codecademy

Hands-On with Pandas

Pandas DataFrame creation

The fundamental Pandas object is called a DataFrame. It is a 2-dimensional size-mutable, potentially heterogeneous, tabular data structure.
A DataFrame can be created multiple ways. It can be created by passing in a dictionary or a list of lists to the `pd.DataFrame()` method, or by reading data from a CSV file.

```
# Ways of creating a Pandas DataFrame
# Passing in a dictionary:
data = {'name':['Anthony', 'Maria'], 'age':
[30, 28]}
df = pd.DataFrame(data)

# Passing in a list of lists:
data = [['Tom', 20], ['Jack', 30], ['Meera',
25]]
df = pd.DataFrame(data, columns = ['Name',
'Age'])

# Reading data from a csv file:
df = pd.read_csv('students.csv')
```

Pandas

Pandas is an open source library that is used to analyze data in Python. It takes in data, like a CSV or SQL database, and creates an object with rows and columns called a data frame. Pandas is typically imported with the alias `pd`.

```
import pandas as pd
```

Selecting Pandas DataFrame rows using logical operators

In pandas, specific rows can be selected if they satisfy certain conditions using Python's logical operators. The result is a DataFrame that is a subset of the original DataFrame.
Multiple logical conditions can be combined with OR (using `|`) and AND (using `&`), and each condition must be enclosed in parentheses.

```
# Selecting rows where age is over 20
df[df.age > 20]

# Selecting rows where name is not John
df[df.name != "John"]

# Selecting rows where age is less than 10
# OR greater than 70
df[(df.age < 10) | (df.age > 70)]
```

Pandas apply() function

The Pandas `apply()` function can be used to apply a function on every value in a column or row of a DataFrame, and transform that column or row to the resulting values.
By default, it will apply a function to all values of a column. To perform it on a row instead, you can specify the argument `axis=1` in the `apply()` function call.

```
# This function doubles the input value
def double(x):
  return 2*x

# Apply this function to double every value in
a specified column
df.column1 = df.column1.apply(double)

# Lambda functions can also be supplied to
`apply()`
df.column2 = df.column2.apply(lambda x : 3*x)

# Applying to a row requires it to be called
on the entire DataFrame
df['newColumn'] = df.apply(lambda row:
  row['column1'] * 1.5 + row['column2'],
  axis=1
)
```

Pandas DataFrames adding columns

Pandas DataFrames allow for the addition of columns after the DataFrame has already been created, by using the format `df['newColumn']` and setting it equal to the new column's value.

```
# Specifying each value in the new column:
df['newColumn'] = [1, 2, 3, 4]

# Setting each row in the new column to the
same value:
df['newColumn'] = 1

# Creating a new column by doing a
# calculation on an existing column:
df['newColumn'] = df['oldColumn'] * 5
```