Pandas Introduction

What is Pandas?

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

What Can Pandas Do?

Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called *cleaning* the data.

Import Pandas

Once Pandas is installed, import it in your applications by adding the `import` keyword:

```
import pandas
```

```
import pandas as pd

mydataset = {  'cars': ["BMW", "Volvo", "Ford"],'passings': [3, 7, 2]}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

Pandas as pd

Pandas is usually imported under the `pd` alias.

alias: In Python alias are an alternate name for referring to the same thing.

Create an alias with the `as` keyword while importing:

```
import pandas as pd
```

Now the Pandas package can be referred to as `pd` instead of `pandas`.

EXAMPLE

```
import pandas as pd

mydataset = {'cars': ["BMW", "Volvo", "Ford"],'passings': [3, 7, 2]}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

Checking Pandas Version

The version string is stored under __version__ attribute.

Example

```
import pandas as pd

print(pd.__version__)
```

Pandas Series

What is a Series?

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

Create a simple Pandas Series from a list:

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a)
print(myvar)
```

Labels

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc.

This label can be used to access a specified value.

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar[0])
```

Create Labels

With the `index` argument, you can name your own labels.

EXAMPLE

Create your own labels:

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a, index = ["x", "y", "z"])
print(myvar)
```

When you have created labels, you can access an item by referring to the label.

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar["y"])
```

Key/Value Objects as Series

You can also use a key/value object, like a dictionary, when creating a Series.

Example

Create a simple Pandas Series from a dictionary:

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

Example

Create a Series using only data from "day1" and "day2":

```
import pandas as pd
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories, index = ["day1", "day2"])
print(myvar)
```

DataFrames

Data sets in Pandas are usually multi-dimensional tables, called DataFrames.

Series is like a column, a DataFrame is the whole table.

Example

Create a DataFrame from two Series:

```
import pandas as pd
data = { "calories": [420, 380, 390], "duration": [50, 40, 45]}
myvar = pd.DataFrame(data)
print(myvar)
```

What is a DataFrame?

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

Example

Create a simple Pandas DataFrame:

```
import pandas as pd
data = { "calories": [420, 380, 390],"duration": [50, 40, 45]}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
```

Locate Row

As you can see from the result above, the DataFrame is like a table with rows and columns.

Pandas use the `loc` attribute to return one or more specified row(s)

```
import pandas as pd

data = {  "calories": [420, 380, 390], "duration": [50, 40, 45]}

df = pd.DataFrame(data)

print(df.loc[0])
```

```
import pandas as pd

data = { "calories": [420, 380, 390],"duration": [50, 40, 45]}

df = pd.DataFrame(data)

print(df.loc[[0, 1]])
```

Named Indexes

With the `index` argument, you can name your own indexes.

Example

```
#Add a list of names to give each row a name:

import pandas as pd
data = {"calories": [420, 380, 390],"duration": [50, 40, 45]}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

Locate Named Indexes

Use the named index in the `loc` attribute to return the specified row(s)

```
            import pandas as pd

data = { "calories": [420, 380, 390], "duration": [50, 40, 45]}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df.loc["day2"])
```