Exploring Pandas
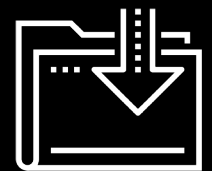
Class Objectives

By the end of today's class, you will be able to:

Navigate through DataFrames using `loc` and `iloc`.

Filter and slice Pandas DataFrames.

Create and access Pandas `groupby` objects.

Sort DataFrames.

Instructor Demonstration

Exploring Data with loc and iloc

Programmers can easily collect specific rows and columns of data from a DataFrame by using the `loc` and `iloc` methods.

Exploring Data with `loc` and `iloc`

- `loc` returns data based on an index of labels/strings

- `loc` is limited to string types and cannot be used on a numerical index. As an alternative solution, you can use the `df.set_index()` function, passing in the desired column header for the index.

- Instead of using labels, `iloc` uses integer-based indexing for selection by position.

```python
# Set new index to STREET NAME
df = original_df.set_index("STREET NAME")
df.head()
```

STREET NAME	STREET NAME ID	STREET FULL NAME	POSTAL COMMUNITY	MUNICIPAL COMMUNITY
PRIVATE STREET	1400342	PRIVATE STREET	BATON ROUGE	BATON ROUGE
4TH	1	N 4TH ST	BATON ROUGE	BATON ROUGE
11TH	10	S 11TH ST	BATON ROUGE	BATON ROUGE
ADDINGTON	100	ADDINGTON AVE	BATON ROUGE	BATON ROUGE
CHALFONT	1000	W CHALFONT DR	BATON ROUGE	PARISH

Exploring Data with `loc` and `iloc`

- Both `loc` and `iloc` use brackets that contain the desired rows, followed by a comma and the desired columns.

- For example: `loc["ADDINGTON", "STREET FULL NAME"]`

  gives the same result as: `iloc[3,1]`

```python
# Set new index to STREET NAME
df = original_df.set_index("STREET NAME")
df.head()
```

STREET NAME	STREET NAME ID	STREET FULL NAME	POSTAL COMMUNITY	MUNICIPAL COMMUNITY
PRIVATE STREET	1400342	PRIVATE STREET	BATON ROUGE	BATON ROUGE
4TH	1	N 4TH ST	BATON ROUGE	BATON ROUGE
11TH	10	S 11TH ST	BATON ROUGE	BATON ROUGE
ADDINGTON	100	ADDINGTON AVE	BATON ROUGE	BATON ROUGE
CHALFONT	1000	W CHALFONT DR	BATON ROUGE	PARISH

`loc` and `iloc` can be used to conditionally filter rows of data based on the values within a column.

Activity: Good Movies

In this activity, you will create an application that searches through IMDb data to find only the best movies out there.

Suggested Time:

20 minutes

Time's Up! Let's Review.

Instructor Demonstration

Cleaning Data

When dealing with massive datasets, it's almost inevitable that we'll encounter duplicate rows, inconsistent spelling, and missing values.

Cleaning Data

```
del <DataFrame>[<columns>]
```

In [4]: `# Preview of the DataFrame`
`# Note that Memo_CD is likely a meaningless column`
`df.head()`

Out[4]:

	Name	Employer	City	State	Zip	Amount	Memo_CD
0	CAREY, JAMES	NOT EMPLOYED	HOCKESSIN	DE	197071618.0	500	NaN
1	OBICI, SILVANA	STONY BROOK	PORT JEFFERSON STATION	NY	117764286.0	250	NaN
2	MAISLIN, KAREN	RETIRED	WILLIAMSVILLE	NY	14221.0	250	NaN
3	MCCLELLAND, CARTER AND STEPHANIE	UNION SQUARE ADVISORS	NEW YORK	NY	10023.0	1000	NaN
4	MCCLUSKEY, MARTHA	STATE UNIVERSITY OF NEW YORK	BUFFALO	NY	14214.0	250	NaN

In [5]: `# Delete extraneous column`
`del df['Memo_CD']`
`df.head()`

Out[5]:

	Name	Employer	City	State	Zip	Amount
0	CAREY, JAMES	NOT EMPLOYED	HOCKESSIN	DE	197071618.0	500
1	OBICI, SILVANA	STONY BROOK	PORT JEFFERSON STATION	NY	117764286.0	250
2	MAISLIN, KAREN	RETIRED	WILLIAMSVILLE	NY	14221.0	250
3	MCCLELLAND, CARTER AND STEPHANIE	UNION SQUARE ADVISORS	NEW YORK	NY	10023.0	1000
4	MCCLUSKEY, MARTHA	STATE UNIVERSITY OF NEW YORK	BUFFALO	NY	14214.0	250

Cleaning Data

```
count()
<DataFrame>.dropna(how='any')
```

```
In [6]:  # Identify incomplete rows
         df.count()

Out[6]:  Name        2000
         Employer    1820
         City        1999
         State       1999
         Zip         1996
         Amount      2000
         dtype: int64
```

```
In [7]:  # Drop all rows with missing information
         df = df.dropna(how='any')
```

```
In [8]:  # Verify dropped rows
         df.count()

Out[8]:  Name        1818
         Employer    1818
         City        1818
         State       1818
         Zip         1818
         Amount      1818
         dtype: int64
```

Cleaning Data

value_counts()

replace()

```
In [12]:  # Display an overview of the Employers column
          df['Employer'].value_counts()

Out[12]:  NOT EMPLOYED              609
          NONE                     321
          SELF-EMPLOYED            132
          SELF                      33
          RETIRED                   32
                                   ...
          INTEL CORPORATION          1
          SLOCUM & SONS              1
          OCPS                       1
          HEALTHCARE PARTNERS        1
          CARBON FIVE                1
          Name: Employer, Length: 519, dtype: int64

In [13]:  # Clean up Employer category. Replace 'SELF' and 'SELF EMPLOYED' with 'SELF-EMPLOYED'
          df['Employer'] = df['Employer'].replace({'SELF': 'SELF-EMPLOYED', 'SELF EMPLOYED': 'SELF-EMPLOYED'})

In [14]:  # Verify clean-up.
          df['Employer'].value_counts()

Out[14]:  NOT EMPLOYED              609
          NONE                     321
          SELF-EMPLOYED            180
          RETIRED                   32
          INGRAM BARGE COMPANY      30
```

Activity: Hong Kong LPG Appliances

In this activity, you will take an LPG appliance dataset from Hong Kong, and clean it so that the DataFrame is consistent, and that there are no rows with missing data.

Suggested Time:

20 minutes

Time's Up! Let's Review.

Activity: Pandas Recap and Data Types

In this activity, we will recap what has been covered in Pandas up to this point.

Suggested Time:

15 minutes

Instructor Demonstration

Pandas Grouping

`groupby` is a simpler method for filtering data.

Pandas Grouping

To split the DataFrame into multiple groups and group by state, we use `df.groupby([<Columns>])`.

The `groupby` method returns a `groupby` object that can only be accessed by using a data function on it.

```python
# Count how many loss incidents occured in each city
grouped_city_df = loss_df.groupby(["Incident City"])

print(grouped_city_df)

grouped_city_df.count().head(10)
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fd919ddadf0>
```

Incident City	Fire Department Name	Incident date	Incident Type Code	Incident Type	Alarm Date and Time	Arrival Date and Time	Last Unit Cleared Date and Time	Property Loss	Contents Loss	Fire Service Deaths	Fire Service Injuries	Other Fire Deaths	Other Fire Injuries	Incident Zip Code	Response Time (seconds)	In Du (se
AMSTON	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
ANSONIA	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
AVON	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	
Andover	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
Ansonia	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	
BERLIN	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	
BETHEL	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	
BLOOMFIELD	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	
BRANFORD	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	
BRIDGEPORT	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	

Pandas Grouping

The `pd.DataFrame()` method makes it possible to create new DataFrames by using only `groupby` data.

A DataFrame can also be created by selecting a single Series from a `groupby` object and passing it in as the values for a specified column.

```python
# Save loss sums as series
city_property_loss = grouped_city_df["Property Loss"].sum()
city_contents_loss = grouped_city_df["Contents Loss"].sum()
city_contents_loss.head()
```

```
Incident City
AMSTON          5000.0
ANSONIA          600.0
AVON            1250.0
Andover          500.0
Ansonia       265100.0
Name: Contents Loss, dtype: float64
```

```python
# Create a new DataFrame using count and loss amounts
city_summary_df = pd.DataFrame({"Number of Loss Incidents": city_loss_counts,
                                "Total Property Loss": city_property_loss,
                                "Total Contents Loss": city_contents_loss})

city_summary_df.head()
```

	Number of Loss Incidents	Total Property Loss	Total Contents Loss
AMSTON	1	65000.0	5000.0
ANSONIA	2	5000.0	600.0
AVON	6	14200.0	1250.0
Andover	3	2500.0	500.0
Ansonia	18	644100.0	265100.0

Pandas Grouping

It's also possible to perform a `df.groupby()` method on multiple columns by passing two or more column references into the list parameter.

```python
# It is also possible to group a DataFrame by multiple columns
# This returns an object with multiple indexes, however, which can be harder to deal with
grouped_city_loss_incidents = loss_df.groupby(["Incident City","Incident Type Code"])

grouped_city_loss_incidents.count().head(10)
```

Incident City	Incident Type Code	Fire Department Name	Incident date	Incident Type	Alarm Date and Time	Arrival Date and Time	Last Unit Cleared Date and Time	Property Loss	Contents Loss	Fire Service Deaths	Fire Service Injuries	Other Fire Deaths	Other Fire Injuries
AMSTON	111	1	1	1	1	1	1	1	1	1	1	1	1
ANSONIA	111	2	2	2	2	2	2	2	2	2	2	2	2
AVON	111	3	3	3	3	3	3	3	3	3	3	3	3
	113	1	1	1	1	1	1	1	1	1	1	1	1
	114	1	1	1	1	1	1	1	1	1	1	1	1
	131	1	1	1	1	1	1	1	1	1	1	1	1
Andover	111	2	2	2	2	2	2	2	2	2	2	2	2
	113	1	1	1	1	1	1	1	1	1	1	1	1
Ansonia	111	12	12	12	12	12	12	12	12	12	12	12	12
	113	2	2	2	2	2	2	2	2	2	2	2	2

Pandas Grouping

A new DataFrame can be created from a groupby object.

```python
# Converting a GroupBy object into a DataFrame
total_city_loss_df = pd.DataFrame(
    grouped_city_loss_incidents[["Property Loss", "Contents Loss"]].sum())
total_city_loss_df.head(10)
```

Incident City	Incident Type Code	Property Loss	Contents Loss
AMSTON	111	65000.0	5000.0
ANSONIA	111	5000.0	600.0
AVON	111	8000.0	1200.0
	113	0.0	50.0
	114	1000.0	0.0
	131	5200.0	0.0
Andover	111	2500.0	200.0
	113	0.0	300.0
Ansonia	111	617100.0	263500.0
	113	5000.0	500.0

Activity: Exploring U.S. Census Data

In this activity, you will revisit the U.S. Census data and create DataFrames with calculated totals and averages of each state by year.

Suggested Time:

25 minutes

Time's Up! Let's Review.

Instructor Demonstration

Sorting Made Easy

Sorting with Pandas

To sort a DataFrame based on the values within a column, use the `df.sort_values()` method and pass in the column name to sort by as a parameter.

The "ascending" parameter is always marked as True by default. Therefore, the `sort_values()` method will always sort from lowest to highest unless the parameter of `ascending=False` is also passed into the `sort_values()` method.

Activity: Search for the Worst

In this activity, you will take a dataset on San Francisco Airports' utility consumption and determine which day in the dataset had the worst consumption for each utility.

Suggested Time:

20 minutes

Time's Up! Let's Review.

Questions?