YData: Introduction to Data Science

Class 08: Intro to pandas

Overview

Review and continuation of functions

pandas Series

pandas DataFrames

- Selecting columns
- Getting subsets of rows
- Sorting data
- Adding new columns
- Getting summary statistics by groups

Announcement: Homework 3

Homework 3has been posted!

It is due on Gradescope on Sunday February 12th at 11pm

- **Be sure to mark each question on Gradescope along with the page that has the answers!**

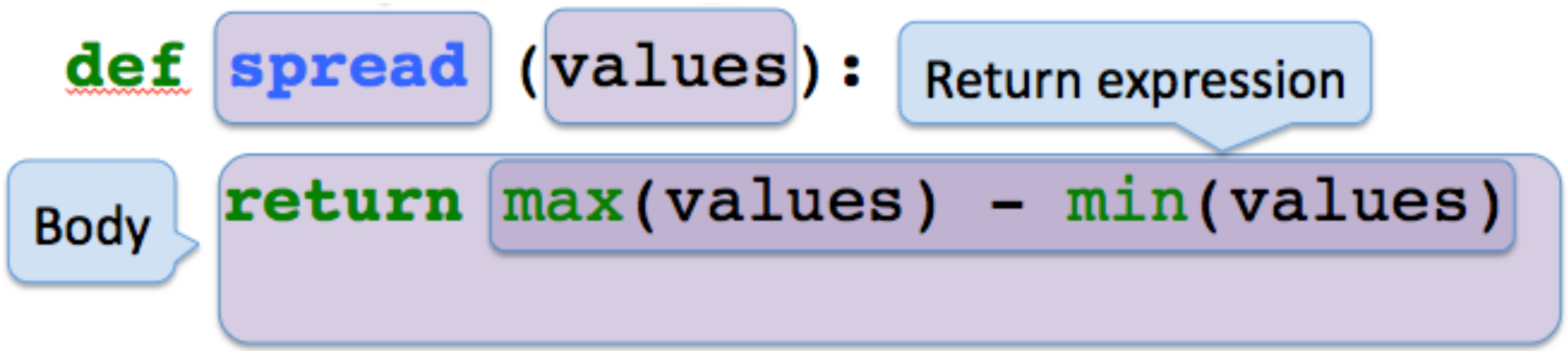
Notes:

- Homework might be a little longer so start early (no Q&R)
- When writing functions, useful to test code outside of the function to make sure it works, then put in into a function

Review: Defining functions

Review: Def statements

User-defined functions give names to blocks of code



```
def spread(values):
    return max(values) - min(values)
```

Let's explore this in Jupyter!

Tuples and returning multiple arguments

Tuples are like lists but they are immutable; i.e., once they are created we can’t change the values in a tuple.

We can create a tuple using:

- my_tuple = (10, 20, 30)

Like lists, we can access elements of tuples using square brackets

- my_tuple[1]

We can’t change values in tuples:

- my_tuple[1] = 50   # Error!!!

Tuples and returning multiple arguments

We can assign values in tuples into regular names using “tuple unpacking”

- my_tuple = (10, 20, 30)
- val1, val2, val3 = my_tuple
- val3

Functions can return tuples which allow us to return multiple names

- val1, val2 = my_function()

Let's explore this in Jupyter!

Passing functions as input arguments

We have passed numbers and arrays as arguments to functions:

- power3(x)   # x here is a number

We can also have functions take other *functions* as input arguments

- my_fun(np.mean)   # passing the np.mean function as an input argument
- my_fun(np.min)   # passing the np.min function as an input argument

Let's explore this in Jupyter!

Series and Tables

Pandas: Series and DataFrames

"pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language."

There are two main data structures in pandas:

- **Series**: represent one-dimensional data
- **DataFrames**: represent data tables
  - i.e., relational data

pandas Series

pandas Series are: One-dimensional ndarray with axis labels

- (including time series)

Example: egg _prices

```
DATE
1980-01-01        0.879
1980-02-01        0.774
1980-03-01        0.812
```

Index

values

pandas Series

We can access elements by Index ***name*** using **.loc**

- egg_prices.loc["1980-01-01"]

We can access elements by Index ***number*** using **.iloc**

- egg_prices.iloc[0]

Let's explore this in Jupyter!

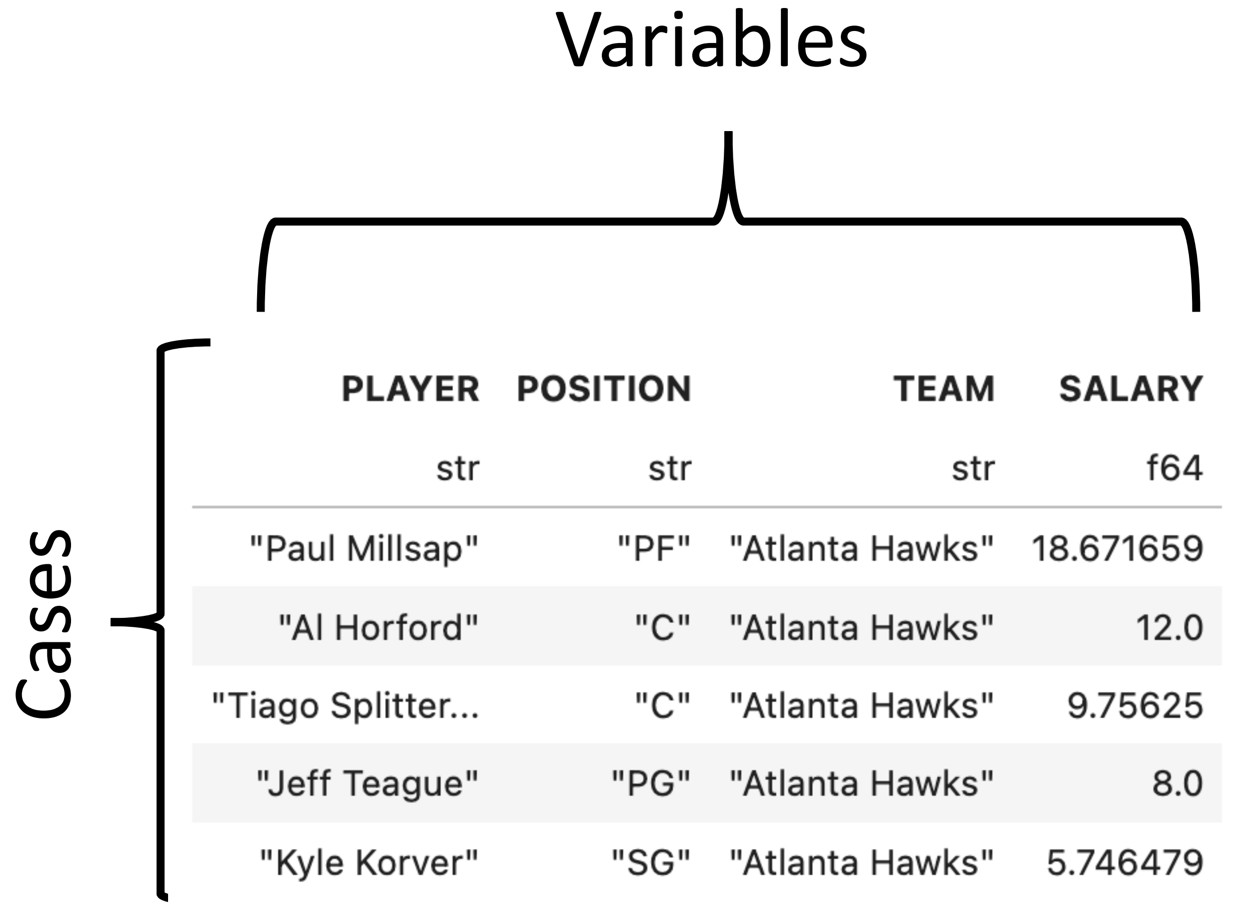
pandas DataFrames

Pandas DataFrame hold Table data

This is one of the most useful formats to extract insights from datasets

Often we read data into a DataFrame using:

- pd.read_csv(“file.csv”)


Variables
Cases

PLAYER	POSITION	TEAM	SALARY
str	str	str	f64
"Paul Millsap"	"PF"	"Atlanta Hawks"	18.671659
"Al Horford"	"C"	"Atlanta Hawks"	12.0
"Tiago Splitter...	"C"	"Atlanta Hawks"	9.75625
"Jeff Teague"	"PG"	"Atlanta Hawks"	8.0
"Kyle Korver"	"SG"	"Atlanta Hawks"	5.746479

Let's explore this in Jupyter!

Selecting columns from a DataFrame

We can select a column from a DataFrame using square brackets:

- my_df[“my_col”]   # returns a Series!

We can select multiple columns from a DataFrame by passing a list into the square brackets

- my_df[[“col1”, “col2”]]

Let's explore this in Jupyter!

Extracting rows from a DataFrame

We can extract rows from a DataFrame by:

1. The position they appear in the DataFrame
2. The Index values

We use the .iloc[] property to extract values by ***position***

- my_df.iloc[0]

We use the .loc[] property to extract values by ***Index value***

- my_df.loc[“index_name”]

Let's explore this in Jupyter!

Sorting rows from a DataFrame

We can sort values in a DataFrame using .sort_values("col_name")

- my_df.sort_values("col_name")

We can sort from highest to lowest by setting the argument ascending = False

- my_df.sort_values("col_name", ascending = False)

Let's explore this in Jupyter!

Adding new columns to a DataFrame

We can add a column to a data frame using square backets. For example:

- my_df[“new_col”] = values_array
- my_df["new col"] = my_df["col1"] + my_df["col2"]`

Let's explore this in Jupyter!

Creating aggregate statistics by group

There are several ways to get statistics by group

- Most methods use the .groupby() and .agg() methods

Perhaps the most useful way is to use the syntax:

```
my_df.groupby("group_col_name").agg(
    new_col1 = ('col_name', 'statistic_name1’),
    new_col2 = ('col_name', 'statistic_name2’),
    new_col3 = ('col_name', 'statistic_name3')
)
```

Let's explore this in Jupyter!

Next class: pandas continued…