In [1]:

```
# A Pandas series is a data structure that stores data in the form
# of a column. A series is normally used to store information
# about a particular attribute in your dataset. Let's see how you
# can create a series in Pandas.
# There are different ways to create a series with Pandas. The
# following script imports the Pandas module and then calls the
# Series() class constructor to create an empty series. Here is
# how to do that:

# empty series
import pandas as pd
my_series = pd.Series()
print(my_series)
```

```
Series([], dtype: float64)

<ipython-input-1-c46511ad4fca>:12: DeprecationWarning: The default dtype f
or empty Series will be 'object' instead of 'float64' in a future version.
Specify a dtype explicitly to silence this warning.
  my_series = pd.Series()
```

In [3]:

```
# You can also create a series using a NumPy array. But, first,
# you need to pass the array to the Series() class constructor, as
# shown in the script below.

# series using numpy array
import pandas as pd
import numpy as np
my_array = np.array([10, 20, 30, 40, 50])
my_series = pd.Series(my_array)
print(my_series)

# In the output, you can see that the indexes for a series
# start from 0 to 1 less than the number of items in the series.
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int32
```

In [4]:

```
# You can also define custom indexes for your series. To do so,
# you need to pass your list of indexes to the index attribute of
# the Series class, as shown in the script below:
# series with custom indexes

import pandas as pd
import numpy as np
my_array = np.array([10, 20, 30, 40, 50])
my_series = pd.Series(my_array, index = ["num1", "num2", "num3", "num4", "num5"])
print(my_series)
```

```
num1    10
num2    20
num3    30
num4    40
num5    50
dtype: int32
```

In [5]:

```
# You can also create a series by directly passing a Python list
# to the Series() class constructor.
# series using a list

import pandas as pd
import numpy as np
my_series = pd.Series([10, 20, 30, 40, 50], index = ["num1", "num2", "num3", "num4", "nu
print(my_series)
```

```
num1    10
num2    20
num3    30
num4    40
num5    50
dtype: int64
```

In [6]:

```
# Finally, a scaler value can also be used to define a series. In
# case you pass a list of indexes, the scaler value will be
# repeated the number of times equal to the items in the index
# list. Here is an example:
# series using a scaler

import pandas as pd
import numpy as np
my_series = pd.Series(25, index = ["num1", "num2", "num3", "num4", "num5"])
print(my_series)
```

```
num1    25
num2    25
num3    25
num4    25
num5    25
dtype: int64
```

In [7]:

```
# Finally, you can also create a series using a dictionary. In this
# case, the dictionary keys will become series indexes while the
# dictionary values are inserted as series items. Here is an
# example:
# series using dictionary

import pandas as pd
import numpy as np
my_dict = {'num1' :6,
'num2' :7,
'num3' :8}
my_series = pd.Series(my_dict)
print(my_series)
```

```
num1    6
num2    7
num3    8
dtype: int64
```

In [8]:

```
# Let's see some of the useful operations you can perform with
# the Pandas series.
# You can use square brackets as well as index labels to access
# series items, as shown in the following script:

## Accessing Items
import pandas as pd
my_series = pd.Series([10, 20, 30, 40, 50], index = ["num1", "num2", "num3", "num4", "nu
print(my_series[0])
print(my_series['num3'])
```

```
10
30
```

In [9]:

```
# Using the min() and max() functions from the NumPy module,
# you can find the maximum and minimum values, respectively,
# from a series. Look at the following script for reference.
# Finding Maximum and Minimum Values

import pandas as pd
import numpy as np
my_series = pd.Series([5, 8, 2, 11, 9])
print(np.min(my_series))
print(np.max(my_series))
```

```
2
11
```

In [10]:

```
# Similarly, the mean() method from the NumPy module can
# find the mean of a Pandas series, as shown in the following
# script.
# Finding Mean

import pandas as pd
import numpy as np
my_series = pd.Series([5, 8, 2, 11, 9])
print(my_series.mean())
```

```
7.0
```

In [12]:

```
# The following script finds the median value of a Pandas series.
# Finding Median

import pandas as pd
import numpy as np
my_series = pd.Series([5, 8, 2, 11, 9])
print(my_series.median())
```

```
8.0
```

In [13]:

```
# You can also find the data type of a Pandas series using the
# dtype attribute. Here is an example:

## Finding Data Type
import pandas as pd
import numpy as np
my_series = pd.Series([5, 8, 2, 11, 9])
print(my_series.dtype)
```

```
int64
```