In [3]:

```
import pandas as pd
import numpy as np
days = pd.Series(['Mon', 'Tue', 'Wed'])
print(days)
```

```
0    Mon
1    Tue
2    Wed
dtype: object
```

In [4]:

```
# creating from numpy array
days_lst = np.array(['Mon', 'Tue', 'Wed'])
pd_days = pd.Series(days_lst)
print(pd_days)
```

```
0    Mon
1    Tue
2    Wed
dtype: object
```

In [5]:

```
# creating from regular python list
days_lst = ['Mon', 'Tue', 'Wed']
pd_days = pd.Series(days_lst)
print(pd_days)
```

```
0    Mon
1    Tue
2    Wed
dtype: object
```

In [6]:

```
days_lst = pd.Series(['Mon', 'Tue', 'Wed'], index=['a', 'b', 'c'])
```

In [7]:

```
days_lst
```

Out[7]:

```
a    Mon
b    Tue
c    Wed
dtype: object
```

In [8]:

```
# creating from dictionary
d1 = pd.Series({'a':'Monday', 'b':'Tuesday', 'c':'Wednesday'})
d1
```

Out[8]:

```
a       Monday
b      Tuesday
c    Wednesday
dtype: object
```

In [9]:

```
d1[0]
```

Out[9]:

```
'Monday'
```

In [10]:

```
d1[1:]
```

Out[10]:

```
b      Tuesday
c    Wednesday
dtype: object
```

In [11]:

```
d1['c']
```

Out[11]:

```
'Wednesday'
```

DataFrame

In [12]:

```
print(pd.DataFrame())
```

```
Empty DataFrame
Columns: []
Index: []
```

In [20]:

```
# create dataframe from dict
df_dict = {'Country':['Ghana', 'Nigeria', 'Togo', 'Kenya'],
           'Capital':['Accra', 'Abuja', 'Lome', 'Nairobi'],
           'Population':['100000', '140000', '67000', '87000'],
           'Age':['60', '70', '75', '80'],
          }
df = pd.DataFrame(df_dict, index=[2,4,6,8])
df
```

Out[20]:

	Country	Capital	Population	Age
2	Ghana	Accra	100000	60
4	Nigeria	Abuja	140000	70
6	Togo	Lome	67000	75
8	Kenya	Nairobi	87000	80

In [16]:

```
# creating from regular python list
df_list = [['Ghana', 'Accra', 12000, 87],
           ['Nigeria', 'Abuja', 45000, 67],
           ['Togo', 'Lome', 23000, 35],
           ['Kenya', 'Nairobi', 10000, 57],
          ]
df1 = pd.DataFrame(df_list, columns=['Country', 'Capital', 'Population', 'Age'], index=[1,2
df1
```

Out[16]:

	Country	Capital	Population	Age
1	Ghana	Accra	12000	87
2	Nigeria	Abuja	45000	67
3	Togo	Lome	23000	35
4	Kenya	Nairobi	10000	57

retrieving values using at, iat, iloc and loc

In [21]:

```
# select the row in the 'at' index 3
df.iloc[3]
```

Out[21]:

```
Country         Kenya
Capital       Nairobi
Population      87000
Age                80
Name: 8, dtype: object
```

In [22]:

```
df.iloc[2:]
```

Out[22]:

	Country	Capital	Population	Age
6	Togo	Lome	67000	75
8	Kenya	Nairobi	87000	80

In [24]:

```
df['Country']
```

Out[24]:

```
2      Ghana
4    Nigeria
6       Togo
8      Kenya
Name: Country, dtype: object
```

In [25]:

```
df.at[6, 'Country'] # select using "at"
```

Out[25]:

```
'Togo'
```

In [31]:

```
df.iat[2, 1] # i.e row 2 col 1
```

Out[31]:

```
'Lome'
```

In [34]:

```
df.iat[3, 3]
```

Out[34]:

```
'80'
```

In [35]:

```
df1.iat[2,3]
```

Out[35]:

```
35
```

In [36]:

```
df1['Capital']
```

Out[36]:

```
1      Accra
2      Abuja
3       Lome
4    Nairobi
Name: Capital, dtype: object
```

In [38]:

```
df['Age'].sum()
```

Out[38]:

```
'60707580'
```

In [39]:

```
df.mean()
```

Out[39]:

```
Population    2.500004e+20
Age           1.517690e+07
dtype: float64
```

In [40]:

```
df.describe()
```

Out[40]:

	Country	Capital	Population	Age
count	4	4	4	4
unique	4	4	4	4
top	Togo	Lome	87000	75
freq	1	1	1	1

In [41]:

```
df1.describe()
```

Out[41]:

	Population	Age
count	4.000000	4.000000
mean	22500.000000	61.500000
std	16051.998837	21.625602
min	10000.000000	35.000000
25%	11500.000000	51.500000
50%	17500.000000	62.000000
75%	28500.000000	72.000000
max	45000.000000	87.000000

In [43]:

```
df1.dtype()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-43-61804cd3531d> in <module>
----> 1 df1.dtype()

~\anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, na
me)
   5272             if self._info_axis._can_hold_identifiers_and_holds_name(
name):
   5273                 return self[name]
-> 5274             return object.__getattribute__(self, name)
   5275 
   5276     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'dtype'
```

missing data

In [45]:

```
df_dict2 = {'Name':['James','Yemen', 'Caro', np.nan],
            'Profession':['Researcher','Trader', 'Teacher', 'Doctor'],
            'Experience':[12, np.nan, 10, 8],
            'Height':[np.nan, 175, 180, 150],
           }
df3 = pd.DataFrame(df_dict2, index=[1,2,3,4])
df3
```

Out[45]:

	Name	Profession	Experience	Height
1	James	Researcher	12.0	NaN
2	Yemen	Trader	NaN	175.0
3	Caro	Teacher	10.0	180.0
4	NaN	Doctor	8.0	150.0

Check the cells with missing values as True

In [46]:

```
df3.isnull()
```

Out[46]:

	Name	Profession	Experience	Height
1	False	False	False	True
2	False	False	True	False
3	False	False	False	False
4	True	False	False	False

In [47]:

```
# remove rows with missing values
df3.dropna()
```

Out[47]:

	Name	Profession	Experience	Height
3	Caro	Teacher	10.0	180.0

In [48]:

```
data = {'apples':[2,4,6,4],
        'oranges':[0,5,3,1]}
p = pd.DataFrame(data)
p
```

Out[48]:

	apples	oranges
0	2	0
1	4	5
2	6	3
3	4	1

In [49]:

```
p.loc[0]
```

Out[49]:

```
apples     2
oranges    0
Name: 0, dtype: int64
```

In [50]:

```
def header(msg):
    print('-' * 50)
    print('[' + msg + ']')
```

In [53]:

```
header("1. load hard coded data into dataframe")
p = pd.DataFrame(data)
p
```

```
--------------------------------------------------
[1. load hard coded data into dataframe]
```

Out[53]:

	apples	oranges
0	2	0
1	4	5
2	6	3
3	4	1

In [52]:

```
p = pd.DataFrame(data)
```

```

```