#Reshaping a pandas dataframe is one of the most common data wrangling tasks in $_$ \hookrightarrow the data analysis world. #It is also referred to as transposing or pivoting/unpivoting a table from long $_$ \hookrightarrow to wide or from wide to long format.

#Reshaping a pandas dataframe is one of the most common data wrangling tasks in $_$ \hookrightarrow the data analysis world. #It is also referred to as transposing or pivoting/unpivoting a table from long $_$ \hookrightarrow to wide or from wide to long format.

```
# import pandas library
```

import pandas as pd

make an array

```
array = [2, 4, 6, 8, 10, 12]
```

create a series

```
series_obj = pd.Series(array)
```

convert series object into array

```
arr = series_obj.values
```

arr

```
[1]: array([ 2, 4, 6, 8, 10, 12], dtype=int64)
```

reshaping series

```
reshaped_arr = arr.reshape((3, 2))
```

show

reshaped_arr

import pandas library

import pandas as pd

make an array

```
array = ["ankit","shaurya", "shivangi", "priya", "jeet","ananya"]
```

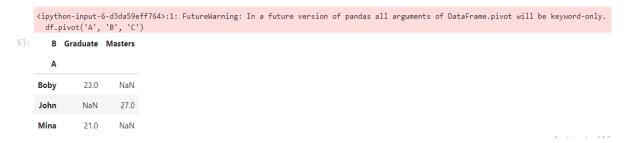
create a series

```
series_obj = pd.Series(array)
```

```
print("Given Series:\n", series_obj)
# convert series object into array
arr = series_obj.values
arr
      Given Series:
       0
                ankit
      1
           shaurya
      2
          shivangi
      3
               priya
                jeet
      4
              ananya
      dtype: object
[3]: array(['ankit', 'shaurya', 'shivangi', 'priya', 'jeet', 'ananya'],
             dtype=object)
# reshaping series
reshaped_arr = arr.reshape((2, 3))
# show
print("After Reshaping: \n", reshaped_arr)
   After Reshaping:
    [['ankit' 'shaurya' 'shivangi']
    ['priya' 'jeet' 'ananya']]
# Create a simple dataframe
# importing pandas as pd
import pandas as pd
# creating a dataframe
df = pd.DataFrame({'A': ['John', 'Boby', 'Mina'], 'B': ['Masters', 'Graduate', 'Graduate'], 'C': [27, 23,
21]})
df
[5]:
            Α
                         C
                      В
      0 John Masters 27
      1 Boby Graduate 23
      2 Mina Graduate 21
```

values can be an object or a list

df.pivot('A', 'B', 'C')



value is a list

df.pivot(index ='A', columns ='B', values =['C', 'A'])

[7]:			Α		
	В	Graduate	Masters	Graduate	Masters
	Α				
	Boby	23	NaN	Boby	NaN
	John	NaN	27	NaN	John
	Mina	21	NaN	Mina	NaN

importing pandas library

import pandas as pd

creating dataframe

```
df = pd.DataFrame({'Name': ['John', 'Sammy', 'Stephan', 'Joe', 'Emily', 'Tom'],
                     'Gender': ['Male', 'Female', 'Male', 'Female', 'Female', 'Male'],
                     'Age': [45, 6, 4, 36, 12, 43]})
print("Dataset")
print(df)
print("-"*40)
```

categorizing in age groups

```
def age_bucket(age):
        if age <= 18:
                return "18"
        else:
```

```
return ">18"
```

df['Age Group'] = df['Age'].apply(age_bucket)

calculating gender percentage

```
gender = pd.DataFrame(df.Gender.value_counts(normalize=True)*100).reset_index()
gender.columns = ['Gender', '%Gender']
```

df = pd.merge(left=df, right=gender, how='inner', on=['Gender'])

creating pivot table

table = pd.pivot_table(df, index=['Gender', '%Gender', 'Age Group'],

values=['Name'], aggfunc={'Name': 'count',})

display table print("Table")

print(table)

Dataset	
---------	--

	Name	Gender	Age
0	John	Male	45
1	Sammy	Female	6
2	Stephan	Male	4
3	Joe	Female	36
4	Emily	Female	12
5	Tom	Male	43

Table

				Name
Gender	%Gender	Age	Group	
Female	50.0	<18		2
		>18		1
Male	50.0	<18		1
		>18		2