 'capital':'oslo', 'population':5.084 } }
print(europe)
#create new dictionary data
data={'capital':'rome','population':59.83}
#add  new  key ‘Italy’ and value is data
europe['italy']=data
#print out europe
print(europe)
```

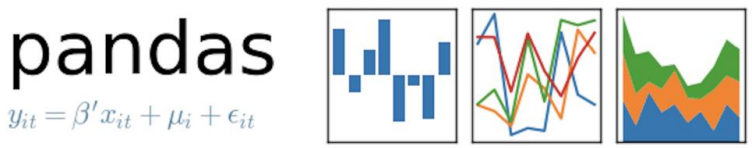
PANDAS

PANDAS

- The pandas package is the most important tool at the disposal of Data Scientists and Analysts working in Python today.

- Pandas is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

- Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming.

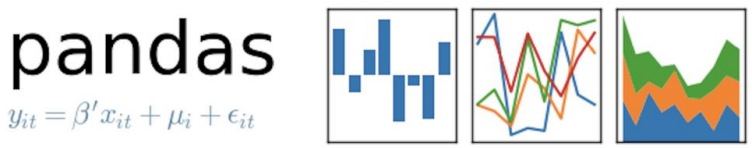


PANDAS - RECAP

- In chapter 2 you learnt how to import the Panda library & how to use read function in Pandas.

Syntax for importing Pandas: import pandas as pd

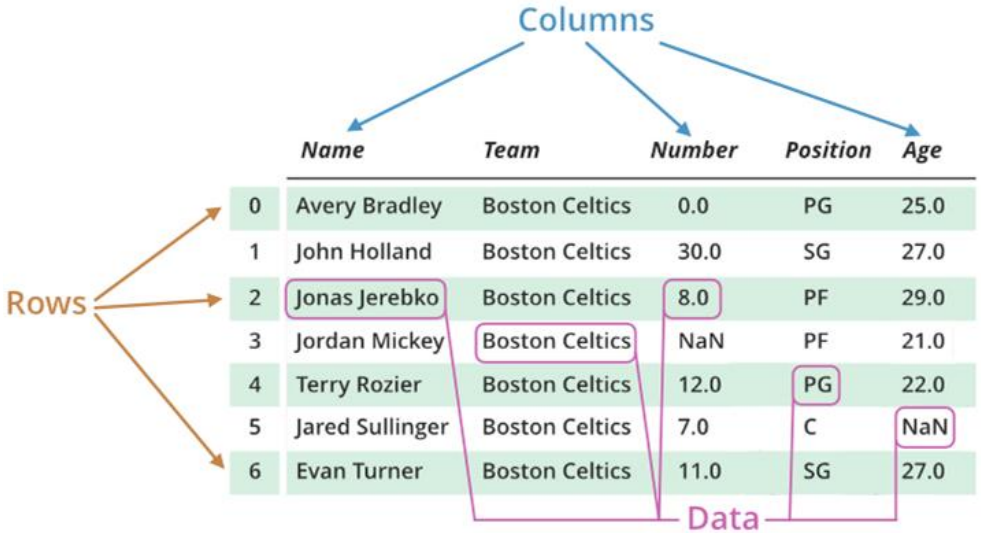
Syntax for using the reading a csv file: pd.read_csv("Source")



DICTIONARY TO DATAFRAME

- Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).
- A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.
- Pandas DataFrame consists of three principal components, the data, rows, and columns.

Syntax: pd.DataFrame("Dictionary Name")



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

DATAFRAME-EXAMPLE

Creating a pandas dataframe by using of employee data

```
# Import pandas package
import pandas as pd
# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}
# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
Print(df)
```

Out:

	Name	Age	Address	Qualification
0	Jai	27	Delhi	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannauj	Phd

DATAFRAME – TASK BACKGROUND

We'll be use these 3 lists – names, dr, cpc

names, containing the country names for which data is available.
dr, a list with booleans that tells whether people drive left or right in the corresponding country.
cpc, the number of motor vehicles per 1000 people in the corresponding country.

```
# Pre-defined lists
names = ['United States', 'Australia', 'Japan', 'India', 'Russia', 'Morocco', 'Egypt']
dr =  [True, False, False, False, True, True, True]
cpc = [809, 731, 588, 18, 200, 70, 45]
```

Each dictionary key is a column label and each value is a list which contains the column elements.

TASK

Use the pre-defined lists to create a dictionary called my_dict. There should be three key value pairs:

key 'country' and value names.

key 'drives_right' and value dr.

key 'cars_per_cap' and value cpc

ANSWER

```
# Pre-defined lists
names = ['United States', 'Australia', 'Japan', 'India', 'Russia', 'Morocco', 'Egypt']
dr =  [True, False, False, False, True, True, True]
cpc = [809, 731, 588, 18, 200, 70, 45]
# Import pandas as pd
import pandas as pd
# Create dictionary my_dict with three key:value pairs: my_dict
my_dict={'country':names,
   'drives_right':dr,
   'cars_per_cap':cpc}
```

TASK

Convert your dictionary my_dict into a DataFrame called cars and print out cars.

ANSWER

```
#import pandas as pd
import pandas as pd
# Build a DataFrame cars from my_dict: cars
cars=pd.DataFrame(my_dict)
# Print cars
print(cars)
```

DATAFRAME – ROW LABELS

Notice that the row labels (i.e. the labels for the different observations) were automatically set to integers from 0 up to 6?

	country	drives_right	cars_per_cap
0	United States	True	809
1	Australia	False	731
2	Japan	False	588
3	India	False	18
4	Russia	True	200
5	Morocco	True	70
6	Egypt	True	45

To solve this a list row_labels has been created. You can use it to specify the row labels of the cars DataFrame. You do this by setting the index attribute of cars, you can use the following syntax: cars.index.

DATAFRAME – ROW LABELS EXAMPLE

Add row_labels in pandas DataFrame

```
# Import pandas package
import pandas as pd
# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}
# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
row_labels = ['emp1', 'emp2', 'emp3', 'emp4']
# Specify row labels of cars
df.index=row_labels
Print(df)
```

Out:

	Name	Age	Address	Qualification
emp1	Jai	27	Delhi	Msc
emp2	Princi	24	Kanpur	MA
emp3	Gaurav	22	Allahabad	MCA
emp4	Anuj	32	Kannauj	Phd

TASK

The row labels on the previous task are not correctly set. Specify the row labels by setting cars.index equal to row_labels.Print out cars again and check if the row labels are correct this time.

Definition of row_labels

row_labels = ['US', 'AUS', 'JPN', 'IN', 'RU', 'MOR', 'EG'].

ANSWER

```
#import pandas
import pandas as pd
# Build cars DataFrame
names = ['United States', 'Australia', 'Japan', 'India', 'Russia', 'Morocco', 'Egypt']
dr =  [True, False, False, False, True, True, True]
cpc = [809, 731, 588, 18, 200, 70, 45]
dict = { 'country':names, 'drives_right':dr, 'cars_per_cap':cpc }
cars = pd.DataFrame(dict)
print(cars)
# Definition of row_labels
row_labels = ['US', 'AUS', 'JPN', 'IN', 'RU', 'MOR', 'EG']
# Specify row labels of cars
cars.index=row_labels
# Print cars again
print(cars)
```

CSV TO DATAFRAME

Putting data in a dictionary and then building a DataFrame works, but it's not very efficient.

What if you're dealing with millions of observations? In those cases, the data is typically available as files with a regular structure.

One of those file types is the CSV file, which is short for "comma-separated values".

To import CSV data into Python as a Pandas DataFrame you can use read_csv().

Let’s refresh how to import a .CSV file in Python.

TASK

Import cars.csv data as a DataFrame. Store this dataframe as cars. Print out cars.

Use the following link to import cars.csv

https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv

ANSWER

```
# Import pandas as pd
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv')
# Print out cars
print(cars)
```

SELECT COLUMNS IN A PANDAS DATAFRAME

Selecting a column or multiple columns from a Pandas dataframe is a common task in exploratory data analysis.

To select a single column, you can use double square brackets [[]], with a single column name inside it.

Syntax: DataFrame[['columnName']]

SELECT COLUMNS -EXAMPLE

Selecting single columns from DataFrame

```
# Import pandas package
import pandas as pd
# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}
# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
#select Name columns form Dataframe
df[['Name']]
```

Out:

	Name
0	Jai
1	Princi
2	Gaurav
3	Anuj

TASK

Print out the country column of cars as a Pandas DataFrame.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv')
# Print out country column as Pandas DataFrame
print(cars[['country']])
```

SELECTING MULTIPLE COLUMNS

We can use double square brackets [[]] to select multiple columns from a data frame in Pandas.

If we want to select multiple columns, we specify the list of column names in the order we want them in the result set.

Syntax: DataFrame[[‘columnName’, ‘columnName’]]

SELECTING MULTIPLE COLUMNS - EXAMPLE

Selecting single columns from DataFrame

```
# Import pandas package
import pandas as pd
# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}
# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
#select Name columns form Dataframe
print(df[['Name', 'Age']])
```

Out:

	Name	Age
0	Jai	27
1	Princi	24
2	Gaurav	22
3	Anuj	32

TASK

Print out a DataFrame with both the country and drives_right columns of cars, in this order.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv')
# Print out DataFrame with country and drives_right column
print(cars[['country','drives_right']])
```

SELECT ROWS

- Square brackets can do more than just selecting columns.
- You can also use them to get rows from a DataFrame.

Syntax: DataFrame[startRow:endRow]

- The result is another DataFrame containing only the rows you specified.

Pay attention: You can only select rows using square brackets if you specify a slice, like 0:4. Also, you're using the integer indexes of the rows here, not the row labels!

SELECT ROWS -EXAMPLE

Selecting first 3 rows from DataFrame

```
# Import pandas package
import pandas as pd
# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}
# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
#select first 3 rows
print(df[0:3])
```

Out:

	Name	Age	Address	Qualification
0	Jai	27	Delhi	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA

TASK

Select the first 2 rows from cars and print them out.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv')
# Print out first 3 observations
print(cars[0:3])
```

TASK

Select the sixth rows and print it out

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv')
# Print out fourth, fifth and sixth observation
print(cars[5:6])
```

WORKING WITH YOUR DATAFRAME RESULT SET

- You must have noticed that by using the print() function on the cars data in the previous task gives you a bland looking result set. What if we wanted the result set in a tabular format.
- To do that you can try the last command without the print function.

Result set in a tabular format

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv')
#result table of fourth, fifth and sixth observation
cars[3:6]
```

	Unnamed: 0	cars_per_cap	country	drives_right
3	IN	18	India	False
4	RU	200	Russia	True
5	MOR	70	Morocco	True

Result set in a Table

```
  Unnamed: 0  cars_per_cap  country  drives_right
3         IN            18    India         False
4         RU           200   Russia          True
5        MOR            70  Morocco          True
```

Regular result set

WORKING WITH YOUR DATAFRAME RESULT SET

Python as per default will assign index numbers to the result data set. You can edit this by using index_col="value" after the file path.

Editing index column

```
# Import cars data
import pandas as pd
#importing cars data set
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv',index_col=0)
#result set
Cars
```

	Unnamed: 0	cars_per_cap	country	drives_right
0	US	809	United States	True
1	AUS	731	Australia	False
2	JAP	588	Japan	False
3	IN	18	India	False
4	RU	200	Russia	True
5	MOR	70	Morocco	True
6	EG	45	Egypt	True

Default Result Set

	cars_per_cap	country	drives_right
US	809	United States	True
AUS	731	Australia	False
JAP	588	Japan	False
IN	18	India	False
RU	200	Russia	True
MOR	70	Morocco	True
EG	45	Egypt	True

Edited Result Set

LOC & ILOC

With loc and iloc you can do practically any data selection operation on DataFrames.

Syntax: loc[[<row selection>, <column selection>]] & iloc[[<row selection>, <column selection>]]

- loc is label-based, which means that you have to specify rows and columns based on their row and column labels.
- iloc is integer index based, so you have to specify rows and columns by their integer index.

Selecting Single Row

```
cars.loc[['RU']]
cars.iloc[[4]]
```

Selecting Multiple Rows

```
cars.loc[['RU', 'AUS']]
cars.iloc[[4, 1]]
```

Each pair of commands here gives the same result.

TASK

Use loc or iloc to select the rows corresponding to Japan. The label of this row is JAP, the index is 2. Print the result set.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv',index_col=0)
# Print out observation for Japan
print(cars.loc[['JAP']])
print(cars.iloc[[2]])
```

TASK

Select the rows for Australia and Egypt as a DataFrame by using both loc / iloc. You can find out about the labels/indexes of these rows by inspecting data in the cars dataset. Print the result.

TASK

Use loc or iloc to select the rows corresponding to Japan. The label of this row is JAP, the index is 2. Print only the Country and Cars Per Cap.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv',index_col=0)
# Print out observations for Australia and Egypt
print(cars.loc[['AUS','EG']])
print(cars.iloc[[1,6]])
```

LOC & ILOC – SELECTING ROWS & COLUMNS

loc and iloc also allow you to select both rows and columns from a DataFrame.

Selecting Single Row & Single Column

```
cars.loc['IN', 'cars_per_cap']
cars.iloc[3, 0]
```

Out: 18

Selecting Multiple Rows & Single Column

```
cars.loc[['IN', 'RU'], 'cars_per_cap']
cars.iloc[[3, 4], 0]
```

Out:
```
IN     18
RU    200
```

Selecting Multiple Rows & Multiple Columns

```
cars.loc[['IN', 'RU'], ['cars_per_cap', 'country']]
cars.iloc[[3, 4], [0, 1]]
```

Out:

	cars_per_cap	country
IN	18	India
RU	200	Russia

TASK

Print out the drives_right value of the row corresponding to Morocco.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv',index_col=0)
# Print out drives_right value of Morocco
print(cars.loc['MOR','drives_right'])
print(cars.iloc[5,2])
```

TASK

Print out the rows containing the observations for Russia and Morocco and the columns country and drives_right.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv',index_col=0)
# Print sub-DataFrame
print(cars.loc[['RU','MOR'],['country','drives_right']])
print(cars.iloc[[4,5],[1,2]])
```

LOC & ILOC – SELECTING COLUMNS

It's also possible to select only columns with loc and iloc. In both cases, you simply put a slice going from beginning to end in front of the comma:

Selecting Single Column

```
cars.loc[:, 'country']
cars.iloc[:, 1]
```

Selecting Multiple Columns

```
cars.loc[:, ['country','drives_right']]
cars.iloc[:, [1, 2]]
```

TASK

Print out the drives_right column as a DataFrame using loc or iloc.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/cars.csv',index_col=0)
# Print out drives_right column as DataFrame
print(cars.loc[:,['drives_right']])
print(cars.iloc[:,[2]])
```

TASK

Print out both the cars_per_cap and drives_right column as a DataFrame using loc or iloc.

ANSWER

```
# Import cars data
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)
# Print out cars_per_cap and drives_right as DataFrame
print(cars.loc[:,['cars_per_cap','drives_right']])
print(cars.iloc[:,[0,2]])
```

TASK

Print out the Country Where Drive Right Value is True. Print the Following Cars_Per_Cap and Country Name

Q&A

python

THANK YOU