Python Packages

Roadmap

1. **Array Data Structure**
2. Data Visualization
3. Pandas
4. Descriptive Analytics in Python

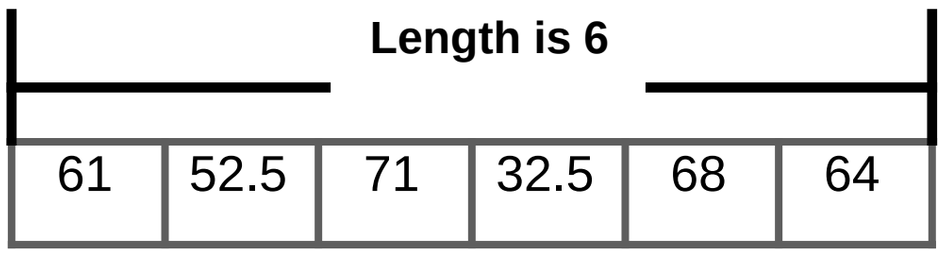
Array Data Structure

“Fighting with a large army under your command is nowise different from fighting with a small one: it is merely a question of instituting signs and signals.”

–Sun Tzu on the Art of War

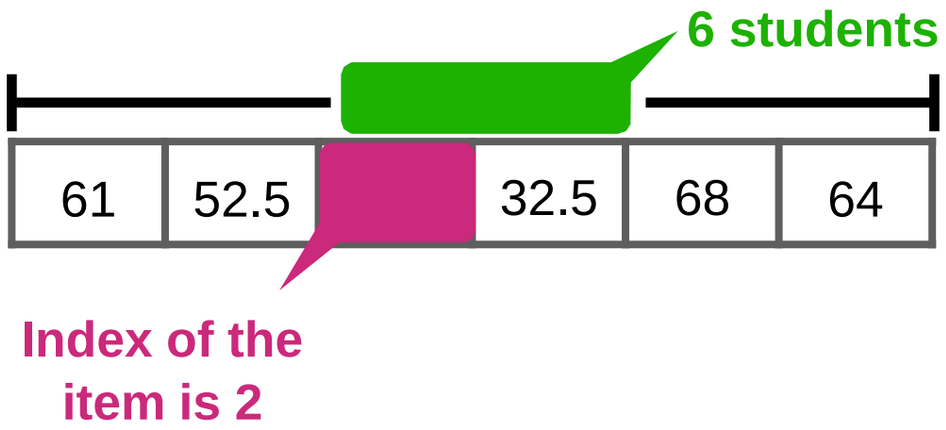
Array Data Structure

- Concepts
  - ▸ One-dimensional arrays
    - ✓ Final exam grades of six students



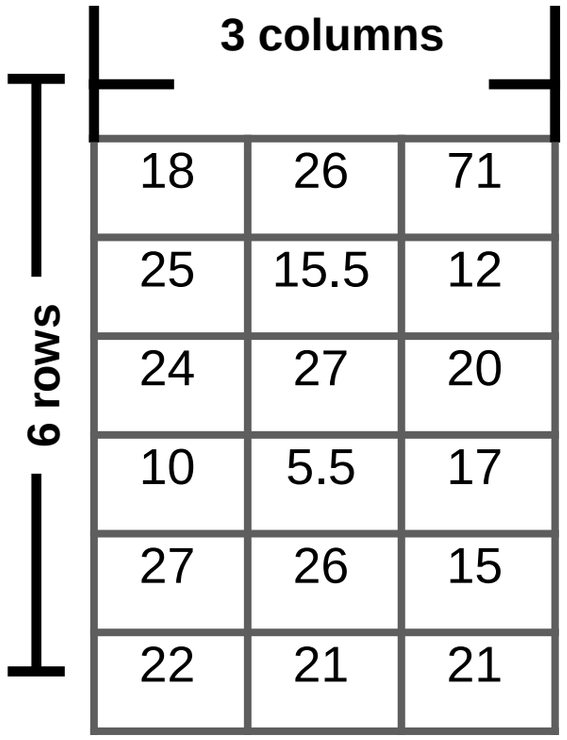
Array Data Structure

- Concepts
  - One-dimensional arrays
    - ✓ Final exam grades of six students



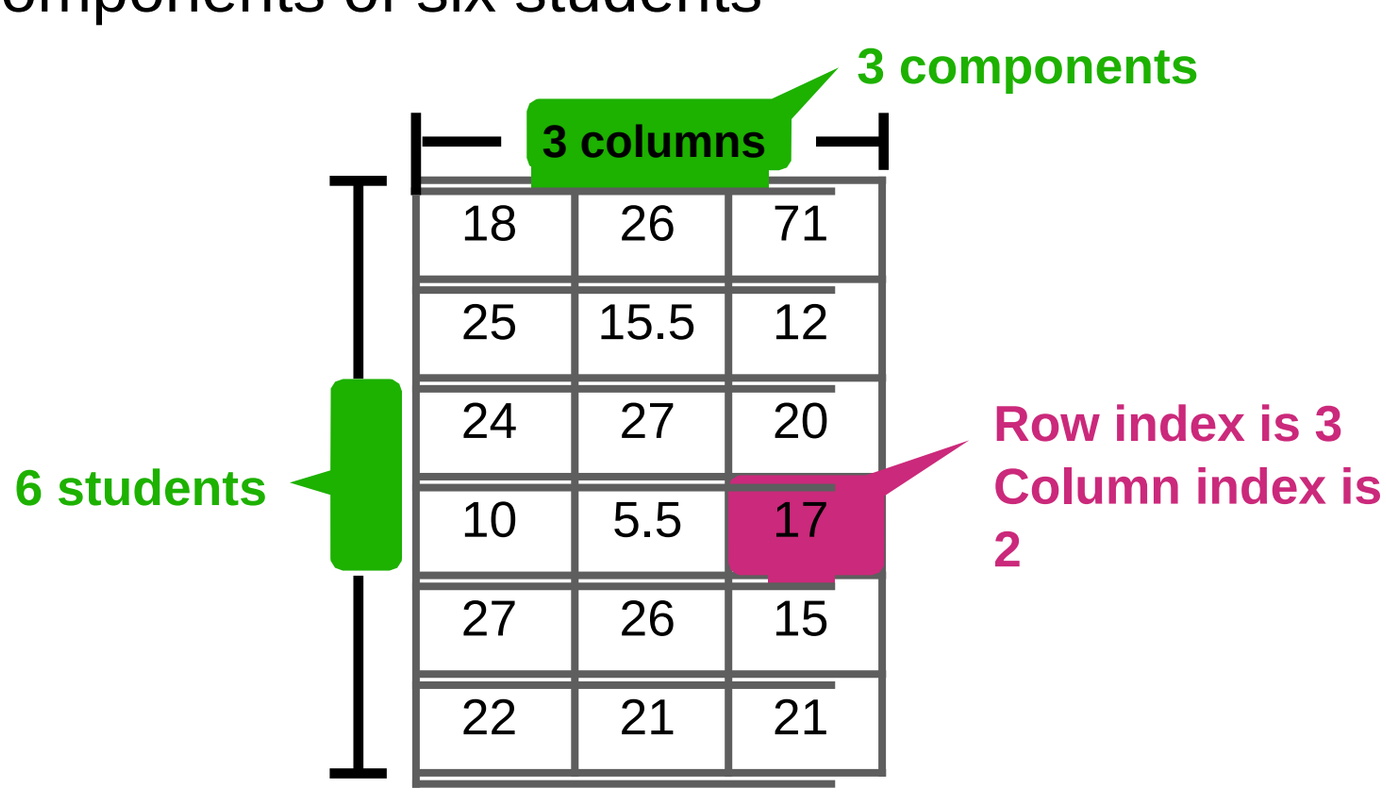
Array Data Structure

- Concepts
  - ▸ Two-dimensional arrays
    - ✓ Grades of three components of six students



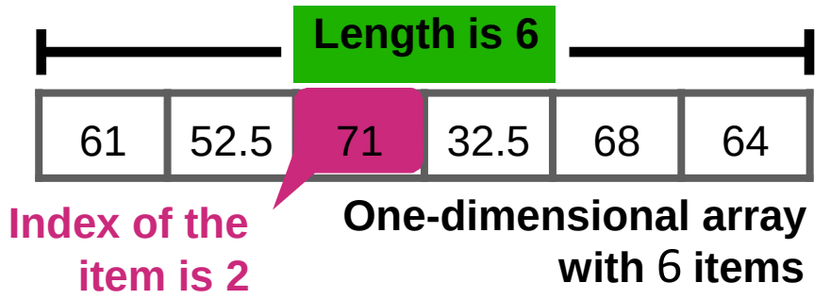
Array Data Structure

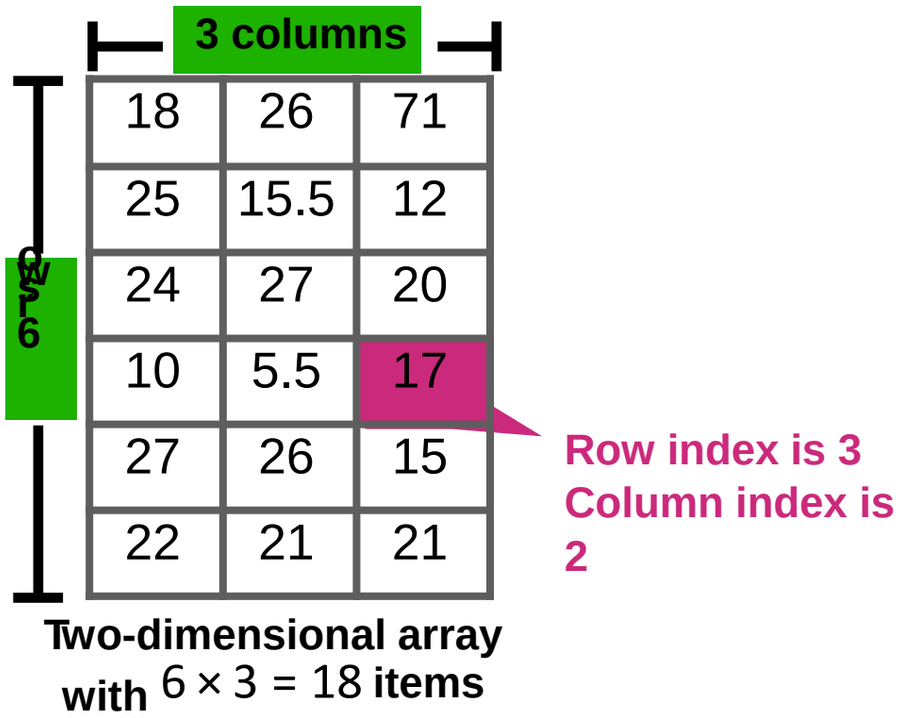
- Concepts
  - ► Two-dimensional arrays
    - ✓ Grades of three components of six students



Array Data Structure

- Concepts
  - *N*-dimensional arrays



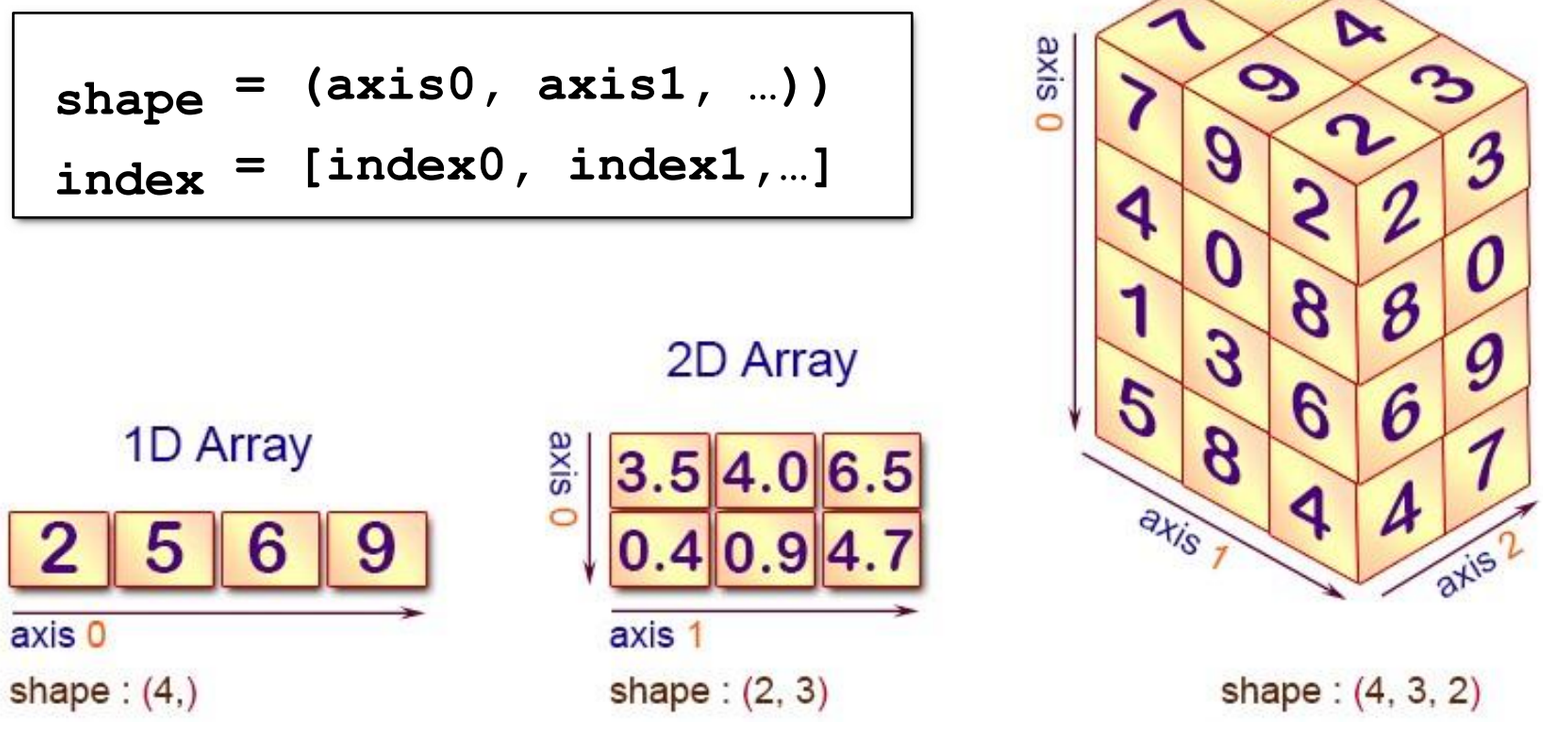


Dimension number	Shape	Index	Item number
1	One number	One index	Shape
2	Two numbers	Two indexes	Shape[0] * Shape[1]
N	*N* numbers	*N* indexes	Shape[0] *…* Shape[N]

Array Data Structure

- Concepts
  - *N*-dimensional arrays

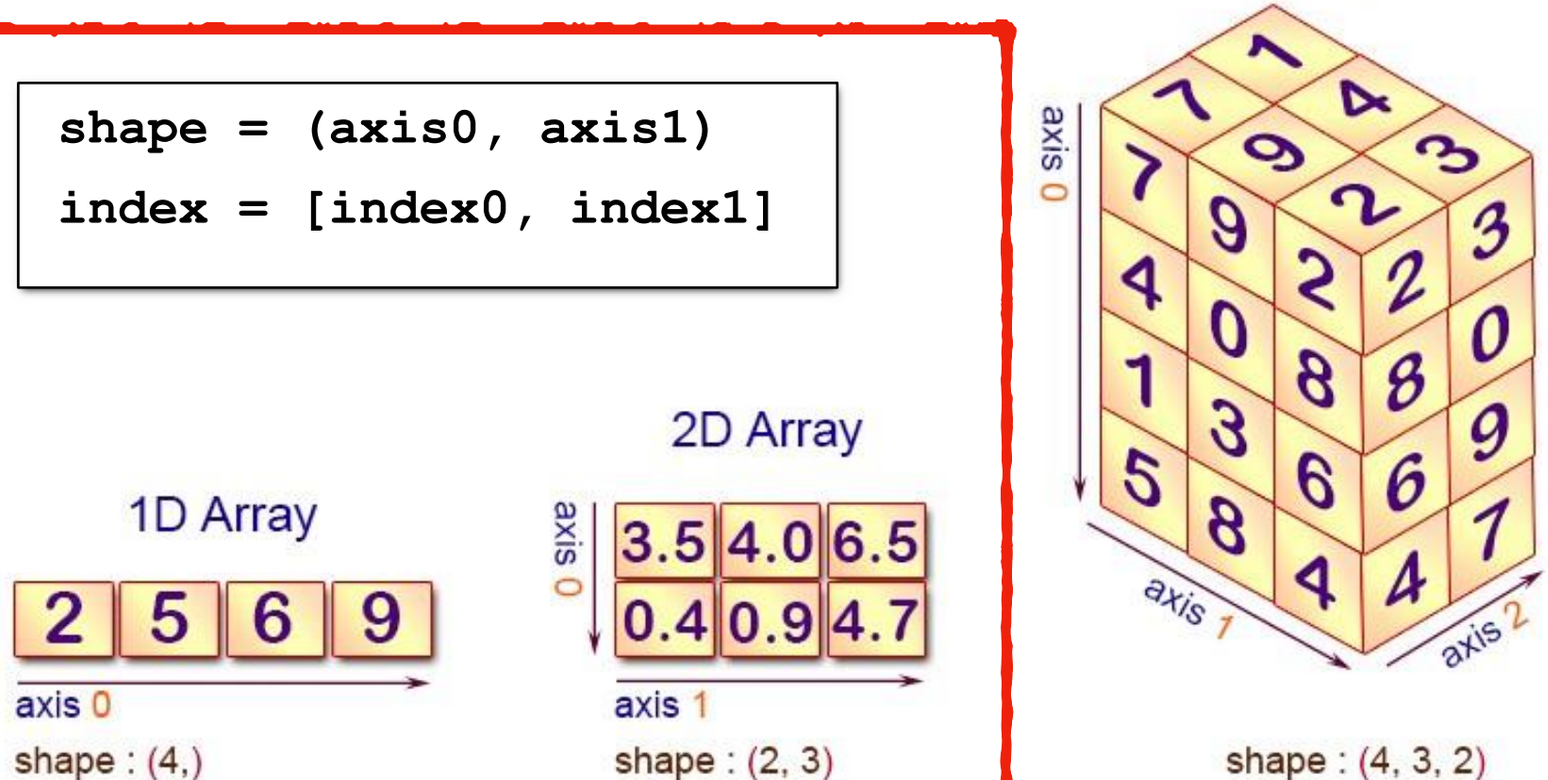
```
shape = (axis0, axis1, …))
index = [index0, index1,…]
```



Array Data Structure

- Concepts
  - *N*-dimensional arrays

```
shape = (axis0, axis1)
index = [index0, index1]
```



Array Data Structure

- NumPy arrays
  - Import the NumPy package

```
import numpy as ■

■.<function name>
```

Array Data Structure

- NumPy arrays
  - Create arrays from Python lists
    - ✓ Function [redacted]

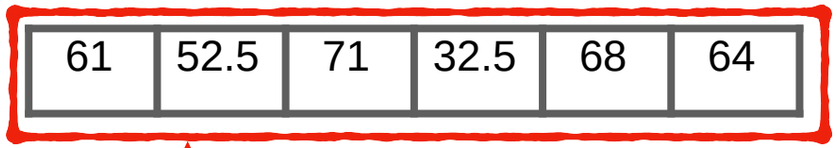
```
array_1d = np.array([61, 52.5, 71, 32.5, 68, 64])   # This is a 1D array
array_2d = np.array([[18, 26, 17],
                     [25, 15.5, 12],
                     [24, 27, 20],
                     [10, 5.5, 17],
                     [27, 26, 15],
                     [22, 21, 21]])                 # This is a 2D array
```

```
print(type(array_1d))
print(type(array_2d))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

Array Data Structure

- NumPy arrays
  - Create arrays from Python lists
    - ✓ Function **`np.array()`**




```
array_1d = np.array(                                    )   # This is a 1D array
array_2d = np.array([[18, 26, 17],
                     [25, 15.5, 12],
                     [24, 27, 20],
                     [10, 5.5, 17],
                     [27, 26, 15],
                     [22, 21, 21]])                         # This is a 2D array
```

Array Data Structure

- NumPy arrays
  - Create arrays from Python lists
    - ✓ Function [redacted]

```
1  array_1d = np.array([61, 52.5, 71, 32.5, 68, 64])   # This is a 1D array
2  array_2d = np.array([redacted]
3
4
5
6
7                                                      is a 2D array
```

18	26	71
25	15.5	12
24	27	20
10	5.5	17
27	26	15
22	21	21

Array Data Structure

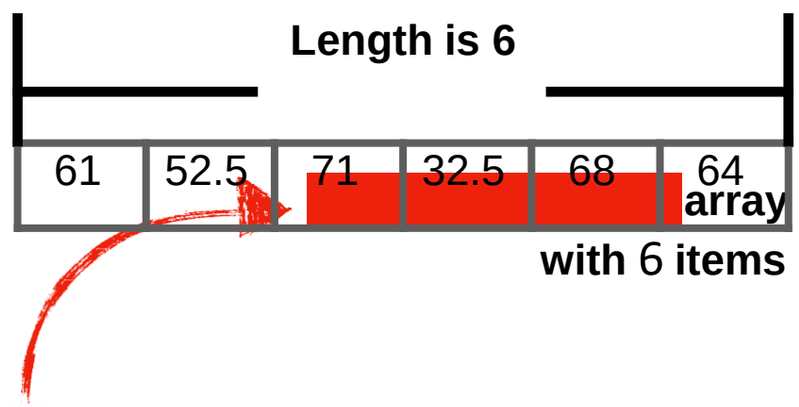
- NumPy arrays
  - ▶ Create arrays from Python lists
    - ✓ Attributes of arrays

```
1  # Attributes of the one-dimensional array
2
3  print('Dimension number:      ', 
4  print('Array shape:           ', 
5  print('Number of data items: ', 
6  print('Data type of items:    ', 
```

```
Dimension number:
Array shape:
Number of data items:
Data type of items:
```

Array Data Structure

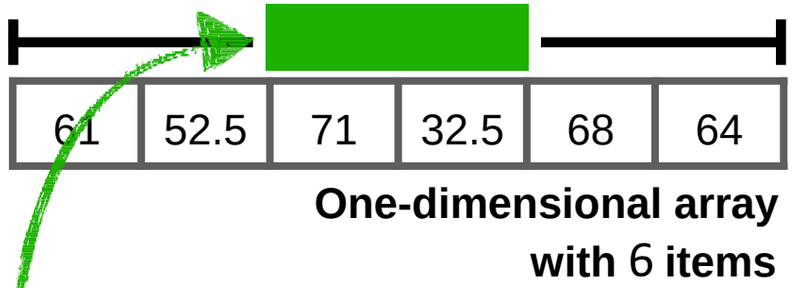
- NumPy arrays
  - ▸ Create arrays from Python lists
    - ✓ Attributes of arrays




```
Dimension number:
Array shape:              (6,)
Number of data items:     6
Data type of items:       float64
```

Array Data Structure

- NumPy arrays
  - ▸ Create arrays from Python lists
    - ✓ Attributes of arrays

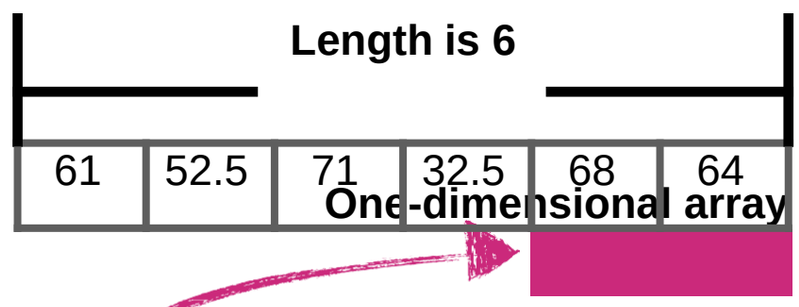



```
Dimension number:      1
Array shape:           
Number of data items:  6
Data type of items:    float64
```

A tuple with one item

Array Data Structure

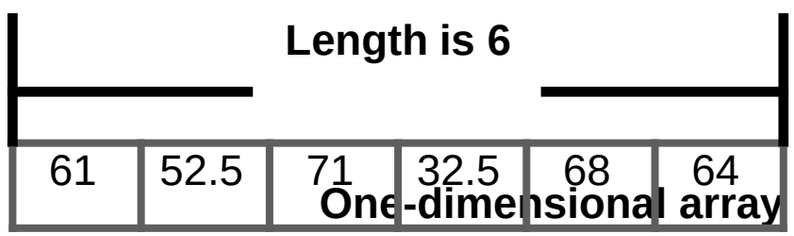
- NumPy arrays
  - ▸ Create arrays from Python lists
    - ✓ Attributes of arrays




```
Dimension number:      1
Array shape:           (6,)
Number of data items:  6
Data type of items:    float64
```

Array Data Structure

- NumPy arrays
  - ▸ Create arrays from Python lists
    - ✓ Attributes of arrays




```
Dimension number:      1
Array shape:           (6,)
Number of data items:  6
Data type of items:
```

Data type: floating point numbers

Array Data Structure

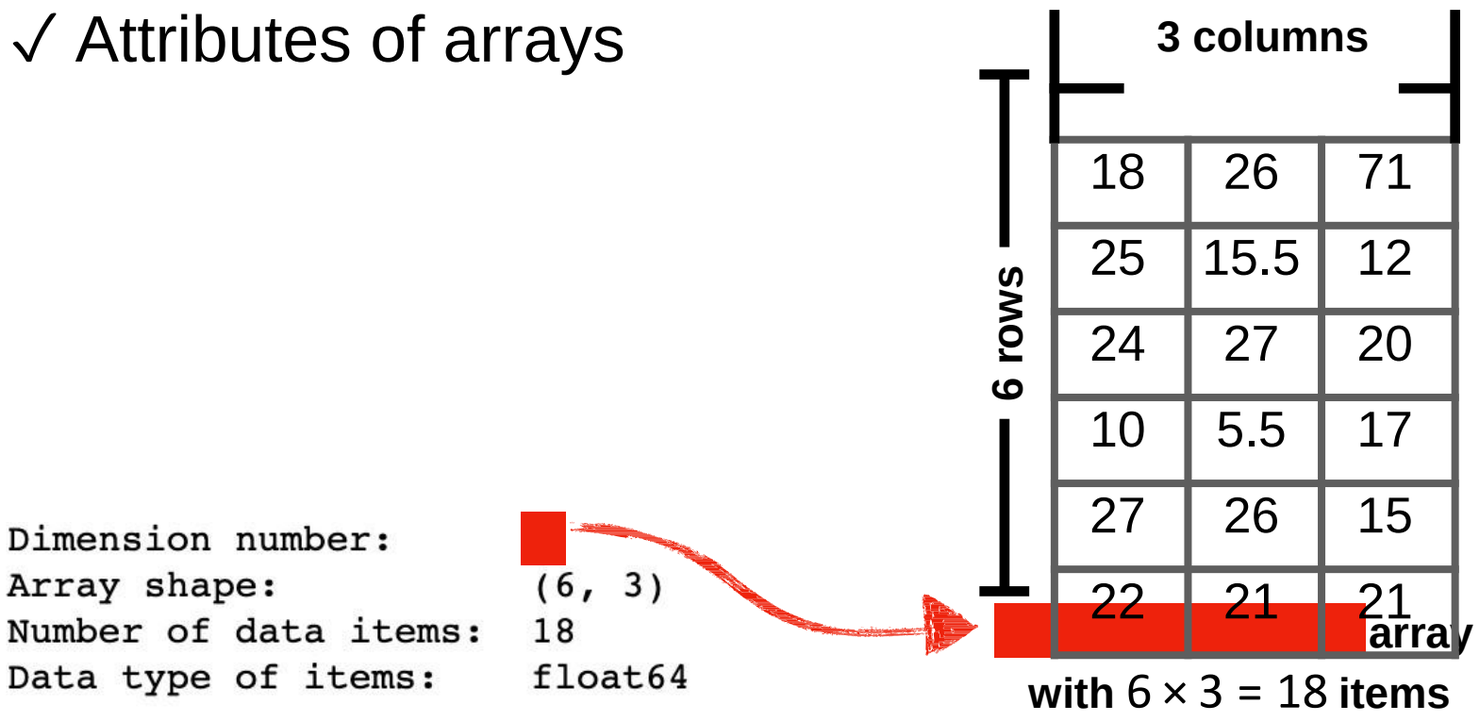
- NumPy arrays
  - ▶ Create arrays from Python lists
    - ✓ Attributes of arrays

```
1  # Attributes of the two-dimensional array
2
3  print('Dimension number:      ',
4  print('Array shape:           ',
5  print('Number of data items: ',
6  print('Data type of items:    ',
```

```
Dimension number:
Array shape:
Number of data items:
Data type of items:
```

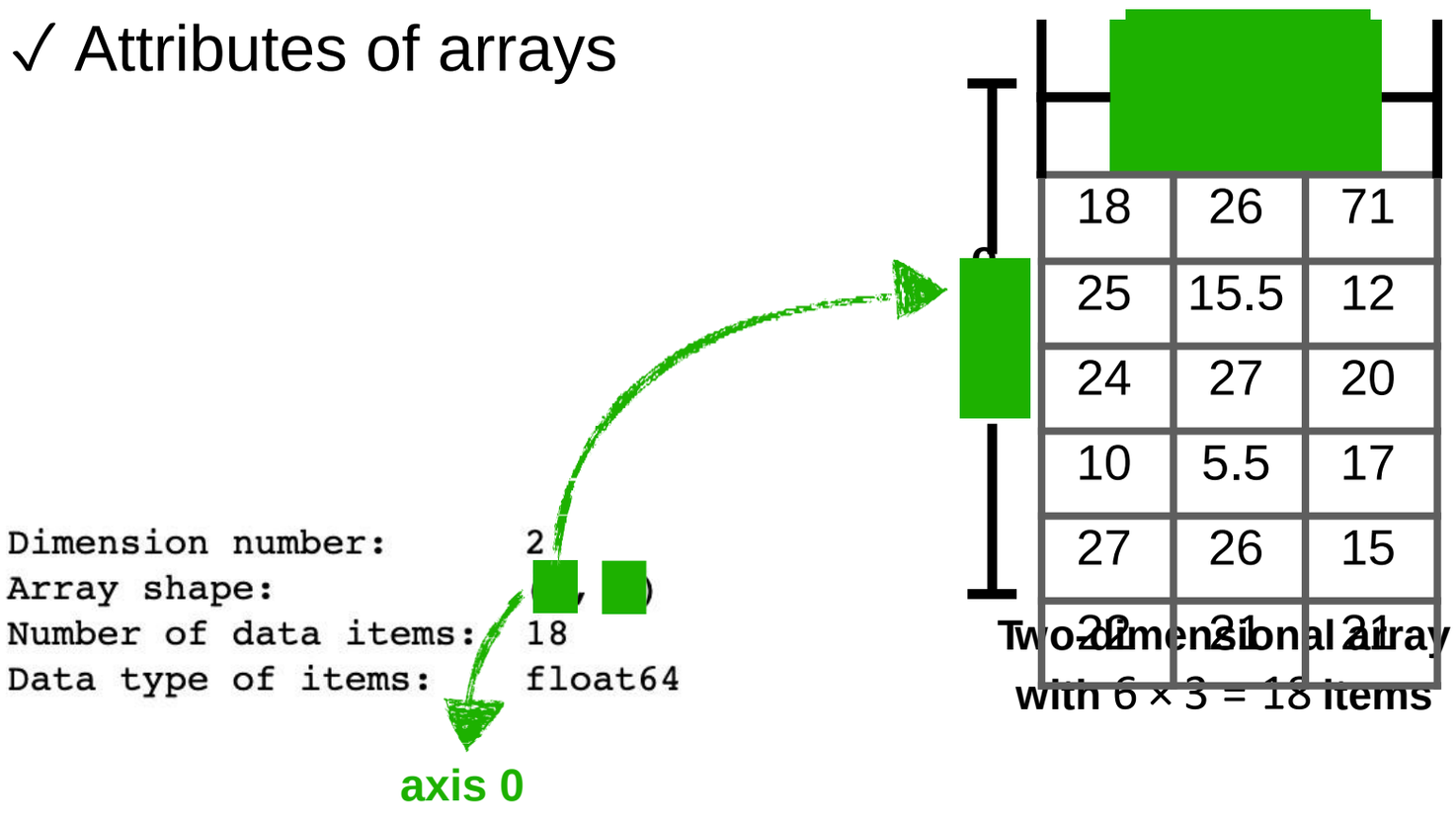
Array Data Structure

- NumPy arrays
  - Create arrays from Python lists
    - ✓ Attributes of arrays



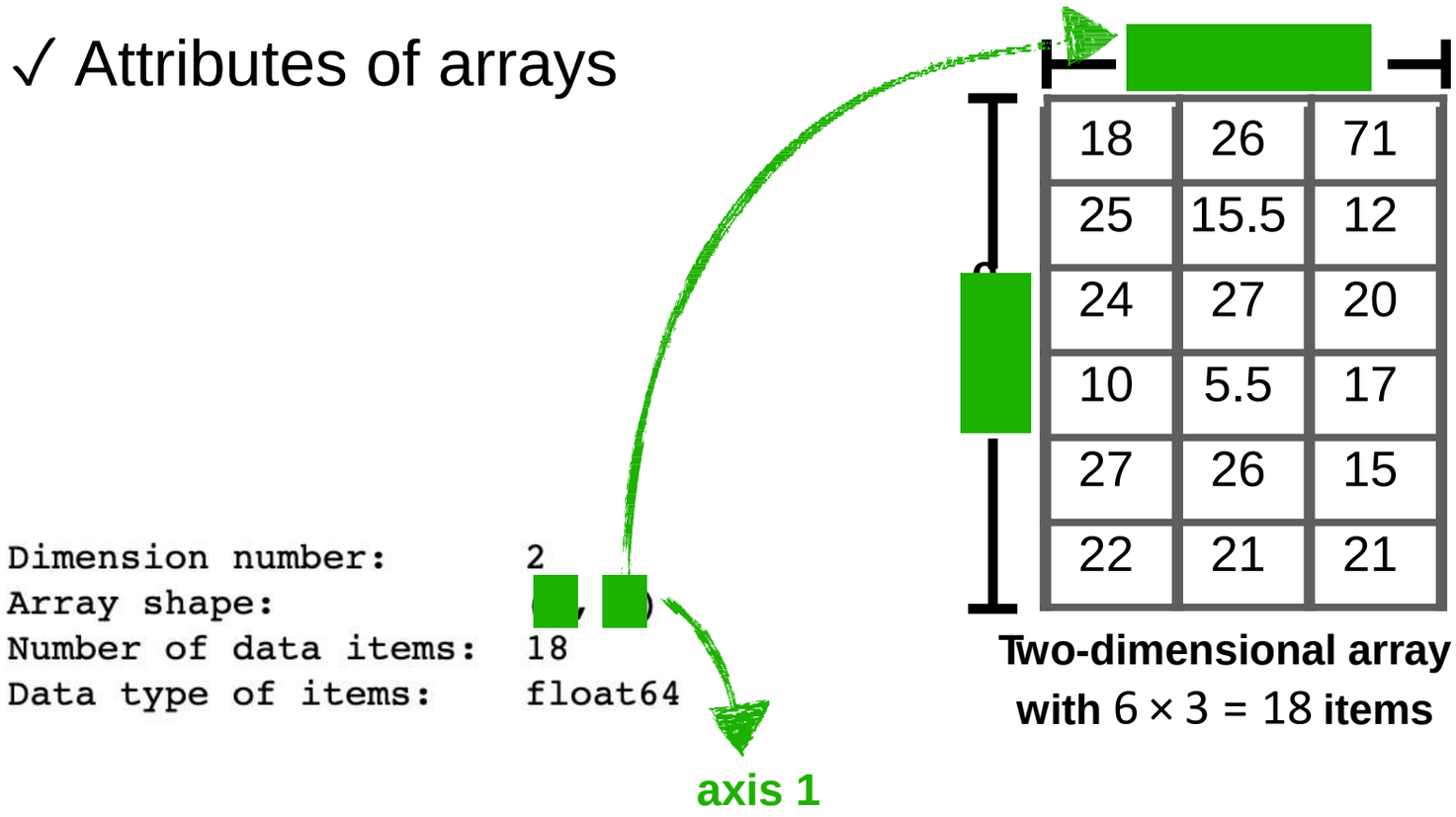
Array Data Structure

- NumPy arrays
  - Create arrays from Python lists
    - ✓ Attributes of arrays



Array Data Structure

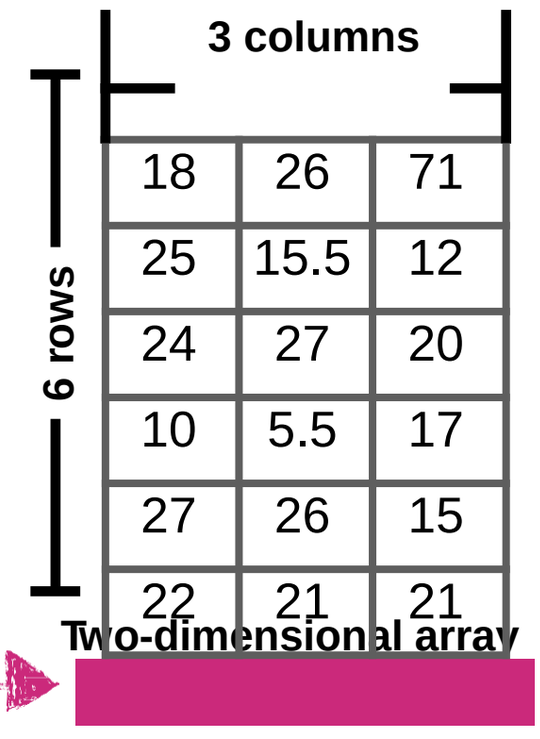
- NumPy arrays
  - Create arrays from Python lists
    - ✓ Attributes of arrays



Array Data Structure

- NumPy arrays
  - Create arrays from Python lists
    - ✓ Attributes of arrays

```
Dimension number:      2
Array shape:           (6, 3)
Number of data items:  18
Data type of items:    float64
```

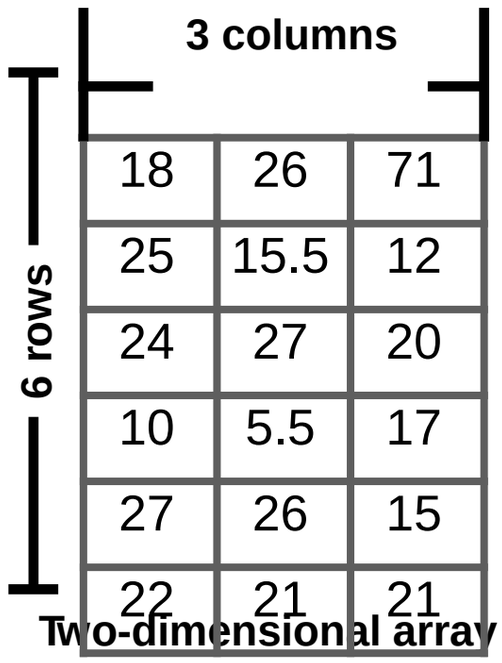


Array Data Structure

- NumPy arrays
  - Create arrays from Python lists
    - ✓ Attributes of arrays

```
Dimension number:       2
Array shape:            (6, 3)
Number of data items:   18
Data type of items:     float64
```

Data type: floating point numbers



Two-dimensional array with 6 × 3 = 18 items

Array Data Structure

- NumPy arrays
  - Create special arrays

```
1  ones_2d = np.ones(                )
2  ones_2d
```

```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

A two-dimensional array:
3 rows and 5 columns

```
1  zeros_1d = np.zeros(
2  zeros_1d
```

A one-dimensional array: length is 5

```
array([0., 0., 0., 0., 0.])
```

```
1  range_array = np.arange
2  range_array
```

(start, stop, step)

```
array([2. , 2.5, 3. , 3.5, 4. , 4.5])
```

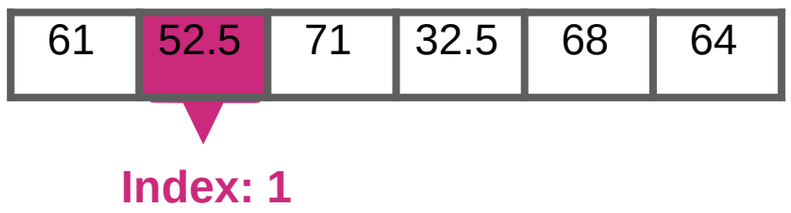
Array Data Structure

- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
 9  print([redacted])
10  print(array_2d[0, 1])
11  print(array_1d[2:])
12  print(array_2d[3:, :])
13  print(array_2d[[0, 2, 1]])
```

[redacted]

```
26.0
[71.  32.5 68.  64. ]
[[10.   5.5 17. ]
 [27.  26.  15. ]
 [22.  21.  21. ]]
[[18.  26.  17. ]
 [24.  27.  20. ]
 [25.  15.5 12. ]]
```



Array Data Structure

- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
 9  print(array_1d[1])
10  print([redacted])
11  print(array_1d[2:])
12  print(array_2d[3:, :])
13  print(array_2d[[0, 2, 1]])
```

```
52.5
[redacted]
[71.  32.5 68.  64. ]
[[10.   5.5 17. ]
 [27.  26.  15. ]
 [22.  21.  21. ]]
[[18.  26.  17. ]
 [24.  27.  20. ]
 [25.  15.5 12. ]]
```

Row index: 0
Column index: 1

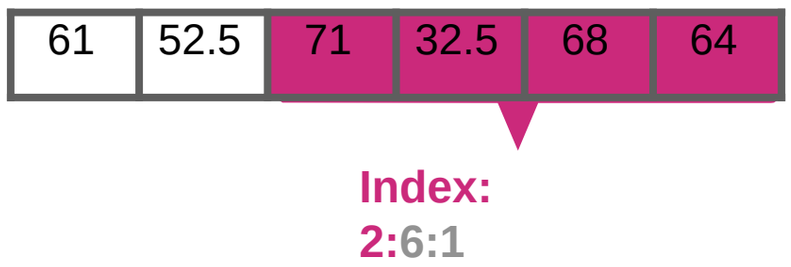
18	26	71
25	15.5	12
24	27	20
10	5.5	17
27	26	15
22	21	21

Array Data Structure

- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
 9  print(array_1d[1])
10  print(array_2d[0, 1])
11  print([redacted])
12  print(array_2d[3:, :])
13  print(array_2d[[0, 2, 1]])
```

```
52.5
26.0
[redacted]
[[10.   5.5 17. ]
 [27.  26.  15. ]
 [22.  21.  21. ]]
[[18.  26.  17. ]
 [24.  27.  20. ]
 [25.  15.5 12. ]]
```



Array Data Structure

- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
 9  print(array_1d[1])
10  print(array_2d[0, 1])
11  print(array_1d[2:])
12  print([redacted])
13  print(array_2d[[0, 2, 1]])
```

```
52.5
26.0
[71.  32.5 68.  64. ]
[redacted]
[[18.  26.  17. ]
 [24.  27.  20. ]
 [25.  15.5 12. ]]
```

18	26	71
25	15.5	12
24	27	20
10	5.5	17
27	26	15
22	21	21

Row index: 3:6:1
Column index: 0:3:1

Array Data Structure

- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
 9  print(array_1d[1])
10  print(array_2d[0, 1])
11  print(array_1d[2:])
12  print(array_2d[3:, :])
13  print([redacted])
```

```
52.5
26.0
[71.  32.5 68.  64. ]
[[10.   5.5 17. ]
 [27.  26.  15. ]
 [22.  21.  21. ]]
[redacted]
```

18	26	71
25	15.5	12
24	27	20
10	5.5	17
27	26	15
22	21	21

Row index: [0, 2, 1]
Column index:
0:3:1

Array Data Structure

- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
 9  print(array_1d[1])
10  print(array_2d[0, 1])
11  print(array_1d[2:])
12  print(array_2d[3:, :])
13  print([redacted]0:3:1)
```

```
52.5
26.0
[71.  32.5 68.  64. ]
[[10.   5.5 17. ]
 [27.  26.  15. ]
 [22.  21.  21. ]]
```

[redacted]

Take all columns if column index is no specified

18	26	71
25	15.5	12
24	27	20
10	5.5	17
27	26	15
22	21	21

Row index: [0, 2, 1]
Column index:

Array Data Structure

- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
array_2d[redacted] = [redacted]
array_2d
```

```
array([[18. , 26. , 17. ],
       [25. , 15.5, 12. ],
       [25. , 28. , 21. ],
       [10. ,  5.5, 17. ],
       [27. , 26. , 15. ],
       [22. , 21. , 21. ]])
```

```
array_1d[2] = 74
array_1d
```

```
array([61. , 52.5, 74. , 32.5, 68. , 64. ])
```

18	26	71
25	15.5	12
24	27	20
10	5.5	17
27	26	15
22	21	21

Items to be changed

Array Data Structure

- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
1  array_2d[redacted] = [redacted]
2  array_2d
```

```
array([[18. , 26. , 17. ],
       [25. , 15.5, 12. ],
       [redacted],
       [10. ,  5.5, 17. ],
       [27. , 26. , 15. ],
       [22. , 21. , 21. ]])
```

```
1  array_1d[2] = 74
2  array_1d
```

```
array([61. , 52.5, 74. , 32.5, 68. , 64. ])
```

18	26	71
25	15.5	12
25	28	21
10	5.5	17
27	26	15
22	21	21

New item values

Array Data Structure

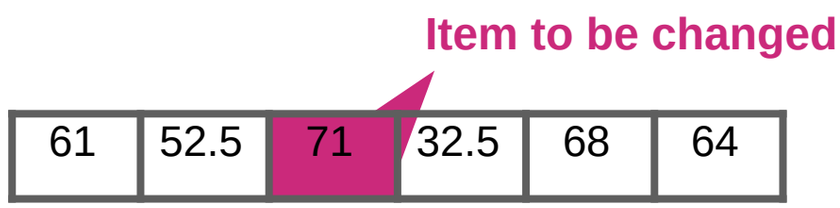
- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
1  array_2d[2, :] = np.array([25, 28, 21])
2  array_2d
```

```
array([[18. , 26. , 17. ],
       [25. , 15.5, 12. ],
       [25. , 28. , 21. ],
       [10. ,  5.5, 17. ],
       [27. , 26. , 15. ],
       [22. , 21. , 21. ]])
```

```
1  array_1d[redacted] = [redacted]
2  array_1d
```

```
array([61. , 52.5, 74. , 32.5, 68. , 64. ])
```



Array Data Structure

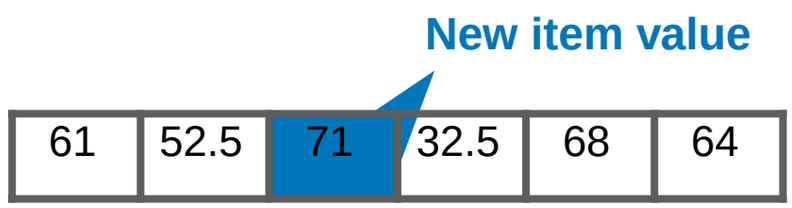
- NumPy arrays
  - Indexing and slicing of arrays
    - ✓ Similar to [redacted] type data

```
1  array_2d[2, :] = np.array([25, 28, 21])
2  array_2d
```

```
array([[18. , 26. , 17. ],
       [25. , 15.5, 12. ],
       [25. , 28. , 21. ],
       [10. ,  5.5, 17. ],
       [27. , 26. , 15. ],
       [22. , 21. , 21. ]])
```

```
1  array_1d[redacted] = [redacted]
2  array_1d
```

```
array([61. , 52.5, [redacted]. , 32.5, 68. , 64. ])
```



Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations
    - ✓ An array and a scalar

1	2			4	5
2	3.5	+	3	=	5	6.5
5	6.5			8	9.5

1	2			2	4
2	3.5	*	2	=	4	7
5	6.5			10	13

```
1  array_2d = np.array([[1, 2],
2                       [2, 3.5],
3                       [5, 6.5]])
4
5
6
7  print(array_2d + array_2d)
8  print(array_2d * array_2d)
```

```
[[ 2.  4.]
 [ 4.  7.]
 [10. 13.]]
[[ 1.    4.  ]
 [ 4.   12.25]
 [25.   42.25]]
```

Array Data Structure

- NumPy arrays
  - ▸ Element-wise arithmetic operations
    - ✓ An array and a scalar

```
1  array_2d = np.array([[1, 2],
2                       [2, 3.5],
3                       [5, 6.5]])
4
5  print(array_2d + 3)
6  print(array_2d * 2)
7
8
```

```
[[4.  5. ]
 [5.  6.5]
 [8.  9.5]]
[[ 2.  4.]
 [ 4.  7.]
 [10. 13.]]
```

1	2		1	2		2	4
2	3.5	+	2	3.5	=	4	7
5	6.5		5	6.5		10	13

1	2		1	2		1	4
2	3.5	*	2	3.5	=	4	12.25
5	6.5		5	6.5		25	42.25

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

Example 1

`usd` is a list of containing five money transactions in US dollars. Transfer each transaction into Singapore dollars.

```
import numpy as np

usd = [2, 3.60, 2.05, 13.50, 18.90]
exchange_rate = 1.37

usd_array = np.array(usd)                  # Convert list into a numpy array
sgd_array = usd_array * exchange_rate      # Element-wise multiplication

print(sgd_array)
```

```
[ 2.74    4.932   2.8085 18.495  25.893 ]
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

2	3.60	2.05	13.50	18.90

```
import numpy as np

[redacted]
exchange_rate = 1.37

[redacted]                                  # Convert list into a numpy array
sgd_array = usd_array * exchange_rate    # Element-wise multiplication

print(sgd_array)
```

```
[ 2.74    4.932   2.8085 18.495  25.893 ]
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

2	3.60	2.05	13.50	18.90	* 1.37

```
import numpy as np

usd = [2, 3.60, 2.05, 13.50, 18.90]
exchange_rate = 1.37

usd_array = np.array(usd)                    # Convert list into a numpy array
                                             # Element-wise multiplication

print(sgd_array)
```

```
[ 2.74    4.932   2.8085 18.495  25.893 ]
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

2.74	4.932	2.8085	18.495	25.893

```
import numpy as np

usd = [2, 3.60, 2.05, 13.50, 18.90]
exchange_rate = 1.37

usd_array = np.array(usd)                  # Convert list into a numpy array
sgd_array = usd_array * exchange_rate      # Element-wise multiplication

print(sgd_array)
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations
    - ✓ More concise
    - ✓ Easier to read
    - ✓ Faster to execute than loops

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

Example 2

The discrete distribution information of two types of newspapers is stored in a two-dimensional array `distr`. The first row represents the probabilities of each weather condition, the second and third rows give the corresponding newspaper demands. Calculate 1) the expected newspaper demands; 2) the standard deviations of newspaper demands; and 3) the expected total profit with the given parameters

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

	Sunny	Cloudy Raining		Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

	Sunny	Cloudy Raining		Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

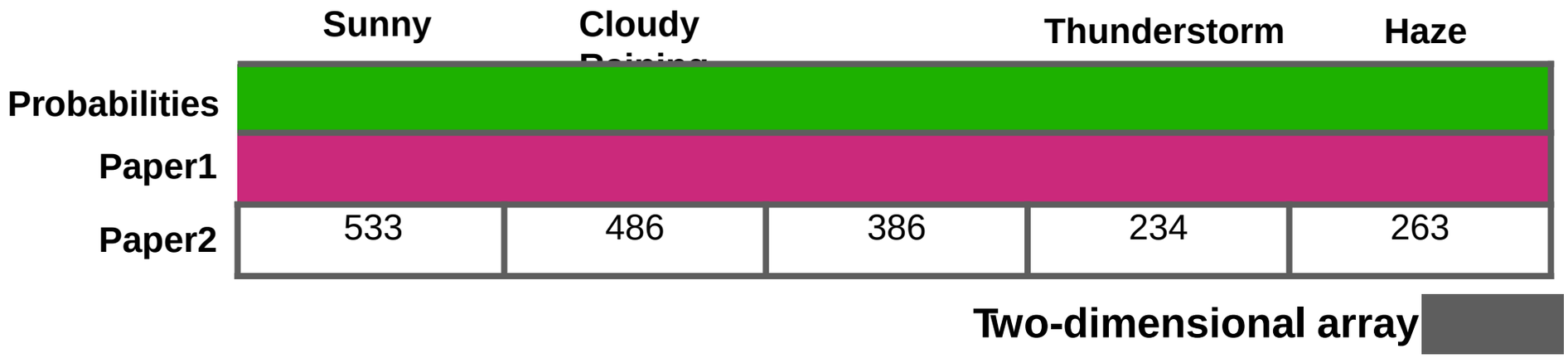
Two-dimensional array

```
1  exp1 = np.sum(distr[0] * distr[1])
2  var1 = np.sum(distr[0] * (distr[1] - exp1)**2)
3  std1 = var1 ** 0.5
4
5  print('Expected demand: {0:0.3f}'.format(exp1))
6  print('Standard deviation: {0:0.3f}'.format(std1))
```

```
Expected demand: 449.249
Standard deviation: 118.908
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

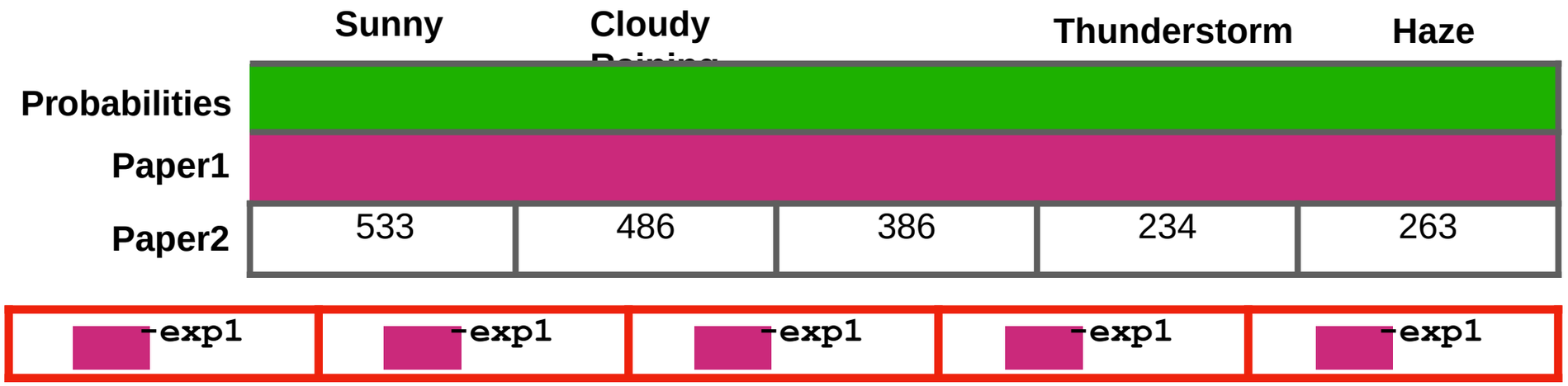



```
1  exp1 = np.sum(          *          )
2  var1 = np.sum(distr[0] * (distr[1] - exp1)**2)
3  std1 = var1 ** 0.5
4
5  print('Expected demand: {0:0.3f}'.format(exp1))
6  print('Standard deviation: {0:0.3f}'.format(std1))
```

```
Expected demand: 449.249
Standard deviation: 118.908
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

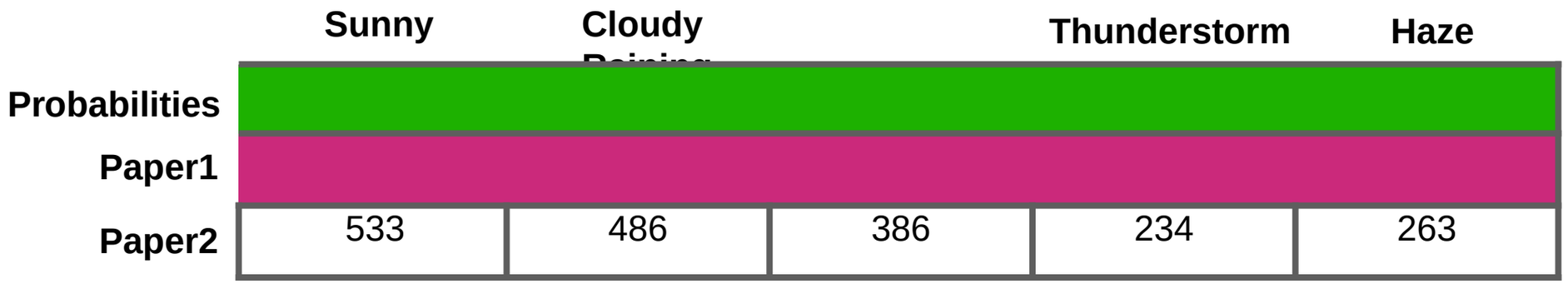



```
1  exp1 = np.sum(distr[0] * distr[1])
2  var1 = np.sum(                 * (                 - exp1)**2)
3  std1 = var1 ** 0.5
4
5  print('Expected demand: {0:0.3f}'.format(exp1))
6  print('Standard deviation: {0:0.3f}'.format(std1))
```

```
Expected demand: 449.249
Standard deviation: 118.908
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

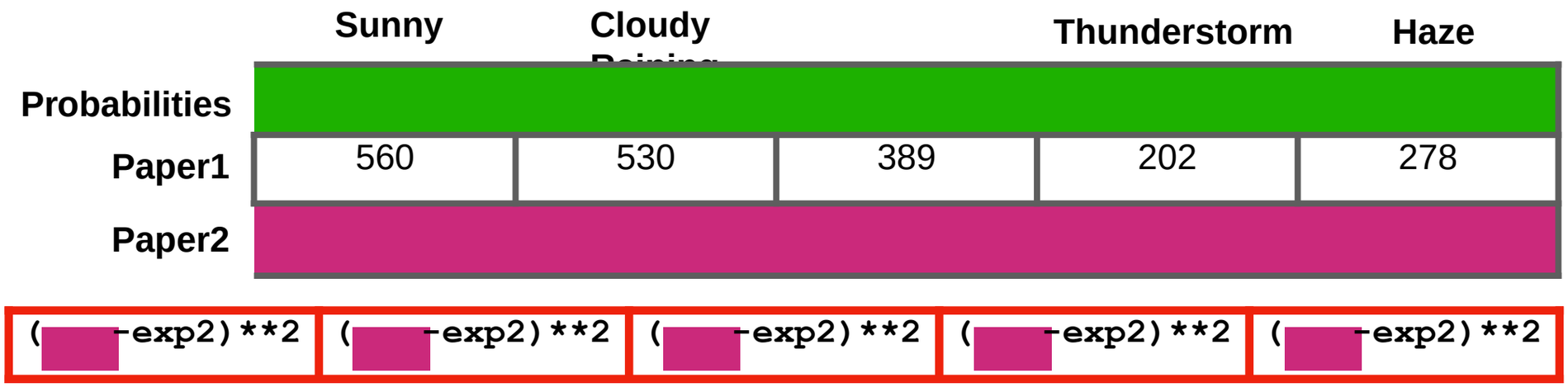



```
1  exp1 = np.sum(distr[0] * distr[1])
2  var1 = np.sum(          * (          - exp1)**2)
3  std1 = var1 ** 0.5
4
5  print('Expected demand: {0:0.3f}'.format(exp1))
6  print('Standard deviation: {0:0.3f}'.format(std1))
```

```
Expected demand: 449.249
Standard deviation: 118.908
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations




```
exp2 = np.sum(                 *                )
var2 = np.sum(                 *  (             - exp2)**2)
std2 = var2 ** 0.5

print('Expected demand: {0:0.3f}'.format(exp2))
print('Standard deviation: {0:0.3f}'.format(std2))
```

```
Expected demand: 431.472
Standard deviation: 101.316
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations


```
1  cost = np.array([0.6, 0.8])
2  price = np.array([1.0, 1.15])
3  order = np.array([    , 380])
4
5  exp_sold1 = np.sum(distr[0] * np.minimum(           ,           ))
6  exp_sold2 = np.sum(distr[0] * np.minimum(order[1], distr[2]))
7  exp_profit = (price[0]*exp_sold1 + price[1]*exp_sold2) - np.sum(order*cost)
8  print('Expected profit: ${0:0.2f}'.format(exp_profit))
```

```
Expected profit: $234.92
```

Array Data Structure

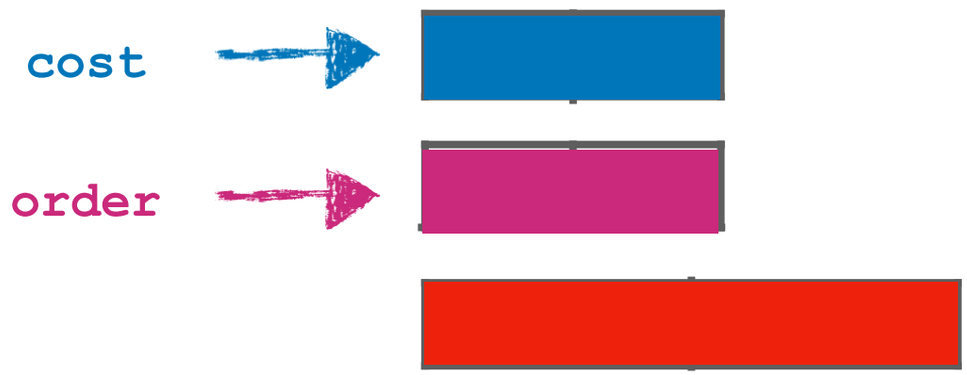
- NumPy arrays
  - Element-wise arithmetic operations


```
1  cost = np.array([0.6, 0.8])
2  price = np.array([1.0, 1.15])
3  order = np.array([430, 380])
4
5  exp_sold1 = np.sum(distr[0] * np.minimum(order[0], distr[1]))
6  exp_sold2 = np.sum(distr[0] * np.minimum(order[1], distr[2]))
7  exp_profit = (price[0]*exp_sold1 + price[1]*exp_sold2) - np.sum(order*cost)
8  print('Expected profit: ${0:0.2f}'.format(exp_profit))
```

```
Expected profit: $234.92
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations




```
cost = np.array(                )
price = np.array([1.0, 1.15])
order = np.array(               )

exp_sold1 = np.sum(distr[0] * np.minimum(order[0], distr[1]))
exp_sold2 = np.sum(distr[0] * np.minimum(order[1], distr[2]))
exp_profit = (price[0]*exp_sold1 + price[1]*exp_sold2) - np.sum(              )
print('Expected profit: ${0:0.2f}'.format(exp_profit))
```

```
Expected profit: $234.92
```

Array Data Structure

- NumPy arrays
  - Element-wise arithmetic operations

```
cost = np.array([0.6, 0.8])
price = np.array([1.0, 1.15])
order = np.array([430, 380])

exp_sold1 = np.sum(distr[0] * np.minimum(order[0], distr[1]))
exp_sold2 = np.sum(distr[0] * np.minimum(order[1], distr[2]))
[redacted]
print('Expected profit: ${0:0.2f}'.format(exp_profit))
```

```
Expected profit: [redacted]
```

Array Data Structure

- NumPy arrays
  - Broadcasting

✓ An array and a scalar

✓ Two arrays of the same shape

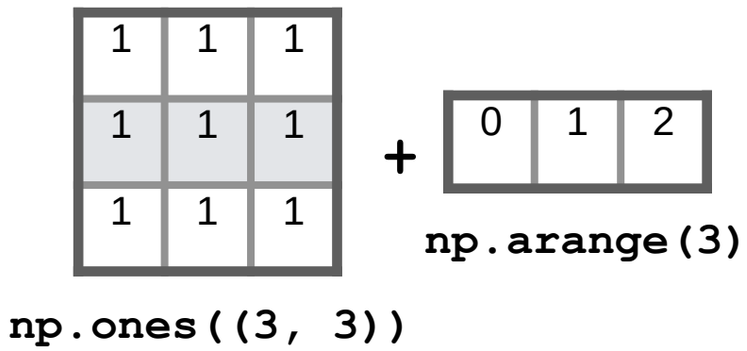
1	2	+	3	=	4	5
2	3.5				5	6.5
5	6.5				8	9.5

1	2	+	1	2	=	2	4
2	3.5		2	3.5		4	7
5	6.5		5	6.5		10	13

Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ Two arrays of different same shapes

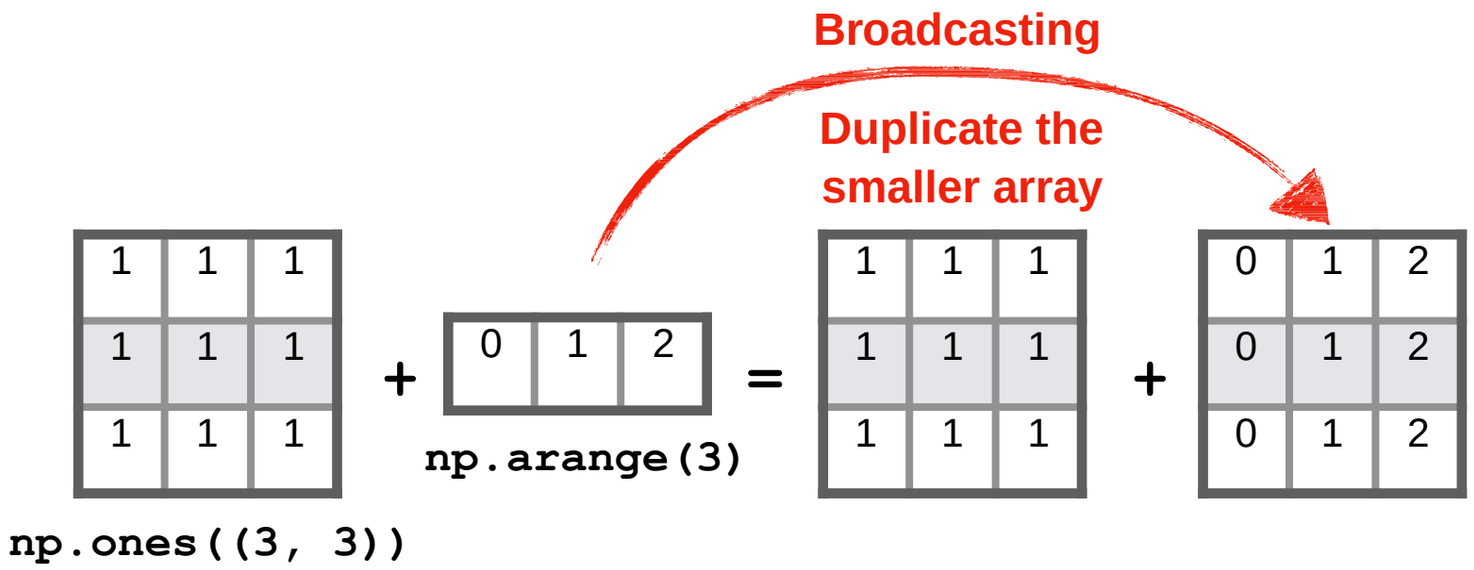
```
1  np.ones((3, 3)) + np.arange(3)
```



Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ Two arrays of different same shapes

```
1  np.ones((3, 3)) + np.arange(3)
```

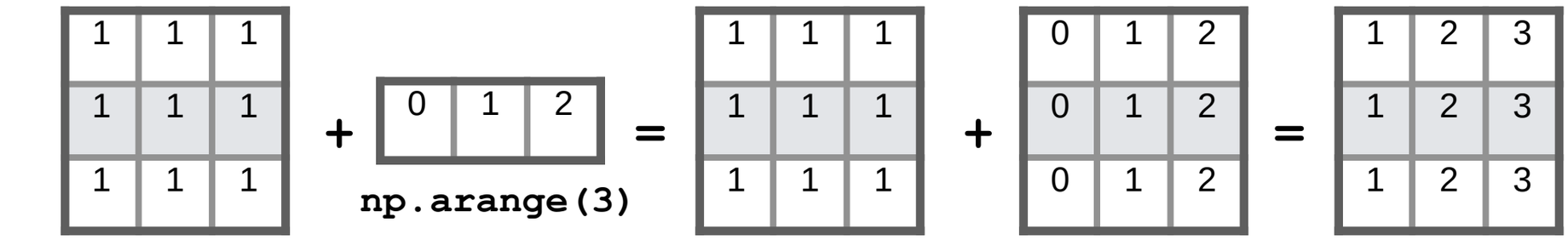


Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ Two arrays of different same shapes

```
1  np.ones((3, 3)) + np.arange(3)
```

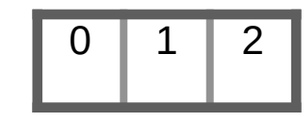
```
array([[1., 2., 3.],
       [1., 2., 3.],
       [1., 2., 3.]])
```



Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ Two arrays of different same shapes

```
1  np.arange(3).reshape((3, 1)) + np.arange(3)
```

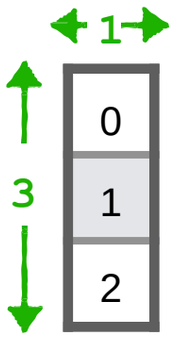


np.arange(3)

Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ Two arrays of different same shapes

```
1  np.arange(3).reshape((3, 1)) + np.arange(3)
```



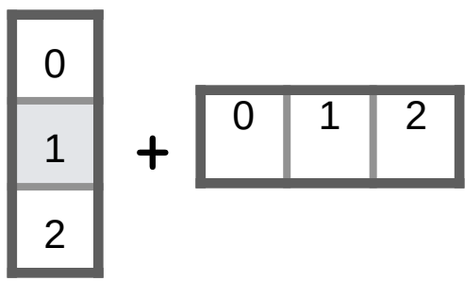
Shape is changed from (3,) to (3, 1)

`np.arange(3).reshape(`[redacted]`)`

Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ Two arrays of different same shapes

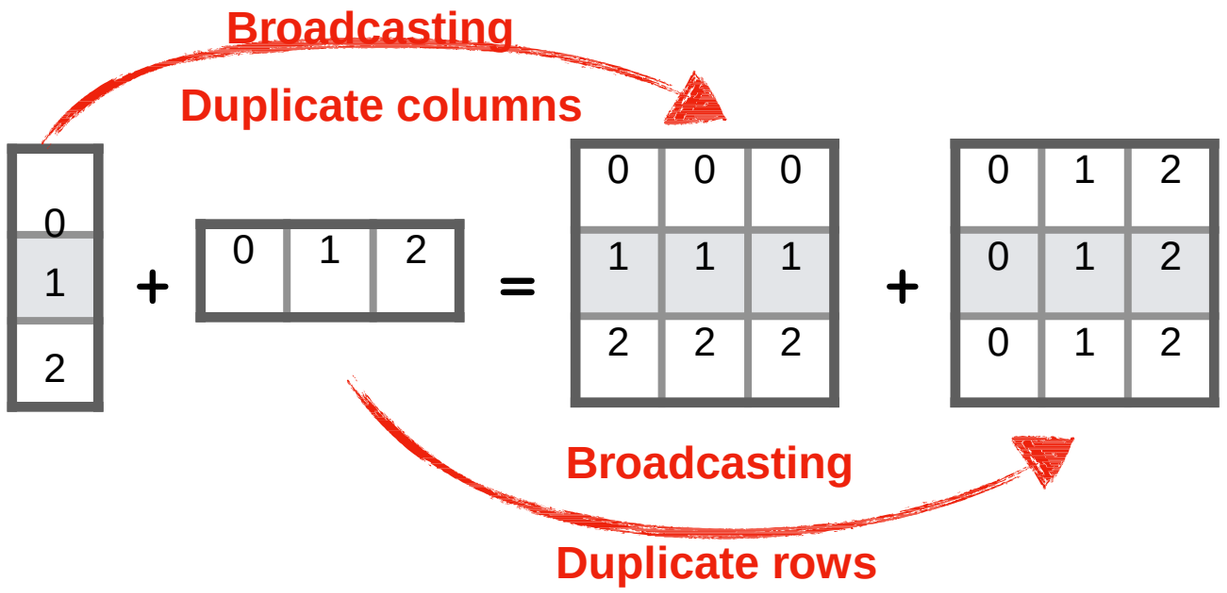
```
1  np.arange(3).reshape((3, 1)) + np.arange(3)
```



Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ Two arrays of different same shapes

```
1  np.arange(3).reshape((3, 1)) + np.arange(3)
```

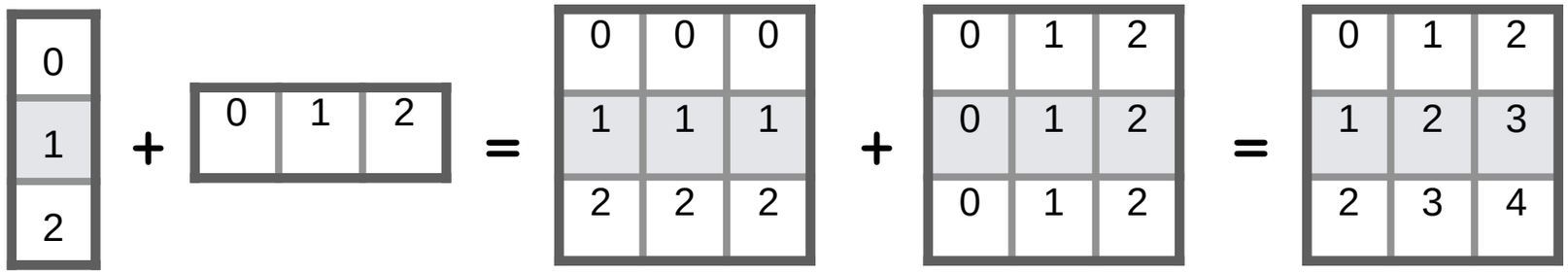


Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ Two arrays of different same shapes

```
1  np.arange(3).reshape((3, 1)) + np.arange(3)
```

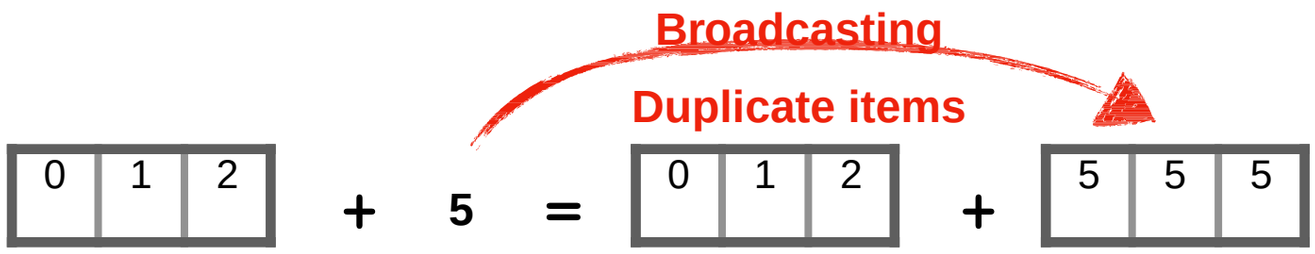
```
array([[0, 1, 2],
       [1, 2, 3],
       [2, 3, 4]])
```



Array Data Structure

- NumPy arrays
  - Broadcasting
    - ✓ An array and a scalar, interpreted by broadcasting

```
1  np.arange(3) + 5
```



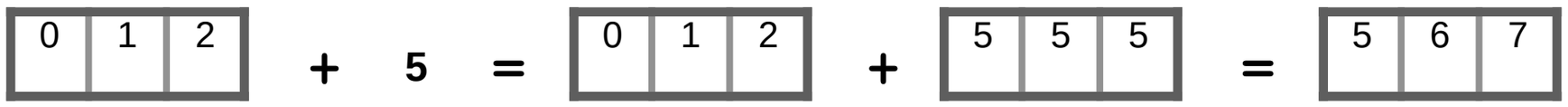
Array Data Structure

- NumPy arrays
  - Broadcasting

✓ An array and a scalar, interpreted by broadcasting

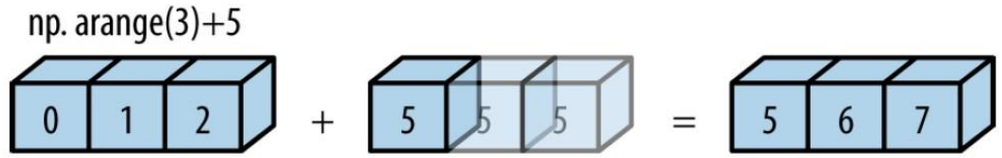
```
1  np.arange(3) + 5
```

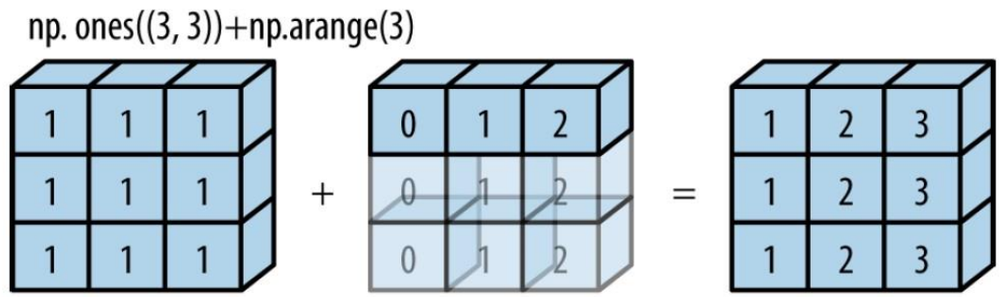
```
array([5, 6, 7])
```

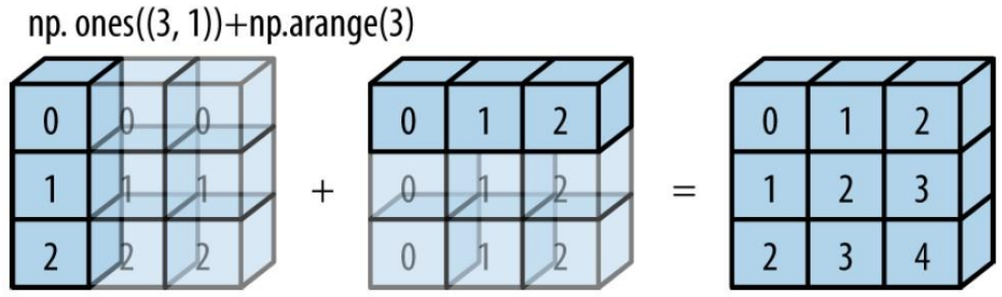


Array Data Structure

- NumPy arrays
  - Broadcasting







Array Data Structure

- NumPy arrays
  - ▶ Functions and array methods
    - ✓ Math functions

```
print(np.log(3))                    # Natural logarithm of 3
print(np.exp(np.arange(4)))         # Exponential of 0, 1, 2, 3
print(np.square(np.arange(3)))      # Squares of 0, 1, 2
print(np.power(np.arange(3), 3))    # Cubes of 0, 1, 2
```

```
1.0986122886681098
[ 1.          2.71828183  7.3890561  20.08553692]
[0 1 4]
[0 1 8]
```

Array Data Structure

- NumPy arrays
  - ▶ Functions and array methods
    - ✓ Aggregation methods

```
array_2d = np.array([[1, 2],
                     [2, 3.5],
                     [5, 6.5]])     # Create a two-dimensional array

print(array_2d.sum())               # Sum of all item
print(array_2d.max())               # The maximum item in the array
print(array_2d.min())               # The minimum item in the array
```

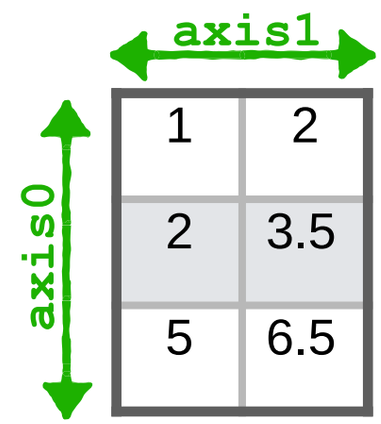
```
20.0
6.5
1.0
```

Array Data Structure

- NumPy arrays
  - ▸ Functions and array methods
    - ✓ Aggregation methods

```
1 print(array_2d.sum(axis=0))
2 print(array_2d.sum(axis=1))
3 print(array_2d.max(axis=0))
4 print(array_2d.min(axis=1))
```

```
[ 8. 12.]
[ 3.   5.5 11.5]
[5.  6.5]
[1. 2. 5.]
```

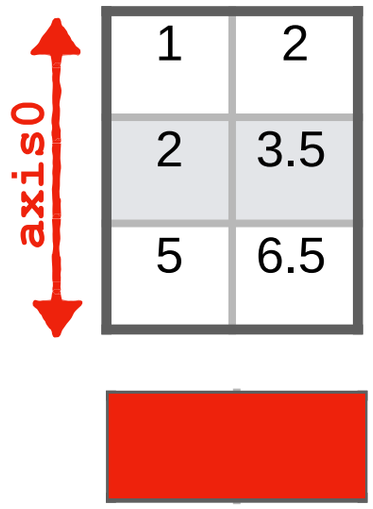


Array Data Structure

- NumPy arrays
  - ► Functions and array methods
    - ✓ Aggregation methods

```
1 print(array_2d.sum( ))
2 print(array_2d.sum(axis=1))
3 print(array_2d.max(axis=0))
4 print(array_2d.min(axis=1))
```

```
[redacted]
[ 3.   5.5 11.5]
[5.  6.5]
[1. 2. 5.]
```

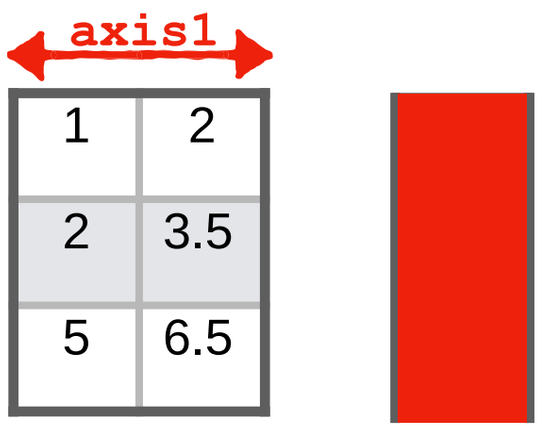


Array Data Structure

- NumPy arrays
  - ▸ Functions and array methods
    - ✓ Aggregation methods

```
1 print(array_2d.sum(axis=0))
2 print(array_2d.sum([redacted]))
3 print(array_2d.max(axis=0))
4 print(array_2d.min(axis=1))
```

```
[ 8. 12.]
[redacted]
[5.  6.5]
[1. 2. 5.]
```

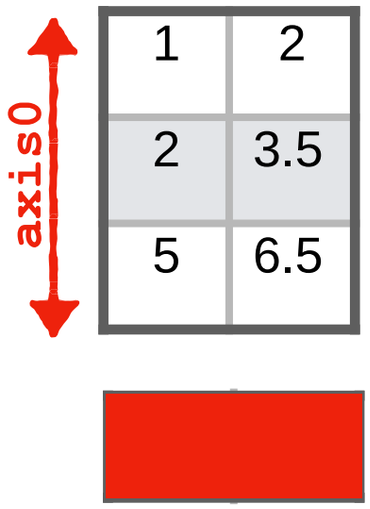


Array Data Structure

- NumPy arrays
  - ▸ Functions and array methods
    - ✓ Aggregation methods

```
1  print(array_2d.sum(axis=0))
2  print(array_2d.sum(axis=1))
3  print(array_2d.max(axis=1))
4  print(array_2d.min(axis=1))
```

```
[ 8. 12.]
[ 3.   5.5 11.5]
[2. 3. 6.5]
[1. 2. 5.]
```

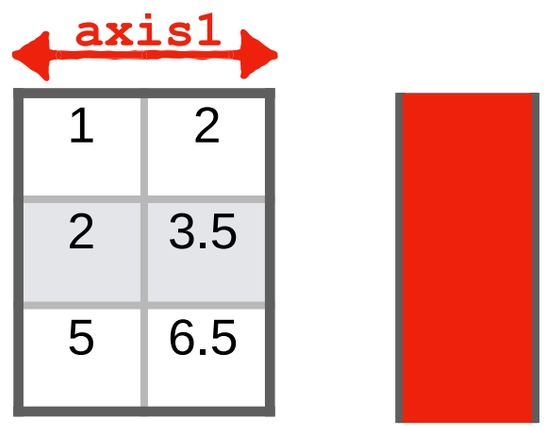


Array Data Structure

- NumPy arrays
  - ▸ Functions and array methods
    - ✓ Aggregation methods

```
1 print(array_2d.sum(axis=0))
2 print(array_2d.sum(axis=1))
3 print(array_2d.max(axis=0))
4 print(array_2d.min [redacted])
```

```
[ 8. 12.]
[ 3.   5.5 11.5]
[5.  6.5]
[redacted]
```



Array Data Structure

- NumPy arrays
  - Functions and array methods

The discrete distribution information of two types of newspapers is stored in a two-dimensional array `distr`. The first row represents the probabilities of each weather condition, the second and third rows give the corresponding newspaper demands. Calculate 1) the expected newspaper demands; 2) the standard deviations of newspaper demands; and 3) the expected total profit with the given parameters

Array Data Structure

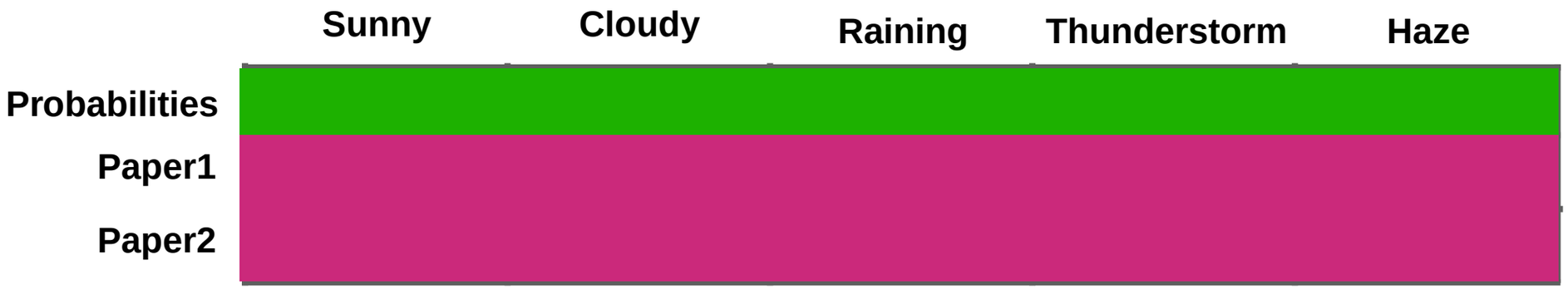
- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array

Array Data Structure

- NumPy arrays
  - Functions and array methods



Two-dimensional array `distr`

0.315	0.226	0.289	0.087	0.083

*

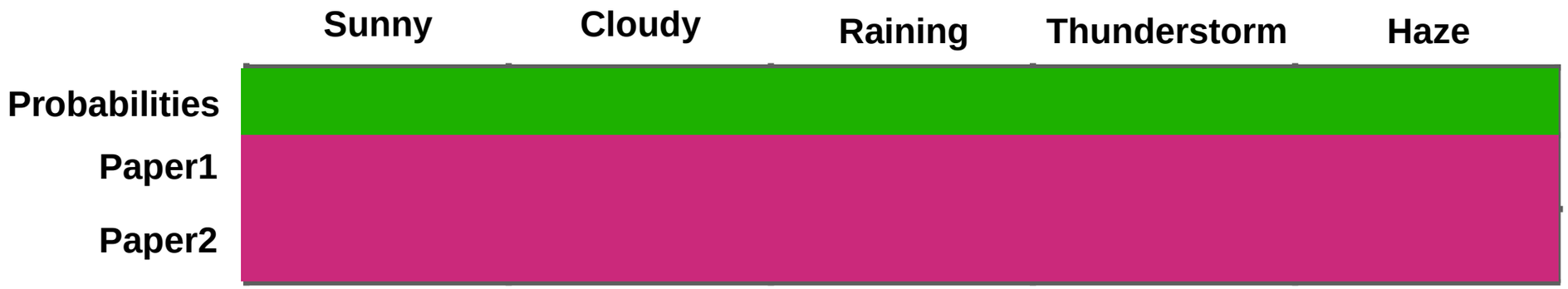
560	530	389	202	278
533	486	386	234	263

```
1  exp_d = (                *                ).sum(axis=1)
2  exp_d
```

```
array([449.249, 431.472])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods



Two-dimensional array `distr`

0.315	0.226	0.289	0.087	0.083
0.315	0.226	0.289	0.087	0.083

*

560	530	389	202	278
533	486	386	234	263

Broadcasting

```
1  exp_d = (          *          ).sum(axis=1)
2  exp_d
```

```
array([449.249, 431.472])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array `distr`

axis1

0.315*560	0.226*530	0.289*389	0.087*202	0.083*278
0.315*533	0.226*486	0.289*386	0.087*234	0.083*263

```
1  exp_d = (                                ).sum(          )
2  exp_d
```

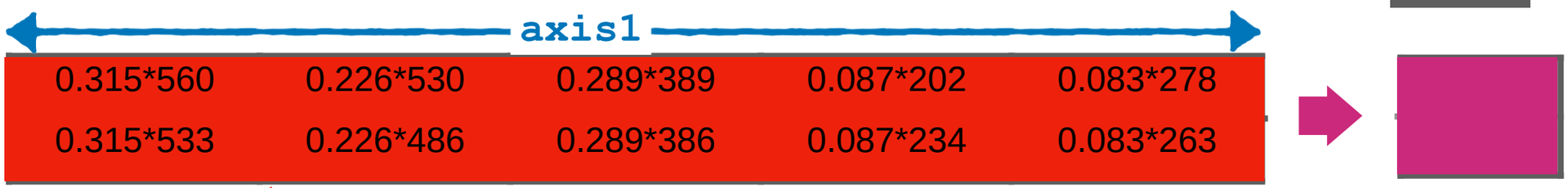
```
array([449.249, 431.472])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array `distr`



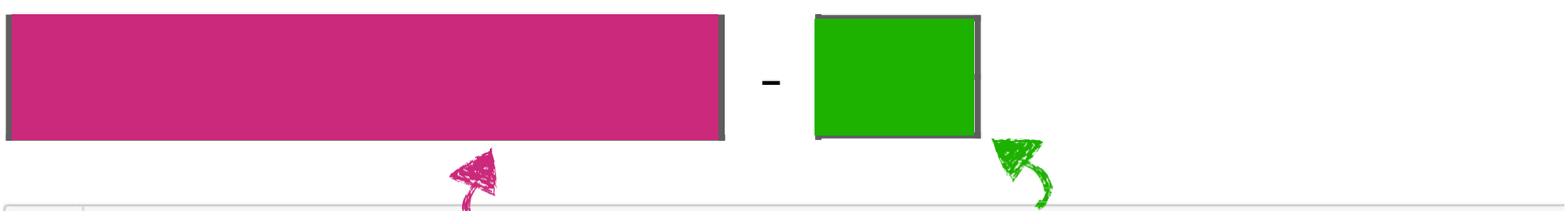

```
1  exp_d = (                    ).sum(      )
2  exp_d
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array



```
1  var_d = (distr[0] *                    )**2).sum(axis=1)
2  std_d = var_d ** 0.5
3  std_d
```

```
array([118.90763222, 101.3157797 ])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array

560	530	389	202	278
533	486	386	234	263

–

449.249	449.249	449.249	449.249	449.249
431.472	431.472	431.472	431.472	431.472

Broadcasting

```
1  var_d = (distr[0] *                  -                                  )**2).sum(axis=1)
2  std_d = var_d ** 0.5
3  std_d
```

```
array([118.90763222, 101.3157797 ])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array

```
1  var_d = (distr[0] *                    )**2).sum(axis=1)
2  std_d = var_d ** 0.5
3  std_d
```

```
array([118.90763222, 101.3157797 ])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array

431.472)**2 (263-431.472)**2

```
1 var_d = (distr[0] *                                    ).sum(axis=1)
2 std_d = var_d ** 0.5
3 std_d
```

```
array([118.90763222, 101.3157797 ])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities					
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array `distr`

axis1

(560-449.249)**2 (530-449.249)**2 (389-449.249)**2 (202-449.249)**2 (278-449.249)**2 (533-431.472)**2 (486-431.472)**2 (386-431.472)**2 (234-431.472)**2 (263-431.472)**2

```
1 var_d = (                  *                  ).sum(          )
2 std_d = var_d ** 0.5
3 std_d
```

```
array([118.90763222, 101.3157797 ])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1					
Paper2					

Two-dimensional array


```
1  order = np.array([430, 380])
2  exp_sold = (distr[0] * np.minimum(                    , 
3  exp_sold
```

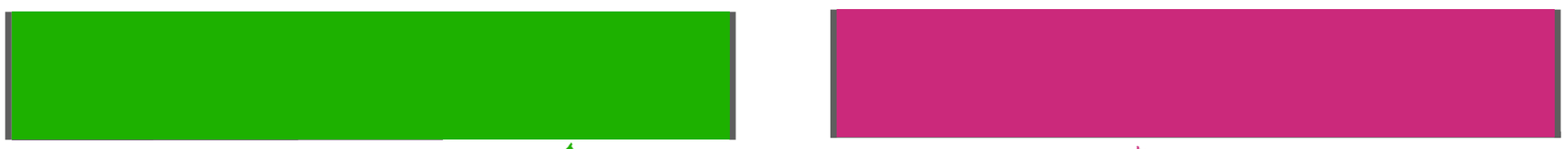
```
array([385.699, 357.587])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1					
Paper2					

Two-dimensional array



Broadcasting

```
1  order = np.array([430, 380])
2  exp_sold = (distr[0] * np.minimum(                    , 
3  exp_sold
```

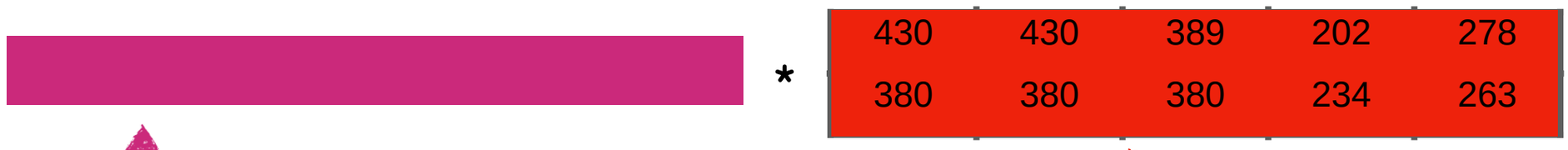
```
array([385.699, 357.587])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities					
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array `distr`




```
1  order = np.array([430, 380])
2  exp_sold = (          *                                    ).sum(axis=1)
3  exp_sold
```

```
array([385.699, 357.587])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities					
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array `distr`

0.315	0.226	0.289	0.087	0.083	*	430	430	389	202	278
0.315	0.226	0.289	0.087	0.083		380	380	380	234	263

Broadcasting

```
1  order = np.array([430, 380])
2  exp_sold = (                  *                                  ).sum(axis=1)
3  exp_sold
```

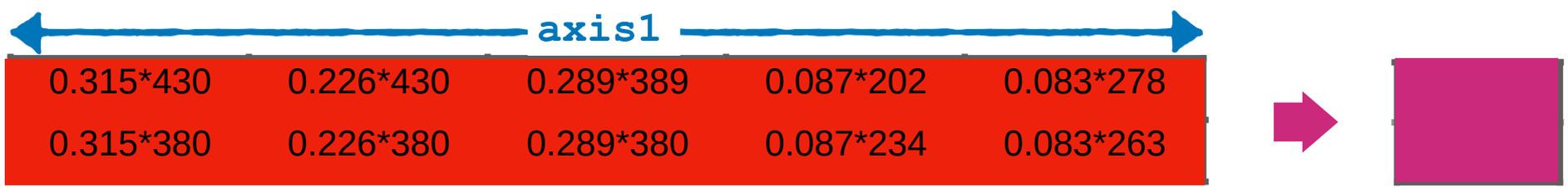
```
array([385.699, 357.587])
```

Array Data Structure

- NumPy arrays
  - Functions and array methods

	Sunny	Cloudy	Raining	Thunderstorm	Haze
Probabilities	0.315	0.226	0.289	0.087	0.083
Paper1	560	530	389	202	278
Paper2	533	486	386	234	263

Two-dimensional array `distr`



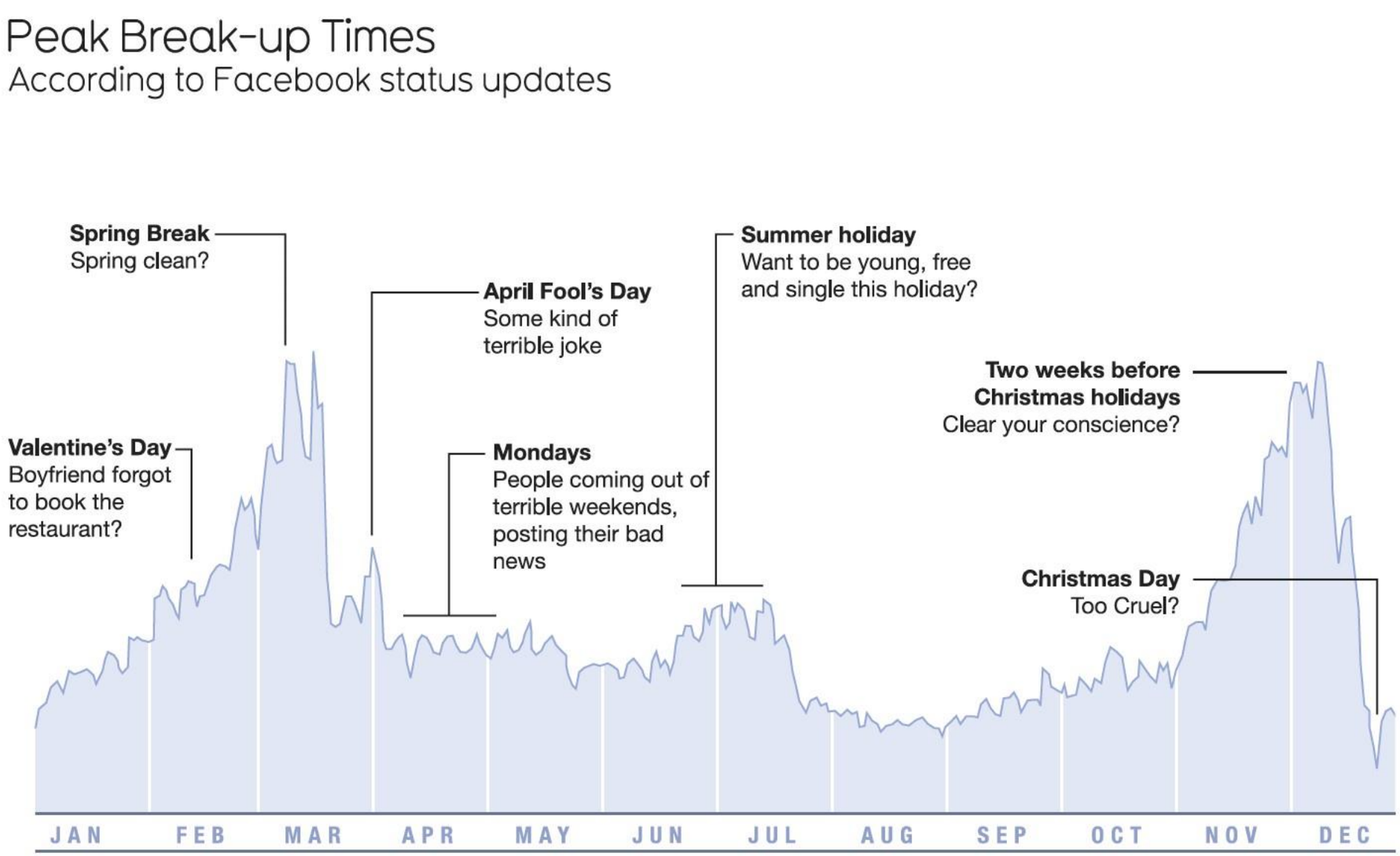

```
1  order = np.array([430, 380])
2  exp_sold = (                                   ).sum
3  exp_sold
```

```
array(                    )
```

Roadmap

1. Array Data Structure
2. **Data Visualization**
3. Pandas
4. Descriptive Analytics in Python

Data Visualization



Data Visualization

- Data Visualization Packages
  - Import the Matplotlib package

```
import matplotlib.pyplot as

.<function name>
```

  - Other package
    - ✓ Plotly
    - ✓ Seaborn

Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Line plot
    - ✓ [redacted] information of [redacted]

```
Parameters
----------
x, y : [redacted]
    The horizontal / vertical coordinates of the data points.
    *x* values are optional and default to `range(len(y))`.

    Commonly, these parameters are 1D arrays.

    They can also be scalars, or two-dimensional (in that case, the
    columns represent separate data sets).

    These arguments cannot be passed as keywords.
```

Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Line plot

Example 3

Plot the graph of an exponential function $y = \exp(x)$**, where** $x = 0,1,2,3,4,5$.

Data Visualization

- Functions [redacted], [redacted], and [redacted]
  - Line plot

Example 3

Plot the graph of an exponential function $y = \exp(x)$**, where** $x = 0,1,2,3,4,5$.

```
step = 1
x = np.arange(0, 5+step, step)        # An array 0, 1, 2, 3, 4, 5
y = np.exp(x)                         # An array of exp(0), exp(1), ..., exp(5)

plt.plot(x, y)                        # Specify the line plot by the x and y data
plt.xlabel('x')                       # Label for x data
plt.ylabel('y')                       # Label for y data
plt.show()                            # Ready for display (optional for Jupyter)
```

Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Line plot
    - x [redacted]

```
1  [redacted]
2                                  # An array 0, 1, 2, 3, 4, 5
3  y = np.exp(x)                   # An array of exp(0), exp(1), ..., exp(5)
4
5  plt.plot(x, y)                  # Specify the line plot by the x and y data
6  plt.xlabel('x')                 # Label for x data
7  plt.ylabel('y')                 # Label for y data
8  plt.show()                      # Ready for display (optional for Jupyter)
```

Data Visualization

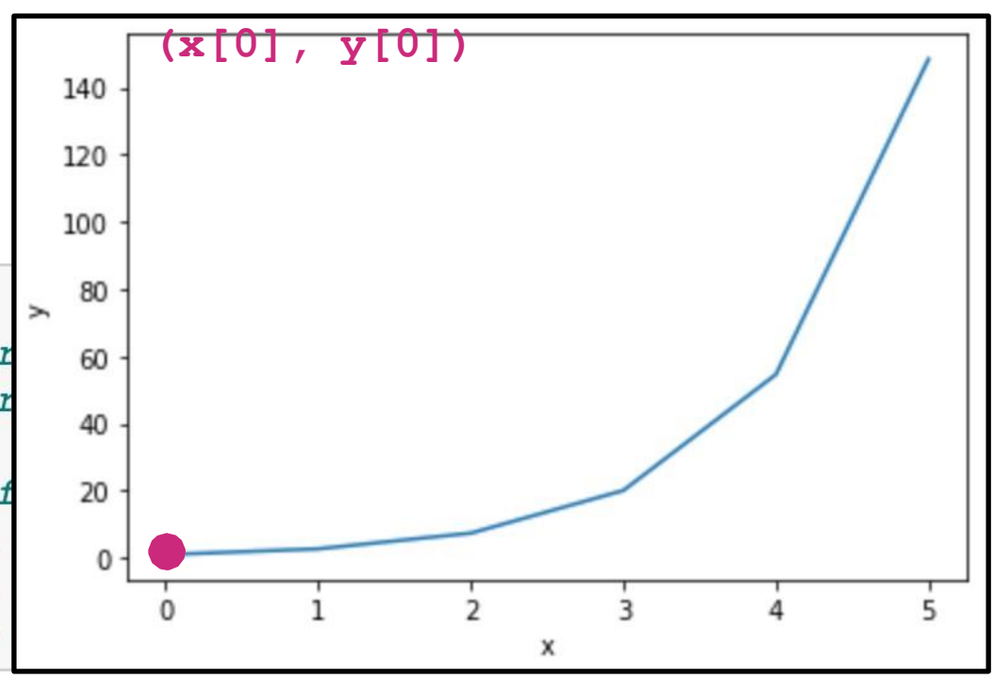
- Functions [redacted], [redacted], and [redacted]
)
  - Line plot
    - x [redacted]
    - y [redacted]

```
1  step = 1
2  x = np.arange(0, 5+step, step)        # An array 0, 1, 2, 3, 4, 5
3  [redacted]                            # An array of exp(0), exp(1), ..., exp(5)
4
5  plt.plot(x, y)                        # Specify the line plot by the x and y data
6  plt.xlabel('x')                       # Label for x data
7  plt.ylabel('y')                       # Label for y data
8  plt.show()                            # Ready for display (optional for Jupyter)
```

Data Visualization

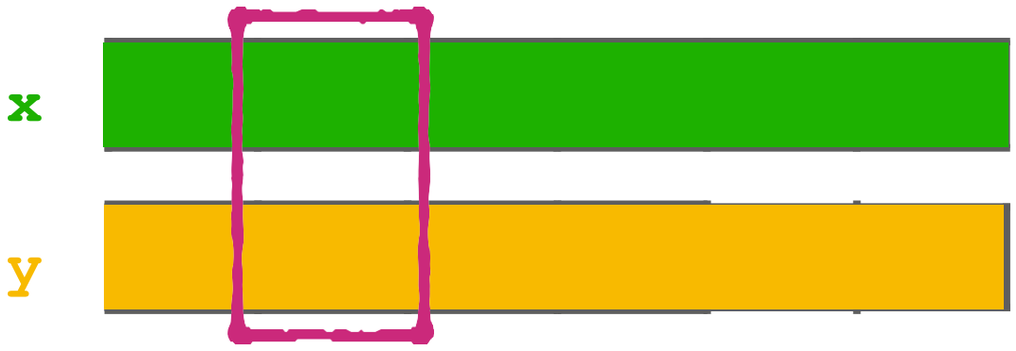
- Functions [redacted], [redacted], and [redacted]
)
  - Line plot


```
1  step = 1
2  x = np.arange(0, 5+step, step)        # An arr
3  y = np.exp(x)                         # An arr
4
5  [redacted]                            # Specif
6  plt.xlabel('x')                       # Label
7  plt.ylabel('y')                       # Label
8  plt.show()                            # Ready
```

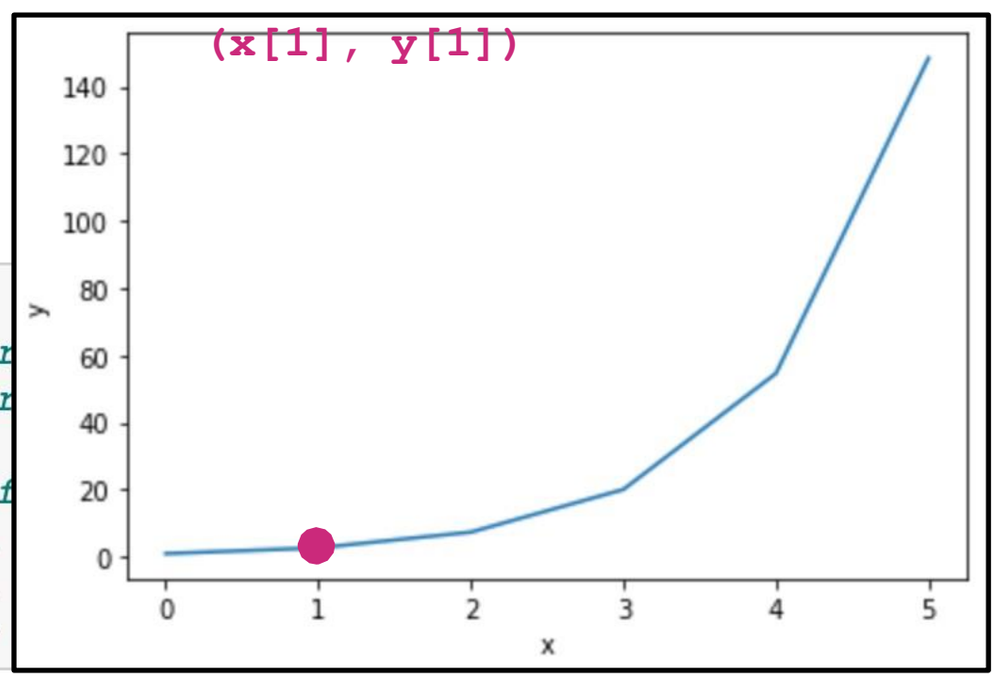


Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Line plot

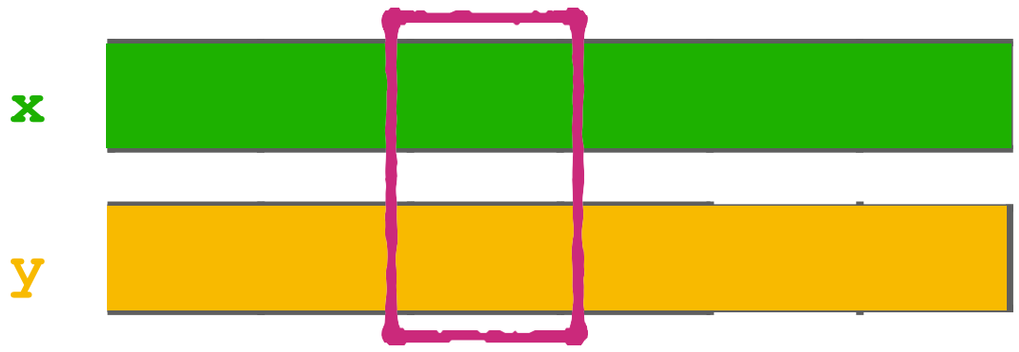



```
1  step = 1
2  x = np.arange(0, 5+step, step)        # An arr
3  y = np.exp(x)                         # An arr
4
5  [redacted]                            # Specif
6  plt.xlabel('x')                       # Label
7  plt.ylabel('y')                       # Label
8  plt.show()                            # Ready
```

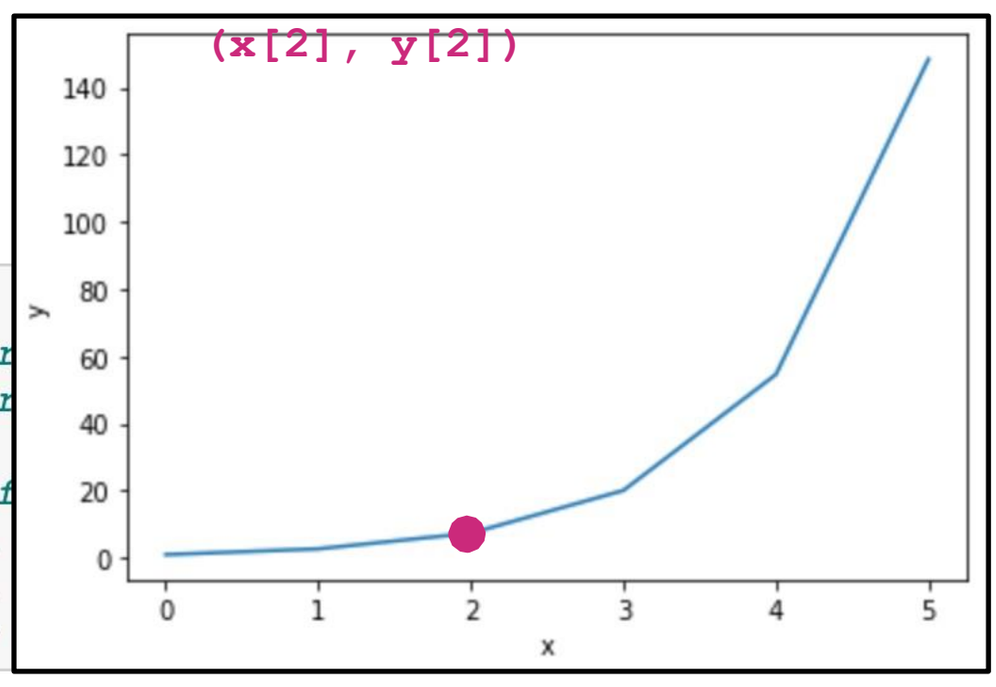


Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Line plot



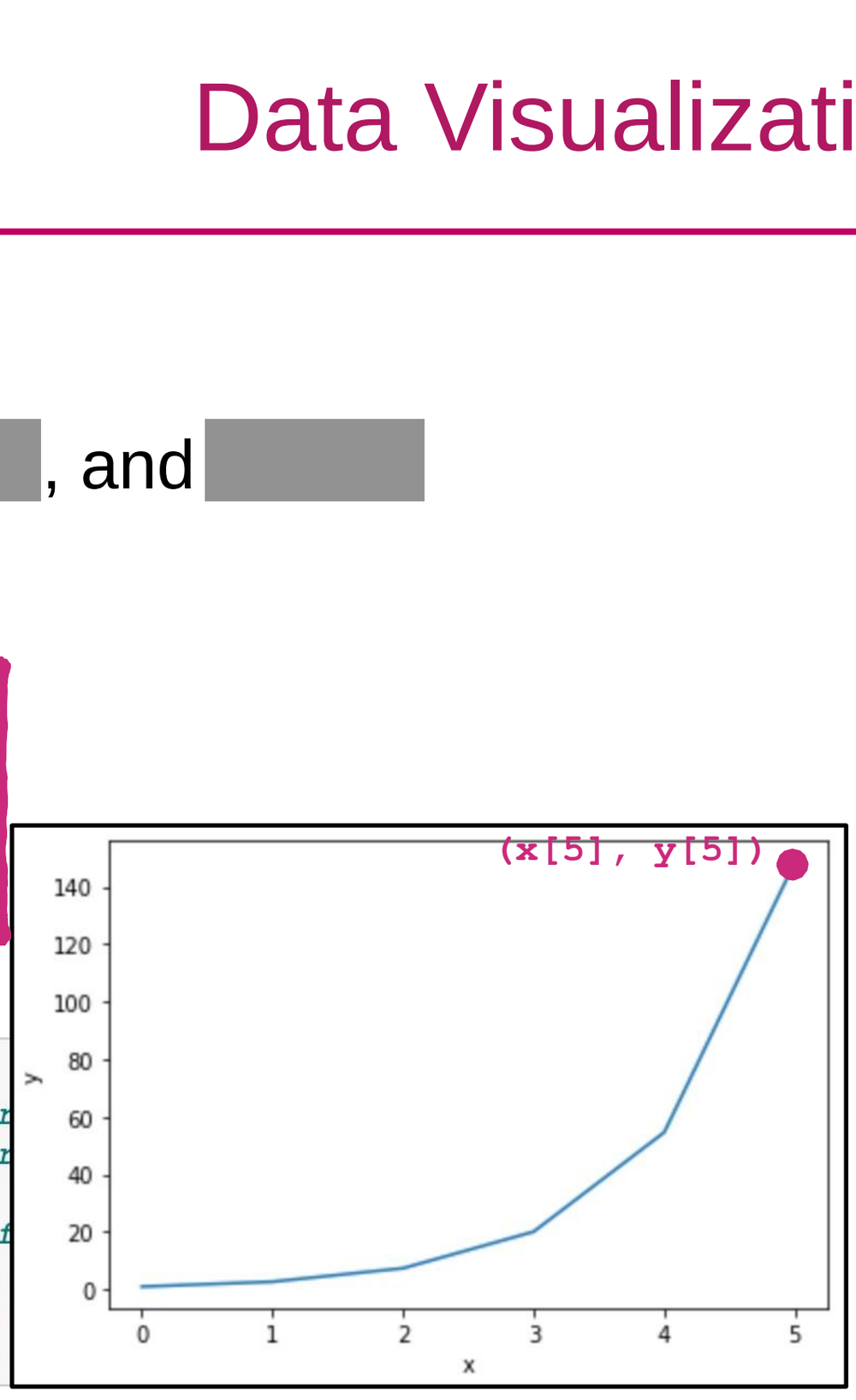

```
1  step = 1
2  x = np.arange(0, 5+step, step)        # An arr
3  y = np.exp(x)                         # An arr
4
5  [redacted]                            # Specif
6  plt.xlabel('x')                       # Label
7  plt.ylabel('y')                       # Label
8  plt.show()                            # Ready
```



Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Line plot

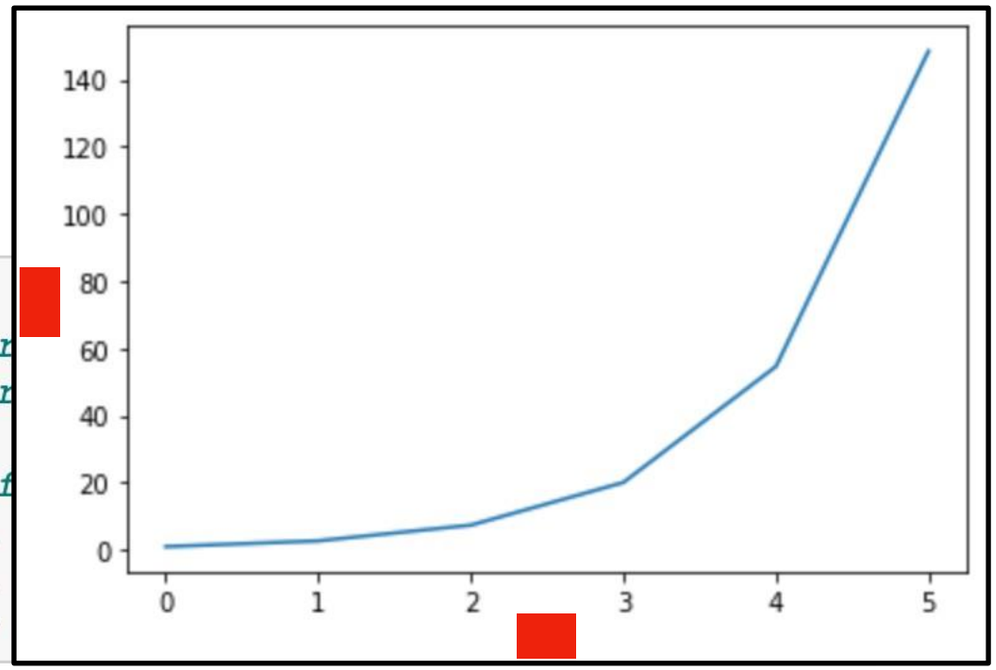

```
1  step = 1
2  x = np.arange(0, 5+step, step)        # An arr
3  y = np.exp(x)                         # An arr
4
5  [redacted]                            # Specif
6  plt.xlabel('x')                       # Label
7  plt.ylabel('y')                       # Label
8  plt.show()                            # Ready
```



Data Visualization

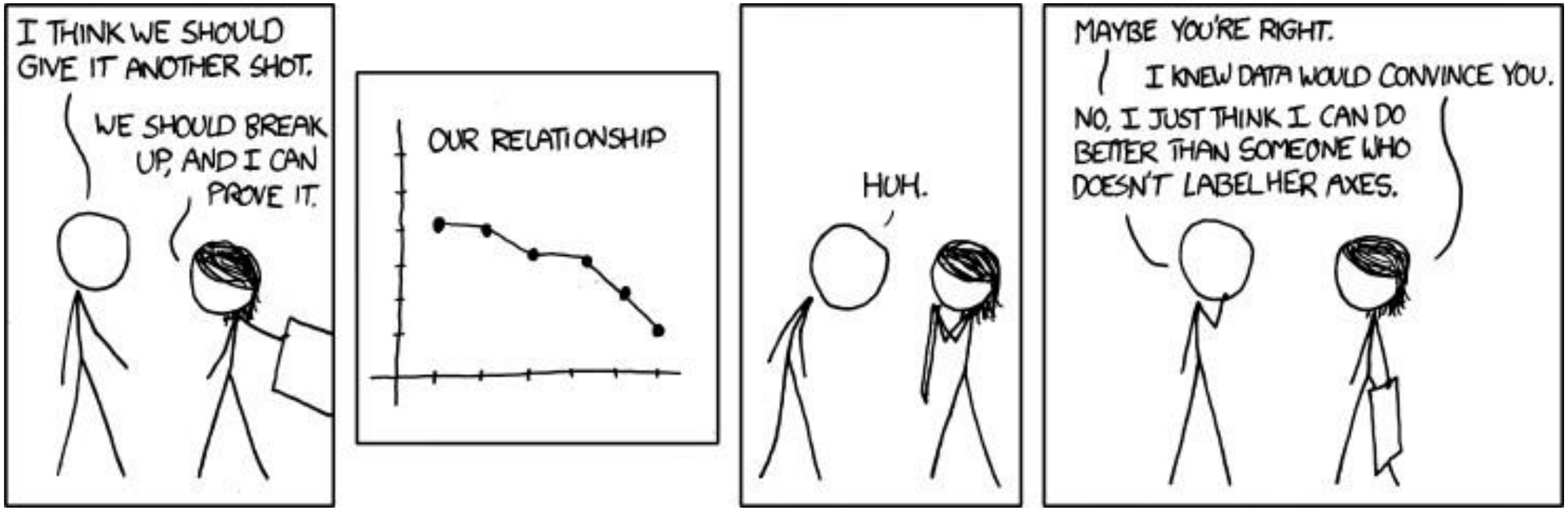
- Functions [redacted], [redacted], and [redacted]
)
  - Line plot
    - x [redacted]
    - y [redacted]

```
1  step = 1
2  x = np.arange(0, 5+step, step)        # An arr
3  y = np.exp(x)                         # An arr
4
5  plt.plot(x, y)                        # Specif
6  [redacted]                            # Label
7                                        # Label
8  plt.show()                            # Ready
```



Data Visualization

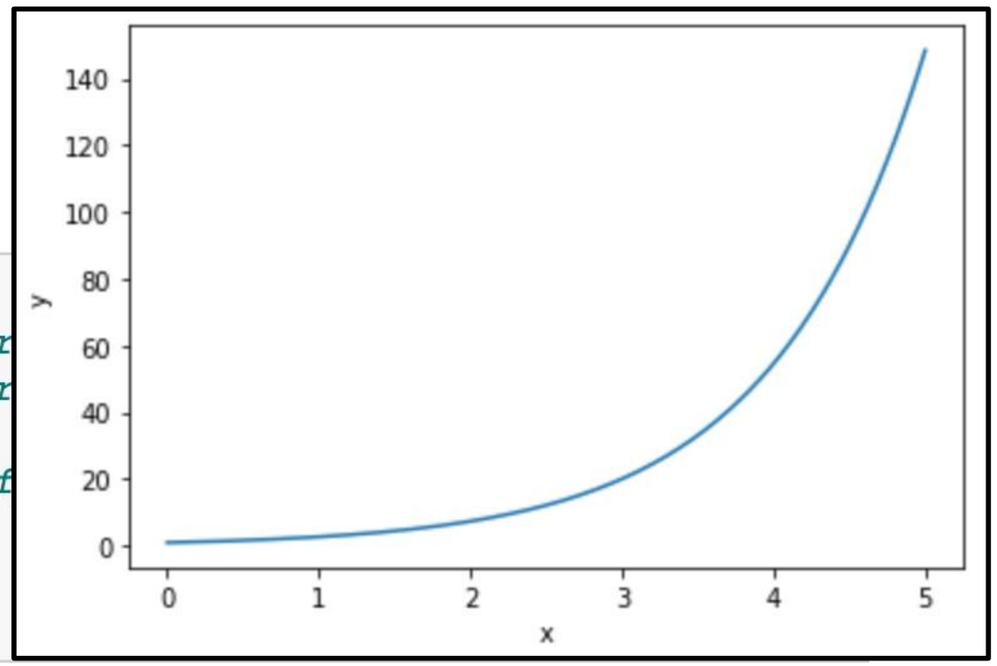
- Functions [redacted], [redacted], and [redacted]
)



Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Line plot
    - x [redacted]
    - y [redacted]

```
1  [redacted]
2  [redacted]                    # An arr
3  [redacted]                    # An arr
4
5  [redacted]                    # Specif
6  plt.xlabel('x')               # Label
7  plt.ylabel('y')               # Label
8  plt.show()                    # Ready
```

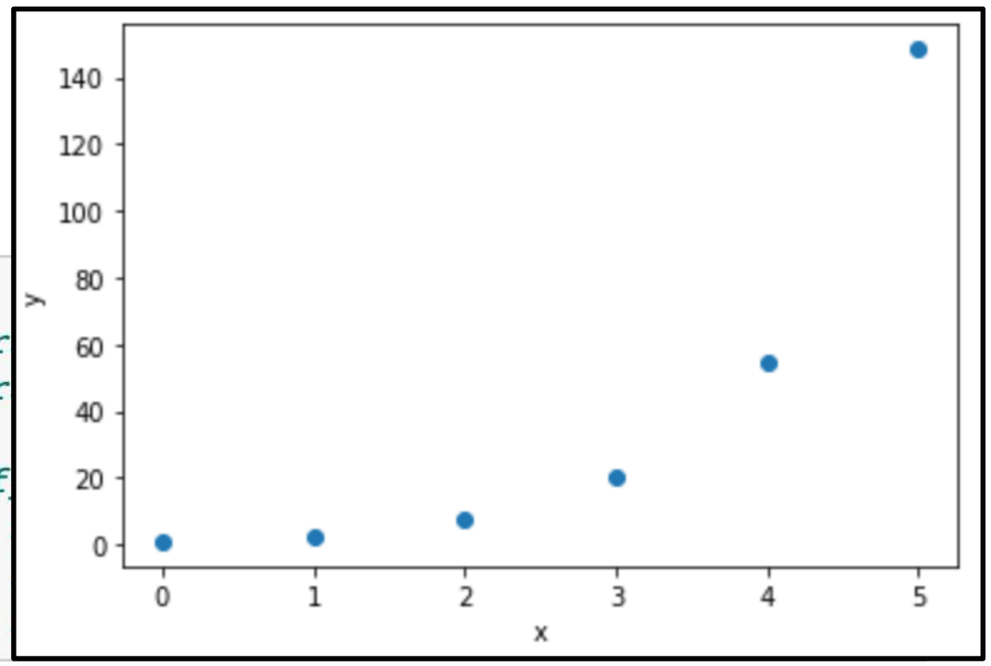


Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Scatter plot
    - x [redacted]

    y [redacted]

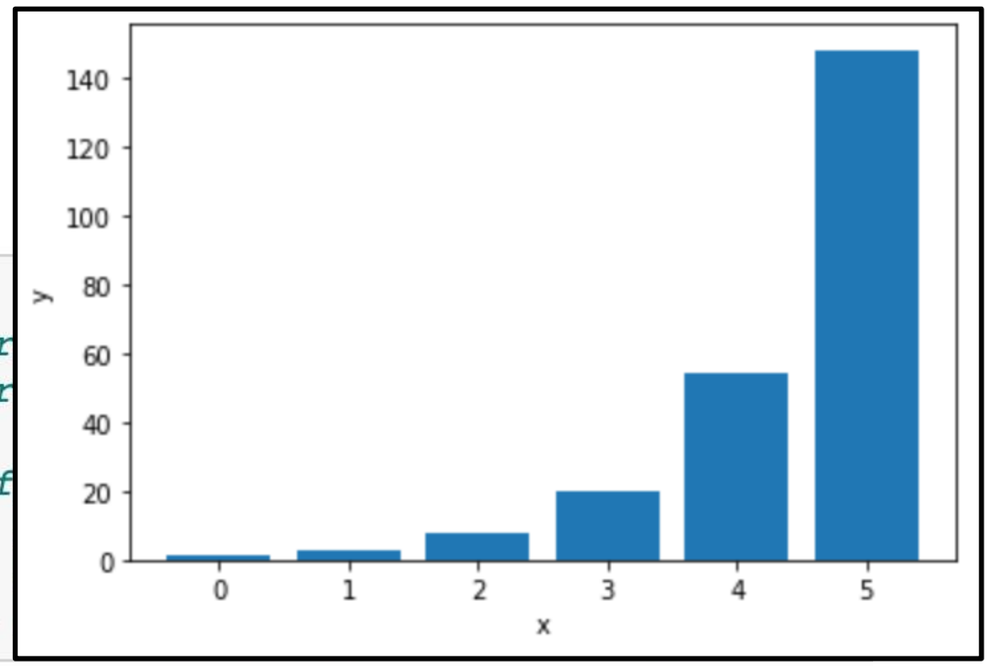
```
1  [redacted]
2  [redacted]                    # An arr
3  [redacted]                    # An arr
4
5  [redacted]                    # Specif
6  plt.xlabel('x')               # Label
7  plt.ylabel('y')               # Label
8  plt.show()                    # Ready
```



Data Visualization

- Functions [redacted], [redacted], and [redacted]
)
  - Bar graph
    - x [redacted]
    - y [redacted]

```
1  [redacted]
2                                    # An arr
3                                    # An arr
4
5                                    # Specif
6  plt.xlabel('x')                   # Label
7  plt.ylabel('y')                   # Label
8  plt.show()                        # Ready
```



Data Visualization

Question

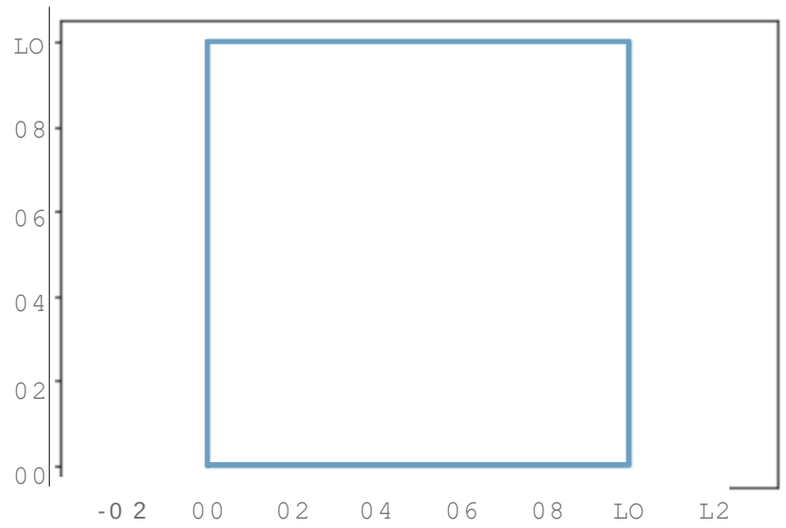
Which of the following code plots a square as the graph shows?

```
A.plt.plot([O, 0, 1, 1, O], [0' 1, 0, 1, O])
```

```
B. plt.plot([O, 1, 1, 0, O], [0' 0, 1, 1, 0])
```

```
C. plt.plot([O, 1, 1, 0]' [0' O, 1, 11)
```

```
D. plt.plot([O, 1, 1, O], [O, 1, 1, O1)
```

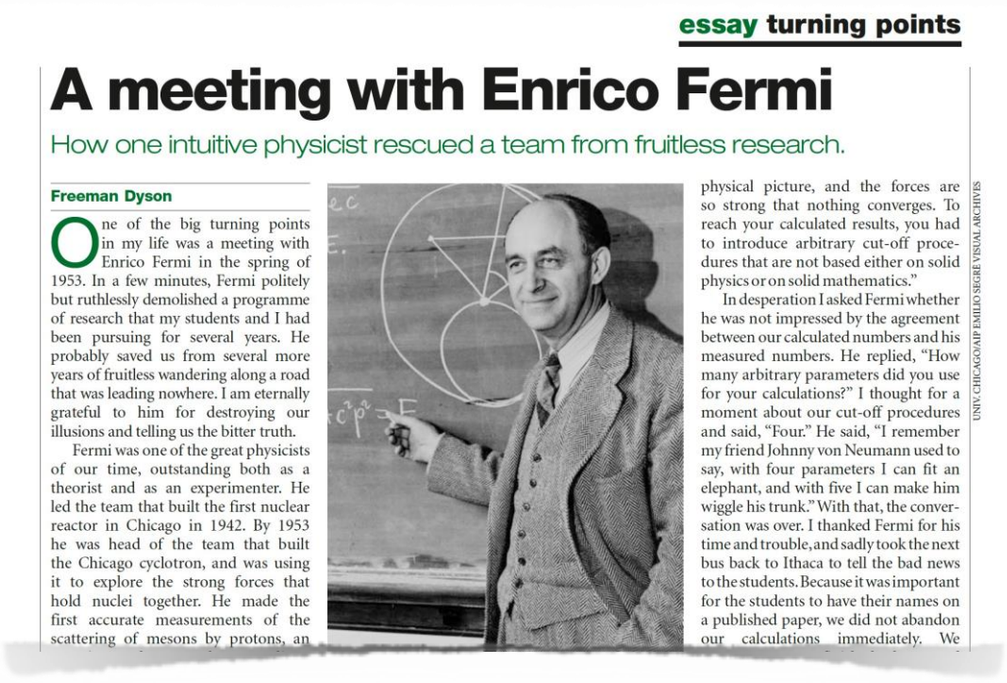


Data Visualization

- Functions [redacted], [redacted], and [redacted]

Example 2

Draw an “elephant” with four parameters.



essay turning points

A meeting with Enrico Fermi

How one intuitive physicist rescued a team from fruitless research.

Freeman Dyson

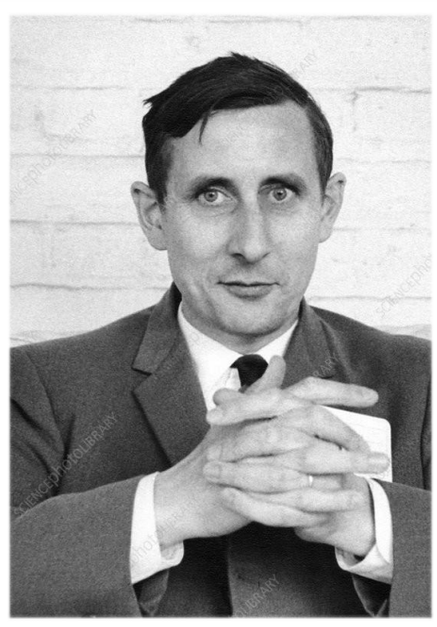
One of the big turning points in my life was a meeting with Enrico Fermi in the spring of 1953. In a few minutes, Fermi politely but ruthlessly demolished a programme of research that my students and I had been pursuing for several years. He probably saved us from several more years of fruitless wandering along a road that was leading nowhere. I am eternally grateful to him for destroying our illusions and telling us the bitter truth.

Fermi was one of the great physicists of our time, outstanding both as a theorist and as an experimenter. He led the team that built the first nuclear reactor in Chicago in 1942. By 1953 he was head of the team that built the Chicago cyclotron, and was using it to explore the strong forces that hold nuclei together. He made the first accurate measurements of the scattering of mesons by protons, an

UNIV. CHICAGO/AIP EMILIO SEGRÈ VISUAL ARCHIVES

physical picture, and the forces are so strong that nothing converges. To reach your calculated results, you had to introduce arbitrary cut-off procedures that are not based either on solid physics or on solid mathematics.”

In desperation I asked Fermi whether he was not impressed by the agreement between our calculated numbers and his measured numbers. He replied, “How many arbitrary parameters did you use for your calculations?” I thought for a moment about our cut-off procedures and said, “Four.” He said, “I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk.” With that, the conversation was over. I thanked Fermi for his time and trouble, and sadly took the next bus back to Ithaca to tell the bad news to the students. Because it was important for the students to have their names on a published paper, we did not abandon our calculations immediately. We



Enrico Fermi

Freeman Dyson

Data Visualization

- Functions [redacted], [redacted], and [redacted]
)

“With four parameters I can fit an elephant, and with five I can make him wiggle his trunk”

-John von Neumann

Data Visualization

- Functions [redacted], [redacted], and [redacted]
)

Drawing an elephant with four complex parameters

American Journal of Physics **78**, 648 (2010); https://doi.org/10.1119/1.3254017

Jürgen Mayer
- Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstr. 108, 01307 Dresden, Germany

Khaled Khairy
- European Molecular Biology Laboratory, Meyerhofstraße. 1, 69117 Heidelberg, Germany

Jonathon Howard
- Max Planck Institute of Molecular Cell Biology and Genetics, Pfotenhauerstr. 108, 01307 Dresden, Germany

Data Visualization

- Functions [redacted], [redacted], and [redacted]
)

Drawing an elephant with four complex parameters

American Journal of Physics **78**, 648 (2010); https://doi.org/10.1119/1.3254017

$$\begin{cases} x = -30\sin(t) + 8\sin(2t) - 10\sin(3t) - 60\cos(t) \\ y = -50\sin(t) - 18\sin(2t) - 12\cos(3t) + 14\cos(5t) \end{cases}$$

where $t \in [0, 2\pi]$**.**

Data Visualization

- Functions [redacted], [redacted], and [redacted]

)

$$x = -30\sin(t) + 8\sin(2t) - 10\sin(3t) - 60\cos(t)$$
$$\{y = -50\sin(t) - 18\sin(2t) - 12\cos(3t) + 14\cos(5t)$$

where $t \in [0, 2\pi]$**.**

```
pi = np.pi
[redacted]

x = -30*np.sin(t) + 8*np.sin(2*t) - 10*np.sin(3*t) - 60*np.cos(t)
y = - 50*np.sin(t) - 18*np.sin(2*t) - 12*np.cos(3*t) + 14*np.cos(5*t)

plt.plot(x, y)           # Plot the curve as the elephant
plt.scatter(20, 20)      # Plot the eye of the elepant as a scatter point

plt.show()
```

An array running from 0 to 2π

Data Visualization

- Functions [redacted], [redacted], and [redacted]

[redacted]

$$\{y = -50\sin(t) - 18\sin(2t) - 12\cos(3t) + 14\cos(5t)$$

where $t \in [0, 2\pi]$**.**

```
pi = np.pi
t = np.arange(0, 2*pi+0.01, 0.01)

[redacted]
y = - 50*np.sin(t) - 18*np.sin(2*t) - 12*np.cos(3*t) + 14*np.cos(5*t)

plt.plot(x, y)           # Plot the curve as the elephant
plt.scatter(20, 20)      # Plot the eye of the elepant as a scatter point

plt.show()
```

Data Visualization

- Functions [redacted], [redacted], and [redacted]

)

$$x = -30\sin(t) + 8\sin(2t) - 10\sin(3t) - 60\cos(t)$$

{y [redacted]

where $t \in [0, 2\pi]$**.**

```
pi = np.pi
t = np.arange(0, 2*pi+0.01, 0.01)

x = -30*np.sin(t) + 8*np.sin(2*t) - 10*np.sin(3*t) - 60*np.cos(t)
[redacted]

plt.plot(x, y)           # Plot the curve as the elephant
plt.scatter(20, 20)      # Plot the eye of the elepant as a scatter point

plt.show()
```

Data Visualization

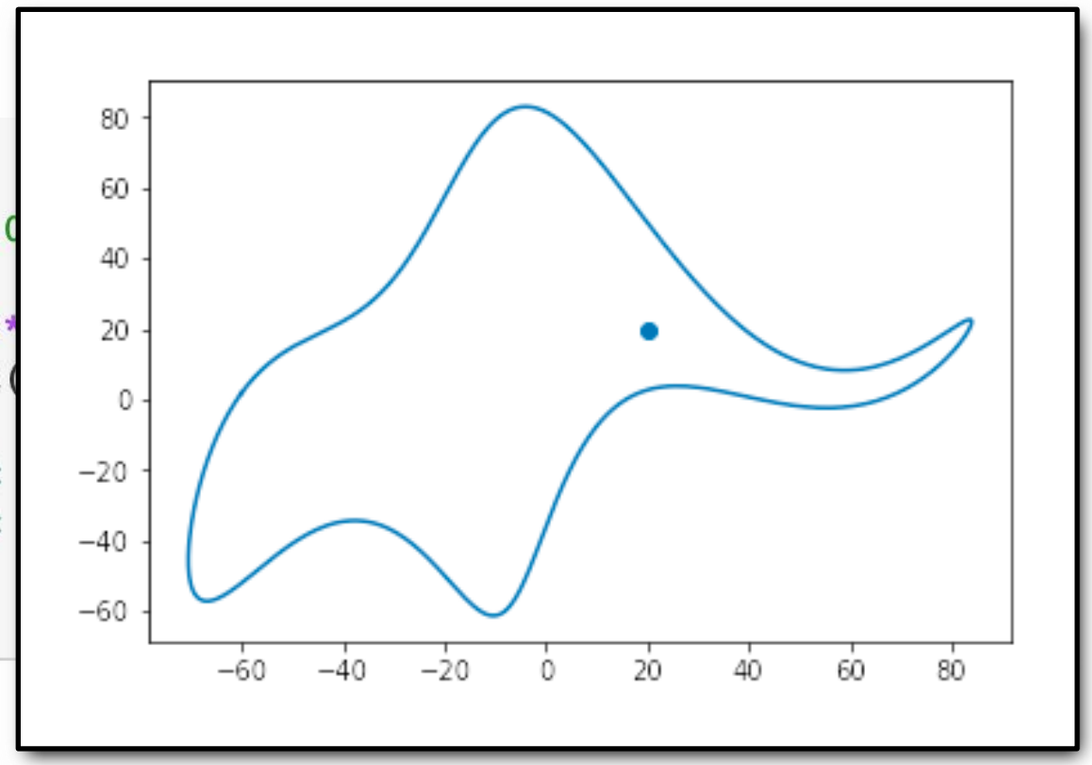
- Functions [redacted], [redacted], and [redacted]

)

$$x = -30\sin(t) + 8\sin(2t) - 10\sin(3t) - 60\cos(t)$$
$$\{y = -50\sin(t) - 18\sin(2t) - 12\cos(3t) + 14\cos(5t)$$

where $t \in [0, 2\pi]$**.**

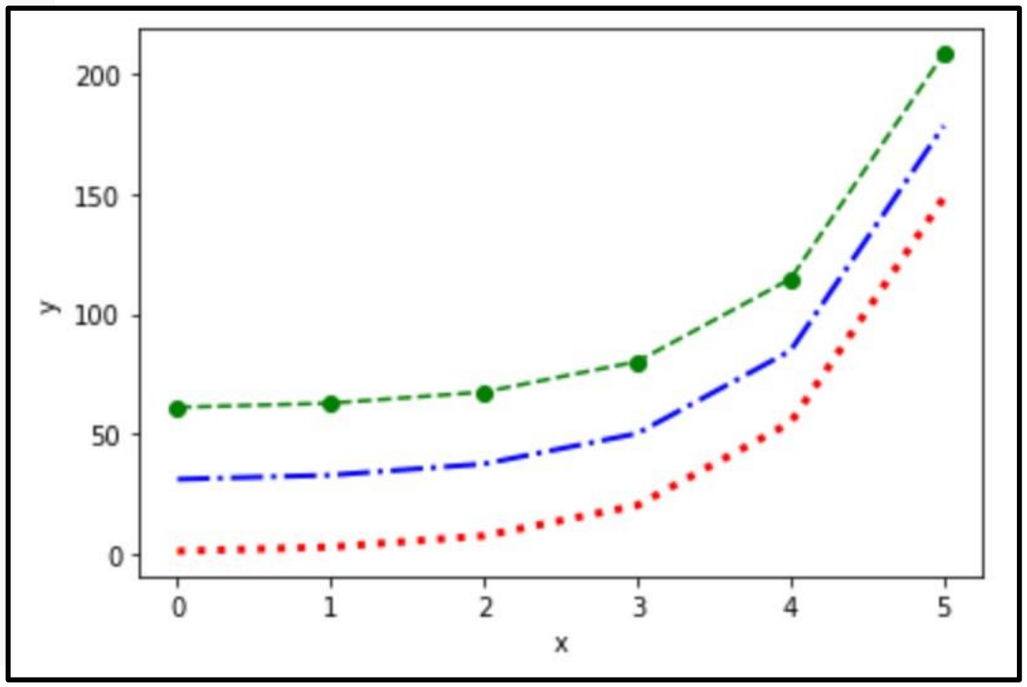
```
pi = np.pi
t = np.arange(0, 2*pi+0.01, 0.0

x = -30*np.sin(t) + 8*np.sin(2*
y = - 50*np.sin(t) - 18*np.sin(

[redacted]          # Plot
                    # Plot

plt.show()
```



Data Visualization

- Configuration of plots and figures
  - Specify the line plot by keyword arguments

```
5  plt.plot(x, y,
6           linewidth=3,
7           linestyle=':',
8           color='r')
9
10 plt.plot(x, y+30,
11          linewidth=2,
12          linestyle='-.',
13          color='b')
14
15 plt.plot(x, y+60,
16          marker='o',
17          linestyle='--',
18          color='g')
19
20 plt.xlabel('x')
21 plt.ylabel('y')
22 plt.show()
```

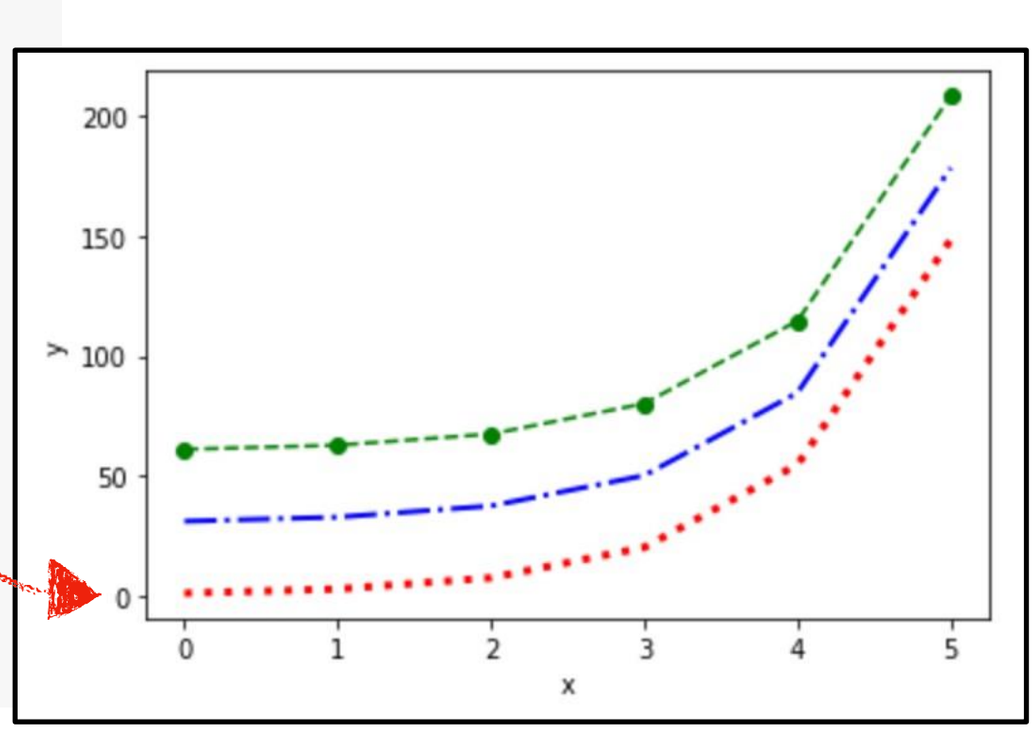


Data Visualization

- Configuration of plots and figures
  - Specify the line plot by keyword arguments

```
5
6
7
8
9
10  plt.plot(x, y+30,
11           linewidth=2,
12           linestyle='-.',
13           color='b')
14
15  plt.plot(x, y+60,
16           marker='o',
17           linestyle='--',
18           color='g')
19
20  plt.xlabel('x')
21  plt.ylabel('y')
22  plt.show()
```

Line width is 3 Dotted line Line color is red

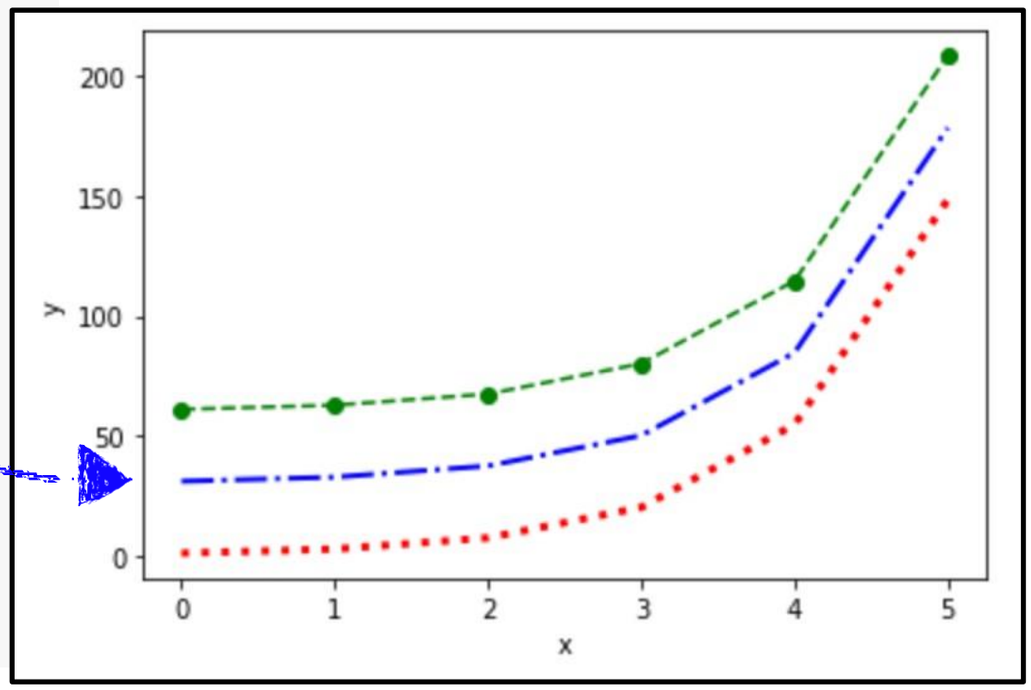


Data Visualization

- Configuration of plots and figures
  - Specify the line plot by keyword arguments

```
5  plt.plot(x, y,
6           linewidth=3,
7           linestyle=':',
8           color='r')
9
10
11
12
13
14
15 plt.plot(x, y+60,
16          marker='o',
17          linestyle='--',
18          color='g')
19
20 plt.xlabel('x')
21 plt.ylabel('y')
22 plt.show()
```

Line width is 2
Dash-dotted line
Line color is blue

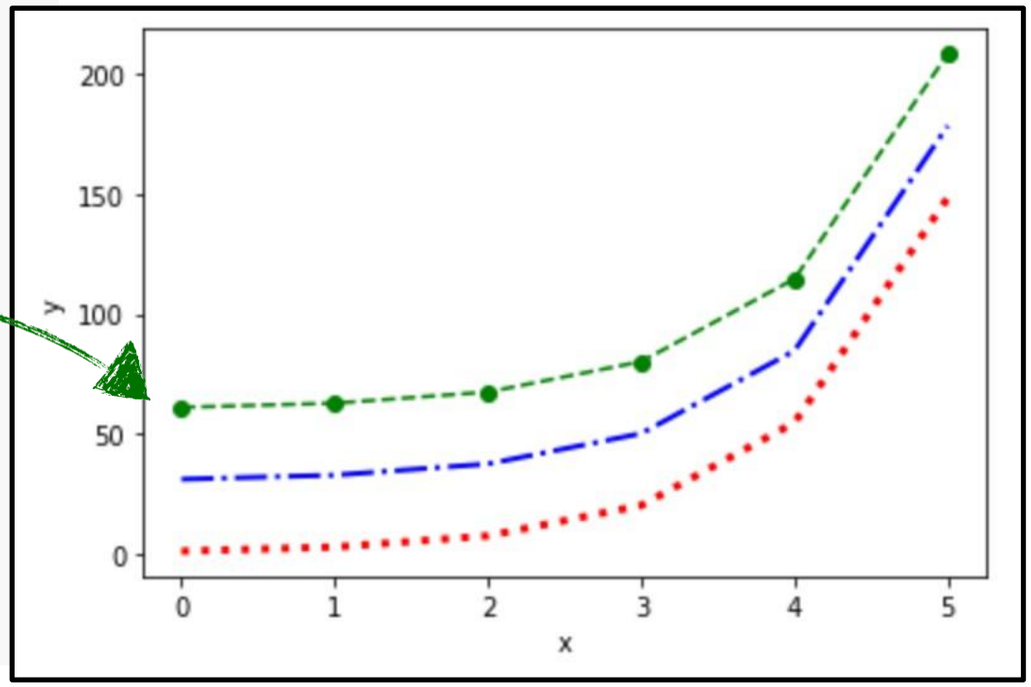


Data Visualization

- Configuration of plots and figures
  - Specify the line plot by keyword arguments

```
5  plt.plot(x, y,
6           linewidth=3,
7           linestyle=':',
8           color='r')
9
10 plt.plot(x, y+30,
11          linewidth=2,
12          linestyle='-.',
13          color='b')
14
15
16
17
18
19
20 plt.xlabel('x')
21 plt.ylabel('y')
22 plt.show()
```

Circle marker
Dashed line
Line color is green

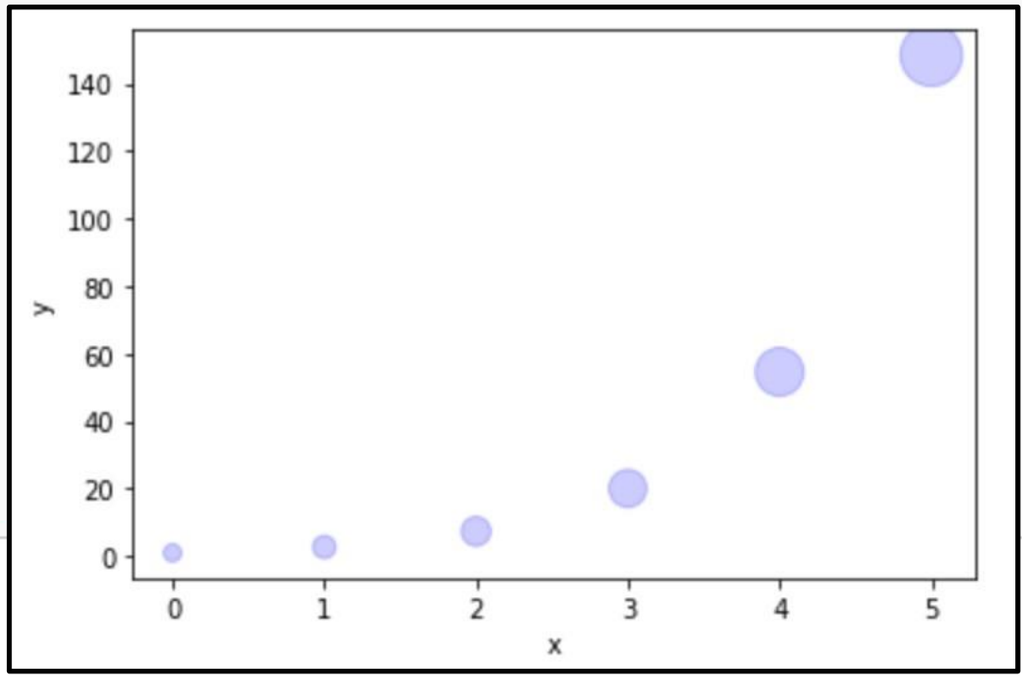


Data Visualization

- Configuration of plots and figures
  - Specify the scatter plot by keyword arguments

```
step = 1
x = np.arange(0, 5+step, step)
y = np.exp(x)
z = np.sqrt(y)

plt.scatter(x, y,
            s=z*50,
            c='b',
            alpha=0.2)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

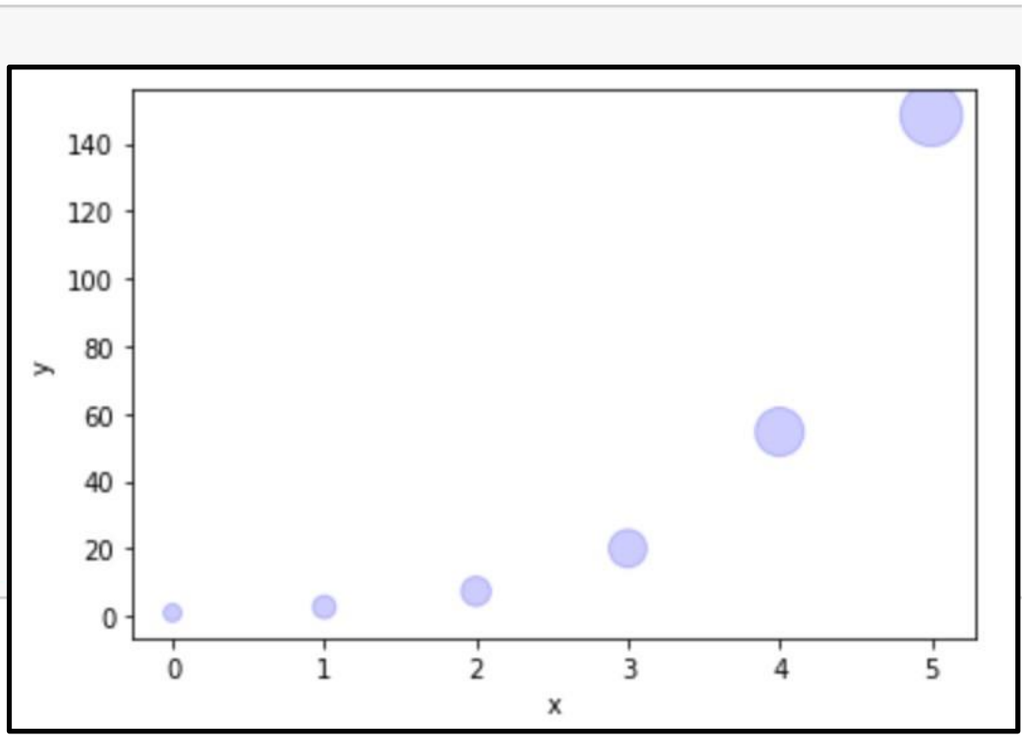
Size of scatter dots



Data Visualization

- Configuration of plots and figures
  - Specify the scatter plot by keyword arguments

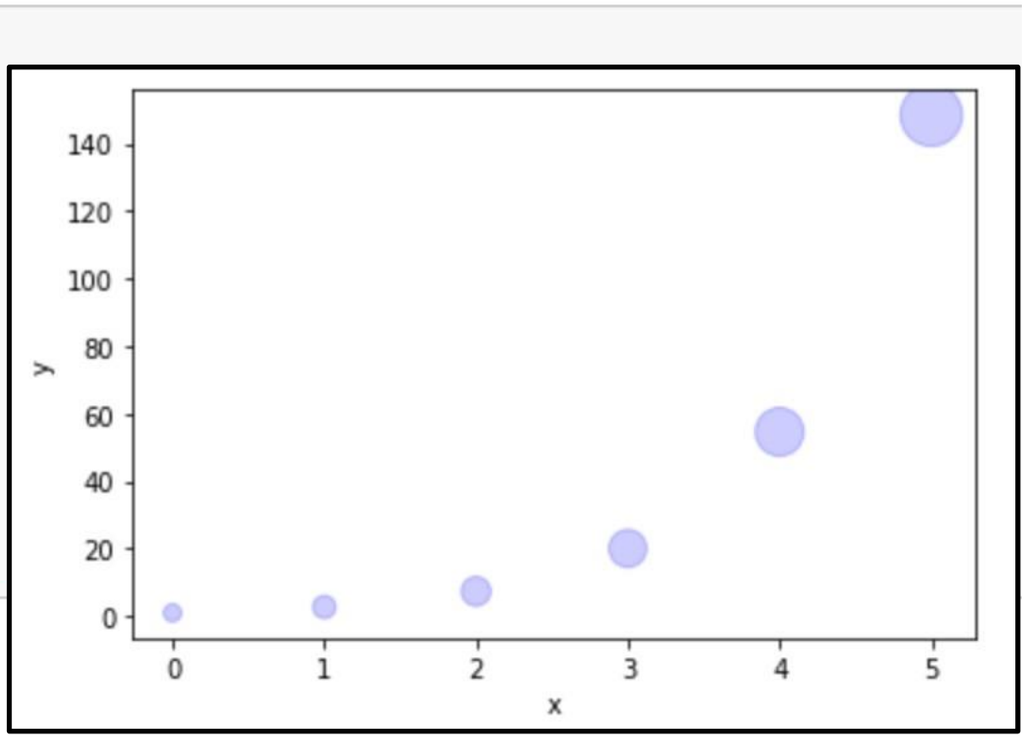
```
step = 1
x = np.arange(0, 5+step, step)
y = np.exp(x)
z = np.sqrt(y)

plt.scatter(x, y,
            s=z*50,
            [redacted], Color of scatter dots
            alpha=0.2)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Data Visualization

- Configuration of plots and figures
  - Specify the scatter plot by keyword arguments

```
step = 1
x = np.arange(0, 5+step, step)
y = np.exp(x)
z = np.sqrt(y)

plt.scatter(x, y,
            s=z*50,
            c='b',
            [redacted]) Opacity of dots
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

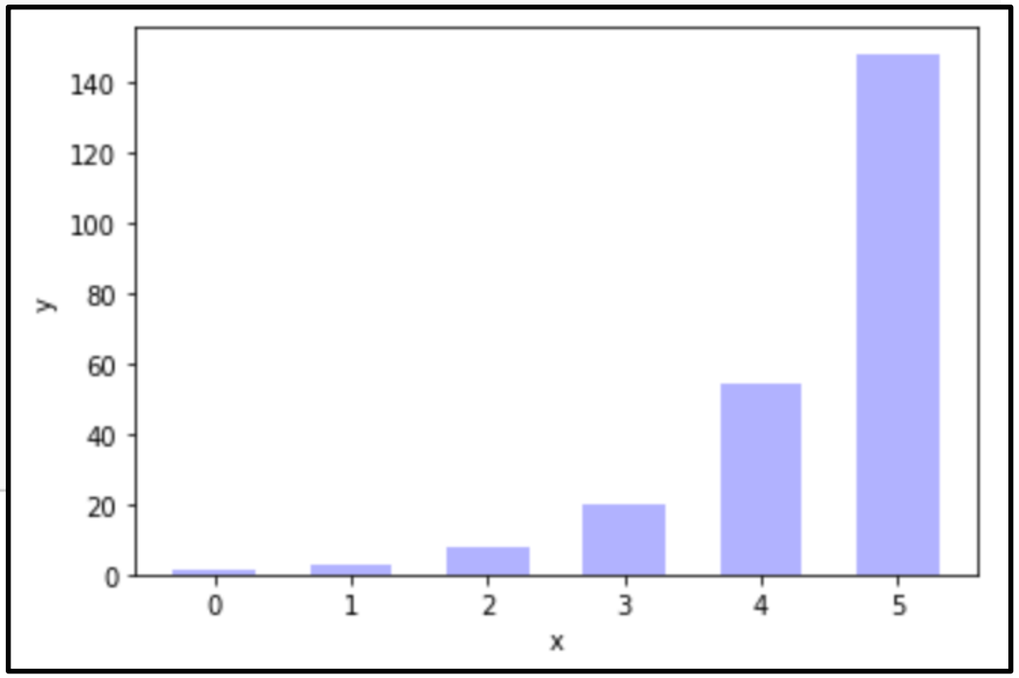


Data Visualization

- Configuration of plots and figures
  - Specify the bar graph by keyword arguments

```
step = 1
x = np.arange(0, 5+step, step)
y = np.exp(x)

plt.bar(x, y,
        [redacted]
        color='b',
        alpha=0.3)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Bar width is 0.6

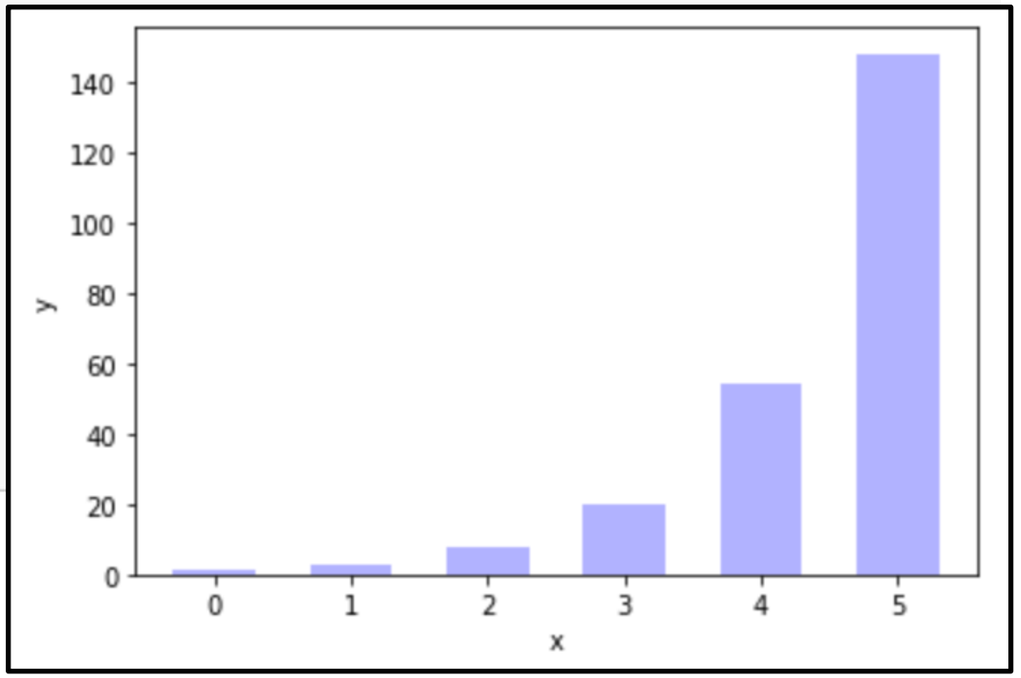


Data Visualization

- Configuration of plots and figures
  - Specify the bar graph by keyword arguments

```
step = 1
x = np.arange(0, 5+step, step)
y = np.exp(x)

plt.bar(x, y,
        width=0.6,
        [redacted]
        alpha=0.3)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Color of bar graph

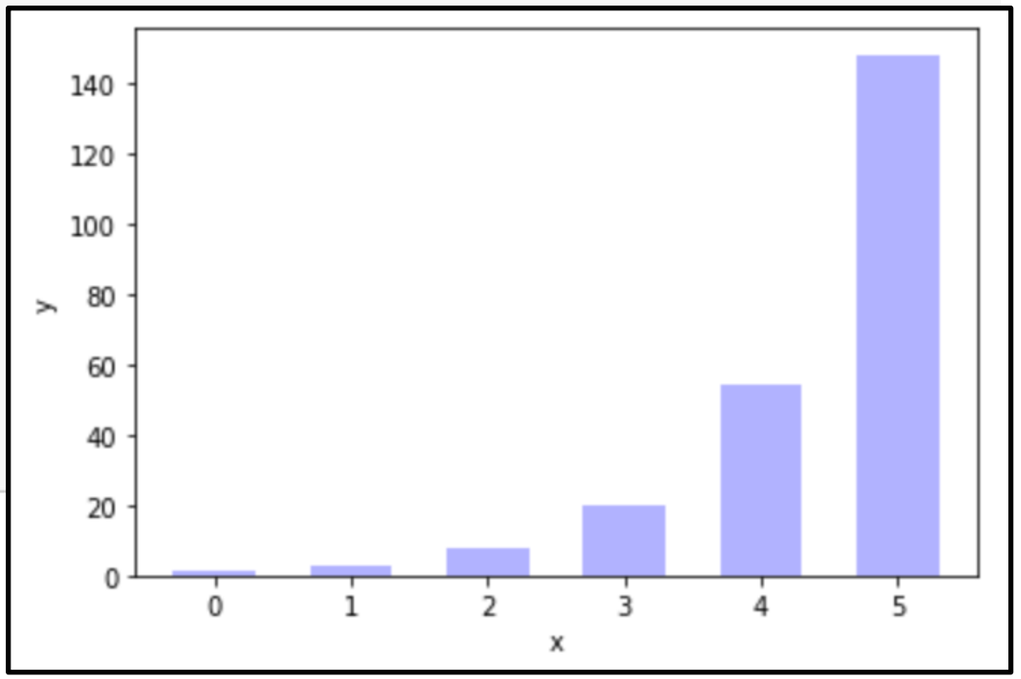


Data Visualization

- Configuration of plots and figures
  - Specify the bar graph by keyword arguments

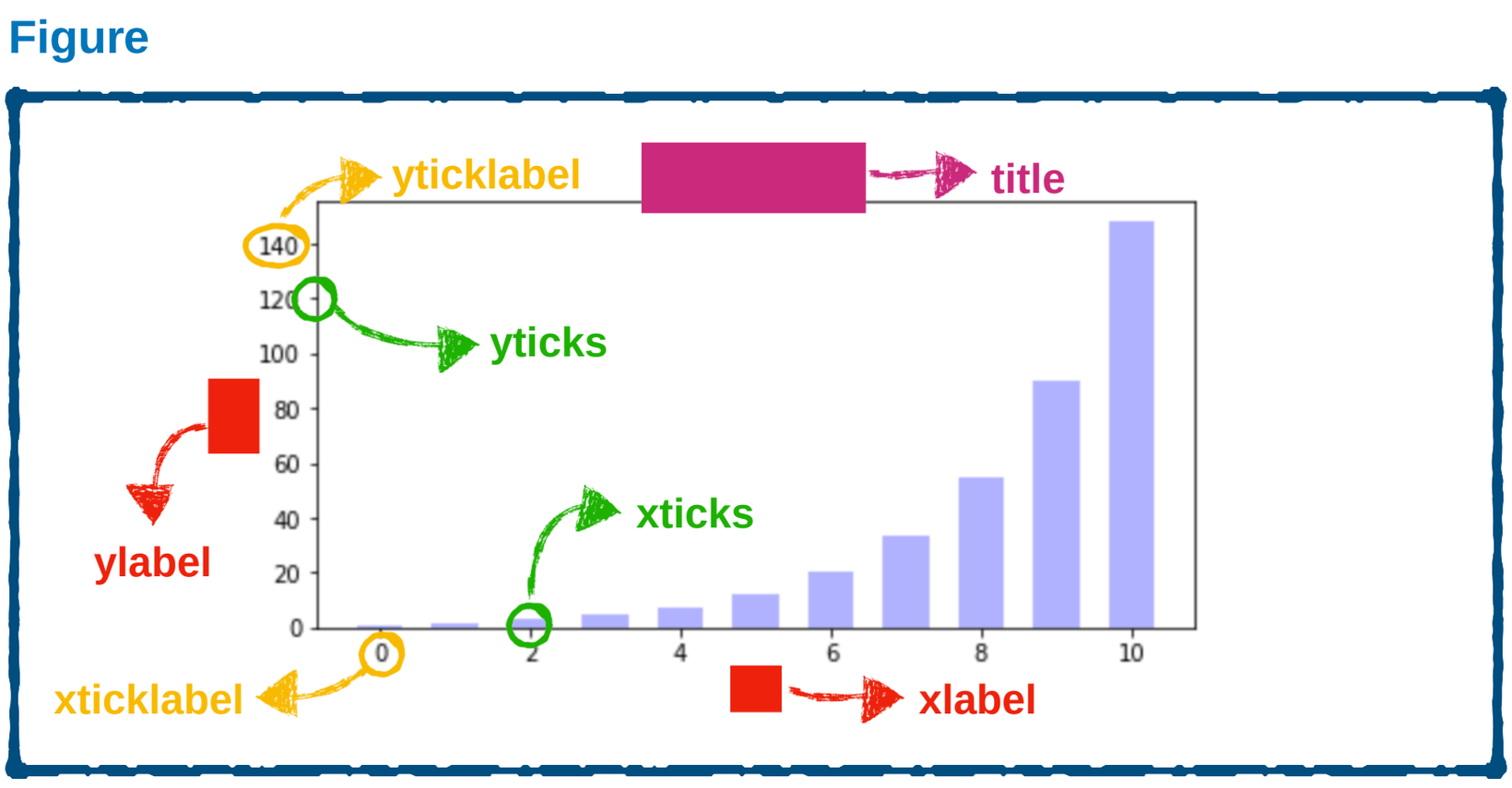
```
step = 1
x = np.arange(0, 5+step, step)
y = np.exp(x)

plt.bar(x, y,
        width=0.6,
        color='b',
        alpha=0.3)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Opacity of bar graph



Data Visualization

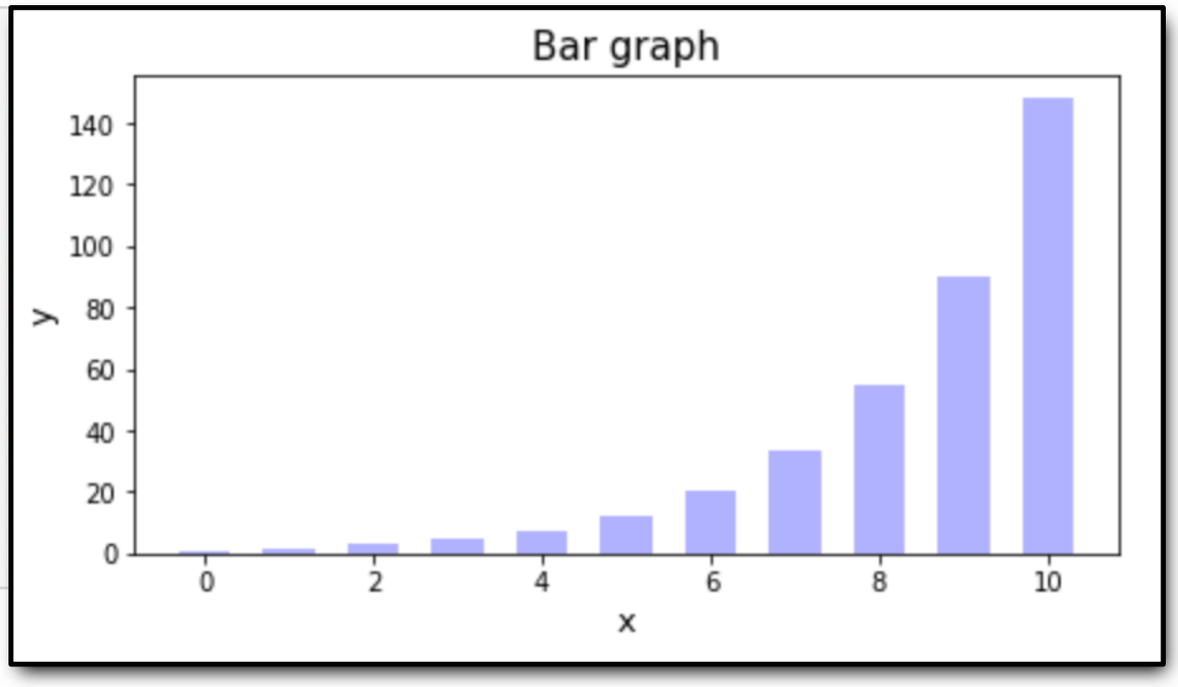
- Configuration of plots and figures
  - Components of a figure



Data Visualization

- Configuration of plots and figures
  - Attributes of a figure

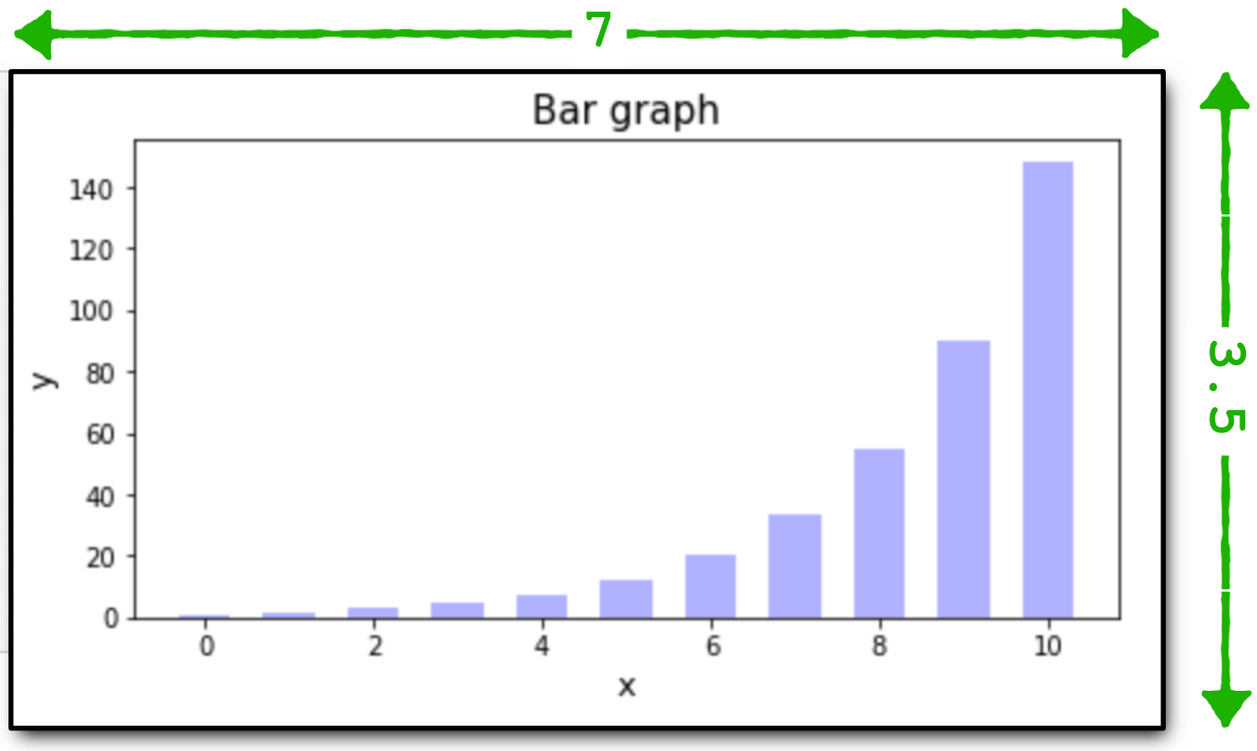
```
step = 1
x = np.arange(0, 10+step, step)
y = np.exp(x/2)

plt.figure(figsize=(7, 3.5))
plt.title('Bar graph', fontsize=15)
plt.bar(x, y,
        width=0.6,
        color='b',
        alpha=0.3)
plt.xlabel('x', fontsize=13)
plt.ylabel('y', fontsize=13)
plt.show()
```



Data Visualization

- Configuration of plots and figures
  - Attributes of a figure

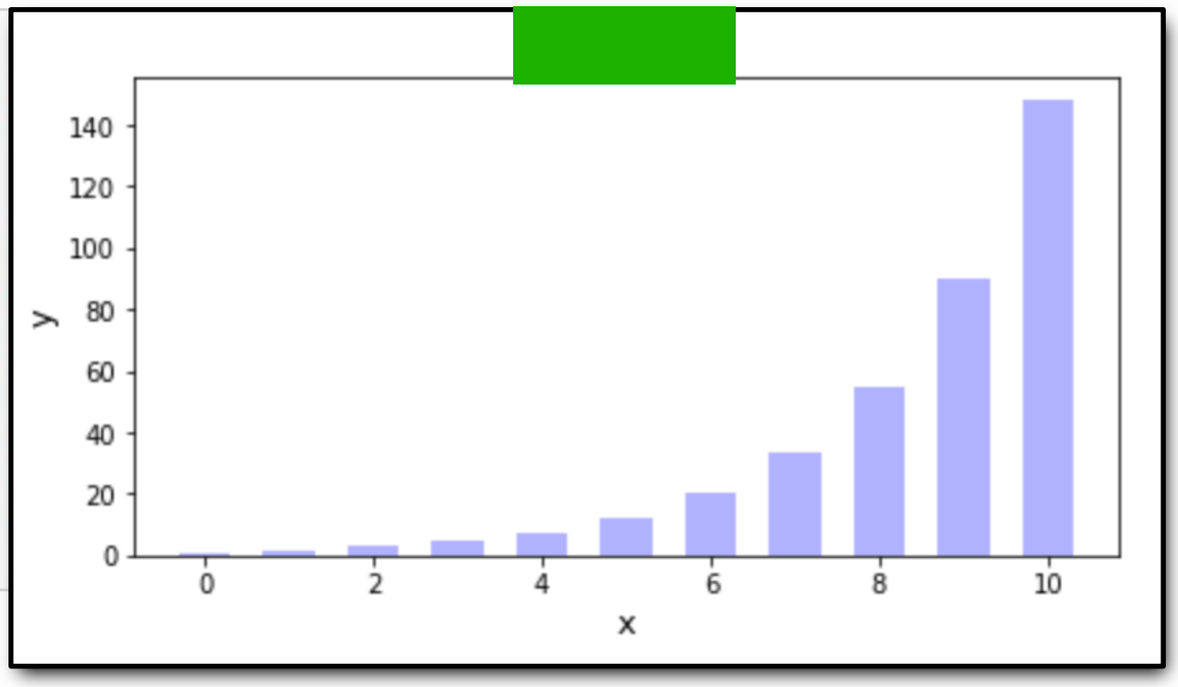
```
1  step = 1
2  x = np.arange(0, 10+step, step)
3  y = np.exp(x/2)
4
5
6  plt.title('Bar graph', fontsize=15)
7  plt.bar(x, y,
8          width=0.6,
9          color='b',
10         alpha=0.3)
11 plt.xlabel('x', fontsize=13)
12 plt.ylabel('y', fontsize=13)
13 plt.show()
```



Data Visualization

- Configuration of plots and figures
  - Attributes of a figure

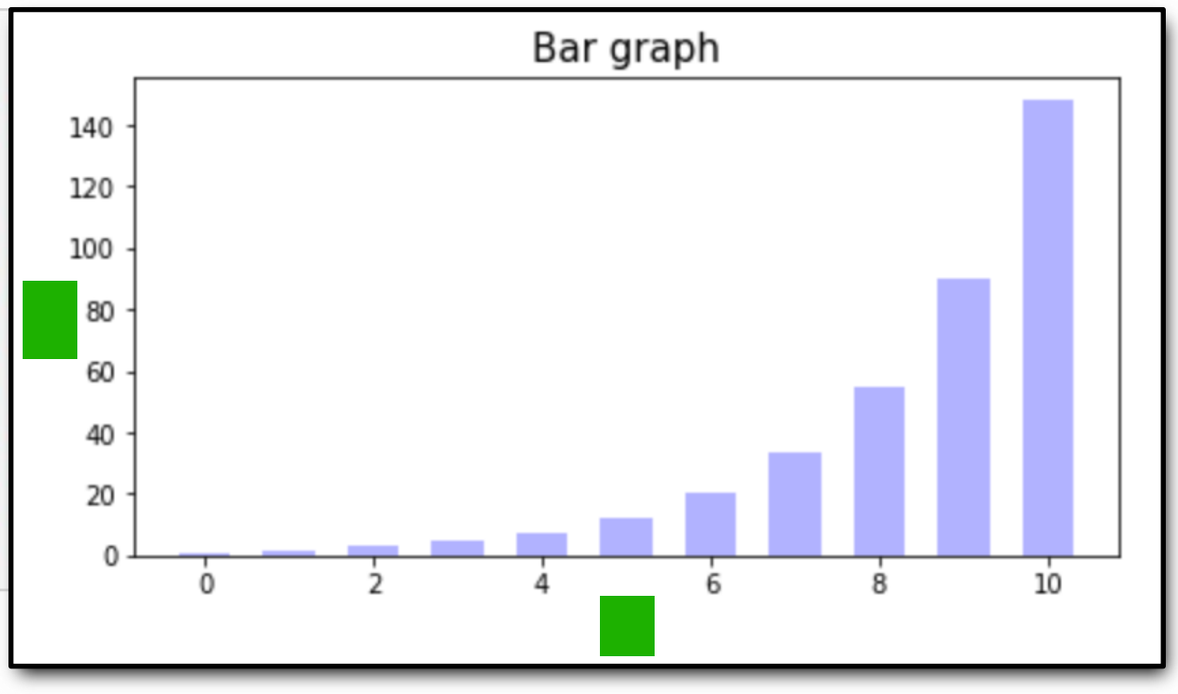
```
step = 1
x = np.arange(0, 10+step, step)
y = np.exp(x/2)

plt.figure(figsize=(7, 3.5))

plt.bar(x, y,
        width=0.6,
        color='b',
        alpha=0.3)
plt.xlabel('x', fontsize=13)
plt.ylabel('y', fontsize=13)
plt.show()
```



Data Visualization

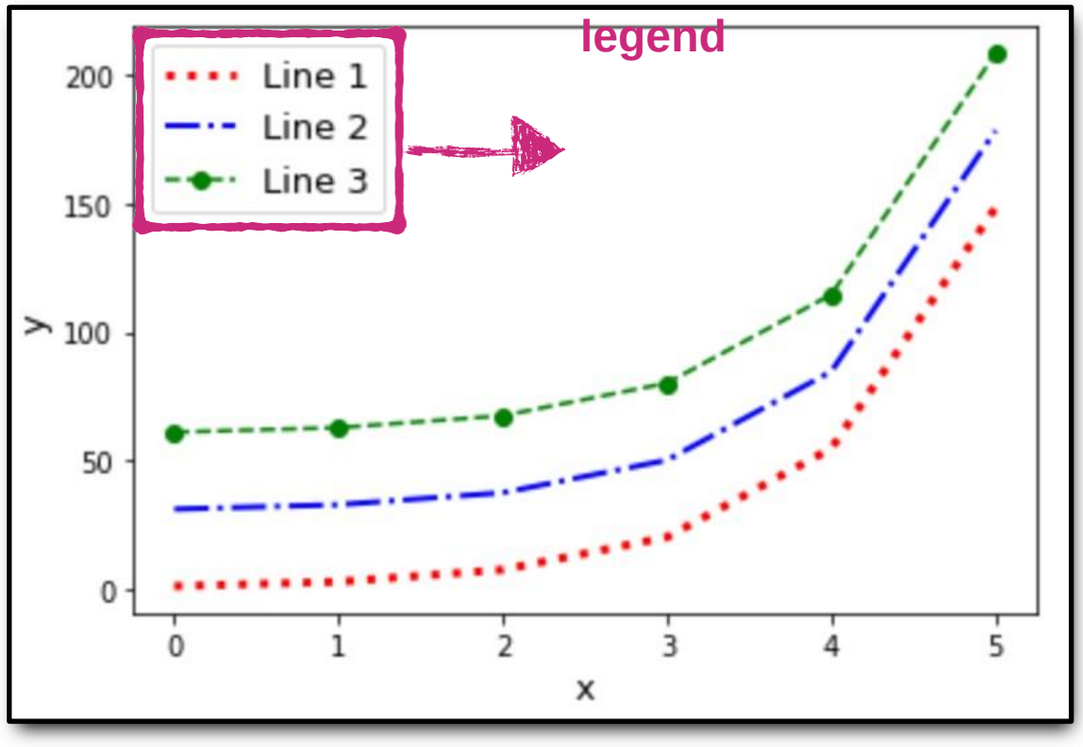
- Configuration of plots and figures
  - Attributes of a figure

```
step = 1
x = np.arange(0, 10+step, step)
y = np.exp(x/2)

plt.figure(figsize=(7, 3.5))
plt.title('Bar graph', fontsize=15)
plt.bar(x, y,
        width=0.6,
        color='b',
        alpha=0.3)
[redacted]
plt.show()
```



Data Visualization

- Labeling plots and the function [redacted]
)

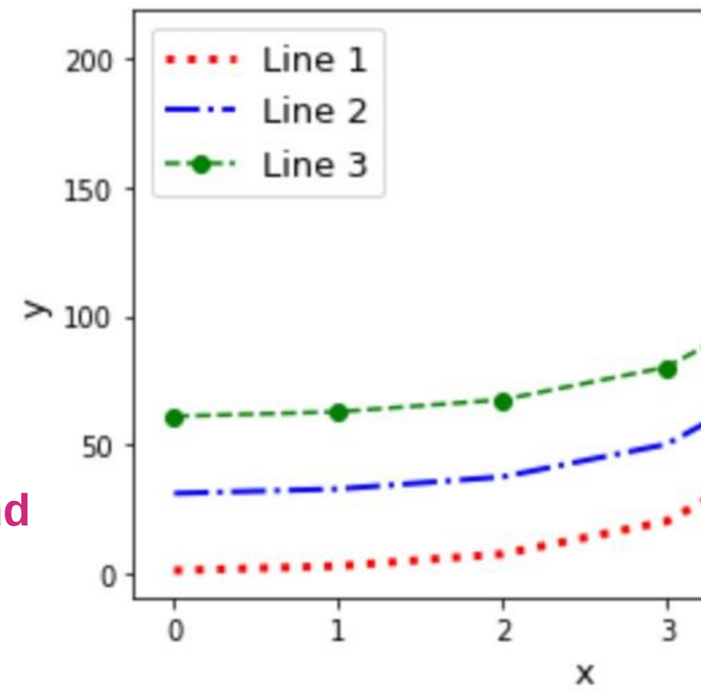


Data Visualization

- Labeling plots and the function [redacted])
  - Connection between plots and labels

```
1  step = 1
2  x = np.arange(0, 5+step, step)
3  y = np.exp(x)
4
5  plt.plot(x, y, linewidth=3, linestyle=':', color='r',
6           label='Line 1')
7
8  plt.plot(x, y+30, linewidth=2, linestyle='-.', color='b',
9           label='Line 2')
10
11 plt.plot(x, y+60, marker='o', linestyle='--', color='g',
12          Label='Line 3')
13
14 plt.legend(fontsize=13)
15 plt.xlabel('x', fontsize=13)
16 plt.ylabel('y', fontsize=13)
17 plt.show()
```

legend



Example 5

The discrete distribution information of two types of newspapers is stored in the dictionary `distr_dict`. Use a bar graph to show the demands of these two types of newspapers under different weather conditions.

```
{'weather': ['Sunny', 'Cloudy', 'Raining',                    'Haze'],
'Thunderstorm',
 'probs': [0.315, 0.226, 0.289, 0.087, 0.083],
 'paper1': [560, 530, 389, 202, 278],
 'paper2': [533, 486, 386, 234, 263]}
```

Data Visualization

```
1  [redacted]  Array 0, 1, 2, …, 4
2  width = 0.3
3
4  plt.figure(figsize=(7.5, 4))
5  plt.bar(x-0.5*width, paper1,
6          color='r', width=width, alpha=0.6,
7          label='paper1')
8  plt.bar(x+0.5*width, paper2,
9          color='b', width=width, alpha=0.6,
10         label='paper2')
11
12 plt.legend(fontsize=13)
13 plt.xticks(x, weather, fontsize=13)
14 plt.ylabel('Demand', fontsize=13)
15 plt.show()
```

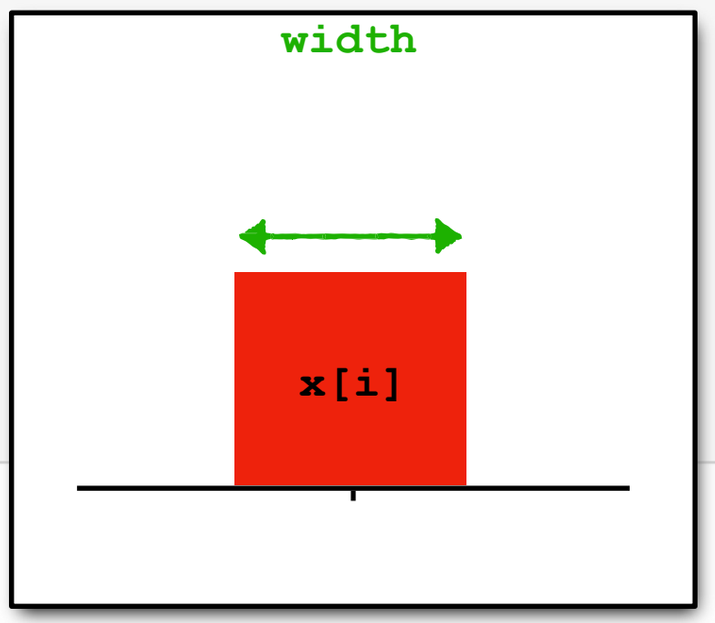
Data Visualization

```
x = np.arange(len(weather))
width = 0.3

plt.figure(figsize=(7.5, 4))
plt.bar(x-0.5*width, paper1,
        color='r', width=width, alpha=0.6,
        label='paper1')
plt.bar(x+0.5*width, paper2,
        color='b', width=width, alpha=0.6,
        label='paper2')

plt.legend(fontsize=13)
plt.xticks(x, weather, fontsize=13)
plt.ylabel('Demand', fontsize=13)
plt.show()
```

Configure the size of figure

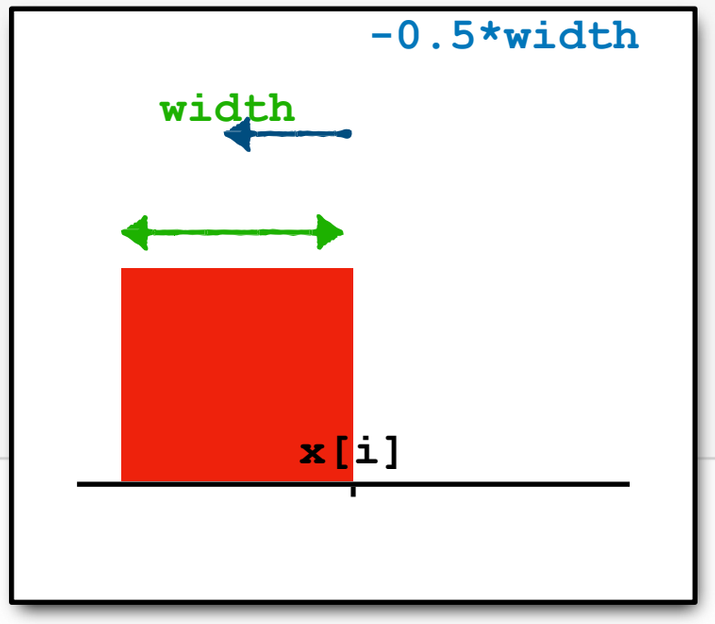
Data Visualization

```
x = np.arange(len(weather))
width = 0.3

plt.figure(figsize=(7.5, 4))
plt.bar(x-0.5*width, paper1,
        color='r', width=width, alpha=0.6,
        label='paper1')
plt.bar(x+0.5*width, paper2,
        color='b', width=width, alpha=0.6,
        label='paper2')

plt.legend(fontsize=13)
plt.xticks(x, weather, fontsize=13)
plt.ylabel('Demand', fontsize=13)
plt.show()
```



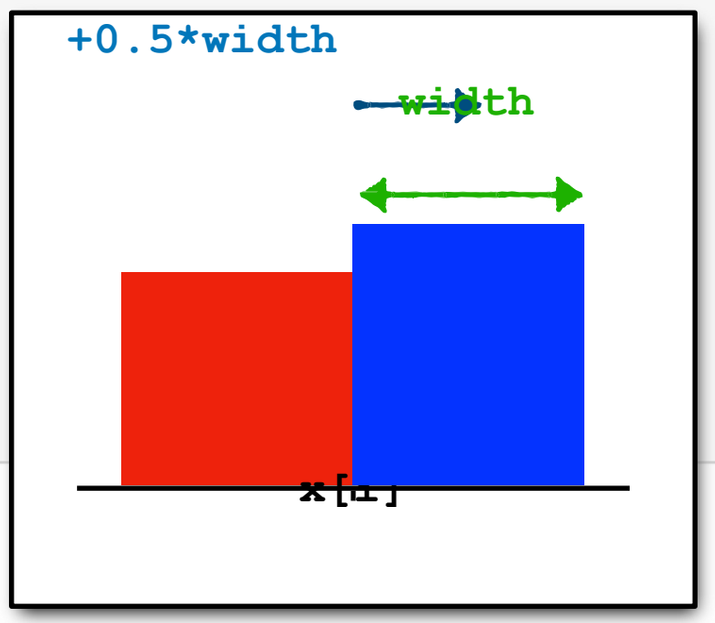
Data Visualization

```
x = np.arange(len(weather))
width = 0.3

plt.figure(figsize=(7.5, 4))
plt.bar(x-0.5*width, paper1,
        color='r', width=width, alpha=0.6,
        label='paper1')
plt.bar(x+0.5*width, paper2,
        color='b', width=width, alpha=0.6,
        label='paper2')

plt.legend(fontsize=13)
plt.xticks(x, weather, fontsize=13)
plt.ylabel('Demand', fontsize=13)
plt.show()
```



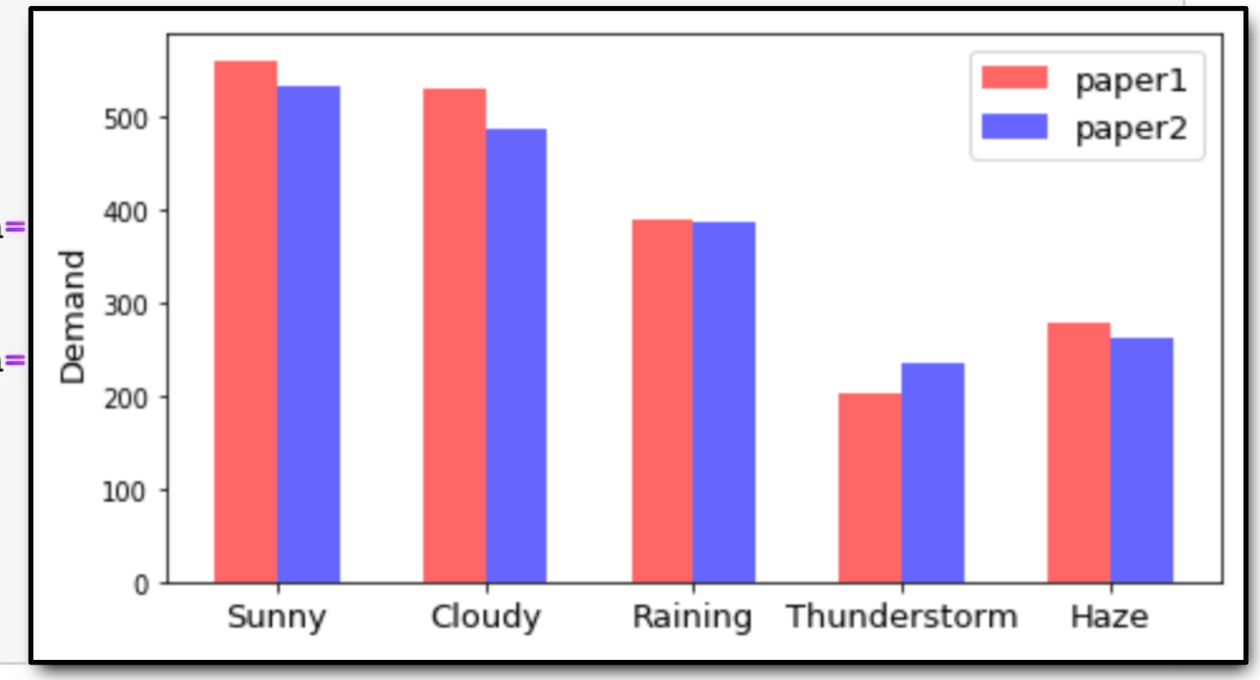
Data Visualization

```
x = np.arange(len(weather))
width = 0.3

plt.figure(figsize=(7.5, 4))
plt.bar(x-0.5*width, paper1,
        color='r', width=width, alpha=0.6,
        label='paper1')
plt.bar(                , paper2,
        color='b', width=width, alpha=0.6,
        label='paper2')

plt.legend(fontsize=13)
plt.xticks(x, weather, fontsize=13)
plt.ylabel('Demand', fontsize=13)
plt.show()
```



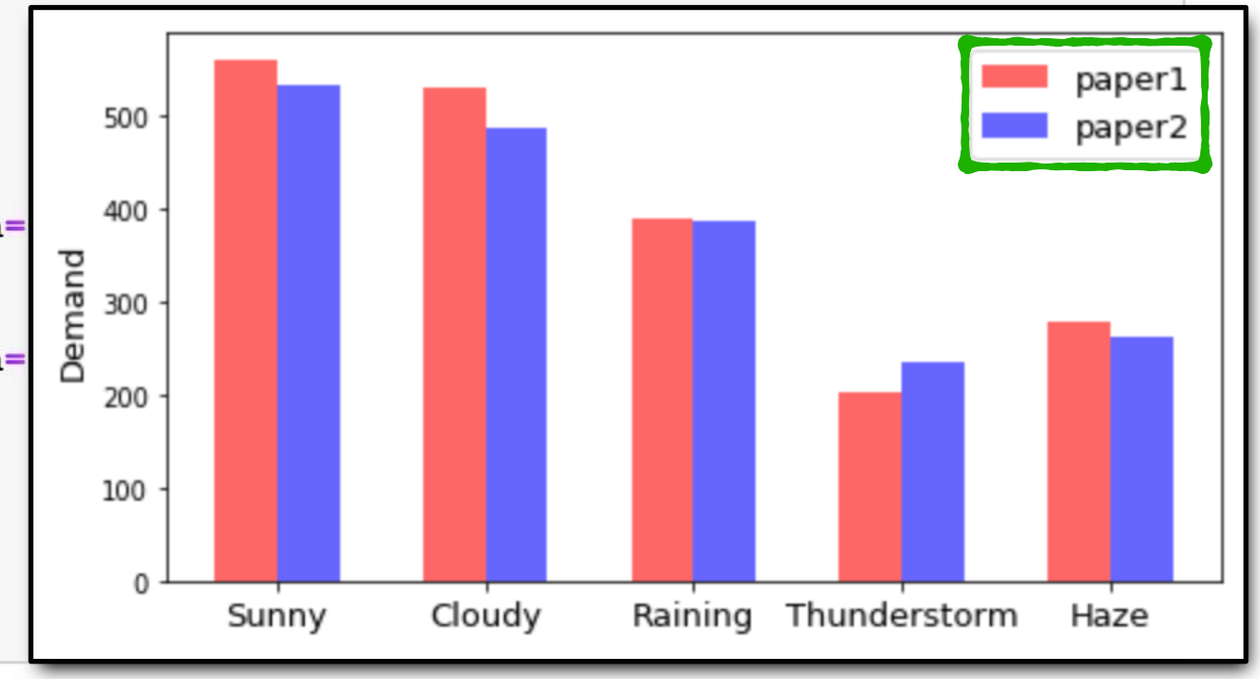
Data Visualization

```
x = np.arange(len(weather))
width = 0.3

plt.figure(figsize=(7.5, 4))
plt.bar(x-0.5*width, paper1,
        color='r', width=width, alpha=
        label='paper1')
plt.bar(x+0.5*width, paper2,
        color='b', width=width, alpha=
        label='paper2')

plt.legend(fontsize=13)
plt.xticks(x, weather, fontsize=13)
plt.ylabel('Demand', fontsize=13)
plt.show()
```



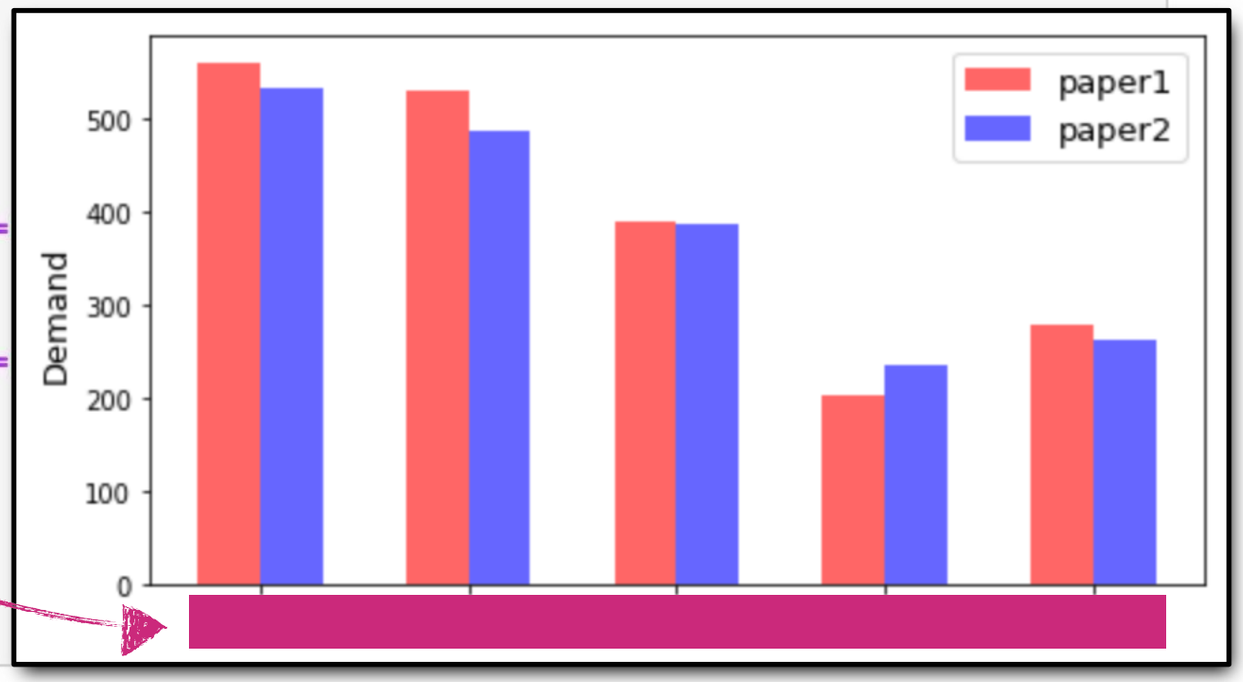
Data Visualization

```
x = np.arange(len(weather))
width = 0.3

plt.figure(figsize=(7.5, 4))
plt.bar(x-0.5*width, paper1,
        color='r', width=width, alpha=
        label='paper1')
plt.bar(x+0.5*width, paper2,
        color='b', width=width, alpha=
        label='paper2')

plt.xticks(x, weather, fontsize=13)
plt.ylabel('Demand', fontsize=13)
plt.show()
```



Data Visualization

```
1  x = np.arange(len(weather))
2  width = 0.3
3
4  plt.figure(figsize=(7.5, 4))
5  plt.bar(x-0.5*width, paper1,
6          color='r', width=width, alpha=
7          label='paper1')
8  plt.bar(x+0.5*width, paper2,
9          color='b', width=width, alpha=
10         label='paper2')
11
12 plt.legend(fontsize=13)
13 [redacted], fontsize=13)
14 plt.ylabel('Demand', fontsize=13)
15 plt.show()
```



Replace integer x-ticks by strings

[redacted]

Python Packages

- Packages NumPy and Matplotlib

Joan Watson: “How did you know he had an affair?”

–Elementary, Season 1, Episode 1

Python Packages

- Packages NumPy and Matplotlib

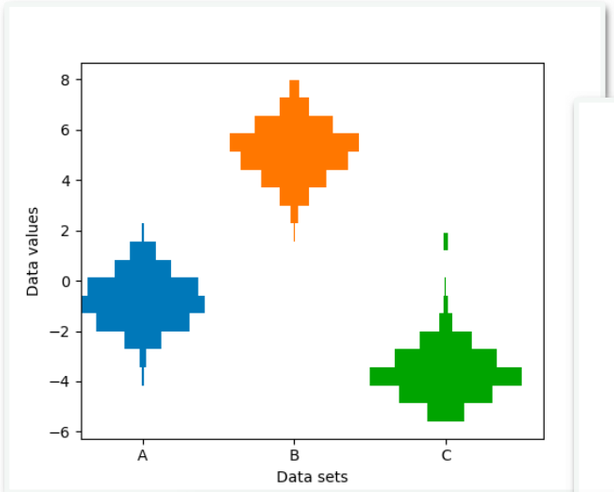
Joan Watson: “How did you know he had an affair?”

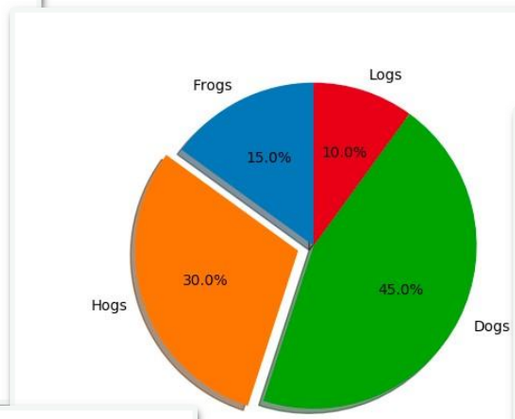
Sherlock Holmes: “Google. Well, not everything is deducible.”

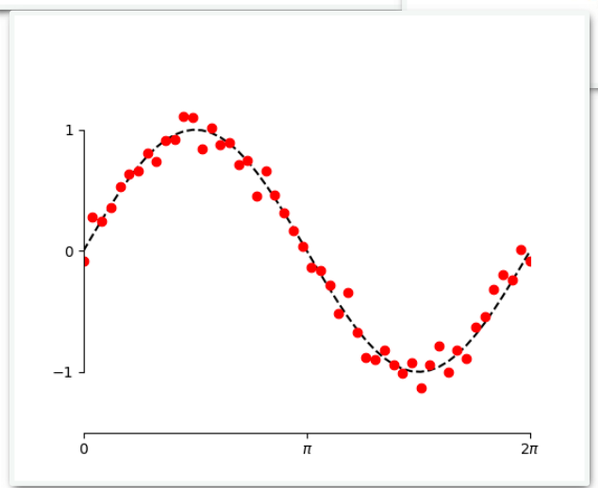
–Elementary, Season 1, Episode 1

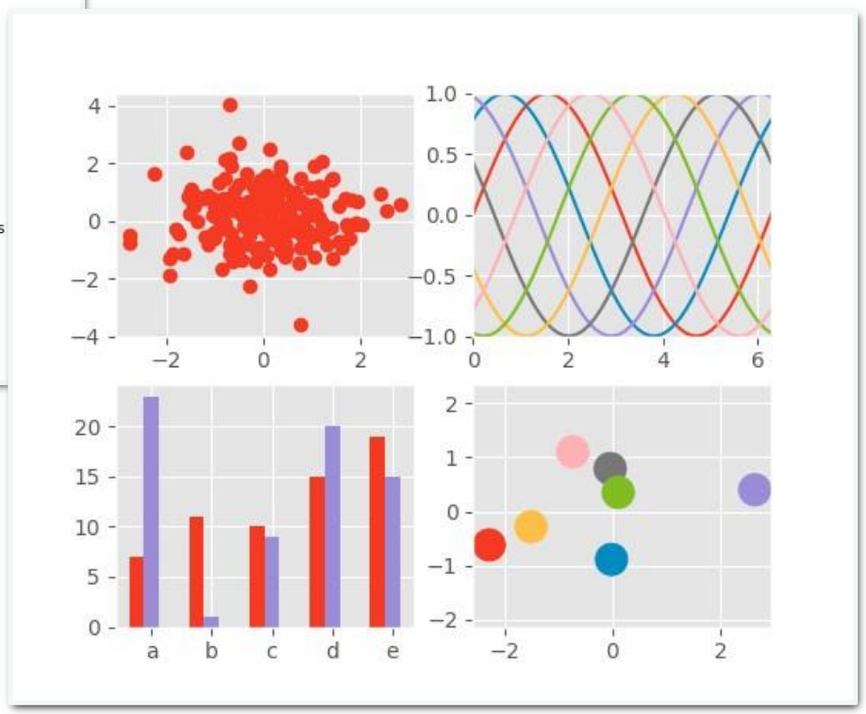
Python Packages

- Packages NumPy and Matplotlib
  - ► [redacted] thumbnail gallery: examples and demos









Roadmap

1. Array Data Structure
2. Data Visualization
3. **Pandas**
4. Descriptive Analytics in Python

“It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.”

–Sherlock Holmes, A Scandal in Bohemia

Datasets

- Data representation

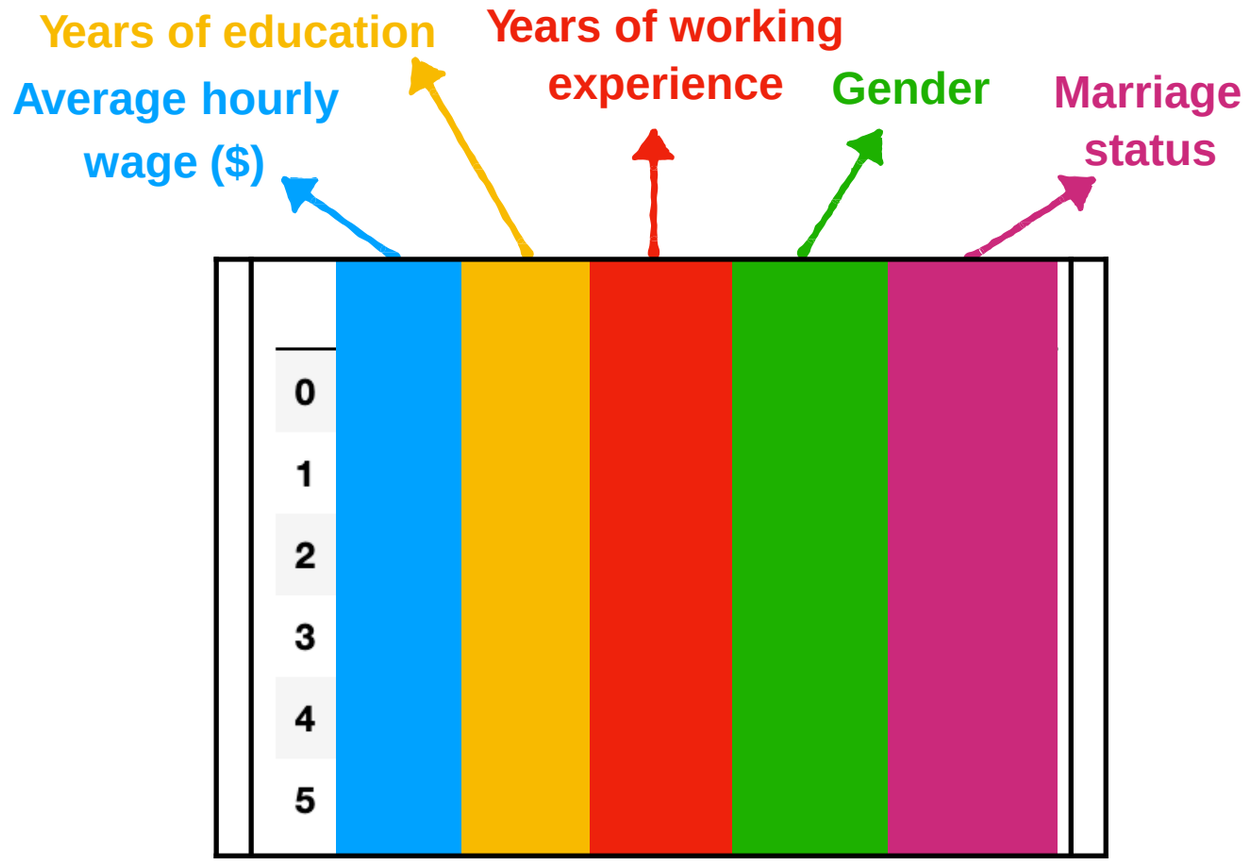
Example 1

The data table contains a cross-sectional dataset on a number of working individuals for the year 1976. Explore the dataset to gain some basic knowledge of the historical data.

	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	`False`
1	3.24	12.0	22.0	Female	`True`
2	3.00	11.0	2.0	Male	`False`
3	6.00	8.0	44.0	Male	`True`
4	5.30	12.0	7.0	Male	`True`
5	8.75	16.0	9.0	Male	`True`

Datasets

- Data representation
  - Variables (fields/attributes) as column labels

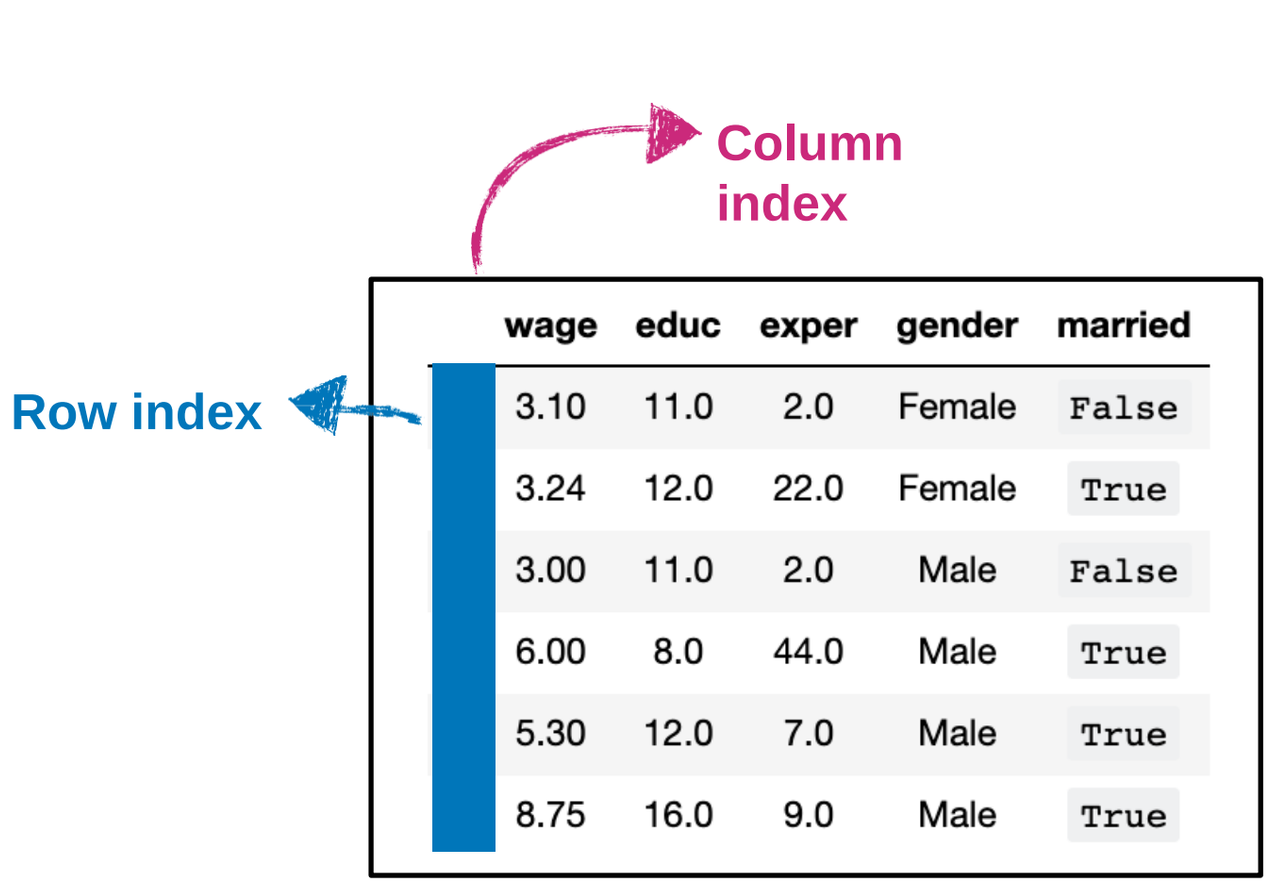


Datasets

- Data representation
  - Variables (fields/attributes) as column labels
  - Observations (cases/records) as rows

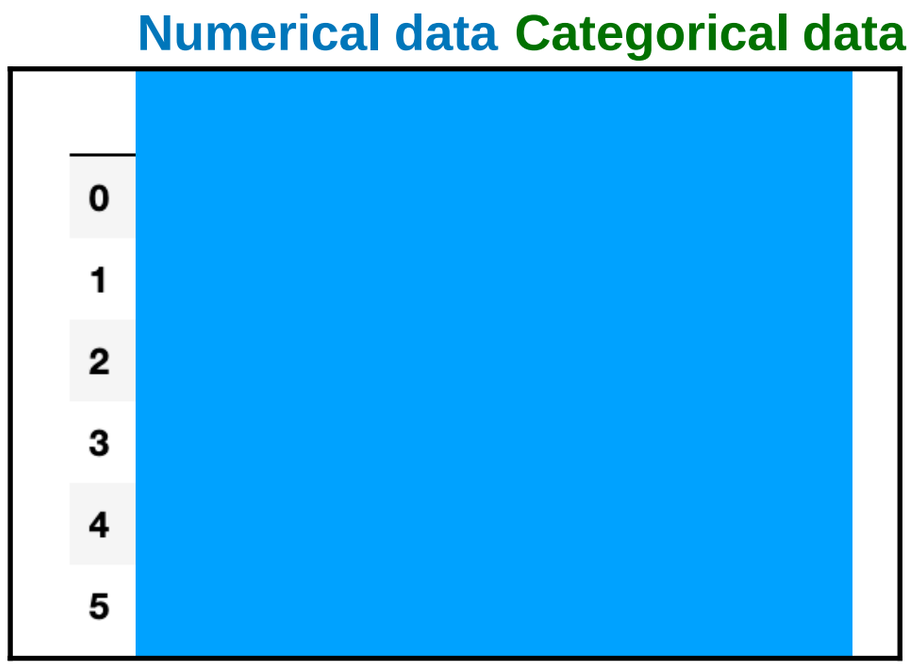
Datasets

- Data representation
  - Column index (label)
  - Row index (label)



Datasets

- Types of variables
  - Numerical data
  - Categorical data

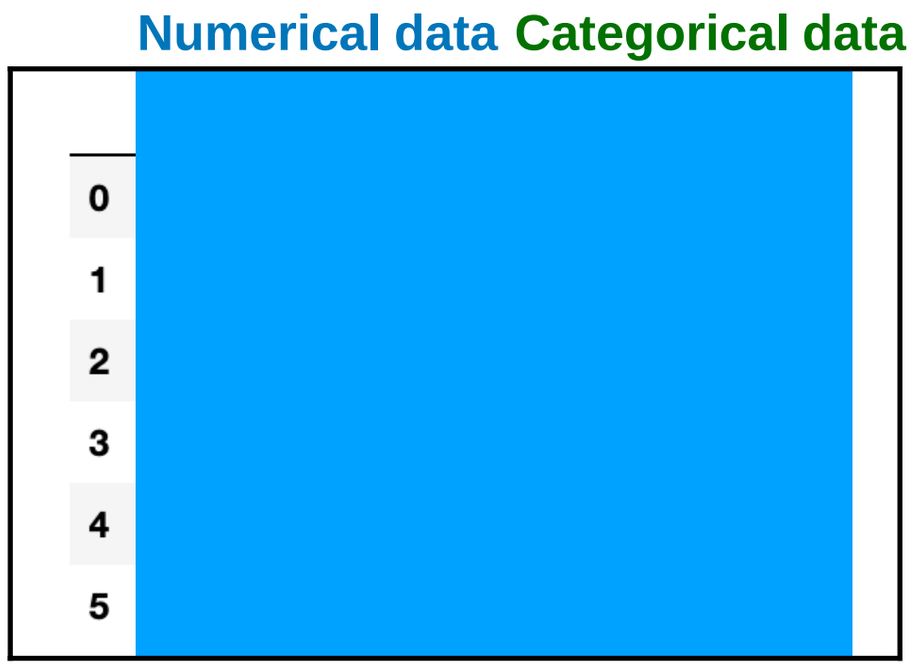


Datasets

- Types of variables
  - Numerical (quantitative) data
  - Categorical (qualitative) data

```
print(int(True))
print(int(False))
```

```
1
0
```



Pandas for Data Analysis

- Introduction to Pandas
  - Labels for variables (or observations)
  - Heterogenous data
  - Possible missing values

	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	`False`
1	3.24	12.0	22.0	Female	`True`
2	3.00	11.0	2.0	Male	`False`
3	6.00	8.0	44.0	Male	`True`
4	5.30	12.0	7.0	Male	`True`
5	8.75	16.0	9.0	Male	`True`

Pandas for Data Analysis

- Introduction to Pandas

```
import pandas as pd

pd.<function name>
```

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - A one-dimensional array of indexed data
  - Create [redacted] object
  a

```
1  wage = pd.Series([3.10, 3.24, 3.00, 6.00, 5.30, 8.75])
2  wage
```

```
0    3.10
1    3.24
2    3.00
3    6.00
4    5.30
5    8.75
dtype: float64
```

	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
5	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - A one-dimensional array of indexed data
  - Create [redacted] object

a

```
1  wage = pd.Series([3.10, 3.24, 3.00, 6.00, 5.30, 8.75])
2  wage
```

```
dtype: float64
```

```
wage.values
```

```
array([3.1 , 3.24, 3.  , 6.  , 5.3 , 8.75])
```

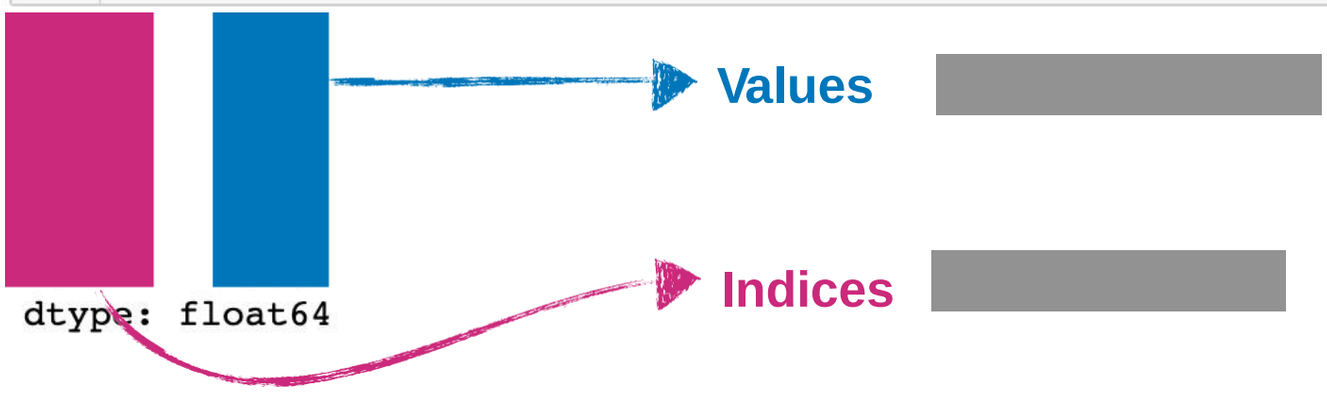
```
wage.index
```

```
RangeIndex(start=0, stop=6, step=1)
```

Pandas for Data Analysis

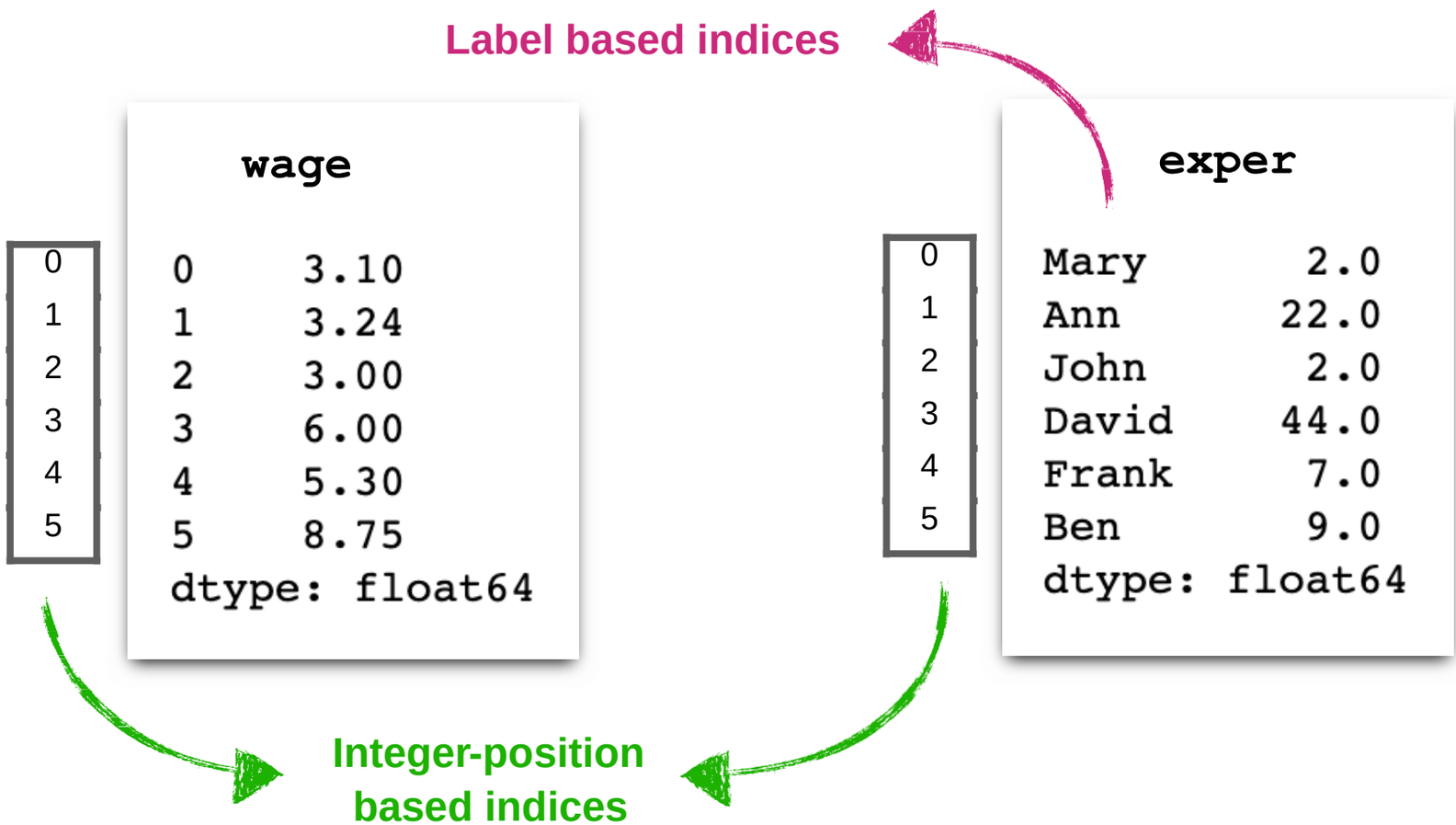
- Th [redacted] data structure
  e
  - A one-dimensional array of indexed data
  - Create **`pandas.Series`** object

```
index = ['Mary', 'Ann', 'John', 'David', 'Frank', 'Ben']
exper = pd.Series([2.0, 22.0, 2.0, 44.0, 7.0, 9.0],
                  index=index)  # Specify series indices

exper
```



Pandas for Data Analysis

- The [redacted] data structure
  - Indexing and slicing of **`pandas.Series`** objects via brackets [redacted]



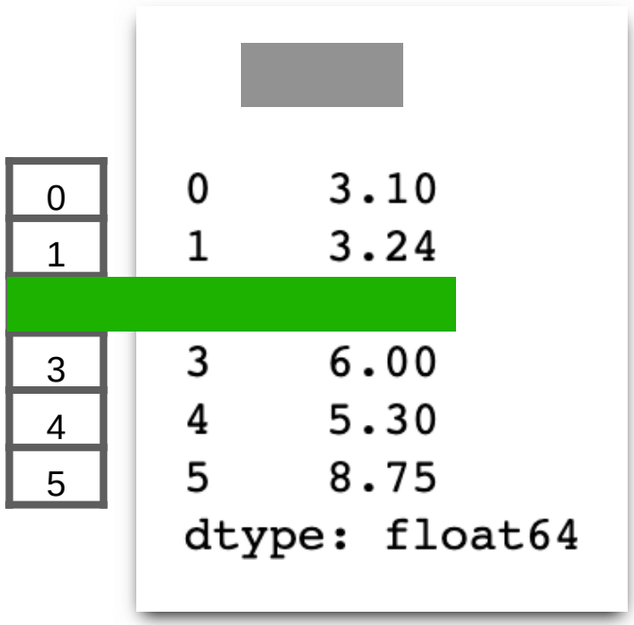
Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices are the same as integer-position based indices

```
1  print(wage[2], '\n')
2  print(wage[:3], '\n')
3  print(wage[-2:])
```

```
3.0

0    3.10
1    3.24
2    3.00
dtype: float64

4    5.30
5    8.75
dtype: float64
```



Integer-position based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices are the same as integer-position based indices

```
1  print(wage[2], '\n')
2  print([redacted], '\n')
3  print(wage[-2:])
```

```
3.0

[redacted]
dtype: float64

4    5.30
5    8.75
dtype: float64
```

[redacted]

3	6.00
4	5.30
5	8.75

dtype: float64

Integer-position based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices are the same as integer-position based indices

```
1  print(wage[2], '\n')
2  print(wage[:3], '\n')
3  print([redacted])
```

```
3.0

0    3.10
1    3.24
2    3.00
dtype: float64

[redacted]
dtype: float64
```

-6	0
-5	1
-4	2
-3	3

[redacted]

```
[redacted]
0    3.10
1    3.24
2    3.00
3    6.00
[redacted]
dtype: float64
```

Integer-position based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

✓ Label based indices different from integer-position based indices

```
1  print(exper['John'])
2  print(exper['Ann'])
```

```
[redacted]
22.0
```

	exper	
0	Mary	2.0
1	Ann	22.0
2	[redacted]	
3	David	44.0
4	Frank	7.0
5	Ben	9.0
	dtype:	float64

Label based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1  print(exper['John'])
2  print([redacted])
```

```
2.0
[redacted]
```

exper

0	Mary 2.0
1	[redacted]
2	John 2.0
3	David 44.0
4	Frank 7.0
5	Ben 9.0

dtype: float64

Label based indices

Pandas for Data Analysis

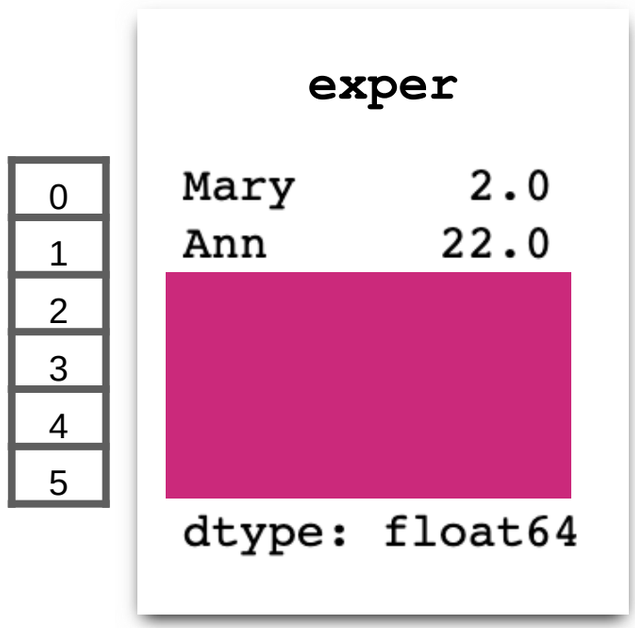
- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1 print(exper['John':], '\n')
2 print(exper[:'Ann'], '\n')
3 print(exper[['John', 'Ann']])
```

[redacted]

```
dtype: float64

Mary     2.0
Ann     22.0
dtype: float64

John     2.0
Ann     22.0
dtype: float64
```



Label based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1  print(exper['John':], '\n')
2  print([redacted], '\n')
3  print(exper[['John', 'Ann']])
```

```
John       2.0
David     44.0
Frank      7.0
Ben        9.0
dtype: float64

[redacted]
dtype: float64

John      2.0
Ann      22.0
dtype: float64
```

0
1
2
3
4
5

exper

[redacted]	
John	2.0
David	44.0
Frank	7.0
Ben	9.0

dtype: float64

Label based indices

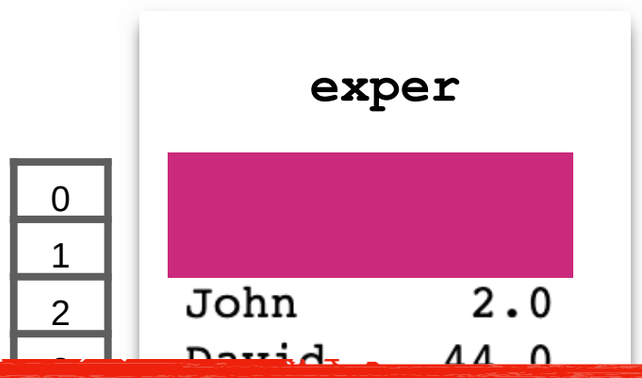
Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1 print(exper['John':], '\n')
2 print([redacted], '\n')
3 print(exper[['John', 'Ann']])
```

```
John       2.0
David     44.0
Frank      7.0
Ben        9.0
dtype: float64

[redacted]
dtype: float64

John       2.0
Ann       22.0
dtype: float64
```



Label based indices

Notes

In the slicing expressions for label based indices, the item indexed by stop are included in the selection. This is different from the list style slicing syntax we learned before.

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1  print(exper['John':], '\n')
2  print(exper[:'Ann'], '\n')
3  print([redacted])
```

```
John       2.0
David     44.0
Frank      7.0
Ben        9.0
dtype: float64

Mary       2.0
Ann       22.0
dtype: float64

[redacted]
dtype: float64
```

0
1
2
3
4
5

exper

```
Mary       2.0
[redacted]
David     44.0
Frank      7.0
Ben        9.0
dtype: float64
```

Label based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1 print(exper[3], '\n')
2 print(exper[:2], '\n')
3 print(exper[-3:], '\n')
4 print(exper[[2, 0, 1]])
```

[redacted]

```
Mary      2.0
Ann      22.0
dtype: float64

David    44.0
Frank     7.0
Ben       9.0
dtype: float64

John      2.0
Mary      2.0
Ann      22.0
dtype: float64
```

exper

0	Mary	2.0
1	Ann	22.0
2	John	2.0
[redacted]		
4	Frank	7.0
5	Ben	9.0
	dtype:	float64

Integer-position based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1 print(exper[3], '\n')
2 print([redacted], '\n')
3 print(exper[-3:], '\n')
4 print(exper[[2, 0, 1]])
```

```
44.0

[redacted]
dtype: float64

David    44.0
Frank     7.0
Ben       9.0
dtype: float64

John     2.0
Mary     2.0
Ann     22.0
dtype: float64
```

exper

[redacted]

2	John 2.0
3	David 44.0
4	Frank 7.0
5	Ben 9.0
	dtype: float64

Integer-position based indices

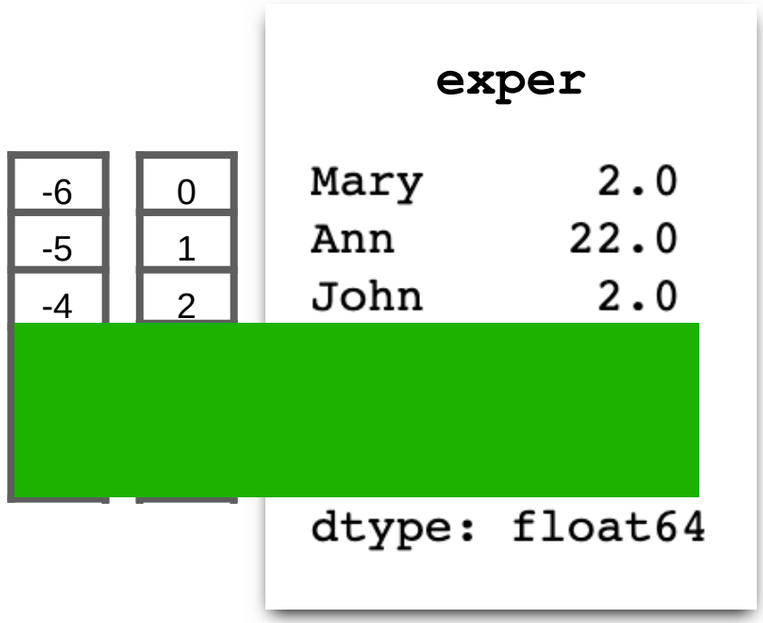
Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1 print(exper[3], '\n')
2 print(exper[:2], '\n')
3 print([redacted], '\n')
4 print(exper[[2, 0, 1]])
```

```
44.0

Mary     2.0
Ann     22.0
dtype: float64

[redacted]
dtype: float64

John     2.0
Mary     2.0
Ann     22.0
dtype: float64
```



Integer-position based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]

    ✓ Label based indices different from integer-position based indices

```
1 print(exper[3], '\n')
2 print(exper[:2], '\n')
3 print(exper[-3:], '\n')
4 print([redacted])
```

```
44.0

Mary      2.0
Ann      22.0
dtype: float64

David    44.0
Frank     7.0
Ben       9.0
dtype: float64

[redacted]
dtype: float64
```

exper

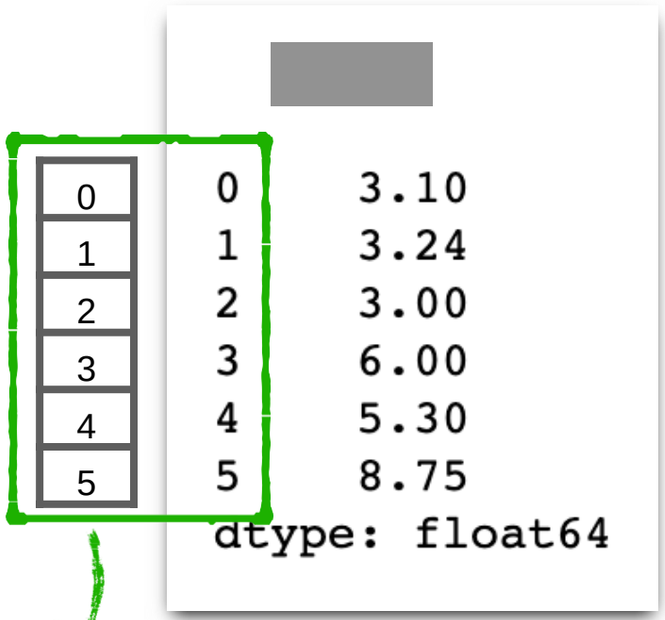
[redacted]

3	David 44.0
4	Frank 7.0
5	Ben 9.0
	dtype: float64

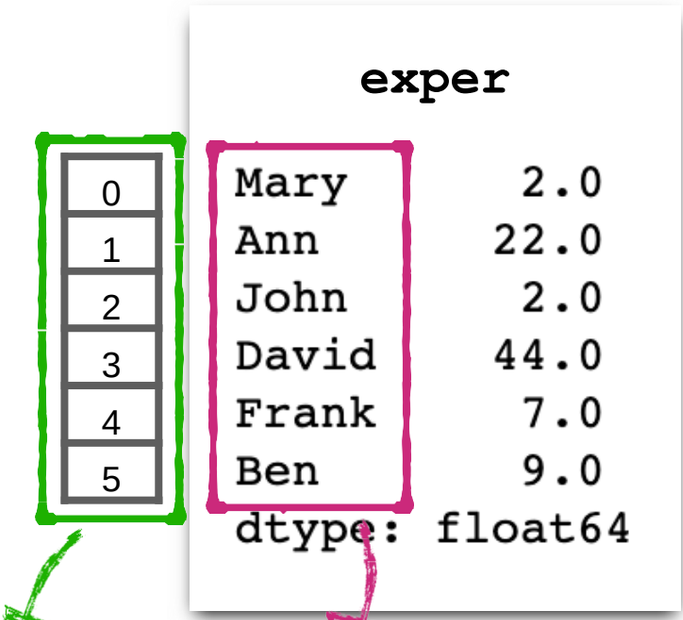
Integer-position based indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexing and slicing of [redacted] objects via brackets [redacted]




```
print(wage[2], '\n')
print(wage[:3], '\n')
print(wage[-2:])
```

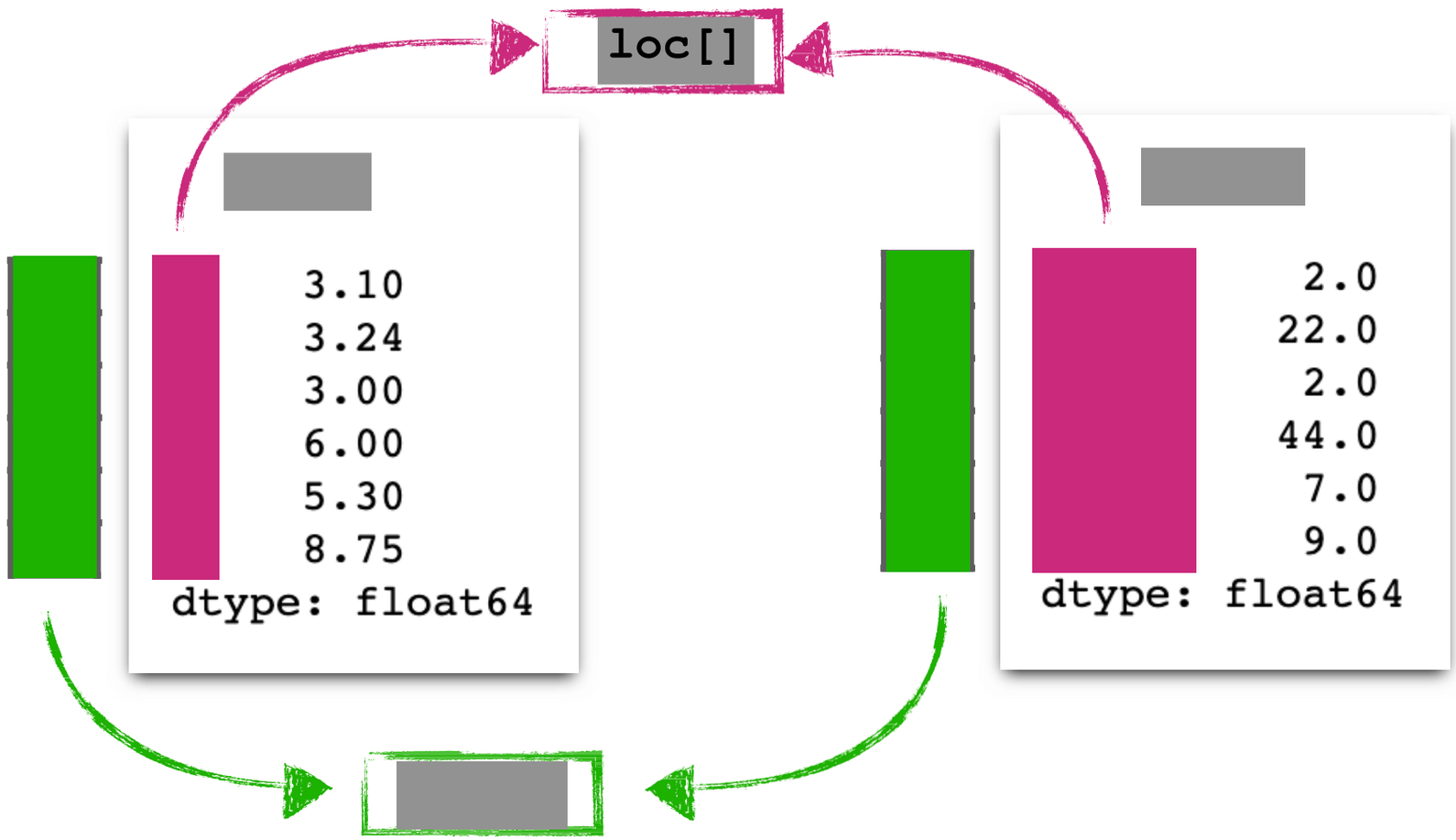



```
print(exper[3], '\n')
print(exper[:2], '\n')
print(exper[-3:], '\n')
print(exper[[2, 0, 1]])
```

```
print(exper['John':], '\n')
print(exper[:'Ann'], '\n')
print(exper[['John', 'Ann']])
```

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Indexers: [redacted] and [redacted]



Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Indexers: [redacted] and [redacted]

```
print(exper.loc['Ann'], '\n')
print(exper.loc['Frank':], '\n')
print(exper.loc[['Mary', 'John']], '\n')
print(exper.loc[::2])
```

```
print([redacted], '\n')
print(exper.iloc[4:], '\n')
print(exper.iloc[[0, 2]], '\n')
print(exper.iloc[::2])
```

```
22.0
```

[redacted]

	[redacted]	
0	Mary	2.0
[redacted]	[redacted]	22.0
2	John	2.0
3	David	44.0
4	Frank	7.0
5	Ben	9.0

dtype: float64

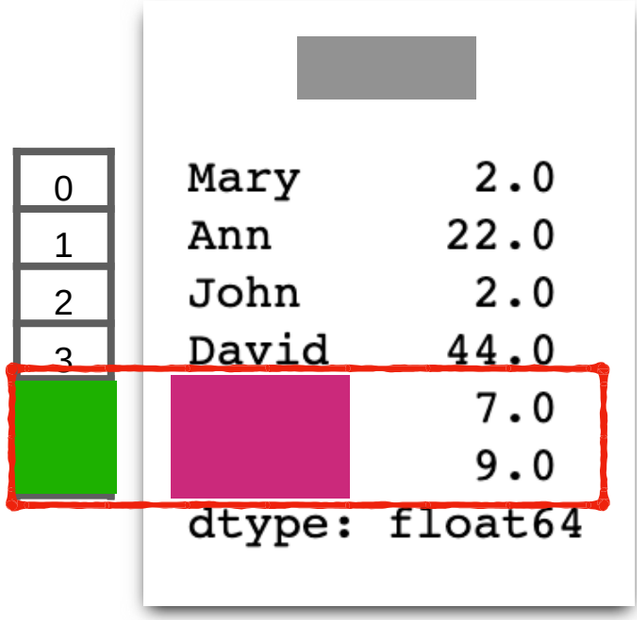
Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Indexers: [redacted] and [redacted]

```
print(exper.loc['Ann'], '\n')
print([redacted], '\n')
print(exper.loc[['Mary', 'John']], '\n')
print(exper.loc[::2])
```

```
print(exper.iloc[1], '\n')
print([redacted] '\n')
print(exper.iloc[[0, 2]], '\n')
print(exper.iloc[::2])
```

```
Frank     7.0
Ben       9.0
dtype: float64
```



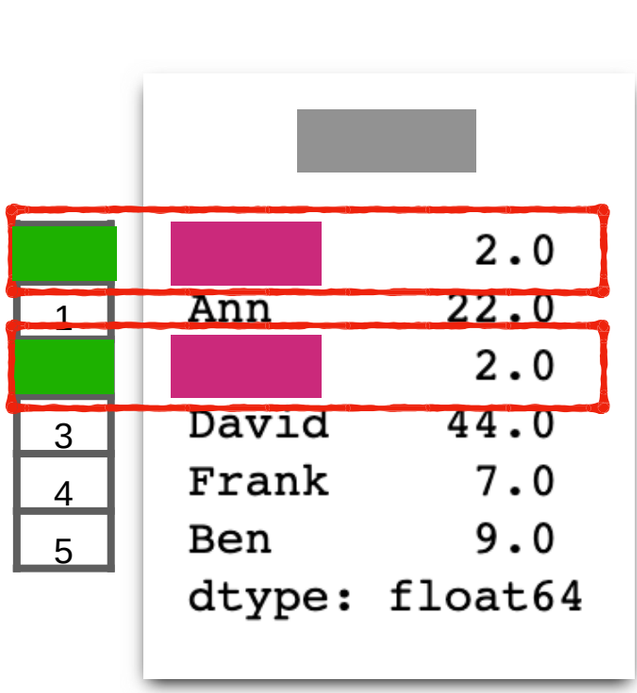
Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Indexers: [redacted] and [redacted]

```
print(exper.loc['Ann'], '\n')
print(exper.loc['Frank':], '\n')
print([redacted], '\n')
print(exper.loc[::2])
```

```
print(exper.iloc[1], '\n')
print(exper.iloc[4:], '\n')
print([redacted], '\n')
print(exper.iloc[::2])
```

```
Mary     2.0
John     2.0
dtype: float64
```



Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Indexers: [redacted] and [redacted]

```
print(exper.loc['Ann'], '\n')
print(exper.loc['Frank':], '\n')
print(exper.loc[['Mary', 'John']], '\n')
print([redacted]
```

```
print(exper.iloc[1], '\n')
print(exper.iloc[4:], '\n')
print(exper.iloc[[0, 2]], '\n')
print([redacted]
```

```
Mary      2.0
John      2.0
Frank     7.0
dtype: float64
```

[redacted]

```
[redacted]   2.0
1    Ann     22.0
[redacted]   2.0
3    David   44.0
[redacted]   7.0
5    Ben     9.0
dtype: float64
```

Pandas for Data Analysis

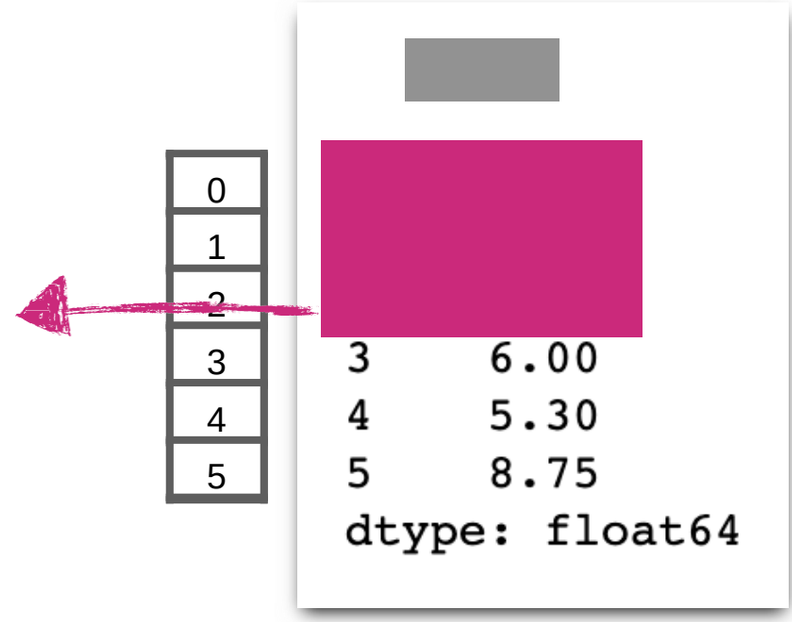
- Th [redacted] data structure
  e
  - Indexers: [redacted] and [redacted]

```
1 print(wage.loc[:2], '\n')
2 print(wage.iloc[:2])
```

[redacted]

```
Name: wage, dtype: float64

0    3.10
1    3.24
Name: wage, dtype: float64
```

[redacted] is included



Pandas for Data Analysis

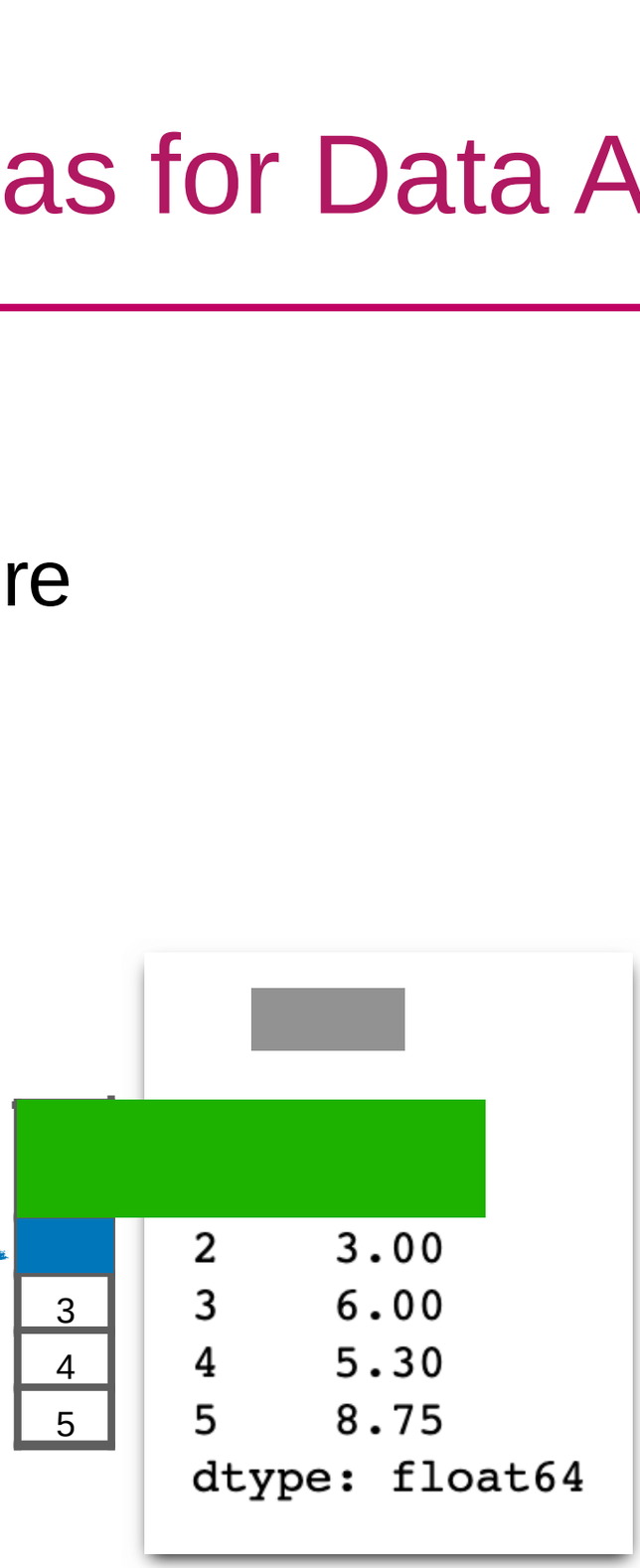
- Th [redacted] data structure
  e
  - Indexers: [redacted] and [redacted]

```
1  print(wage.loc[:2], '\n')
2  print([redacted])
```

```
0    3.10
1    3.24
2    3.00
Name: wage, dtype: float64
```

[redacted] [redacted] **is excluded**

```
Name: wage, dtype: float64
```



Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexers: **`loc[]`** and **`iloc[]`**

    ✓ Label based indices are expressed by [redacted], with the [redacted] index included in the selection

    ✓ Integer-position based indices are expressed by [redacted] with the [redacted] index excluded from the selection

Coding Style

One guiding principle of Python code is that "explicit is better than implicit." The explicit natu[redacted] and [redacted] make them very useful in maintaining clean and readable code; especially in the case of integer indexes.

Pandas for Data Analysis

- The **`pandas.DataFrame`** data structure
  - Create **`pandas.DataFrame`** object

```
1  data_dict = {'wage': [3.10, 3.24, 3.00, 6.00, 5.30, 8.75],
2               'educ': [11.0, 12.0, 11.0, 8.0, 12.0, 16.0],
3               'exper': [2.0, 22.0, 2.0, 44.0, 7.0, 9.0],
4               'gender': ['Female', 'Female', 'Male', 'Male', 'Male', 'Male'],
5               'married': [False, True, False, True, True, True]}
6
7  data_frame = pd.DataFrame(data_dict)
8  data_frame
```

	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
5	8.75	16.0	9.0	Male	True

```
1  type(
```

```
1  type(data_frame)
```

```
pandas.core.frame.DataFrame
```

Pandas for Data Analysis

- The [redacted] data structure
  - Create [redacted] object

```
1  data_dict = {'wage': [3.10, 3.24, 3.00, 6.00, 5.30, 8.75],
2               'educ': [11.0, 12.0, 11.0, 8.0, 12.0, 16.0],
3               'exper': [2.0, 22.0, 2.0, 44.0, 7.0, 9.0],
4               'gender': ['Female', 'Female', 'Male', 'Male', 'Male', 'Male'],
5               'married': [False, True, False, True, True, True]}
6
7  [redacted] = [redacted](data_dict)
8  data_frame
```

Constructor

```
1  type(data_dict)
```

```
dict
```

```
1  type([redacted]
```

[redacted]

	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
5	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Create [redacted] object

```
data_dict = {[redacted]: [3.10, 3.24, 3.00, 6.00, 5.30, 8.75],
             [redacted]: [11.0, 12.0, 11.0, 8.0, 12.0, 16.0],
             [redacted]: [2.0, 22.0, 2.0, 44.0, 7.0, 9.0],
             [redacted]: ['Female', 'Female', 'Male', 'Male', 'Male', 'Male'],
             [redacted]: [False, True, False, True, True, True]}

data_frame = pd.DataFrame(data_dict)
data_frame
```

```
type(data_dict)
```

```
dict
```

```
type(data_frame)
```

```
pandas.core.frame.DataFrame
```

Column index [redacted]

	[redacted]				
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
5	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- The [redacted] data structure
  - Create [redacted] object

```
1  data_dict = {'wage': [3.10, 3.24, 3.00, 6.00, 5.30, 8.75],
2               'educ': [11.0, 12.0, 11.0, 8.0, 12.0, 16.0],
3               'exper': [2.0, 22.0, 2.0, 44.0, 7.0, 9.0],
4               'gender': ['Female', 'Female', 'Male', 'Male', 'Male', 'Male'],
5               'married': [False, True, False, True, True, True]}
6
7  data_frame = pd.DataFrame(data_dict)
8  data_frame
```

```
1  type(data_dict)
```

```
dict
```

```
1  type(data_frame)
```

```
pandas.core.frame.DataFrame
```

Row index [redacted]

	wage	educ	exper	gender	married
	3.10	11.0	2.0	Female	False
	3.24	12.0	22.0	Female	True
	3.00	11.0	2.0	Male	False
	6.00	8.0	44.0	Male	True
	5.30	12.0	7.0	Male	True
	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- The [redacted] data structure
  - Create [redacted] object

```
1  data_dict = {'wage': [redacted],
2               'educ': [11.0, 12.0, 11.0, 8.0, 12.0, 16.0],
3               'exper': [2.0, 22.0, 2.0, 44.0, 7.0, 9.0],
4               'gender': ['Female', 'Female', 'Male', 'Male', 'Male', 'Male'],
5               'married': [False, True, False, True, True, True]}
6
7  data_frame = pd.DataFrame(data_dict)
8  data_frame
```

	wage	educ	exper	gender	married
0	[redacted]	11.0	2.0	Female	False
1		12.0	22.0	Female	True
2		11.0	2.0	Male	False
3		8.0	44.0	Male	True
4		12.0	7.0	Male	True
5		16.0	9.0	Male	True

```
1  type(data_dict)
```

```
dict
```

```
1  type(data_frame)
```

```
pandas.core.frame.DataFrame
```

Pandas for Data Analysis

- The [redacted] data structure
  - Create **`pandas.DataFrame`** object

```
index = ['Mary', 'Ann', 'John', 'David', 'Frank', 'Ben']
data_frame_new = pd.DataFrame(data_dict, index=index)

data_frame_new
```

	wage	educ	exper	gender	married
Mary	3.10	11.0	2.0	Female	False
Ann	3.24	12.0	22.0	Female	True
John	3.00	11.0	2.0	Male	False
David	6.00	8.0	44.0	Male	True
Frank	5.30	12.0	7.0	Male	True
Ben	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Accessing columns and rows via brackets [redacted]

    ✓ Columns: a single column index or a list of column indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Accessing columns and rows via brackets [redacted]

✓ Columns: a single column index or a list of column indices

```
wage = data_frame['wage']
wage
```

```
0    3.10
1    3.24
2    3.00
3    6.00
4    5.30
5    8.75
Name: wage, dtype: float64
```

```
type(wage)
```

```
pandas.core.series.Series
```

	[redacted]	educ	exper	gender	married
Mary		11.0	2.0	Female	False
Ann		12.0	22.0	Female	True
John		11.0	2.0	Male	False
David		8.0	44.0	Male	True
Frank		12.0	7.0	Male	True
Ben		16.0	9.0	Male	True

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Accessing columns and rows via brackets [redacted]

    ✓ Columns: a single column index or [redacted]

```
skills = data_frame[['educ', 'exper']]
skills
```

	educ	exper
0	11.0	2.0
1	12.0	22.0
2	11.0	2.0
3	8.0	44.0
4	12.0	7.0
5	16.0	9.0

	wage	[redacted]	gender	married
Mary	3.10		Female	False
Ann	3.24		Female	True
John	3.00		Male	False
David	6.00		Male	True
Frank	5.30		Male	True
Ben	8.75		Male	True

Pandas for Data Analysis

- The **`pandas.DataFrame`** data structure
  - Accessing columns and rows via brackets `[]`

    ✓ Rows: a slicing expression of the label based or integer-position based indices

Pandas for Data Analysis

- The [redacted] data structure
  - ► Accessing columns and rows via brackets [redacted]

    ✓ Rows: a slicing expression of the label based or integer-position

```
1 john = data_frame_new[2:3]
2 john
```

	wage	educ	exper	gender	married
John	3.0	11.0	2.0	Male	False

	wage	educ	exper	gender	married
Mary	3.10	11.0	2.0	Female	False
Ann	3.24	12.0	22.0	Female	True
David	6.00	8.0	44.0	Male	True
Frank	5.30	12.0	7.0	Male	True
Ben	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Accessing columns and rows via brackets [redacted]

    ✓ Rows: [redacted] of the label based or integer-position

```
david = data_frame_new['David':'Frank']
david
```

	wage	educ	exper	gender	married
David	6.0	8.0	44.0	Male	True
Frank	5.3	12.0	7.0	Male	True

	wage	educ	exper	gender	married
Mary	3.10	11.0	2.0	Female	False
Ann	3.24	12.0	22.0	Female	True
John	3.00	11.0	2.0	Male	False
[redacted]					
Ben	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- The [redacted] data structure
  - Accessing columns and rows via brackets `[]`

Question

Given a data table. how to access the highlighted subset?

A.
```
1 data[1][[2, 3]]
```

B.
```
1 data[2][[1, 2]]
```

C.
```
1 data[2:3][[1, 2]]
```

D.
```
1 data[1:2][[1, 2]]
```

E.
```
1 data[1:2][[2, 3]]
```

F.
```
1 data[2:3][[1, 2]]
```

	1	2	3	4
1	1	2	3	4
2	5			8
3	9	10	11	12
4	13	14	15	16

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Accessing columns and rows via brackets [redacted]

    ✓ Columns: a single column index or a list of column indices

    ✓ Rows: a slicing expression of the label based or integer-position based indices

Pandas for Data Analysis

- The [redacted] data structure
  - Indexers: [redacted] and [redacted]

.iloc selections - position based selection

data.iloc[<row selection>, <column selection>]

Integer list of rows: [0,1,2]
Slice of rows: [4:7]
Single values: 1

Integer list of columns: [0,1,2]
Slice of columns: [4:7]
Single column selections: 1

loc selections - label based selection

data.loc[<row selection>, <column selection>]

Index/Label value: 'john'
List of labels: ['john', 'sarah']
Logical/Boolean index: data['age'] == 10

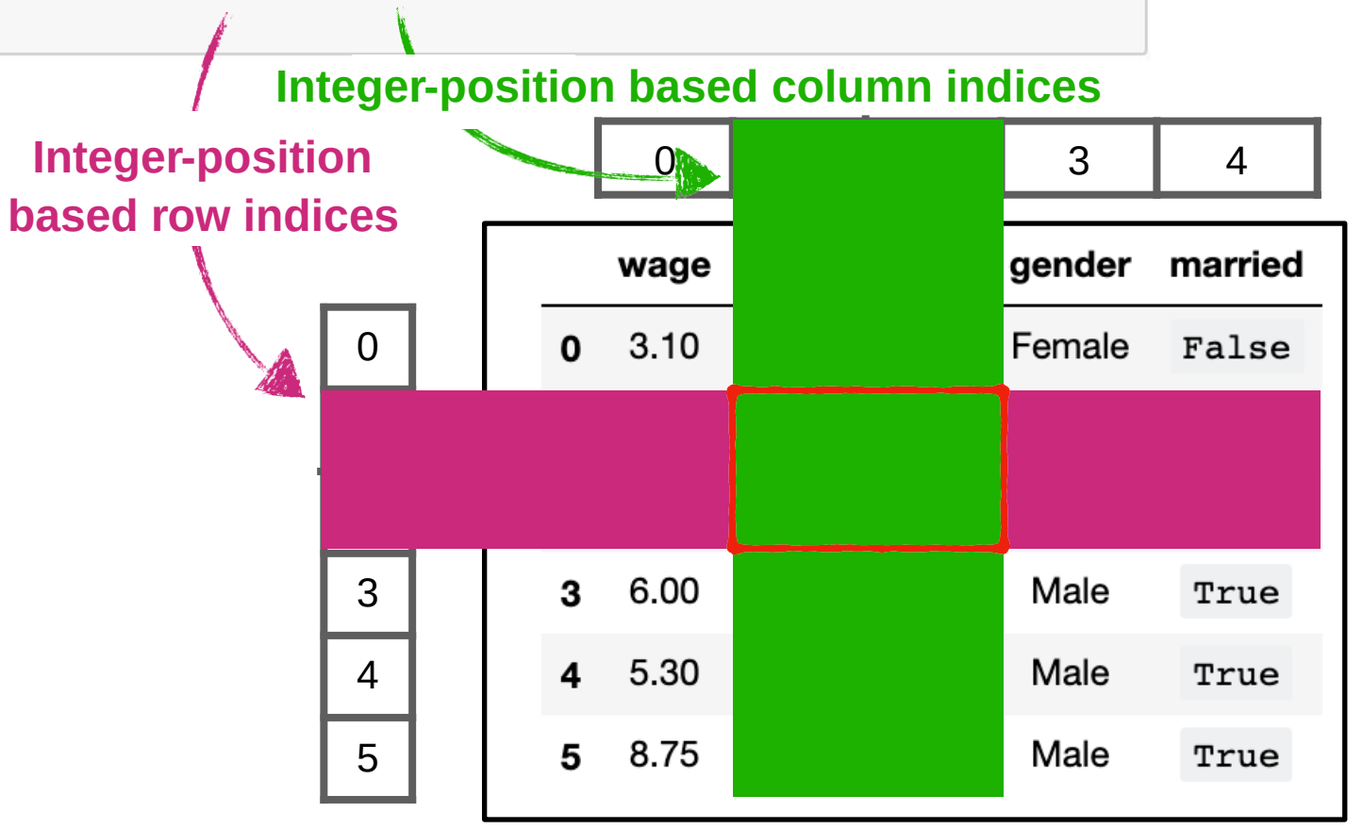
Named column: 'first_name'
List of column names: ['first_name', 'age']
Slice of columns: 'first_name':'address'

Pandas for Data Analysis

- The [redacted] data structure
  - Indexers: [redacted] and [redacted]

```
6 data_subset = data_frame.iloc[1:3, 1:3]
7 data_subset
```

	educ	exper
1	12.0	22.0
2	11.0	2.0

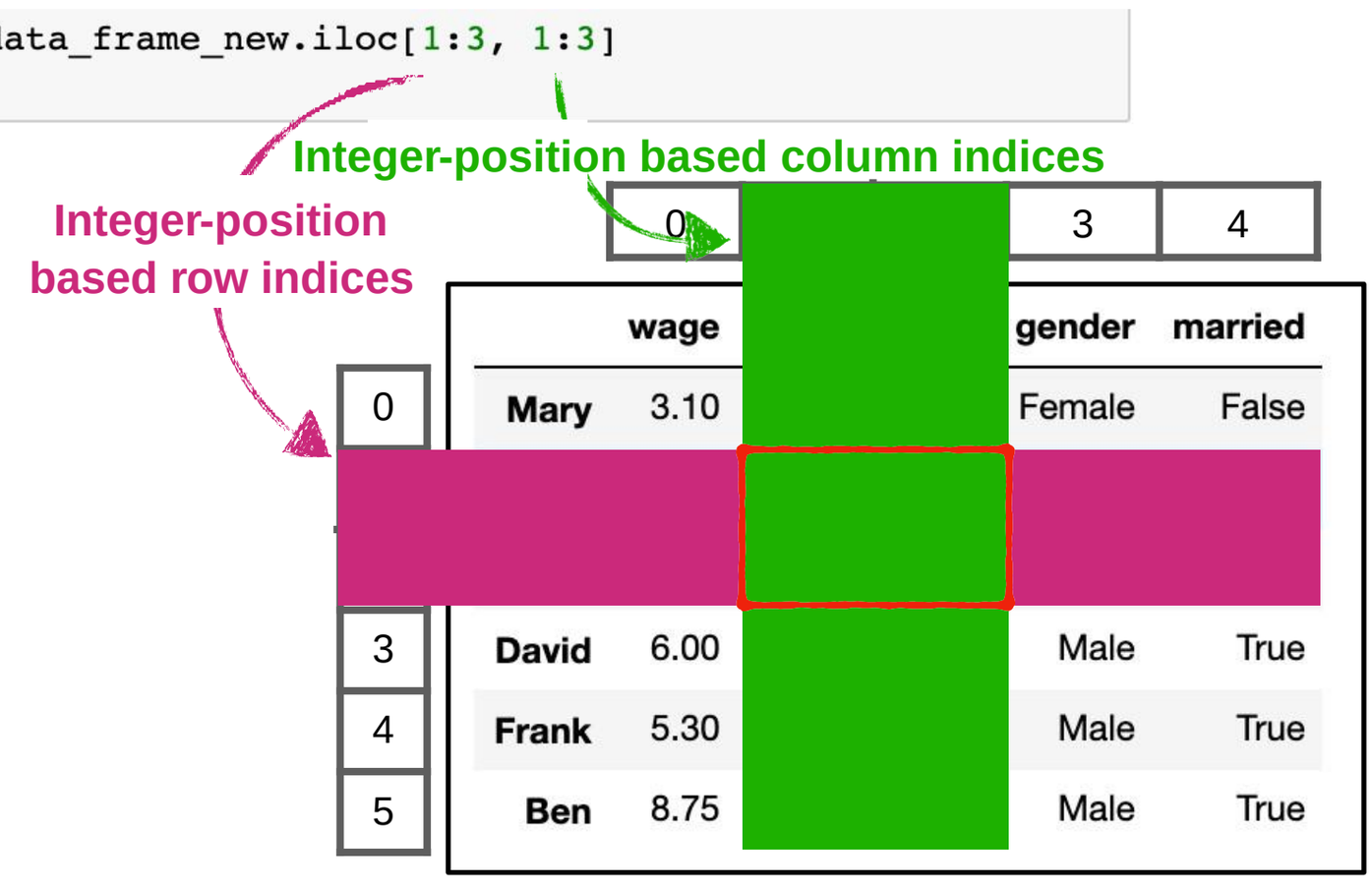


Pandas for Data Analysis

- The [redacted] data structure
  - Indexers: [redacted] and [redacted]

```
6 data_new_subset = data_frame_new.iloc[1:3, 1:3]
7 data_new_subset
```

	educ	exper
Ann	12.0	22.0
John	11.0	2.0



Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Indexers: [redacted] and [redacted]

```
6 data_subset = data_frame.loc[1:2, 'educ':'exper']
7 data_subset
```

	educ	exper
1	12.0	22.0
2	11.0	2.0

Label based row indices

Label based column indices

	wage	[redacted]	gender	married
0	3.10		Female	`False`
[redacted]				
3	6.00		Male	`True`
4	5.30		Male	`True`
5	8.75		Male	`True`

Pandas for Data Analysis

- The [redacted] data structure
  - Indexers: [redacted] and [redacted]

```
6 data_subset = data_frame.loc[1:2, 'educ':'exper']
7 data_subset
```

	educ	exper
1	12.0	22.0
2	11.0	2.0

Label based row indices

Label based column indices

	wage		gender	married
0	3.10		Female	False
3	6.00		Male	True
4	5.30		Male	True
5	8.75		Male	True

Notes

In the slicing expressions for label based indices (`loc[]`), the item indexed by stop are included in the selection. This is different from the list style slicing syntax we learned before.

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexers: [redacted] and [redacted]
    - ✓ Access all rows or columns via [redacted]

```
1  data_frame.iloc[:, 2]
```

```
0     2.0
1    22.0
2     2.0
3    44.0
4     7.0
5     9.0
Name: exper, dtype: float64
```

All rows

Column indices

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexers: [redacted] and [redacted]
    - ✓ Access all rows or columns via [redacted]

```
1  rows = data_frame_new.loc['John':'Ben', :]
2  rows
```

All columns

All indices

	wage	educ	exper	gender	married
John	3.00	11.0	2.0	Male	False
David	6.00	8.0	44.0	Male	True
Frank	5.30	12.0	7.0	Male	True
Ben	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Indexers: [redacted] and [redacted]
    - ✓ Access all rows or columns via [redacted]

```
1  rows = data_frame_new.loc[                ]
2  rows
```

All indices

All columns as default

	wage	educ	exper	gender	married
John	3.00	11.0	2.0	Male	False
David	6.00	8.0	44.0	Male	True
Frank	5.30	12.0	7.0	Male	True
Ben	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- The [redacted] data structure
  - Indexers: [redacted] and [redacted]
    - ✓ Data types of indices

```
1  data_frame_new.loc['John']
```

```
wage            3
educ           11
exper           2
gender       Male
married     False
Name: John, dtype: object
```

```
1  data_frame_new.loc[['John']]
```

	wage	educ	exper	gender	married
John	3.0	11.0	2.0	Male	False

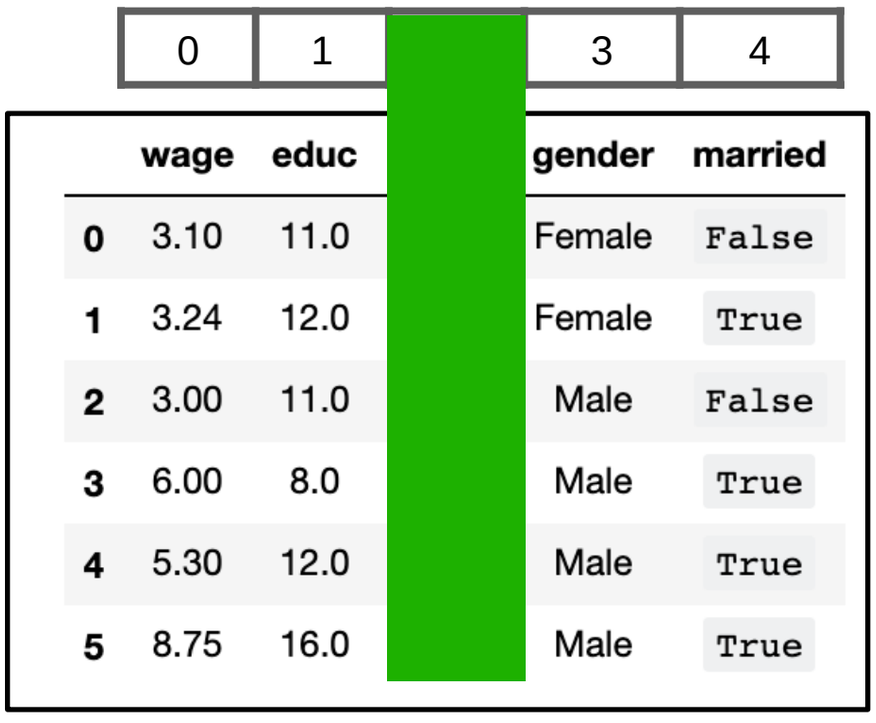
	wage	educ	exper	gender	married
Mary	3.10	11.0	2.0	Female	False
Ann	3.24	12.0	22.0	Female	True
David	6.00	8.0	44.0	Male	True
Frank	5.30	12.0	7.0	Male	True
Ben	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- The [redacted] data structure
  - Indexers: [redacted] and [redacted]
    - ✓ Data types of indices

```
1  data_frame.iloc[:, 2]
```

```
0     2.0
1    22.0
2     2.0
3    44.0
4     7.0
5     9.0
Name: exper, dtype: float64
```



Pandas for Data Analysis

- The [redacted] data structure
  - Indexers: [redacted] and [redacted]
    - ✓ Data types of indices

```
1  data_frame.iloc[:, 2:3]
```

	exper
0	2.0
1	22.0
2	2.0
3	44.0
4	7.0
5	9.0

0	1		3	4

	wage	educ		gender	married
0	3.10	11.0		Female	False
1	3.24	12.0		Female	True
2	3.00	11.0		Male	False
3	6.00	8.0		Male	True
4	5.30	12.0		Male	True
5	8.75	16.0		Male	True

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - ► Modify a subset of a **`pandas.DataFrame`** object

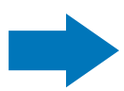
    ✓ Similar to modifying a part of a built-in compound data object (list/dictionary)

Pandas for Data Analysis

- The [redacted] data structure
  - Modify a subset of a [redacted] object

```
1 data_frame.loc[2:3, 'educ'] = 9.0
2 data_frame
```

	wage	[redacted]	exper	gender	married
0	3.10		2.0	Female	False
1	3.24		22.0	Female	True
[redacted]			2.0	Male	False
[redacted]			44.0	Male	True
4	5.30		7.0	Male	True
5	8.75		9.0	Male	True



	wage	[redacted]	exper	gender	married
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
[redacted]	3.00	9.0	2.0	Male	False
[redacted]	6.00	9.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
5	8.75	16.0	9.0	Male	True

Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Modify a subset of a [redacted] object

```
1 data_frame.iloc[ , ] = 1.0
2 data_frame
```

		0	[redacted]	3	4
		wage		gender	married
0	**0**	3.10		Female	False
1	**1**	3.24		Female	True
[redacted]					
3	**3**	6.00		Male	True
4	**4**	5.30		Male	True
5	**5**	8.75		Male	True

		0	[redacted]	3	4
		wage	educ	exper	gender	married
0	**0**	3.10	11.0	2.0	Female	False
1	**1**	3.24	12.0	22.0	Female	True
[redacted]	**2**	3.00	1.0	1.0	Male	False
3	**3**	6.00	9.0	44.0	Male	True
4	**4**	5.30	12.0	7.0	Male	True
5	**5**	8.75	16.0	9.0	Male	True

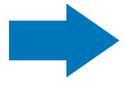
Pandas for Data Analysis

- Th [redacted] data structure
  e
  - Modify a subset of a **`pandas.DataFrame`** object

```
1  data_frame.loc[ , ] = 'none'
2  data_frame
```

New label added as a new column

	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	9.0	2.0	Male	False
3	6.00	9.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
5	8.75	16.0	9.0	Male	True



	wage	educ	exper	gender	married	
	3.10	11.0	2.0	Female	False	none
	3.24	12.0	22.0	Female	True	none
	3.00	1.0	1.0	Male	False	none
	6.00	9.0	44.0	Male	True	none
	5.30	12.0	7.0	Male	True	none
	8.75	16.0	9.0	Male	True	none

Pandas for Data Analysis

- The [redacted] data structure
  - Boolean **`pandas.Series`** and boolean indexing

A subset for female workers?

	wage	educ	exper	gender	married	remarks
0						
1						
2						
3						
4						
5						

Pandas for Data Analysis

- The [redacted] data structure
  - Boolean [redacted] and boolean indexing

```
1 is_female = data_frame['gender'] == 'Female'
2 is_female
```

```
0     True
1     True
2    False
3    False
4    False
5    False
Name: gender, dtype: bool
```

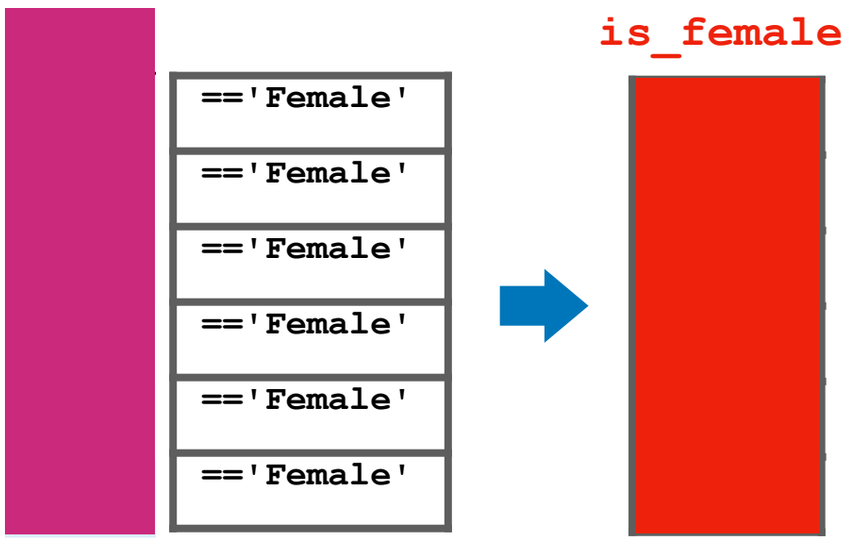
	wage	educ	exper	gender	married	remarks
0	3.10	11.0	2.0	Female	False	none
1	3.24	12.0	22.0	Female	True	none
2	3.00	1.0	1.0	Male	False	none
3	6.00	9.0	44.0	Male	True	none
4	5.30	12.0	7.0	Male	True	none
5	8.75	16.0	9.0	Male	True	none

Pandas for Data Analysis

- The [redacted] data structure
  - Boolean [redacted] and boolean indexing

```
1 is_female = data_frame['gender'] == 'Female'
2 is_female
```

```
0     True
1     True
2    False
3    False
4    False
5    False
Name: gender, dtype: bool
```

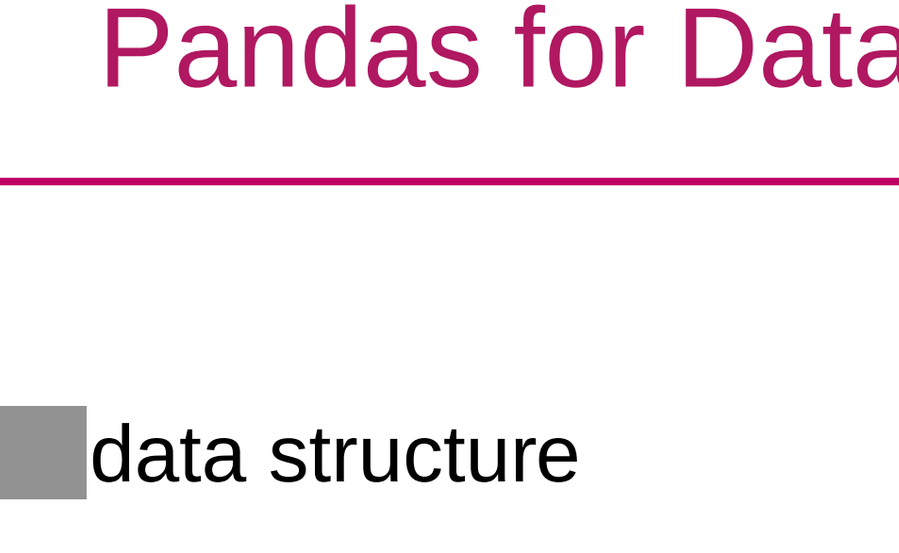


Pandas for Data Analysis

- The [redacted] data structure
  - Boolean [redacted] and boolean indexing

```
1  is_high_wage = data_frame['wage'] > 4
2  is_high_wage
```

```
0    False
1    False
2    False
3     True
4     True
5     True
Name: wage, dtype: bool
```

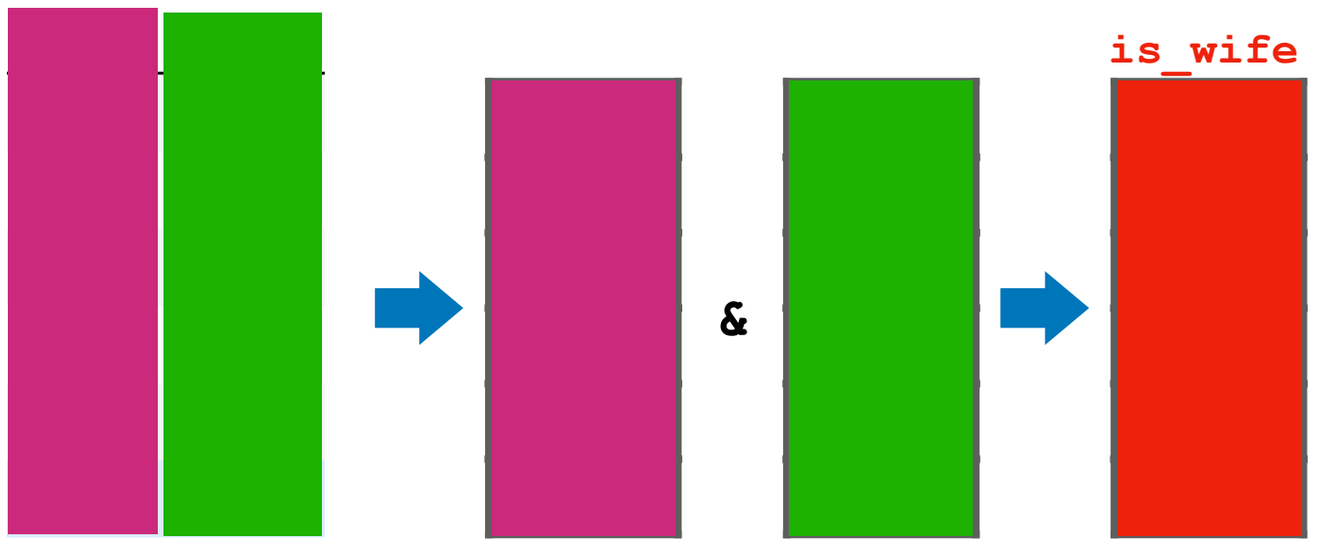


Pandas for Data Analysis

- The [redacted] data structure
  - Boolean [redacted] and boolean indexing

```
3  is_wife = (data_frame['gender']=='Female') & (data_frame['married'])
4  is_wife
```

```
0    False
1     True
2    False
3    False
4    False
5    False
dtype: bool
```



Pandas for Data Analysis

- The [redacted] data structure
  - Boolean [redacted] and boolean indexing

```
1 is_female = data_frame['gender'] == 'Female'
2 female = data_frame.loc[is_female]
3 female
```

is_female		wage	educ	exper	gender	married	remarks
FALSE	**2**	3.00	1.0	1.0	Male	False	none
FALSE	**3**	6.00	9.0	44.0	Male	True	none
FALSE	**4**	5.30	12.0	7.0	Male	True	none
FALSE	**5**	8.75	16.0	9.0	Male	True	none

Pandas for Data Analysis

- The [redacted] data structure
  - Boolean [redacted] and boolean indexing

```
1 [redacted] = data_frame['gender'] == 'Female'
2 female = data_frame.loc[redacted]
3 female
```

	wage	educ	exper	gender	married	remarks
0	3.10	11.0	2.0	Female	False	none
1	3.24	12.0	22.0	Female	True	none

	wage	educ	exper	gender	married	remarks
[redacted]						
2	3.00	1.0	1.0	Male	False	none
3	6.00	9.0	44.0	Male	True	none
4	5.30	12.0	7.0	Male	True	none
5	8.75	16.0	9.0	Male	True	none

Pandas for Data Analysis

- The [redacted] data structure
  - Boolean [redacted] and boolean indexing

```
1 is_wife = (data_frame['gender']=='Female') & (data_frame['married'])
2 data_frame.loc[is_wife, 'remarks'] = 'Wife'
3 data_frame
```

is_wife		wage	educ	exper	gender	married	[redacted]
FALSE	0	3.10	11.0	2.0	Female	False	[redacted]
[redacted]							
FALSE	2	3.00	1.0	1.0	Male	False	none
FALSE	3	6.00	9.0	44.0	Male	True	none
FALSE	4	5.30	12.0	7.0	Male	True	none
FALSE	5	8.75	16.0	9.0	Male	True	none

Pandas for Data Analysis

- The [redacted] data structure
  - Boolean [redacted] and boolean indexing

```
1 [redacted] = (data_frame['gender']=='Female') & (data_frame['married'])
2 data_frame.loc[redacted] = 'Wife'
3 data_frame
```

is_wife		wage	educ	exper	gender	married	[redacted]
FALSE	0	3.10	11.0	2.0	Female	False	[redacted]
[redacted]							
FALSE	2	3.00	1.0	1.0	Male	False	none
FALSE	3	6.00	9.0	44.0	Male	True	none
FALSE	4	5.30	12.0	7.0	Male	True	none
FALSE	5	8.75	16.0	9.0	Male	True	none

Pandas for Data Analysis

- The **`pandas.DataFrame`** data structure
  - Boolean **`pandas.Series`** and boolean indexing

Question

Please change values of the column “remarks” by yourself according to the following rule:

-The value for married male is changed to the string 'Husband';

-The value for unmarried male or female is changed to the string ‘Single'.

Roadmap

1. Array Data Structure
2. Data Visualization
3. Pandas
4. **Descriptive Analytics in Python**

Descriptive Analytics in Python

"You can, for example, never foretell what any one man will do, but you can say with precision what an average number will be up to. Individuals vary, but percentages remain constant"

–Sherlock Holmes, A Sign of Four

Descriptive Analytics in Python

- Read Data from files

	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
...	...	...	...	...	...
521	15.00	16.0	14.0	Female	True
522	2.27	10.0	2.0	Female	False
523	4.67	15.0	13.0	Male	True
524	11.56	16.0	5.0	Male	True
525	3.50	14.0	5.0	Female	False

526 rows × 5 columns

Descriptive Analytics in Python

- Read Data from files

```
data = pd.read_csv('wage.csv')  # Read data from a file "wage.csv"
data.head(6)                    # Return the first six rows of data
```

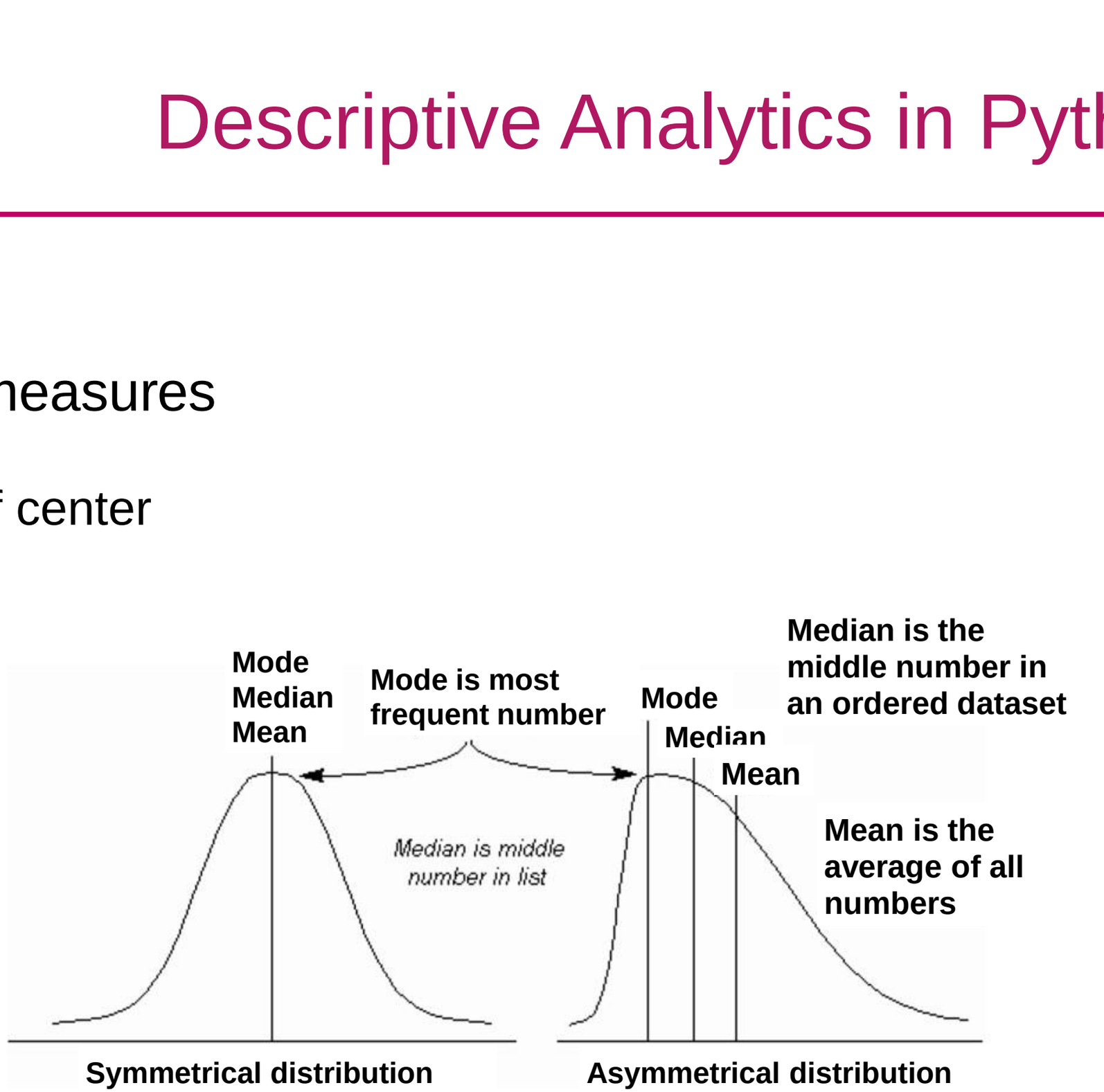
	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
5	8.75	16.0	9.0	Male	True

	wage	educ	exper	gender	married
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
...	...	...	...	...	...
521	15.00	16.0	14.0	Female	True
522	2.27	10.0	2.0	Female	False
523	4.67	15.0	13.0	Male	True
524	11.56	16.0	5.0	Male	True
525	3.50	14.0	5.0	Female	False

526 rows × 5 columns

Descriptive Analytics in Python

- Descriptive measures
  - Measures of center
    - ✓ **Mean**
    - ✓ **Median**
    - ✓ **Mode**



Descriptive Analytics in Python

- Descriptive Measures
  - ► Measures of center
    - ✓ **Mean**

```
1  print(data.mean(), '\n')      # Mean value
2  print(type(data.mean()))      # Show the d
```

```
dtype: float64
```

				gender	
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
...	...	...	...	...	...
521	15.00	16.0	14.0	Female	True
522	2.27	10.0	2.0	Female	False
523	4.67	15.0	13.0	Male	True
524	11.56	16.0	5.0	Male	True
525	3.50	14.0	5.0	Female	False

Descriptive Analytics in Python

- Descriptive Measures
  - Measures of center
    - ✓ **Mean**

```
1  print(data.mean(), '\n')     # Mean value of each column
2  print(type(data.mean()))     # Show the data type of the results
```

```
wage          5.896103
educ         12.562738
exper        17.017110
[redacted]
dtype: float64

<class 'pandas.core.series.Series'>
```

$$\text{[redacted]} = \frac{1}{\text{Total case number}} \text{[redacted]}$$

Number of married cases

$$X_i = \begin{cases} 1, & \text{if True} \\ 0, & \text{if False} \end{cases}$$

Descriptive Analytics in Python

- Descriptive Measures
  - ► Measures of center
    - ✓ **Median**

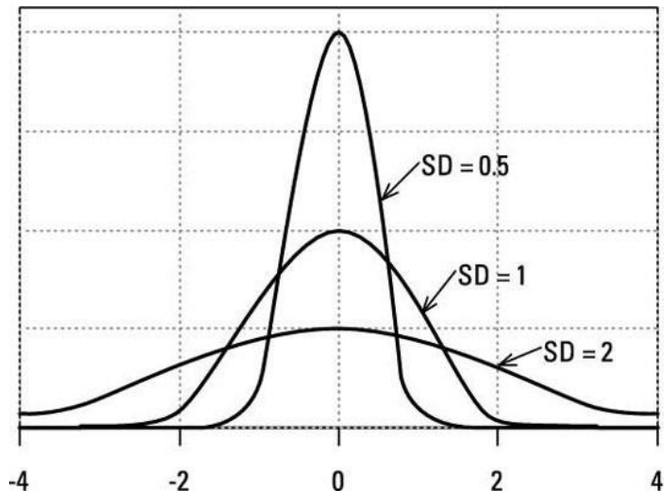
```
1  print(data.median(), '\n')          # Medi
2  print(                          )   # Show the d
```

```
dtype: float64
```

				gender	
0	3.10	11.0	2.0	Female	False
1	3.24	12.0	22.0	Female	True
2	3.00	11.0	2.0	Male	False
3	6.00	8.0	44.0	Male	True
4	5.30	12.0	7.0	Male	True
...	...	...	...	...	...
521	15.00	16.0	14.0	Female	True
522	2.27	10.0	2.0	Female	False
523	4.67	15.0	13.0	Male	True
524	11.56	16.0	5.0	Male	True
525	3.50	14.0	5.0	Female	False

Descriptive Analytics in Python

- Descriptive Measures
  - ► Measures of variation
    - ✓ **Standard deviation**
    - ✓ **Variance**

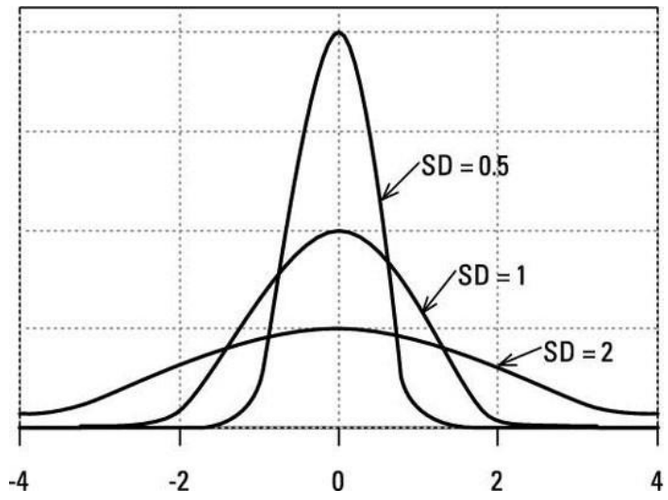



```
1  data.std()          # Sample Standard deviation of each column
```

```
wage         3.693086
educ         2.769022
exper       13.572160
married      0.488580
dtype: float64
```

Descriptive Analytics in Python

- Descriptive Measures
  - ► Measures of variation
    - ✓ **Standard deviation**
    - ✓ **Variance**




```
1  data.var()          # Sample Variance of each column
```

```
wage         13.638884
educ          7.667485
exper       184.203516
married       0.238711
dtype: float64
```

Descriptive Analytics in Python

- Descriptive Measures
  - Extreme points

✓ **Maximum**

```
1 data.max()          # Maximum value of each column
```

```
wage       24.98
educ          18
exper         51
gender      Male
married     True
dtype: object
```

✓ **Minimum**

```
1 data.min()          # Minimum value of each column
```

```
wage          0.53
educ             0
exper            1
gender      Female
married      False
dtype: object
```

Descriptive Analytics in Python

- Descriptive Measures
  - Counts of categorical values

```
1  data['gender'].value_counts()
```

```
Male      274
Female    252
Name: gender, dtype: int64
```

Actual count of each category (as default)

```
1  data['gender'].value_counts(normalize=True)
```

```
Male      0.520913
Female    0.479087
Name: gender, dtype: float64
```

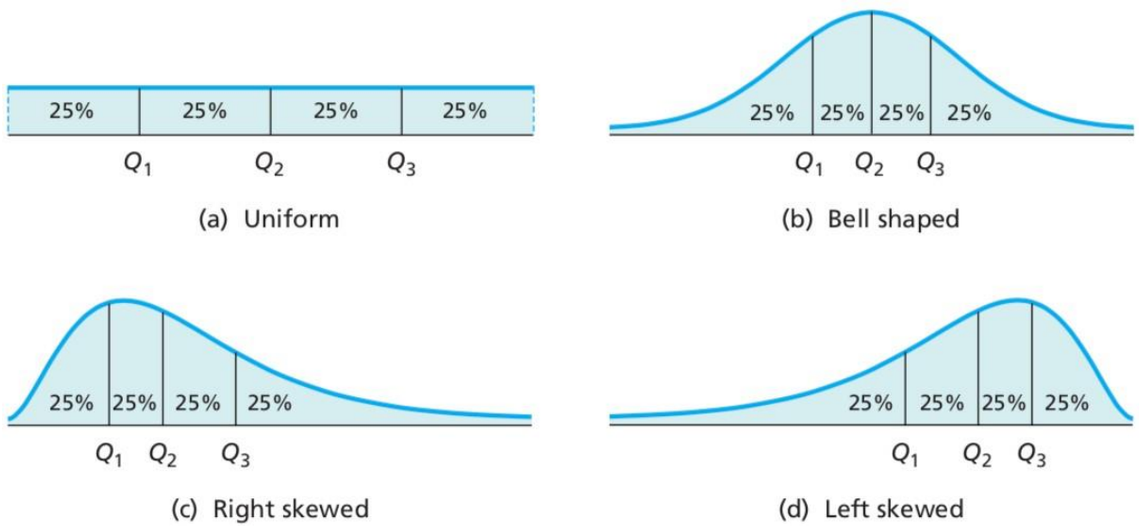
Proportion of each category

Descriptive Analytics in Python

- Descriptive Measures
  - Method **`describe()`**

```
1  wage_summary = data.describe()
2  wage_summary
```

	wage	educ	exper
count	526.000000	526.000000	526.00000
mean	5.896103	12.562738	17.01711
std	3.693086	2.769022	13.57216
min	0.530000	0.000000	1.00000
max	24.980000	18.000000	51.00000



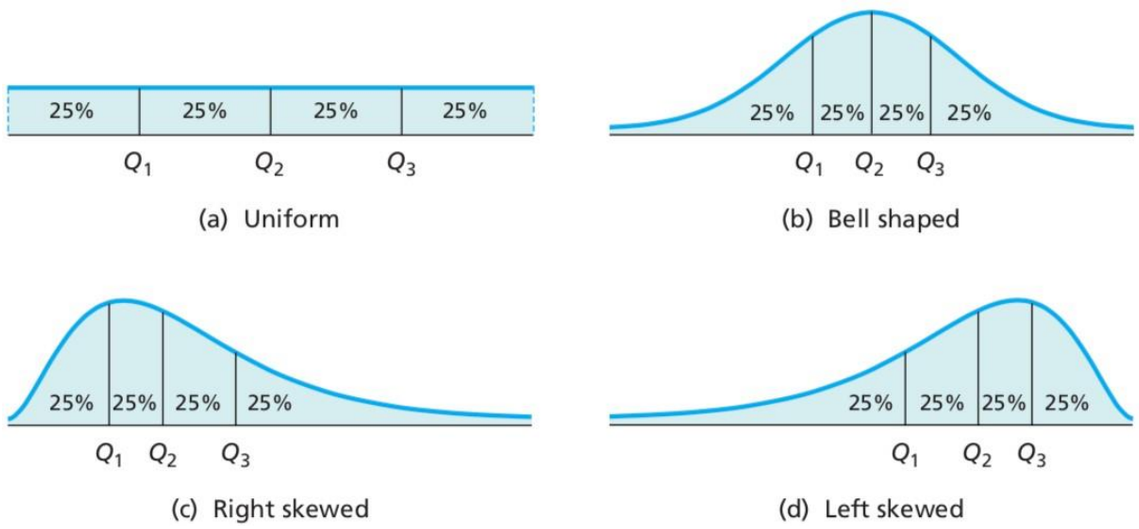
Q1 **Quartiles: values**
Q2 **that divide a dataset**
Q3 **into quarters:**

Descriptive Analytics in Python

- Descriptive Measures
  - Method [redacted]

```
1  wage_summary = data.describe()
2  wage_summary
```

	wage	educ	exper
count	526.000000	526.000000	526.00000
mean	5.896103	12.562738	17.01711
std	3.693086	2.769022	13.57216
min	0.530000	0.000000	1.00000
[redacted]			
max	24.980000	18.000000	51.00000



(a) Uniform

(b) Bell shaped

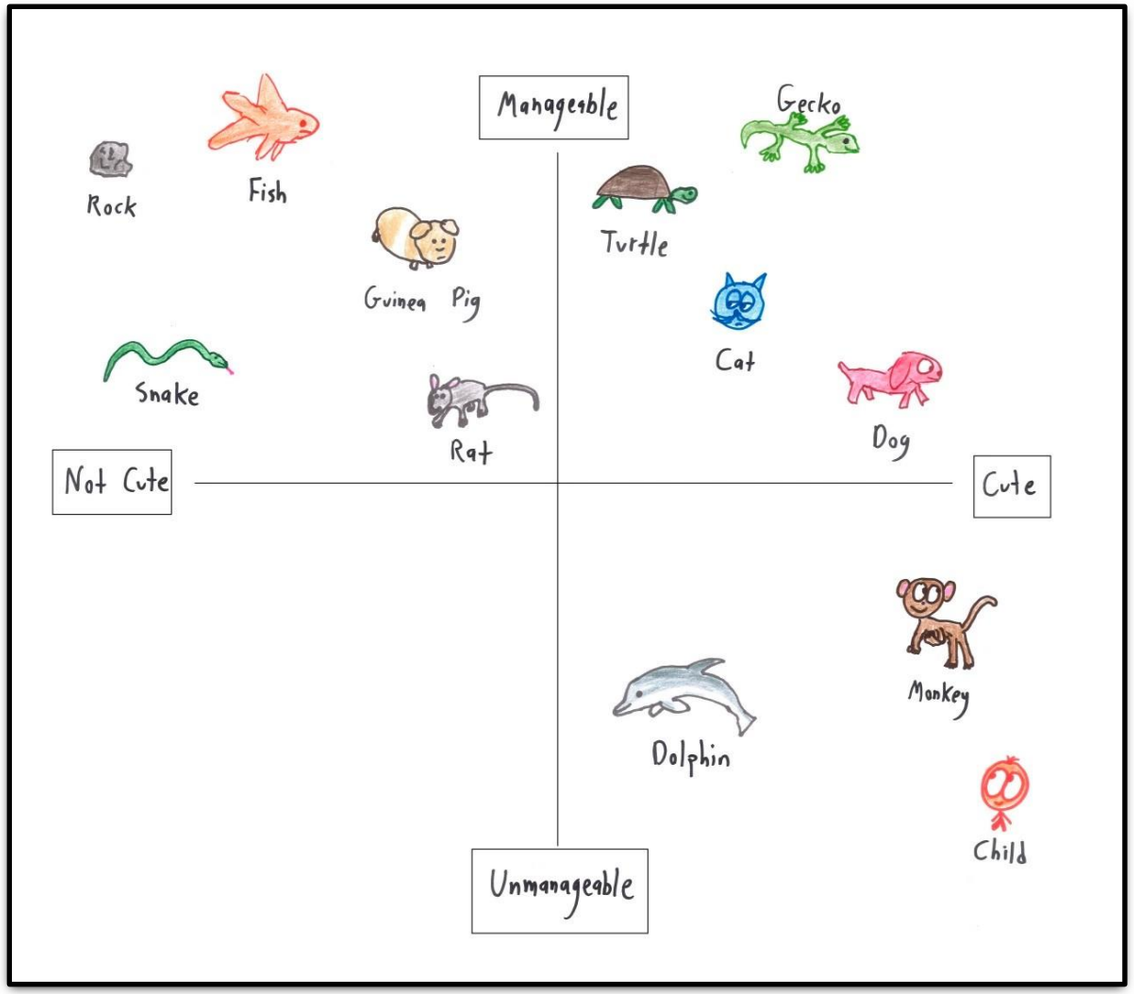
(c) Right skewed

(d) Left skewed

The value Q3 - Q1 is called the interquartile range (IQR).

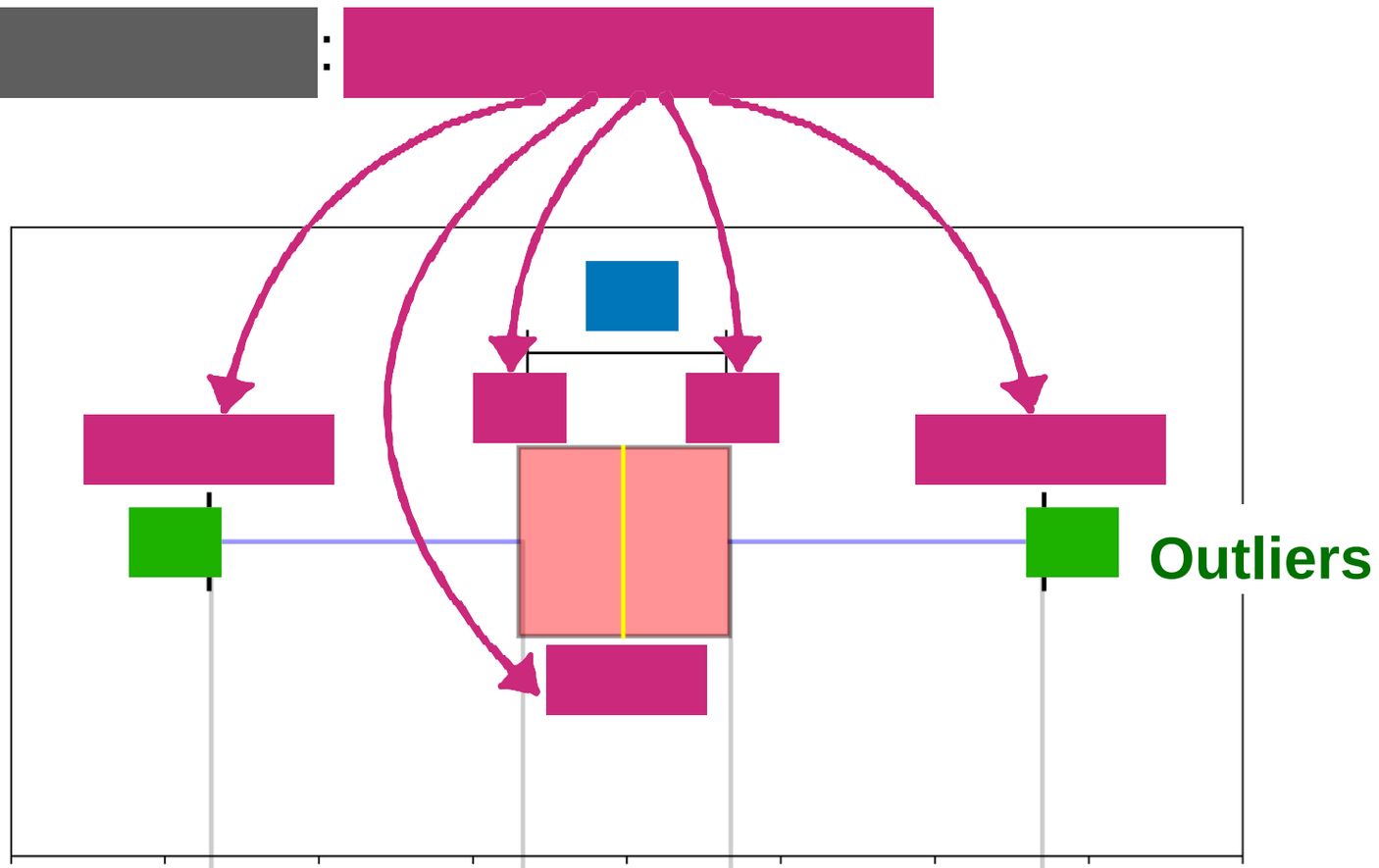
Descriptive Analytics in Python

- Data Visualization



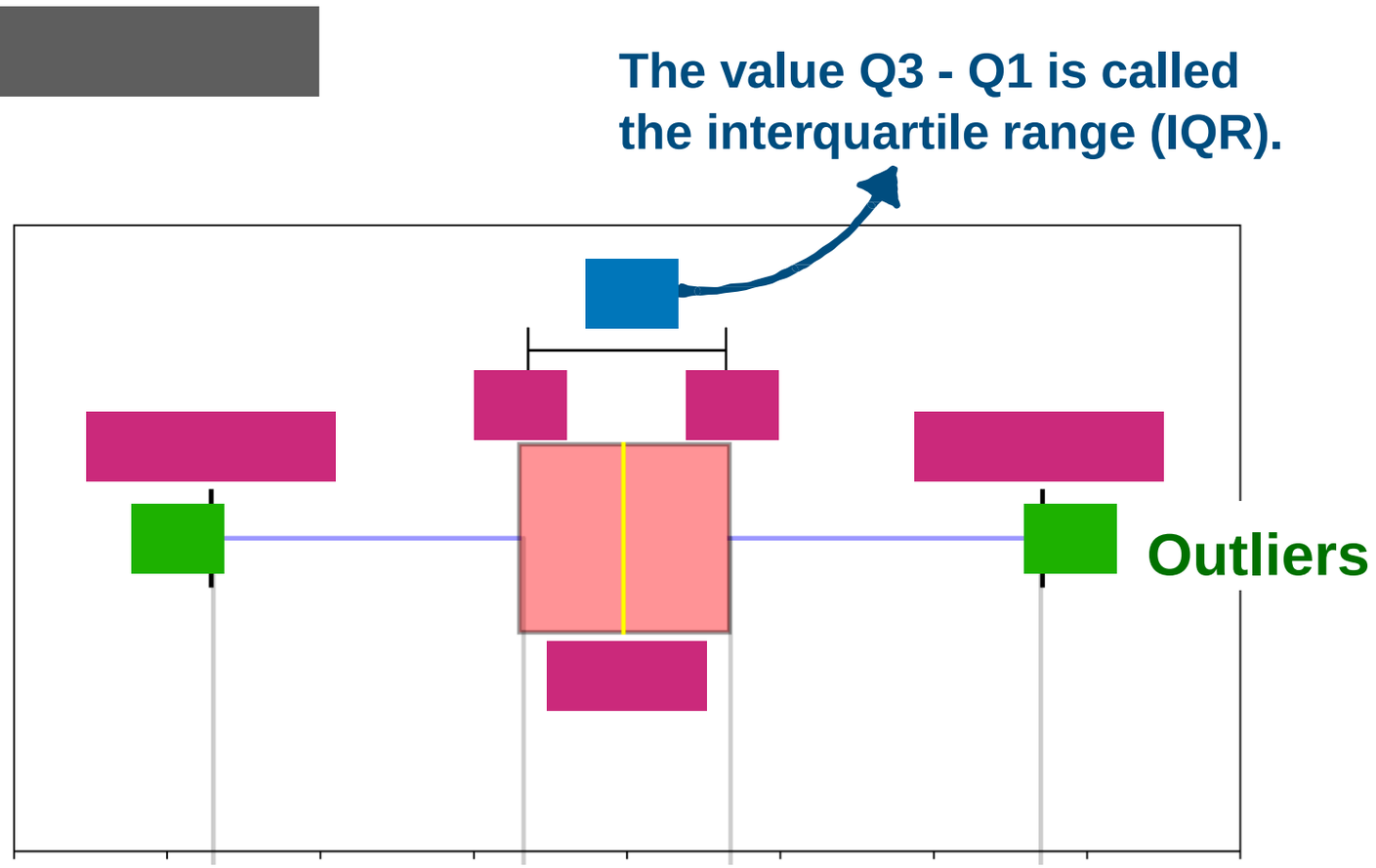
Descriptive Analytics in Python

- Data Visualization
  - Distributions of variables
    - ✓ Function [redacted]:
)



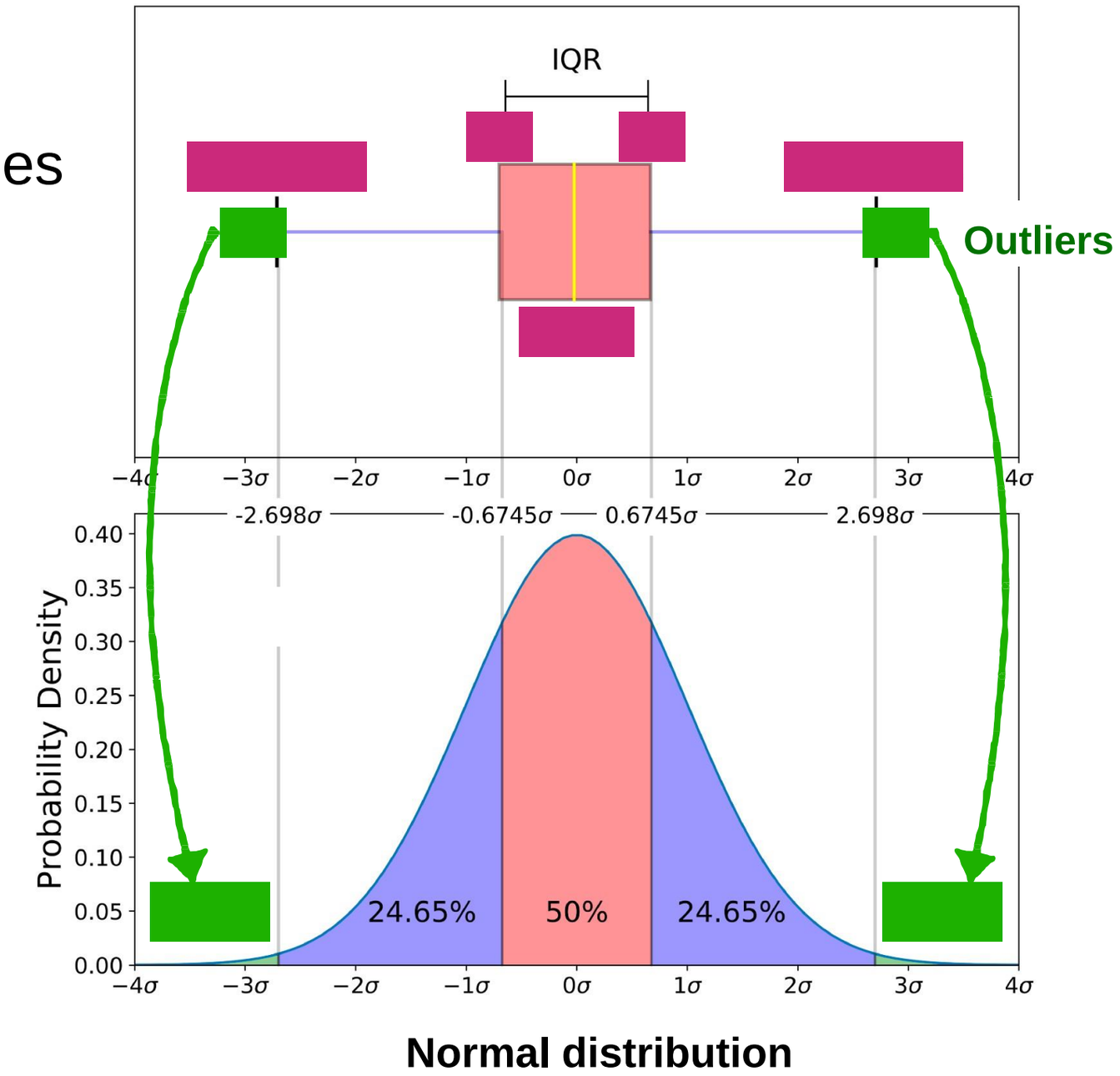
Descriptive Analytics in Python

- Data Visualization
  - Distributions of variables
    - ✓ Function [redacted])



Descriptive Analytics in Python

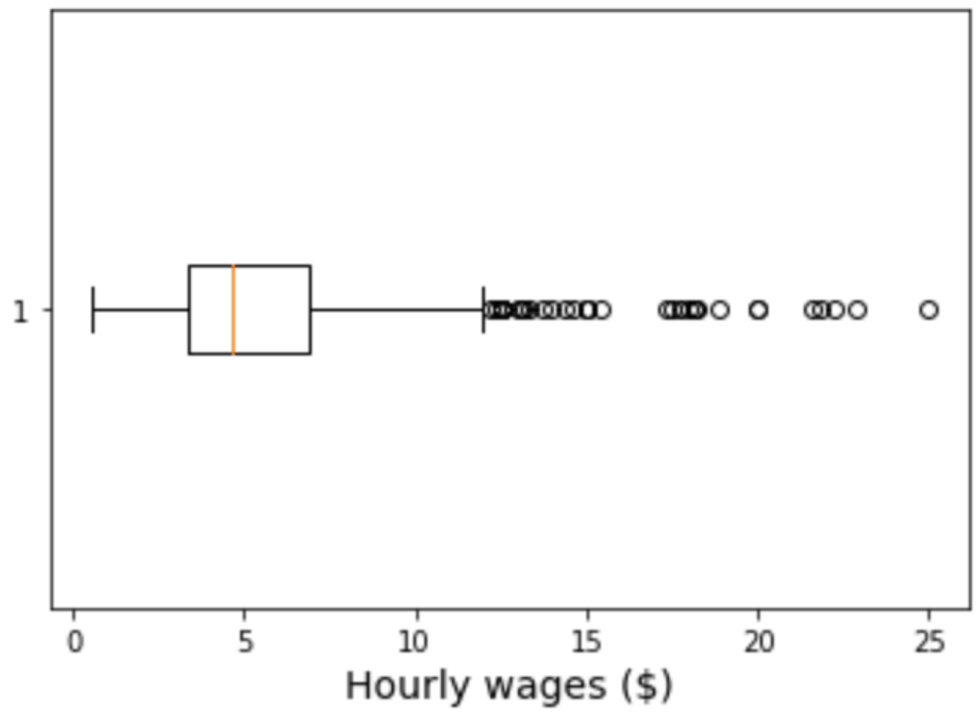
- Data Visualization
  - Distributions of variables
    - ✓ Function **`boxplot()`**



Normal distribution

Descriptive Analytics in Python

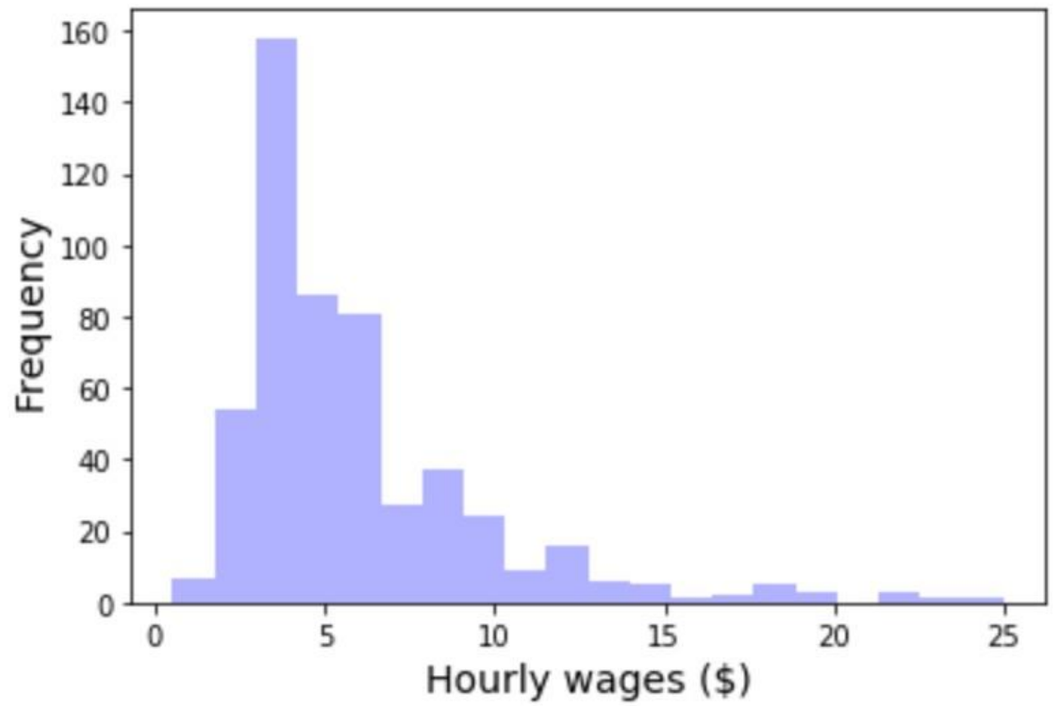
- Data Visualization
  - Distributions of variables
    - ✓ Package **`matplotlib`**
    - ✓ Function **`boxplot()`**




```
1  plt.boxplot(data['wage'],          # Create a box plot of the wages
2              vert=False);           # Make the plot horizontal
3
4  plt.xlabel('Hourly wages ($)',
5             fontsize=14)            # Label of the x axis
6
7  plt.show()
```

Descriptive Analytics in Python

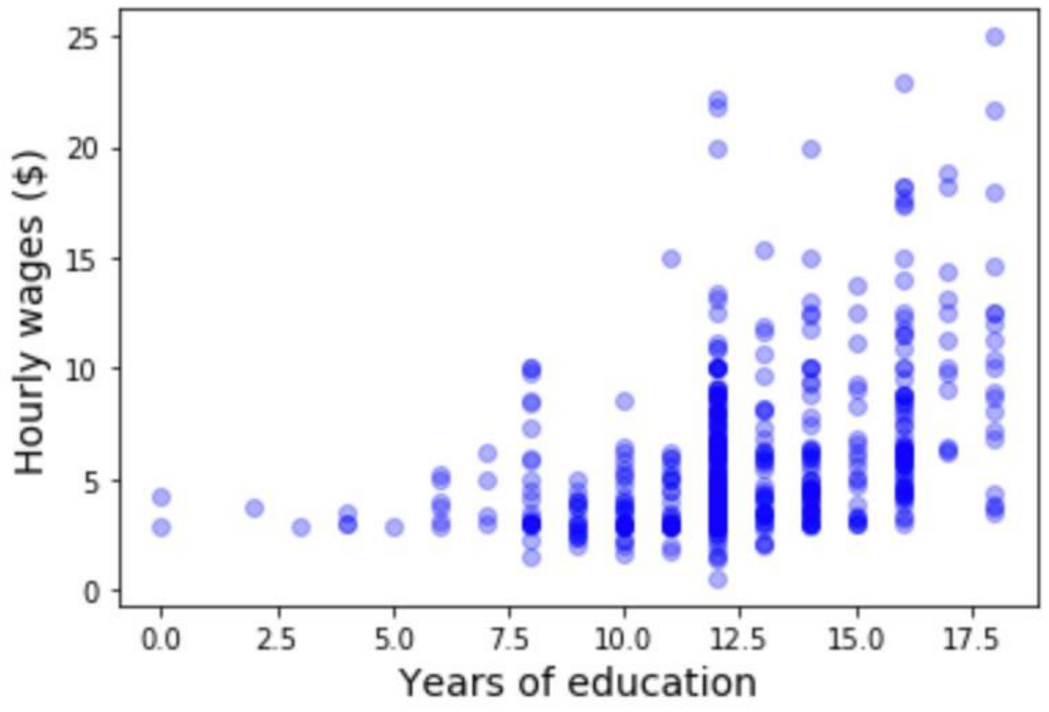
- Data Visualization
  - ▸ Distributions of variables
    - ✓ Package [redacted]
    - ✓ Function [redacted]




```
plt.hist(data['wage'], 20,       # Histogram of wages with 20 bins
         color='b', alpha=0.3); # Color is blue, opacity is 0.3

plt.xlabel('Hourly wages ($)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)

plt.show()
```

Descriptive Analytics in Python

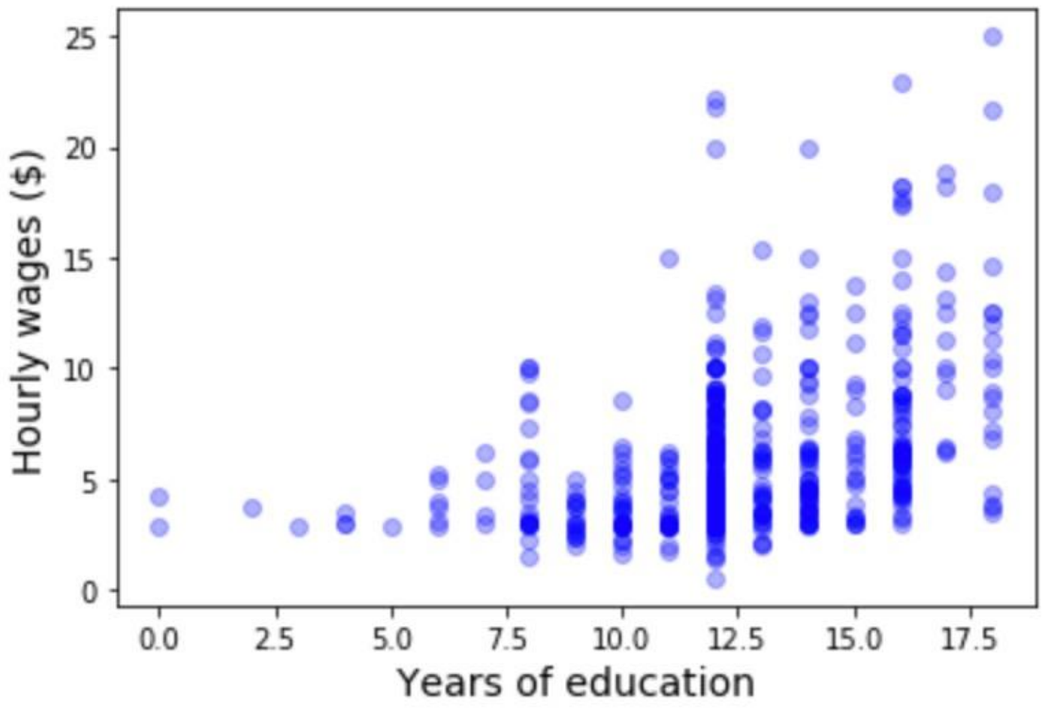
- Data Visualization
  - Relationship between variables




```
plt.scatter(data['educ'],        # x data is the years of education
            data['wage'],        # y data is the hourly wages
            c='b', alpha=0.3)    # Color is blue, opacity is 0.3

plt.xlabel('Years of education', fontsize=14)
plt.ylabel('Hourly wages ($)', fontsize=14)

plt.show()
```

Descriptive Analytics in Python

- Data Visualization
  - Relationship between variables




```
1 data.corr()
```

	wage	educ	exper	married
wage	1.000000		0.112903	0.228817
educ		1.000000	-0.299542	0.068881
exper	0.112903	-0.299542	1.000000	0.316984
married	0.228817	0.068881	0.316984	1.000000