Programming Principles in Python (CSCI 503/490)

Data

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.

NumPy Arrays

- import numpy as np
- Creating:
  - `data1 = [6, 7, 8, 0, 1]`
  - `arr1 = np.array(data1)`
  - `arr1_float = np.array(data1, dtype='float64')`
  - `np.ones((4,2)) # 2d array of ones`
  - `arr1_ones = np.ones_like(arr1) # [1, 1, 1, 1, 1]`
- Type and Shape Information:
  - `arr1.dtype # int64 # type of values stored in array`
  - `arr1.ndim # 1 # number of dimensions`
  - `arr1.shape # (5,) # shape of the array`

Array Operations

- `a = np.array([1,2,3])`
  `b = 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!`

Assignment 7

- Coming Soon…
- Downloading and finding files
- Processing data

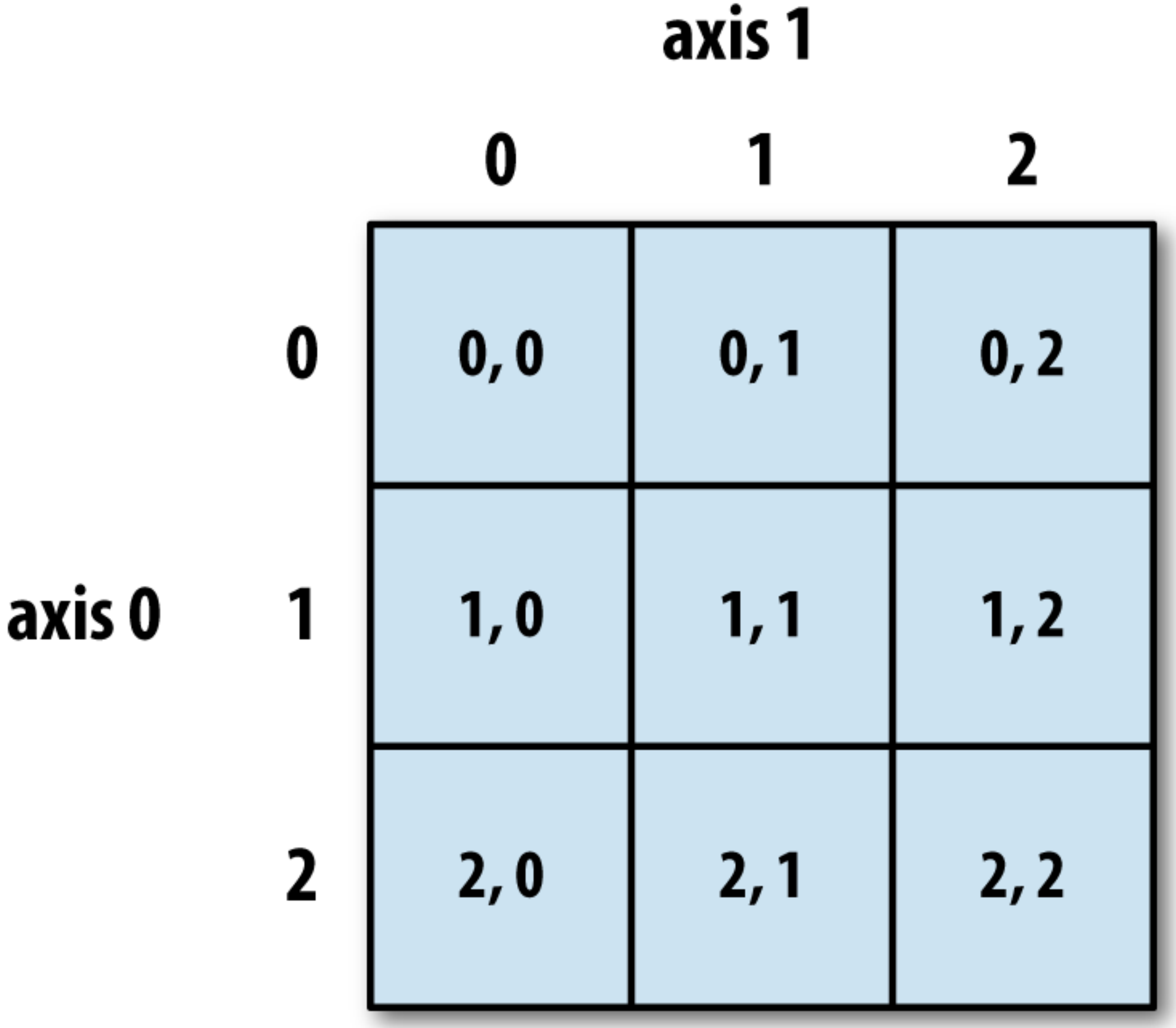
Teaching Evaluations

- This Tuesday (November 15) in class

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`

2D Indexing



[W. McKinney, Python for Data Analysis]

Slicing

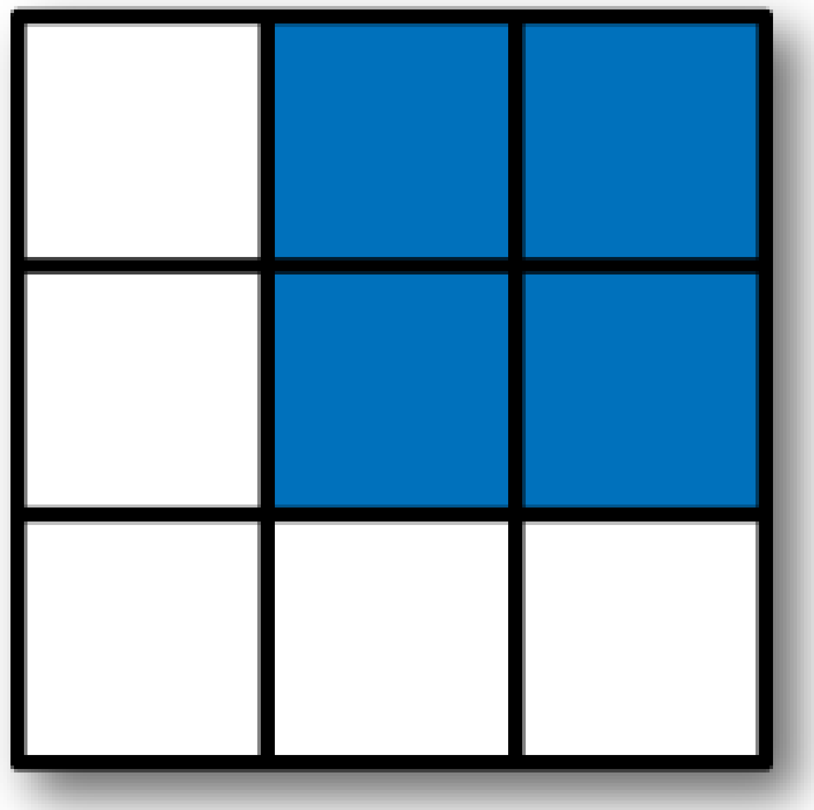
- 1D: Similar to lists
  - `arr1 = np.array([6, 7, 8, 0, 1])`
  - `arr1[2:5] # np.array([8, 0, 1]), sort of`
- Can **mutate** original array:
  - `arr1[2:5] = 3 # supports assignment`
  - `arr1 # the original array changed`
- Slicing returns **views** (copy the array if original array shouldn't change)
  - `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

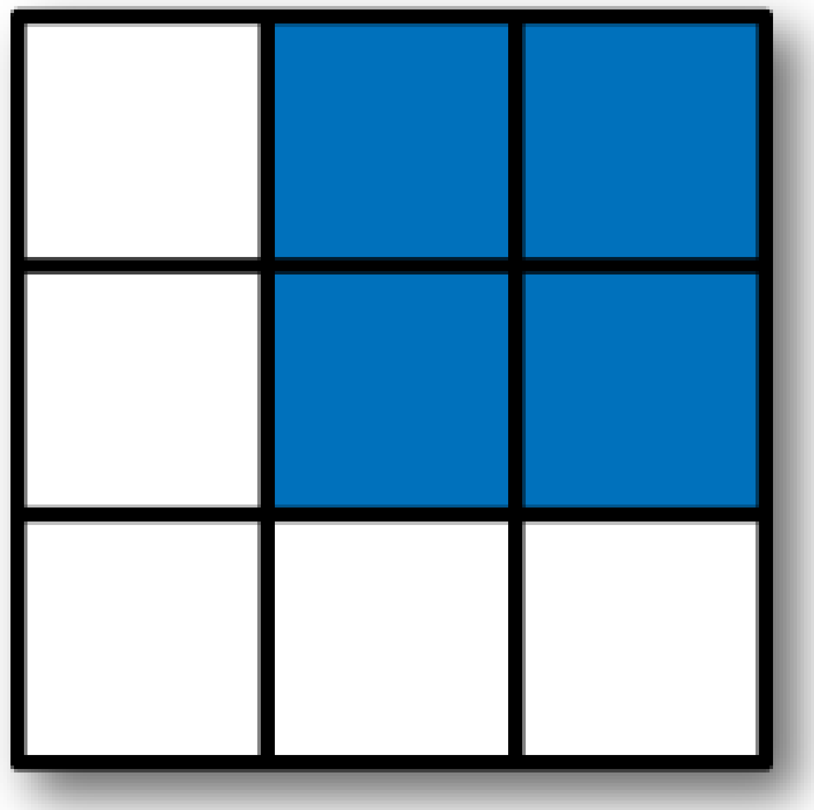
How to obtain the blue slice from array `arr`?



[W. McKinney, Python for Data Analysis]

2D Array Slicing

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

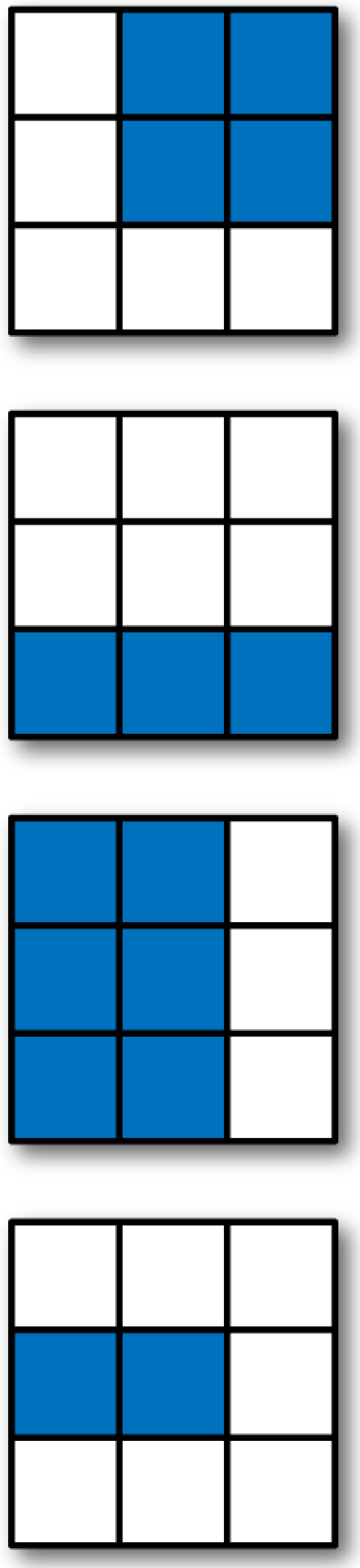


`arr[:2,1:]`

[W. McKinney, Python for Data Analysis]

2D Array Slicing

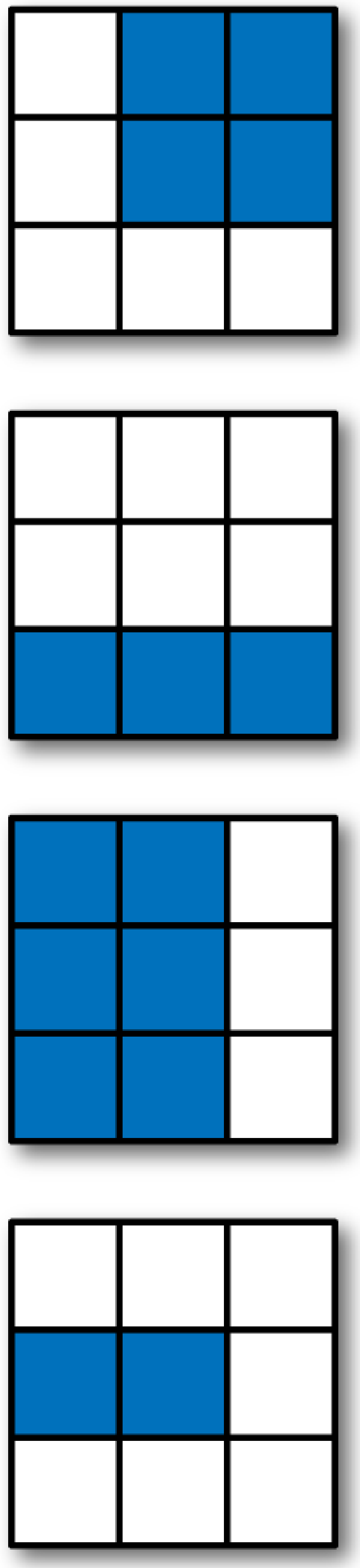
How to obtain the blue slice from array `arr`?



[W. McKinney, Python for Data Analysis]

2D Array Slicing

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



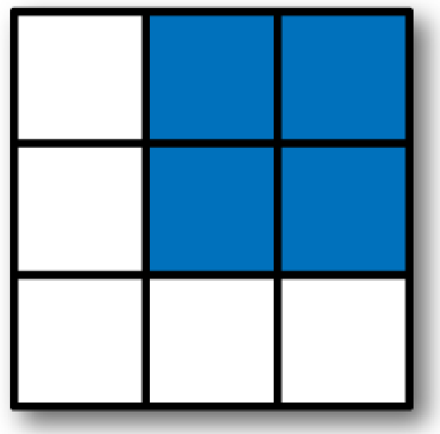
Expression	Shape
`arr[:2, 1:]`	(2, 2)

[W. McKinney, Python for Data Analysis]

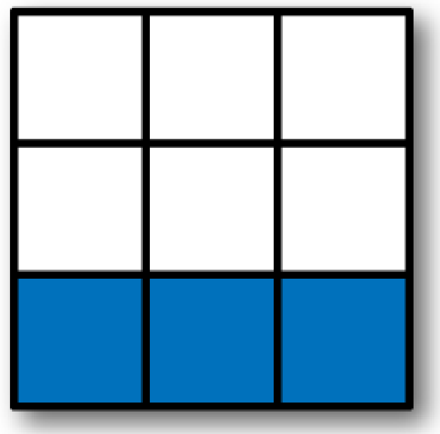
2D Array Slicing

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

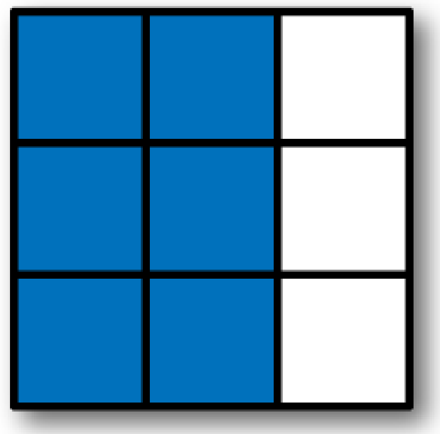
Expression	Shape

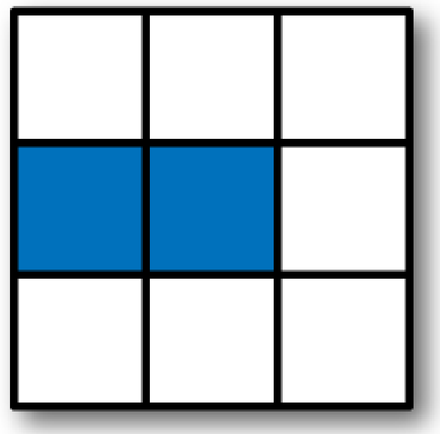


`arr[:2, 1:]`	`(2, 2)`



`arr[2]`	`(3,)`
`arr[2, :]`	`(3,)`
`arr[2:, :]`	`(1, 3)`

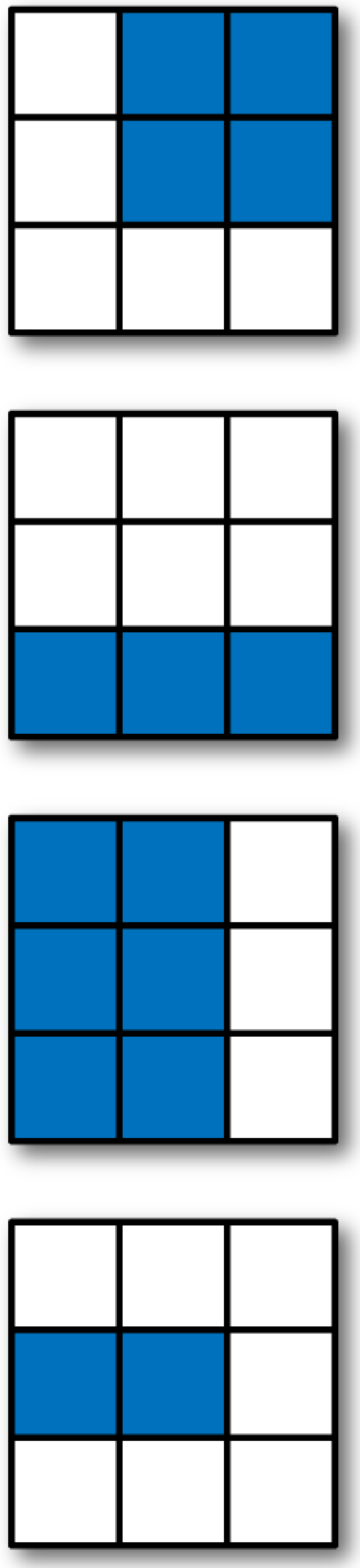




[W. McKinney, Python for Data Analysis]

2D Array Slicing

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



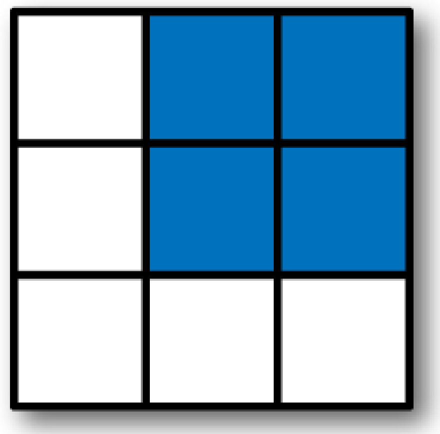
Expression	Shape
`arr[:2, 1:]`	(2, 2)
`arr[2]`	(3,)
`arr[2, :]`	(3,)
`arr[2:, :]`	(1, 3)
`arr[:, :2]`	(3, 2)

[W. McKinney, Python for Data Analysis]

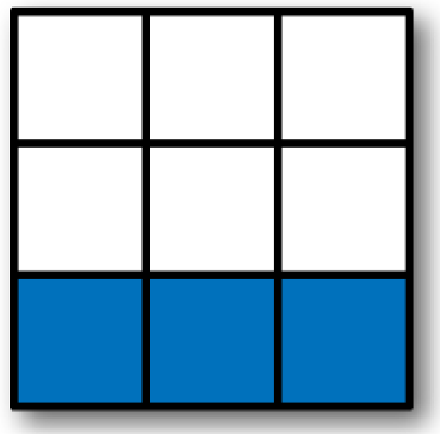
2D Array Slicing

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

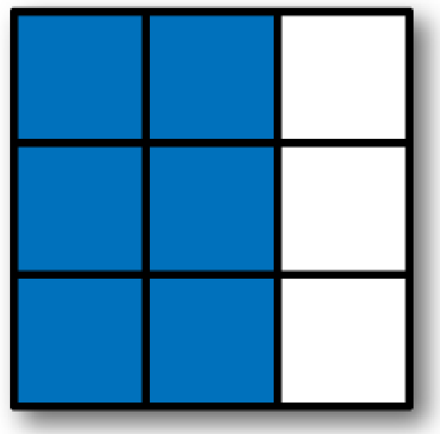
Expression	Shape



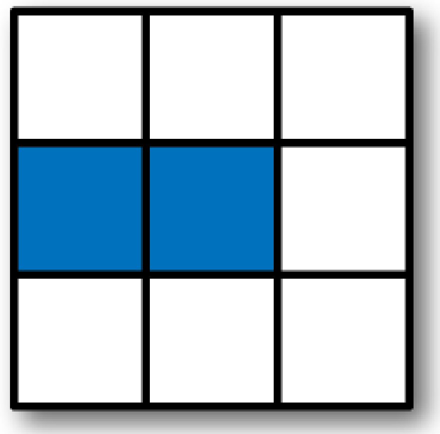
Expression	Shape
`arr[:2, 1:]`	(2, 2)



Expression	Shape
`arr[2]`	(3,)
`arr[2, :]`	(3,)
`arr[2:, :]`	(1, 3)



Expression	Shape
`arr[:, :2]`	(3, 2)



Expression	Shape
`arr[1, :2]`	(2,)
`arr[1:2, :2]`	(1, 2)

[W. McKinney, Python for Data Analysis]

More 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**

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]) # ???`

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`

pandas

- Contains high-level data structures and manipulation tools designed to make data analysis fast and easy in Python
- Built on top of NumPy
- Built with the following requirements:
  - Data structures with labeled axes (aligning data)
  - Support time series data
  - Do arithmetic operations that include metadata (labels)
  - Handle missing data
  - Add 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
- `obj = pd.Series([7,14,-2,1])`
- Basically two arrays: `obj.values` and `obj.index`
- Can specify the index explicitly and use strings
- ```
  obj2 = pd.Series([4, 7, -5, 3],
                   index=['d', 'b', 'a', 'c'])
  ```
- Kind of like fixed-length, ordered dictionary + can create from a dictionary
- ```
  obj3 = pd.Series({'Ohio': 35000, 'Texas': 71000,
                    '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
- Arithmetic operations **align**:

```
In [28]: obj3
Out[28]:
Ohio      35000
Oregon    16000
Texas     71000
Utah       5000
dtype: int64
```

```
In [29]: obj4
Out[29]:
California      NaN
Ohio          35000
Oregon        16000
Texas         71000
dtype: float64
```

```
In [30]: obj3 + obj4
Out[30]:
California       NaN
Ohio           70000
Oregon         32000
Texas         142000
Utah             NaN
dtype: float64
```

[W. McKinney, Python for Data Analysis]

Data Frame

- A dictionary of Series (labels for each series)
- A spreadsheet with row keys (the index) and column headers
- Has an index shared with each series
- Allows easy reference to any cell
- ```
  df = DataFrame({'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada'],
                  'year': [2000, 2001, 2002, 2001],
                  '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
  ```

DataFrame Constructor Inputs

Type	Notes
2D ndarray	A matrix of data, passing optional row and column labels
dict of arrays, lists, or tuples	Each sequence becomes a column in the DataFrame. All sequences must be the same length.
NumPy structured/record array	Treated as the "dict of arrays" case
dict of Series	Each value becomes a column. Indexes from each Series are unioned together to form the result's row index if no explicit index is passed.
dict of dicts	Each inner dict becomes a column. Keys are unioned to form the row index as in the "dict of Series" case.
list of dicts or Series	Each item becomes a row in the DataFrame. Union of dict keys or Series indexes become the DataFrame's column labels
List of lists or tuples	Treated as the "2D ndarray" case
Another DataFrame	The DataFrame's indexes are used unless different ones are passed
NumPy MaskedArray	Like the "2D ndarray" case except masked values become NA/missing in the DataFrame result

[W. McKinney, Python for Data Analysis]

DataFrame Access and Manipulation

- df.values → 2D NumPy array
- Accessing a column:
  - df["<column>"]
  - df.<column>
  - Both return Series
  - Dot syntax only works when the column is a valid identifier
- Assigning to a column:

```
- df["<column>"] = <scalar> # all cells set to same value
- df["<column>"] = <array>  # values set in order
- df["<column>"] = <series> # values set according to match
                            # between df and series indexes
```

Data Frame

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

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
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

344 rows × 17 columns

Data Frame

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

Column Names

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
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

344 rows × 17 columns

Data Frame

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

Column Names

Index

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
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

344 rows × 17 columns

Data Frame

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

Column Names

Index

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
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

344 rows × 17 columns

Column: `df['Island']`

Data Frame

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

Column Names

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
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

Row: `df.loc[2]`

Index

344 rows × 17 columns

Column: `df['Island']`

Data Frame

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

Column Names

Row: `df.loc[2]`

Index

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

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
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
			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

344 rows × 17 columns

Column: `df['Island']`

Data Frame

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

Column Names

Row: `df.loc[2]`

Index

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

Missing Data

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
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	
...	...	...	...	...	...	...	...	...	...	...
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
			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

344 rows × 17 columns

Column: `df['Island']`