Advanced Data Management (CSCI 640/490)

Data & Pandas

Dr. David Koop
Arrays

• Usually a fixed size—lists are meant to change size
• Are mutable—tuples are not
• Store only one type of data—lists and tuples can store anything
• Are faster to access and manipulate than lists or tuples
• Can be multidimensional:
  - Can have list of lists or tuple of tuples but no guarantee on shape
  - Multidimensional arrays are rectangles, cubes, etc.
Why NumPy?

- Fast **vectorized** array operations for data munging and cleaning, subsetting and filtering, transformation, and any other kinds of computations
- Common array algorithms like sorting, unique, and set operations
- Efficient descriptive statistics and aggregating/summarizing data
- Data alignment and relational data manipulations for merging and joining together heterogeneous data sets
- Expressing conditional logic as array expressions instead of loops with **if-elif-else** branches
- Group-wise data manipulations (aggregation, transformation, function application).

[W. McKinney, Python for Data Analysis]
Speed Benefits

• Compare random number generation in pure Python versus numpy

• Python:

  - import random
    - %timeit rolls_list = [random.randrange(1,7) for i in range(0, 60_000)]

• With NumPy:

  - %timeit rolls_array = np.random.randint(1, 7, 60_000)

• Significant speedup (80x+)
Assignment 2

- Assignment 1 Questions with pandas, DuckDB, and polars
- CS 640 students do all, CS 490 do pandas & DuckDB (polars is EC)
- Can work by framework or by query
- Most questions can be answered with a single statement… but that statement can take a while to write
  - Read documentation
  - Check hints
Array Shape

- Our normal way of checking the size of a collection is \texttt{len}.

- How does this work for arrays?
  - \texttt{arr1 = np.array([1,2,3,6,9])}
    - \texttt{len(arr1)} # 5
  - \texttt{arr2 = np.array([[1.5,2,3,4],[5,6,7,8]])}
    - \texttt{len(arr2)} # 2

- All dimension lengths $\rightarrow$ shape: \texttt{arr2.shape} # (2,4)

- Number of dimensions: \texttt{arr2.ndim} # 2

- Can also reshape an array:
  - \texttt{arr2.reshape(4,2)}
  - \texttt{arr2.reshape(-1,2)} # what happens here?
Array Programming

• Lists:
  - c = []
    for i in range(len(a)):
      c.append(a[i] + b[i])

• How to improve this?
Array Programming

• Lists:
  - `c = []`
    ```python
    for i in range(len(a)):
        c.append(a[i] + b[i])
    ```
  - `c = [aa + bb for aa, bb in zip(a, b)]`

• NumPy arrays:
  - `c = a + b`

• More functional-style than imperative

• Internal iteration instead of external
Operations

- \( a = \text{np.array}([1, 2, 3]) \)
  \( b = \text{np.array}([6, 4, 3]) \)

- (Array, Array) Operations (Element-wise)
  - Addition, Subtraction, Multiplication
  - \( a + b \) # array([7, 6, 6])

- (Scalar, Array) Operations (Broadcasting):
  - Addition, Subtraction, Multiplication, Division, Exponentiation
  - \( a ** 2 \) # array([1, 4, 9])
  - \( b + 3 \) # array([9, 7, 6])
More on Array Creation

- **Zeros**: `np.zeros(10)`
- **Ones**: `np.ones((4,5))` # shape
- **Empty**: `np.empty((2,2))`
- _like versions: pass an existing array and matches shape with specified contents
- **Range**: `np.arange(15)` # constructs an array, not iterator!
Indexing

• Same as with lists plus shorthand for 2D+
  - arr1 = np.array([6, 7, 8, 0, 1])
  - arr1[1]
  - arr1[-1]

• What about two dimensions?
  - arr2 = np.array([[1.5, 2, 3, 4], [5, 6, 7, 8]])
  - arr[1][1]
  - arr[1,1] # shorthand
In multidimensional arrays, if you omit later indices, the returned object will be a lower dimensional ndarray consisting of all the data along the higher dimensions. So in the $2 \times 2 \times 3$ array $\text{arr3d}$:

In

$$\text{arr3d} = \text{np.array}([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])$$

Out

$$\text{arr3d}[0]$$

is a $2 \times 3$ array:

In

$$\text{arr3d}[0] = 42$$

Out

$$\text{arr3d}[0]$$

Both scalar values and arrays can be assigned to $\text{arr3d}[0]$:
Slicing

• 1D: Similar to lists
  - `arr1 = np.array([6, 7, 8, 0, 1])`
  - `arr1[2:5]` # np.array([8,0,1]), sort of

• Can **mutate** original array:
  - `arr1[2:5] = 3` # supports assignment
  - `arr1` # the original array changed

• Slicing returns **views** (copy the array if original array shouldn't change)
  - `arr1[2:5]` # a view
  - `arr1[2:5].copy()` # a new array
Slicing

• 2D+: comma separated indices as shorthand:
  - arr2 = np.array([[1.5,2,3,4],[5,6,7,8]])
  - a[1:3,1:3]
  - a[1:3,:] # works like in single-dimensional lists

• Can combine index and slice in different dimensions
  - a[1,:] # gives a row
  - a[:,1] # gives a column
2D Array Slicing

Suppose each name corresponds to a row in the data array and we wanted to select all the rows with corresponding name 'Bob'. Like arithmetic operations, comparisons (such as $==$) with arrays are also vectorized. Thus, comparing names with the string 'Bob' yields a boolean array:

```
In [87]: names == 'Bob'
Out[87]: array([ True, False, False, True, False, False, False], dtype=bool)
```

This boolean array can be passed when indexing the array:

```
In [88]: data[names == 'Bob']
Out[88]:
array([[-0.048 ,  0.5433, -0.2349,  1.2792],
       [ 2.1452,  0.8799, -0.0523,  0.0672]])
```

The boolean array must be of the same length as the axis it's indexing. You can even mix and match boolean arrays with slices or integers (or sequences of integers, more on this later):

```
In [89]: data[names == 'Bob', 2:]
Out[89]:
array([[-0.2349,  1.2792]])
```

How to obtain the blue slice from array $\text{arr}$?
Suppose each name corresponds to a row in the data array and we wanted to select all the rows with corresponding name 'Bob'. Like arithmetic operations, comparisons (such as \(==\)) with arrays are also vectorized. Thus, comparing names with the string 'Bob' yields a boolean array:

\[
\text{In [87]: names == 'Bob'}
\]
\[
\text{Out[87]: array([True, False, False, True, False, False, False], dtype=bool)}
\]

This boolean array can be passed when indexing the array:

\[
\text{In [88]: data[names == 'Bob']}
\]
\[
\text{Out[88]: array([[-0.048, 0.5433, -0.2349, 1.2792],
                   [2.1452, 0.8799, -0.0523, 0.0672]])}
\]

The boolean array must be of the same length as the axis it's indexing. You can even mix and match boolean arrays with slices or integers (or sequences of integers, more on this later):

\[
\text{In [89]: data[names == 'Bob', 2:]}\]
\[
\text{Out[89]: array([[-0.2349, 1.2792],
                   [2.1452, 0.8799]])}
\]

How to obtain the blue slice from array arr?
### 2D Array Slicing

How to obtain the blue slice from array `arr`?

<table>
<thead>
<tr>
<th>Expression</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arr[2:1:]</code></td>
<td><code>(2, 2)</code></td>
</tr>
<tr>
<td><code>arr[2]</code></td>
<td><code>(3,)</code></td>
</tr>
<tr>
<td><code>arr[2, :]</code></td>
<td><code>(3,)</code></td>
</tr>
<tr>
<td><code>arr[2:, :]</code></td>
<td><code>(1, 3)</code></td>
</tr>
</tbody>
</table>

[W. McKinney, Python for Data Analysis]
2D Array Slicing

<table>
<thead>
<tr>
<th>Expression</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>arr[:2, 1:]</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>arr[2]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2, :]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2:, :]</td>
<td>(1, 3)</td>
</tr>
<tr>
<td>arr[:, 2]</td>
<td>(3, 2)</td>
</tr>
</tbody>
</table>

How to obtain the blue slice from array `arr`?

[W. McKinney, Python for Data Analysis]
Suppose each name corresponds to a row in the data array and we wanted to select all the rows with corresponding name 'Bob'. Like arithmetic operations, comparisons (such as ==) with arrays are also vectorized. Thus, comparing names with the string 'Bob' yields a boolean array:

```
In [87]: names == 'Bob'
Out[87]: array([ True, False, False, True, False, False, False], dtype=bool)
```

This boolean array can be passed when indexing the array:

```
In [88]: data[names == 'Bob']
Out[88]:
array([[-0.048 ,  0.5433, -0.2349,  1.2792],
       [ 2.1452,  0.8799, -0.0523,  0.0672]])
```

The boolean array must be of the same length as the axis it's indexing. You can even mix and match boolean arrays with slices or integers (or sequences of integers, more on this later):

```
In [89]: data[names == 'Bob', 2:]
Out[89]:
array([[-0.2349,  1.2792],
       [ 2.1452,  0.8799]])
```

How to obtain the blue slice from array `arr`? 

---

[W. McKinney, Python for Data Analysis]
Reshaping

• reshape:
  - arr2.reshape(4,2) # returns new view

• resize:
  - arr2.resize(4,2) # no return, modifies arr2 in place

• flatten:
  - arr2.flatten() # array([1.5, 2., 3., 4., 5., 6., 7., 8.])

• ravel:
  - arr2.ravel() # array([1.5, 2., 3., 4., 5., 6., 7., 8.])

• flatten and ravel look the same, but ravel is a view
Boolean Indexing

- `names == 'Bob'` gives back booleans that represent the element-wise comparison with the array `names`.
- Boolean arrays can be used to index into another array:
  - `data[names == 'Bob']`
- Can even mix and match with integer slicing.
- Can do boolean operations (`&`, `|`) between arrays (just like addition, subtraction):
  - `data[(names == 'Bob') | (names == 'Will')]`
- Note: `or` and `and` do not work with arrays.
- We can set values too! `data[data < 0] = 0`
Array Transformations

• Transpose
  - arr2.T # flip rows and columns

• Stacking: take iterable of arrays and stack them horizontally/vertically
  - arrh1 = np.arange(3)
  - arrh2 = np.arange(3,6)
  - np.vstack([arrh1, arrh2])
  - np.hstack([arr1.T, arr2.T]) # ???
numpy Functions

• Unary: abs, sqrt, log, ceil, sin, cos, tan, arccos, arcsin, ...
• Binary: add, subtract, multiple, divide, ... <, >, >=, <=, ==, !=
• Statistics: sum, mean, std, min, max, argmin, argmax
• Boolean: any, all
• Others: sort, unique
• Linear Algebra (numpy.linalg)
• Pseudorandom Number Generation (numpy.random)
pandas

• Contains high-level data structures and manipulation tools designed to make data analysis fast and easy in Python
• Built on top of NumPy
• Requirements:
  - Data structures with labeled axes (aligning data)
  - Time series data
  - Arithmetic operations that include metadata (labels)
  - Handle missing data
  - Merge and relational operations
Pandas Code Conventions

• Universal:
  - import pandas as pd

• Also used:
  - from pandas import Series, DataFrame
Series

- A one-dimensional array (with a type) with an index
- Index defaults to numbers but can also be text (like a dictionary)
- Allows easier reference to specific items
- \( \text{obj} = \text{pd.Series}([7,14,-2,1]) \)
- Basically two arrays: `obj.values` and `obj.index`
- Can specify the index explicitly and use strings
- \( \text{obj2} = \text{pd.Series}([4, 7, -5, 3], \)
  \hspace{1cm} \text{index}=['d', 'b', 'a', 'c']) \)
- Kind of like fixed-length, ordered dictionary + can create from a dictionary
- \( \text{obj3} = \text{pd.Series}({'Ohio': 35000, 'Texas': 71000,} \)
  \hspace{1cm} {'Oregon': 16000, 'Utah': 5000}) \)
Series

- **Indexing:** `s[1]` or `s['Oregon']`
- **Can check for missing data:** `pd.isnull(s)` or `pd.notnull(s)`
- **Both index and values can have an associated name:**
  - `s.name = 'population'; s.index.name = 'state'`
- **Addition and NumPy ops work as expected and preserve the index-value link**
- **These operations align:**

<table>
<thead>
<tr>
<th>In [28]: obj3</th>
<th>In [29]: obj4</th>
<th>In [30]: obj3 + obj4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out[28]:</td>
<td>Out[29]:</td>
<td>Out[30]:</td>
</tr>
<tr>
<td>Ohio</td>
<td>California</td>
<td>California</td>
</tr>
<tr>
<td>35000</td>
<td>35000</td>
<td>70000</td>
</tr>
<tr>
<td>Oregon</td>
<td>Ohio</td>
<td>Ohio</td>
</tr>
<tr>
<td>16000</td>
<td>35000</td>
<td>70000</td>
</tr>
<tr>
<td>Texas</td>
<td>Oregon</td>
<td>Oregon</td>
</tr>
<tr>
<td>71000</td>
<td>16000</td>
<td>32000</td>
</tr>
<tr>
<td>Utah</td>
<td>Texas</td>
<td>Texas</td>
</tr>
<tr>
<td>5000</td>
<td>71000</td>
<td>142000</td>
</tr>
<tr>
<td>dtype: int64</td>
<td>dtype: float64</td>
<td>dtype: float64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[W. McKinney, Python for Data Analysis]
Data Frame

• A dictionary of Series (labels for each series)
• A spreadsheet with column headers
• Has an index shared with each series
• Allows easy reference to any cell
• `df = DataFrame({'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada'],
                 'pop': [1.5, 1.7, 3.6, 2.4]})`

• Index is automatically assigned just as with a series but can be passed in as well via index kwarg
• Can reassign column names by passing columns kwarg
df = pd.read_csv('penguins_lter.csv')

<table>
<thead>
<tr>
<th>studyName</th>
<th>Sample Number</th>
<th>Species</th>
<th>Region</th>
<th>Island</th>
<th>Stage</th>
<th>Individual ID</th>
<th>Clutch Completion</th>
<th>Date Egg</th>
<th>Date Egg</th>
<th>Culmen Length (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PAL0708</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A1</td>
<td>Yes</td>
<td>11/11/07</td>
<td>11/11/07</td>
<td>39.1</td>
</tr>
<tr>
<td>1</td>
<td>PAL0708</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A2</td>
<td>Yes</td>
<td>11/11/07</td>
<td>11/11/07</td>
<td>39.5</td>
</tr>
<tr>
<td>2</td>
<td>PAL0708</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>11/16/07</td>
<td>40.3</td>
</tr>
<tr>
<td>3</td>
<td>PAL0708</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A2</td>
<td>Yes</td>
<td>11/16/07</td>
<td>11/16/07</td>
<td>NaN</td>
</tr>
<tr>
<td>4</td>
<td>PAL0708</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N3A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>11/16/07</td>
<td>36.7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>339</td>
<td>PAL0910</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N38A2</td>
<td>No</td>
<td>12/1/09</td>
<td>12/1/09</td>
<td>NaN</td>
</tr>
<tr>
<td>340</td>
<td>PAL0910</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>11/22/09</td>
<td>46.8</td>
</tr>
<tr>
<td>341</td>
<td>PAL0910</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>11/22/09</td>
<td>50.4</td>
</tr>
<tr>
<td>342</td>
<td>PAL0910</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>11/22/09</td>
<td>45.2</td>
</tr>
<tr>
<td>343</td>
<td>PAL0910</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>11/22/09</td>
<td>49.9</td>
</tr>
</tbody>
</table>

344 rows x 17 columns
## Data Frame

```python
df = pd.read_csv('penguins_lter.csv')
```

### Column Names

<table>
<thead>
<tr>
<th>studyName</th>
<th>Sample Number</th>
<th>Species</th>
<th>Region</th>
<th>Island</th>
<th>Stage</th>
<th>Individual ID</th>
<th>Clutch Completion</th>
<th>Date Egg</th>
<th>Culmen Length (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A1</td>
<td>Yes</td>
<td>11/11/07</td>
<td>39.1</td>
</tr>
<tr>
<td>PAL0708</td>
<td>2</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A2</td>
<td>Yes</td>
<td>11/11/07</td>
<td>39.5</td>
</tr>
<tr>
<td>PAL0708</td>
<td>3</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>40.3</td>
</tr>
<tr>
<td>PAL0708</td>
<td>4</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A2</td>
<td>Yes</td>
<td>11/16/07</td>
<td>NaN</td>
</tr>
<tr>
<td>PAL0708</td>
<td>5</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N3A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>36.7</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>PAL0910</td>
<td>120</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N38A2</td>
<td>No</td>
<td>12/1/09</td>
<td>NaN</td>
</tr>
<tr>
<td>PAL0910</td>
<td>121</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>46.8</td>
</tr>
<tr>
<td>PAL0910</td>
<td>122</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>50.4</td>
</tr>
<tr>
<td>PAL0910</td>
<td>123</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>45.2</td>
</tr>
<tr>
<td>PAL0910</td>
<td>124</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>49.9</td>
</tr>
</tbody>
</table>

344 rows x 17 columns
A data frame is shown with columns for **studyName**, **Sample Number**, **Species**, **Region**, **Island**, **Stage**, **Individual ID**, **Clutch Completion**, **Date Egg**, and **Culmen Length (mm)**.

The data frame contains information on penguin species, specifically Adelie and Gentoo penguins, with details such as location, stage of development, and measurements.

For example, the first record shows a studyName of PAL0708 with a Sample Number of 1, species Adelie Penguin (Pygoscelis adeliae), region Anvers, island Torgersen, stage Adult, 1 Egg Stage, individual ID N1A1, clutch completion Yes, date egg 11/11/07, and culmen length 39.1 mm.

The data frame includes 344 rows and 17 columns.
## Data Frame

```
df = pd.read_csv('penguins_lter.csv')
```

### Column Names

<table>
<thead>
<tr>
<th></th>
<th>studyName</th>
<th>Sample Number</th>
<th>Species</th>
<th>Region</th>
<th>Island</th>
<th>Stage</th>
<th>Individual ID</th>
<th>Clutch Completion</th>
<th>Date Egg</th>
<th>Culmen Length (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PAL0708</td>
<td>1</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A1</td>
<td>Yes</td>
<td>11/11/07</td>
<td>39.1</td>
</tr>
<tr>
<td>1</td>
<td>PAL0708</td>
<td>2</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A2</td>
<td>Yes</td>
<td>11/11/07</td>
<td>39.5</td>
</tr>
<tr>
<td>2</td>
<td>PAL0708</td>
<td>3</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>40.3</td>
</tr>
<tr>
<td>3</td>
<td>PAL0708</td>
<td>4</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A2</td>
<td>Yes</td>
<td>11/16/07</td>
<td>NaN</td>
</tr>
<tr>
<td>4</td>
<td>PAL0708</td>
<td>5</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N3A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>36.7</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>339</td>
<td>PAL0910</td>
<td>120</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N38A2</td>
<td>No</td>
<td>12/1/09</td>
<td>NaN</td>
</tr>
<tr>
<td>340</td>
<td>PAL0910</td>
<td>121</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>46.8</td>
</tr>
<tr>
<td>341</td>
<td>PAL0910</td>
<td>122</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>50.4</td>
</tr>
<tr>
<td>342</td>
<td>PAL0910</td>
<td>123</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>45.2</td>
</tr>
<tr>
<td>343</td>
<td>PAL0910</td>
<td>124</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>49.9</td>
</tr>
</tbody>
</table>

344 rows x 17 columns

### Column: `df['Island']`
Data Frame

```python
df = pd.read_csv('penguins_lter.csv')
```

## Column Names

<table>
<thead>
<tr>
<th>studyName</th>
<th>Sample Number</th>
<th>Species</th>
<th>Region</th>
<th>Island</th>
<th>Stage</th>
<th>Individual ID</th>
<th>Clutch Completion</th>
<th>Date Egg</th>
<th>Culmen Length (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A1</td>
<td>Yes</td>
<td>11/11/07</td>
<td>39.1</td>
</tr>
<tr>
<td>PAL0708</td>
<td>2</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A2</td>
<td>Yes</td>
<td>11/11/07</td>
<td>39.5</td>
</tr>
<tr>
<td>PAL0708</td>
<td>3</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>40.3</td>
</tr>
<tr>
<td>PAL0708</td>
<td>4</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A2</td>
<td>Yes</td>
<td>11/16/07</td>
<td>NaN</td>
</tr>
<tr>
<td>PAL0708</td>
<td>5</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N3A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>36.7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>PAL0910</td>
<td>120</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N38A2</td>
<td>No</td>
<td>12/1/09</td>
<td>NaN</td>
</tr>
<tr>
<td>PAL0910</td>
<td>121</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>46.8</td>
</tr>
<tr>
<td>PAL0910</td>
<td>122</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>50.4</td>
</tr>
<tr>
<td>PAL0910</td>
<td>123</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>45.2</td>
</tr>
<tr>
<td>PAL0910</td>
<td>124</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>49.9</td>
</tr>
</tbody>
</table>

344 rows x 17 columns

## Row: `df.loc[2]`

## Index

Column: `df['Island']`
## Data Frame

```python
import pandas as pd
df = pd.read_csv('penguins_lter.csv')
```

### Column Names

<table>
<thead>
<tr>
<th>StudyName</th>
<th>Sample Number</th>
<th>Species</th>
<th>Region</th>
<th>Island</th>
<th>Stage</th>
<th>Individual ID</th>
<th>Clutch Completion</th>
<th>Date Egg</th>
<th>Culmen Length (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A1</td>
<td>Yes</td>
<td>11/11/07</td>
<td>39.1</td>
</tr>
<tr>
<td>PAL0708</td>
<td>2</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N1A2</td>
<td>Yes</td>
<td>11/11/07</td>
<td>39.5</td>
</tr>
<tr>
<td>PAL0708</td>
<td>3</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>40.3</td>
</tr>
<tr>
<td>PAL0708</td>
<td>4</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N2A2</td>
<td>Yes</td>
<td>11/16/07</td>
<td>NaN</td>
</tr>
<tr>
<td>PAL0708</td>
<td>5</td>
<td>Adelie Penguin (Pygoscelis adeliae)</td>
<td>Anvers</td>
<td>Torgersen</td>
<td>Adult, 1 Egg Stage</td>
<td>N3A1</td>
<td>Yes</td>
<td>11/16/07</td>
<td>36.7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>PAL0910</td>
<td>120</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N38A2</td>
<td>No</td>
<td>12/1/09</td>
<td>NaN</td>
</tr>
<tr>
<td>PAL0910</td>
<td>121</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>46.8</td>
</tr>
<tr>
<td>PAL0910</td>
<td>122</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N39A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>50.4</td>
</tr>
<tr>
<td>PAL0910</td>
<td>123</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A1</td>
<td>Yes</td>
<td>11/22/09</td>
<td>45.2</td>
</tr>
<tr>
<td>PAL0910</td>
<td>124</td>
<td>Gentoo penguin (Pygoscelis papua)</td>
<td>Anvers</td>
<td>Biscoe</td>
<td>Adult, 1 Egg Stage</td>
<td>N43A2</td>
<td>Yes</td>
<td>11/22/09</td>
<td>49.9</td>
</tr>
</tbody>
</table>

344 rows x 17 columns

### Row: `df.loc[2]`

```
0 PAL0708 1 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen Adult, 1 Egg Stage N1A1 Yes 11/11/07 39.1
1 PAL0708 2 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen Adult, 1 Egg Stage N1A2 Yes 11/11/07 39.5
2 PAL0708 3 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen Adult, 1 Egg Stage N2A1 Yes 11/16/07 40.3
3 PAL0708 4 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen Adult, 1 Egg Stage N2A2 Yes 11/16/07 NaN
4 PAL0708 5 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen Adult, 1 Egg Stage N3A1 Yes 11/16/07 36.7
... ... ... ... ... ... ... ... ... ...
339 PAL0910 120 Gentoo penguin (Pygoscelis papua) Anvers Biscoe Adult, 1 Egg Stage N38A2 No 12/1/09 NaN
340 PAL0910 121 Gentoo penguin (Pygoscelis papua) Anvers Biscoe Adult, 1 Egg Stage N39A1 Yes 11/22/09 46.8
341 PAL0910 122 Gentoo penguin (Pygoscelis papua) Anvers Biscoe Adult, 1 Egg Stage N39A2 Yes 11/22/09 50.4
342 PAL0910 123 Gentoo penguin (Pygoscelis papua) Anvers Biscoe Adult, 1 Egg Stage N43A1 Yes 11/22/09 45.2
343 PAL0910 124 Gentoo penguin (Pygoscelis papua) Anvers Biscoe Adult, 1 Egg Stage N43A2 Yes 11/22/09 49.9
```

### Cell: `df.loc[341, 'Species']`

```
Gentoo penguin (Pygoscelis papua)
```

### Column: `df['Island']`

<table>
<thead>
<tr>
<th>Island</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torgersen</td>
</tr>
<tr>
<td>Biscoe</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

D. Koop, CSCI 640/490, Spring 2024
Data Frame

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>studyName</strong></td>
<td><strong>Sample Number</strong></td>
</tr>
<tr>
<td>0</td>
<td>PAL0708</td>
</tr>
<tr>
<td>1</td>
<td>PAL0708</td>
</tr>
<tr>
<td>2</td>
<td>PAL0708</td>
</tr>
<tr>
<td>3</td>
<td>PAL0708</td>
</tr>
<tr>
<td>4</td>
<td>PAL0708</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>339</td>
<td>PAL0910</td>
</tr>
<tr>
<td>340</td>
<td>PAL0910</td>
</tr>
<tr>
<td>342</td>
<td>PAL0910</td>
</tr>
<tr>
<td>343</td>
<td>PAL0910</td>
</tr>
</tbody>
</table>

344 rows x 17 columns

Cell: df.loc[341,'Species']

Column: df['Island']

Missing Data

D. Koop, CSCI 640/490, Spring 2024
Chicago Food Inspections Example

- Use pandas to analyze food inspection data
  - (see notebook)