Notes on pandas for beginners

Author --> Ashbab khan

All this pdf is created by me

Importing pandas

In [1]:
```
# importing pandas as pd
import pandas as pd
```

Creating series with pandas

In [2]:
```
# Creating a series with the use pandas
# and series is one dimension
names = pd.Series(['Ashbab Khan','Sarah','Harry'])
names
```

Out[2]:
```
0    Ashbab Khan
1          Sarah
2          Harry
dtype: object
```

In [3]:
```
occupation = pd.Series(['Data Scientist','Web developer','Programmer'])
occupation
```

Out[3]:
```
0    Data Scientist
1     Web developer
2        Programmer
dtype: object
```

Creating dataframe

In [4]:
```
# Creating a data frame using pandas
# and it is two dimensional
info_table = pd.DataFrame({'names':names,'occupation':occupation})
# printing our info_table
info_table
```

Out[4]:

	names	occupation
0	Ashbab Khan	Data Scientist
1	Sarah	Web developer
2	Harry	Programmer

Importing local csv files

In [5]:

```
# Lets import some csv files using pandas
# we are reading our csv file
# using pandas
data = pd.read_csv('emp_info.csv')
data
```

Out[5]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24
1	David	Australia	Web developer	29
2	Sarah	New Zealand	Doctor	22
3	Andrew	Poland	Technician	35
4	Joe	Romania	Data engineer	31
5	Vicky	India	Java developer	27

Importing csv file from a url

In [6]:

```
# Importing csv file which are not
# in your present in your computer or you want to
# import from some url
```

```
url_data = pd.read_csv('https://raw.githubusercontent.com/ashbabkhan2/new_repo/main/emp_info.csv')
url_data
```

Out[6]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24
1	David	Australia	Web developer	29
2	Sarah	New Zealand	Doctor	22
3	Andrew	Poland	Technician	35
4	Joe	Romania	Data engineer	31
5	Vicky	India	Java developer	27

Exporting Data

In [7]:

```
# exporting data

url_data.to_csv('data_info.csv')
```

Detail about of our data column

and some important functions

In [8]:

```
# This will give us an overview of our column
# and its data types
url_data.dtypes
```

Out[8]:

```
name          object
country       object
occupation    object
age            int64
dtype: object
```

In [9]:

```
# describe() gives us some insights from our data but it
# is only shown from a numerical column that's why it shows
# insight of age column

url_data.describe()
```

Out[9]:

	age
count	6.000000
mean	28.000000
std	4.732864
min	22.000000
25%	24.750000
50%	28.000000
75%	30.500000
max	35.000000

In [10]:

```
# info about our data

url_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        6 non-null      object
 1   country     6 non-null      object
 2   occupation  6 non-null      object
 3   age         6 non-null      int64 
dtypes: int64(1), object(3)
memory usage: 320.0+ bytes
```

In [11]:

```
# you can select a particular column
# using [""] or using .

url_data['name']
```

Out[11]:

```
0    Ashbab khan
```

```
1         David
2          Sarah
3         Andrew
4            Joe
5          Vicky
Name: name, dtype: object
```

In [12]:

```
# you can select a particular column
# using [""] or using but the main
# disadvantage of using . is that if
# your column contain some space then
# it show error so it recommended to use [""]

url_data.name
```

Out[12]:

```
0    Ashbab khan
1         David
2          Sarah
3         Andrew
4            Joe
5          Vicky
Name: name, dtype: object
```

In [13]:

```
# Calculating mean of one of our column
url_data["age"].mean()
```

Out[13]:

```
28.0
```

In [14]:

```
# If you want to count the number of rows then this
# len() function will help you

len(url_data)
```

Out[14]:

```
6
```

In [15]:

```
# this head() function give the first 5 rows
# of our data if you doesn't give any arguments

url_data.head()
```

Out[15]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24
1	David	Australia	Web developer	29
2	Sarah	New Zealand	Doctor	22
3	Andrew	Poland	Technician	35
4	Joe	Romania	Data engineer	31

In [16]:

```
# But if you want custom rows then
# simply pass your custom value to head(3) funtion

url_data.head(3)
```

Out[16]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24
1	David	Australia	Web developer	29
2	Sarah	New Zealand	Doctor	22

In [17]:

```
# You can also use tail() function to get
# 5 bottom rows

url_data.tail()
```

Out[17]:

	name	country	occupation	age
1	David	Australia	Web developer	29
2	Sarah	New Zealand	Doctor	22
3	Andrew	Poland	Technician	35
4	Joe	Romania	Data engineer	31
5	Vicky	India	Java developer	27

In [18]:

```
# or use custom such as tail(2) to get
# two rows from bottom

url_data.tail(2)
```

Out[18]:

	name	country	occupation	age
4	Joe	Romania	Data engineer	31
5	Vicky	India	Java developer	27

In [19]:

```
# getting rows with the use of index
# using loc this will return the index 3 row

url_data.loc[3]
```

Out[19]:

```
name                  Andrew
country               Poland
occupation        Technician
age                       35
Name: 3, dtype: object
```

In [20]:

```
# getting rows with the use of position

url_data.iloc[1]
```

Out[20]:

```
name                     David
country              Australia
occupation       Web developer
age                         29
Name: 1, dtype: object
```

In [21]:

```
# If you want to filter rows such as only
# select those rows which have an age =  31
# or name='ashbab' or whatever you want

# Example 1

url_data[url_data['age'] == 31]
```

Out[21]:

	name	country	occupation	age

	name	country	occupation	age
4	Joe	Romania	Data engineer	31

In [22]:
```
# Example 2

url_data[url_data['name'] == 'Ashbab khan']
```

Out[22]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24

In [23]:
```
# Example 3

url_data[url_data['age']>=30]
```

Out[23]:

	name	country	occupation	age
3	Andrew	Poland	Technician	35
4	Joe	Romania	Data engineer	31

In [24]:
```
# Creating a cross tab this generally help
# if you are making cross info
# between the table

pd.crosstab(url_data['name'],url_data['age'])
```

Out[24]:

age	**22**	**24**	**27**	**29**	**31**	**35**
name						
Andrew	0	0	0	0	0	1
Ashbab khan	0	1	0	0	0	0
David	0	0	0	1	0	0
Joe	0	0	0	0	1	0
Sarah	1	0	0	0	0	0

age	22	24	27	29	31	35
name						
Vicky	0	0	1	0	0	0

Manipulating Data

One thing to keep in mind that if you want to change anything permanently then you need to reassign to our data frame

In [27]:

```
# Converting our data into small letters
# this will only work for temporary basis

url_data['name'].str.lower()
```

Out[27]:

```
0    ashbab khan
1          david
2          sarah
3         andrew
4            joe
5          vicky
Name: name, dtype: object
```

In [28]:

```
# You can see that the data is still
# not in lower case and it is because
# it change into lower case only for
# temporary basis

url_data
```

Out[28]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24

	name	country	occupation	age
1	David	Australia	Web developer	29
2	Sarah	New Zealand	Doctor	22
3	Andrew	Poland	Technician	35
4	Joe	Romania	Data engineer	31
5	Vicky	India	Java developer	27

In [29]:

```
# If you want change on permanent basis
# you need to reassign just like this

url_data['name'] = url_data['name'].str.lower()
```

In [30]:

```
# Now all our name column is converted into
# lower case

url_data
```

Out[30]:

	name	country	occupation	age
0	ashbab khan	India	Data scientist	24
1	david	Australia	Web developer	29
2	sarah	New Zealand	Doctor	22
3	andrew	Poland	Technician	35
4	joe	Romania	Data engineer	31
5	vicky	India	Java developer	27

Let's import some another data

which contain some Null values

In [42]:

```
# importing data

missing_data = pd.read_csv('missing_info_data.csv')
```

In [44]:

```
# In pandas Nan is used instead of NULL
# So all the missing value showing in Nan

missing_data
```

Out[44]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24.0
1	David	NaN	Web developer	29.0
2	Sarah	New Zealand	Doctor	NaN
3	Andrew	Poland	NaN	35.0
4	Joe	NaN	Data engineer	31.0
5	Vicky	India	Java developer	NaN

In [49]:

```
# So how to fill that Nan value with
# some meaningful values let's
# say we want to fill missing countries
# with unknown

missing_data['country'] = missing_data['country'].fillna('Unknown')
```

In [50]:

```
missing_data
```

Out[50]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24.0
1	David	Unknown	Web developer	29.0
2	Sarah	New Zealand	Doctor	NaN
3	Andrew	Poland	NaN	35.0

	name	country	occupation	age
4	Joe	Unknown	Data engineer	31.0
5	Vicky	India	Java developer	NaN

In [51]:
```
# we can also use inplace = true for
# assigning into our dataframe
```

In [54]:
```
# So what if we want to remove entire
# row which contain Nan value

missing_data.dropna(inplace=True)
```

In [56]:
```
# It drop the rows which contains
# Nan value

missing_data
```

Out[56]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24.0
1	David	Unknown	Web developer	29.0
4	Joe	Unknown	Data engineer	31.0

In [57]:
```
# If you want to export you can do that
# with to_csv() syntax

missing_data.to_csv('cleaned_data.csv')
```

In [64]:
```
# How to add new column to our data frame

gender_info = pd.Series(['M','M','M'])
gender_info
```

Out[64]:
```
0    M
```

```
1    M
2    M
dtype: object
```

In [65]:

```
missing_data['gender'] = gender_info
missing_data
```

Out[65]:

	name	country	occupation	age	gender
0	Ashbab khan	India	Data scientist	24.0	M
1	David	Unknown	Web developer	29.0	M
4	Joe	Unknown	Data engineer	31.0	NaN

In [71]:

```
# how to drop column from data frames
missing_data.drop('gender',axis=1,inplace=True)
```

In [79]:

```
missing_data
```

Out[79]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24.0
1	David	Unknown	Web developer	29.0
4	Joe	Unknown	Data engineer	31.0

In [81]:

```
# we can also shuffle our row with sample function
# frac is used to get how many rows we want
# 0.4 means 40% and 1 means 100%

missing_data.sample(frac=1)
```

Out[81]:

	name	country	occupation	age
4	Joe	Unknown	Data engineer	31.0
0	Ashbab khan	India	Data scientist	24.0

	name	country	occupation	age
1	David	Unknown	Web developer	29.0

In [82]:

```
missing_data
```

Out[82]:

	name	country	occupation	age
0	Ashbab khan	India	Data scientist	24.0
1	David	Unknown	Web developer	29.0
4	Joe	Unknown	Data engineer	31.0

In [ ]:

```

```