COMS BC1016

Introduction to Computational Thinking and Data Science

Lecture 7: Histograms and Functions (Continued)

Sept. 3, 2025

February 11, 2026

Logistics

- This week: My office hours are **Today** from 3-5
- HW 1 is due tonight at 11:59pm
  - Please remember to submit it as a **.ipynb** file
    - If you’ve already submitted as a PDF please resubmit
    - I’m sorry it’s confusing that Labs are PDFs and HWs are .ipynb
  - HW 2 is due next week Wednesday
- General reminder for religious holidays: please contact me or your class dean ASAP if you require religious accommodations

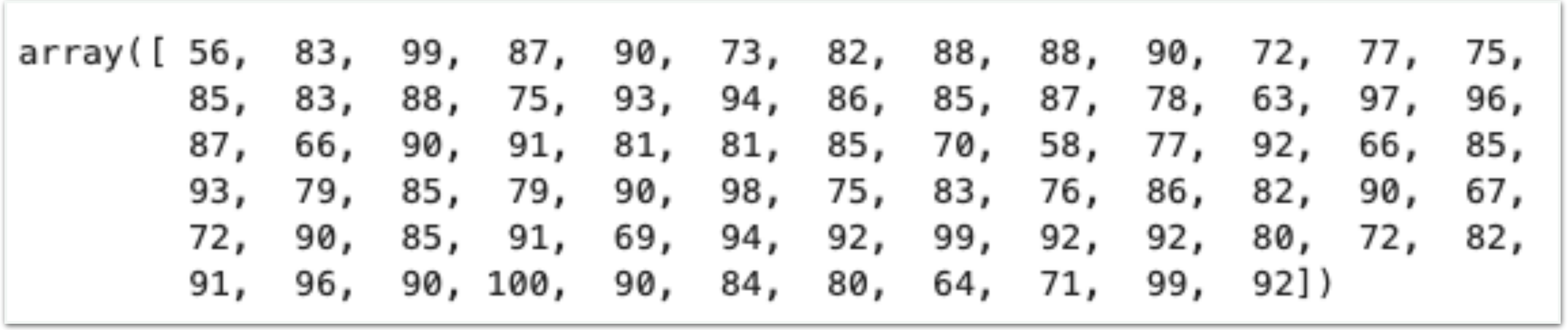
Midterm Info

- Midterm is in roughly 4 weeks on **March 11**
    - Exam will cover material up until Wednesday, March 4
        - Depending on how we're doing, I may convert this lecture into another review session
    - TAs will lead a midterm review during class Monday, March 9
- Paper exam (you will *not* be asked to program on a computer or write code on paper)
    - Can create and bring a single sheet with notes on the exam (to be submitted along with your exam)
    - Please contact CARDS/ODS to arrange disability accommodations

Last Time: Histograms

Visualizing Numerical Distributions

Let’s say we have a data set containing grades students scored on an exam:



```
array([ 56,  83,  99,  87,  90,  73,  82,  88,  88,  90,  72,  77,  75,
        85,  83,  88,  75,  93,  94,  86,  85,  87,  78,  63,  97,  96,
        87,  66,  90,  91,  81,  81,  85,  70,  58,  77,  92,  66,  85,
        93,  79,  85,  79,  90,  98,  75,  83,  76,  86,  82,  90,  67,
        72,  90,  85,  91,  69,  94,  92,  99,  92,  92,  80,  72,  82,
        91,  96,  90, 100,  90,  84,  80,  64,  71,  99,  92])
```

What if we want to know generally how students on the exam?

- How many students got between 90 and 100?
- What range of values did the majority of students fall into?

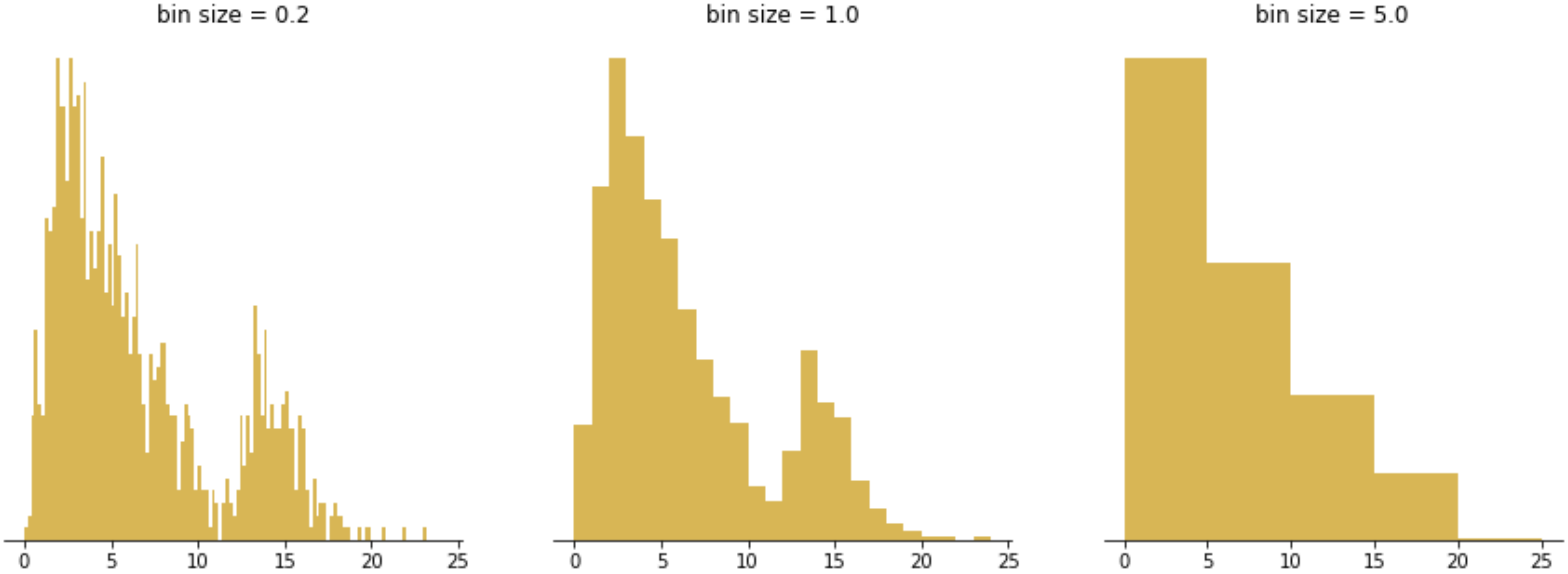
Visualizing Numerical Distributions

Histograms display the distribution of a numerical value

- Makes use of **bins** (each bar corresponds to an individual bin)
  - A **bin** refers to a range of numerical values and **binning** counts the number of values that lie within that range
  - Binning converts a numerical distribution into a categorical distribution
- Makes use of the **area principle**

Choosing Bin Size

Choose so that it's representative of your data



bin size = 0.2

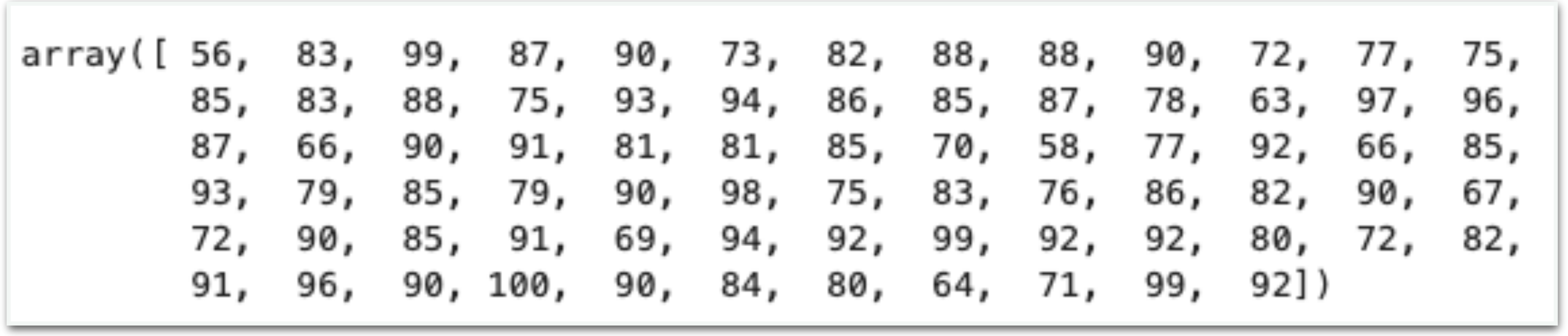
bin size = 1.0

bin size = 5.0

https://www.atlassian.com/data/charts/histogram-complete-guide

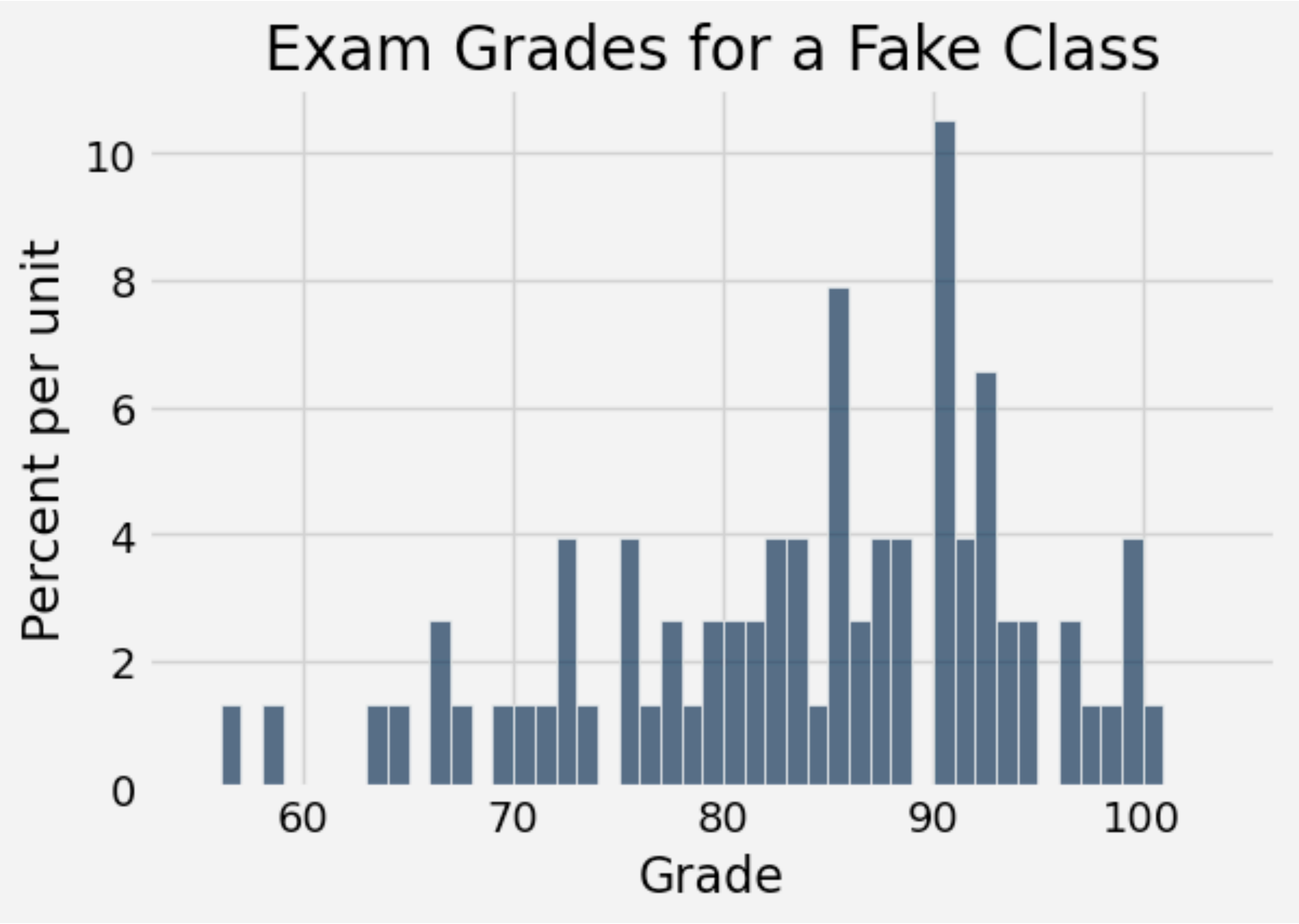
Choosing Bin Size

Let’s go back to our data from before:

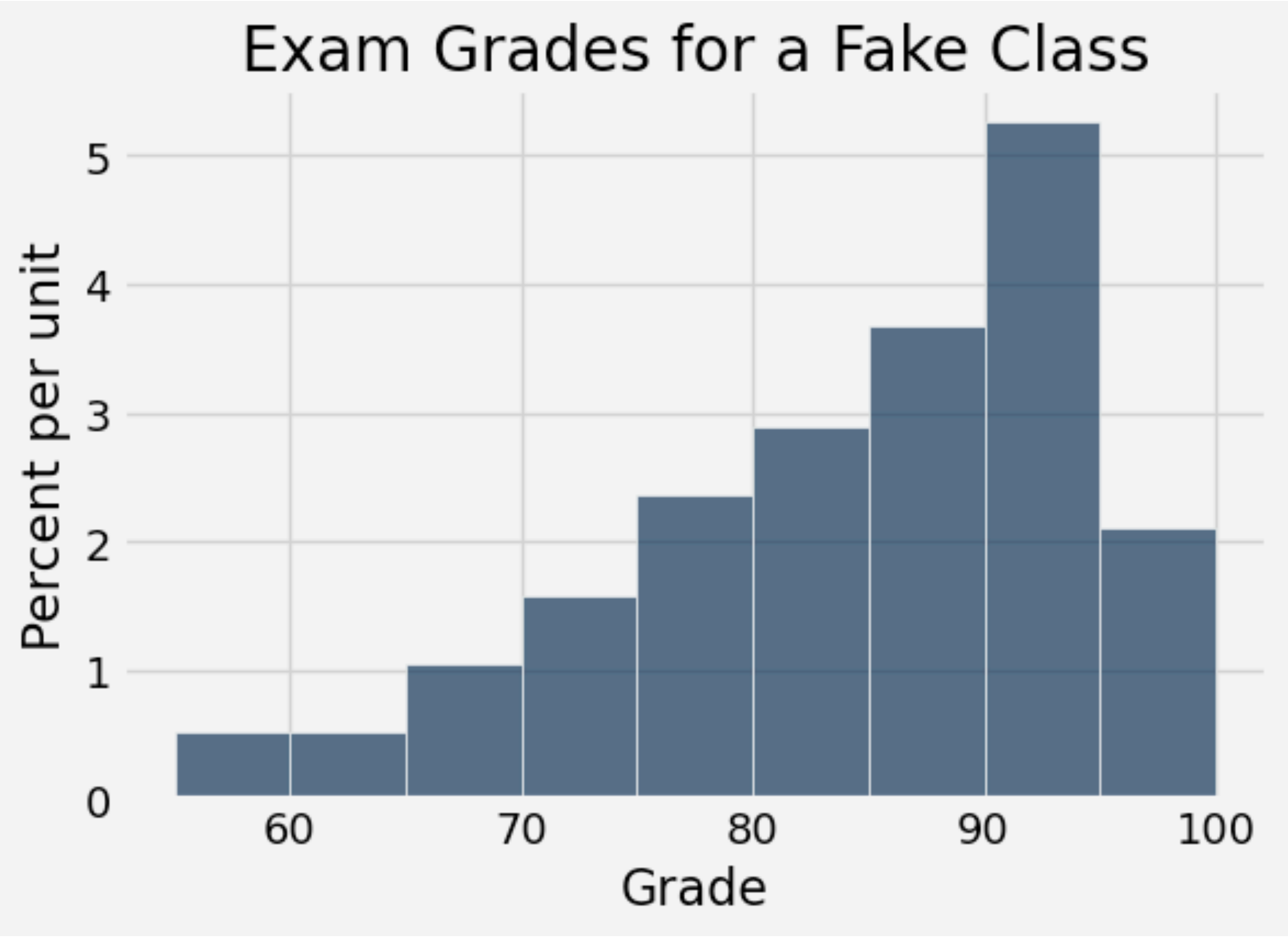


```
array([ 56,  83,  99,  87,  90,  73,  82,  88,  88,  90,  72,  77,  75,
        85,  83,  88,  75,  93,  94,  86,  85,  87,  78,  63,  97,  96,
        87,  66,  90,  91,  81,  81,  85,  70,  58,  77,  92,  66,  85,
        93,  79,  85,  79,  90,  98,  75,  83,  76,  86,  82,  90,  67,
        72,  90,  85,  91,  69,  94,  92,  99,  92,  92,  80,  72,  82,
        91,  96,  90, 100,  90,  84,  80,  64,  71,  99,  92])
```

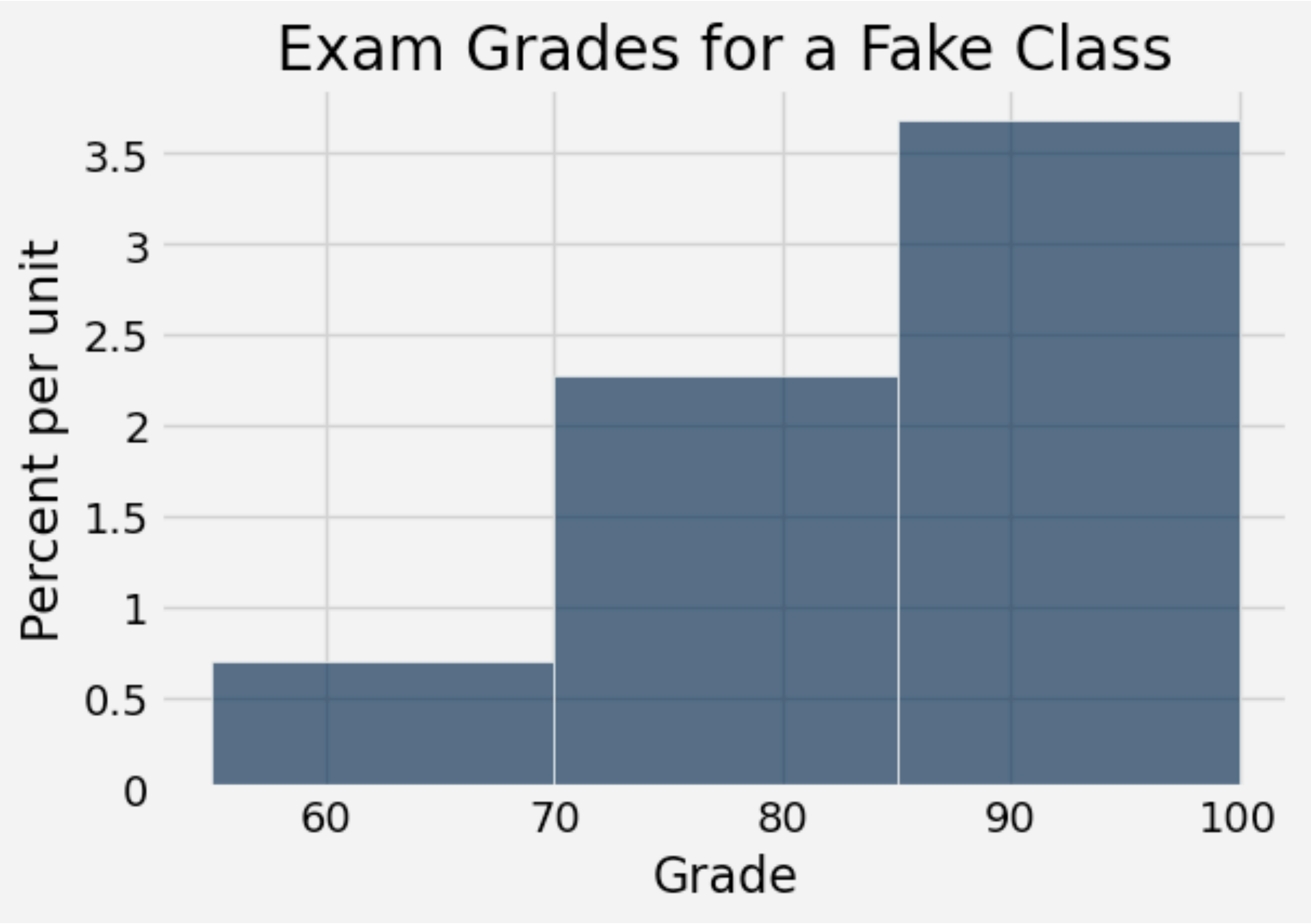
Bin size = 1



Bin size = 5



Bin size = 15



bin

Group values in column `c` into 10 equally sized intervals:

- `tbl.bin(c)`

Create `n` equally wide bins:

- `tbl.bin(c, bins=n)`

Create bins of size `step` from `start` to `end`:

- `tbl.bin(c, bins=np.arange(start, end, step))`

`hist`

Create a histogram of numerical values in column `c` with 10 equal bins:

- `tbl.hist(c)`

Create a histogram with `u` as the x-axis:

- `tbl.hist(c, unit=u)`

Create a histogram with specified bins:

- `tbl.hist(c, bins=np.arange(start, end, step))`

Create a histogram with x-axis `u` and specified bins:

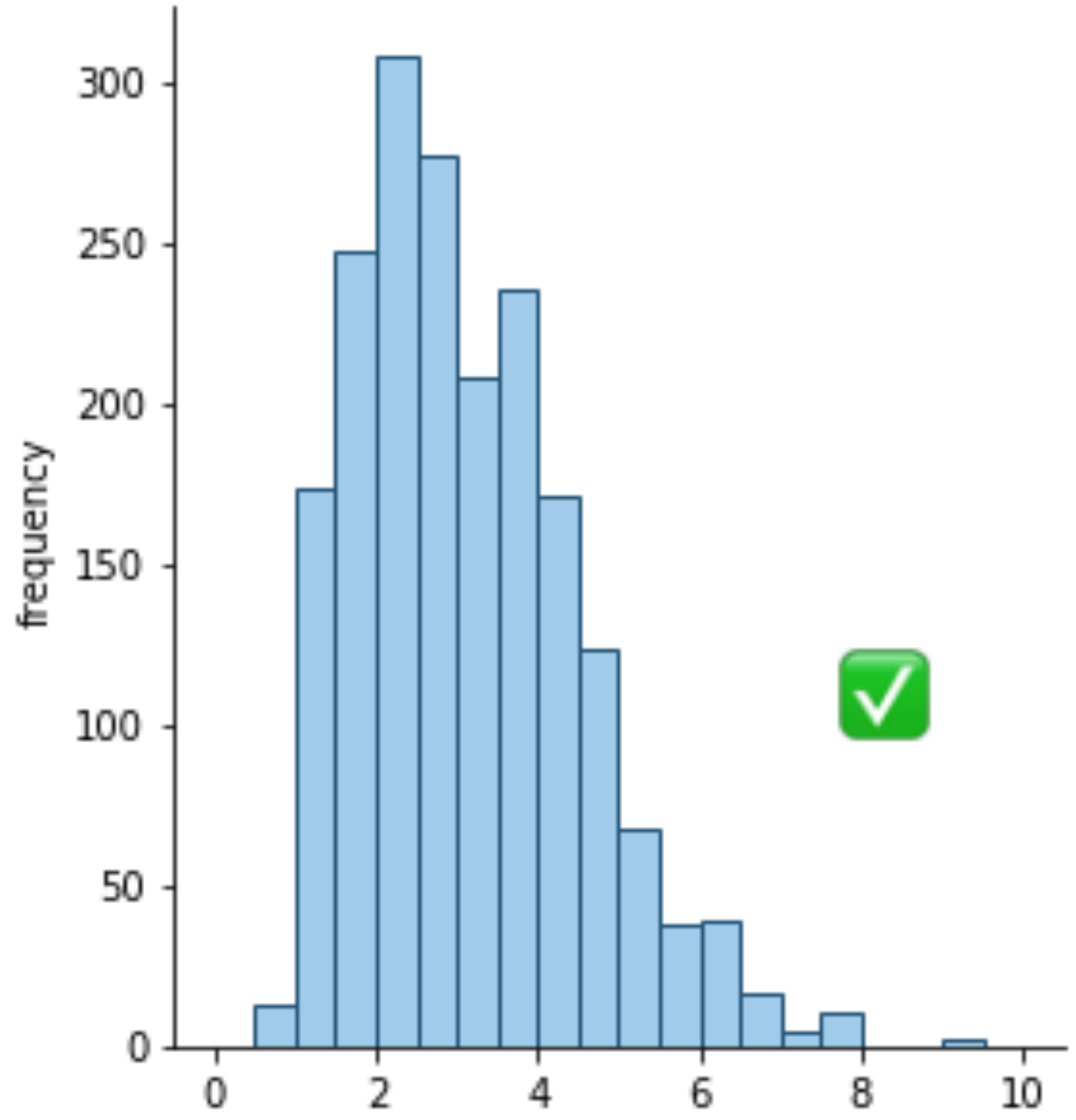
- `tbl.hist(c, unit=u, bins=np.arange(start, end, step))`

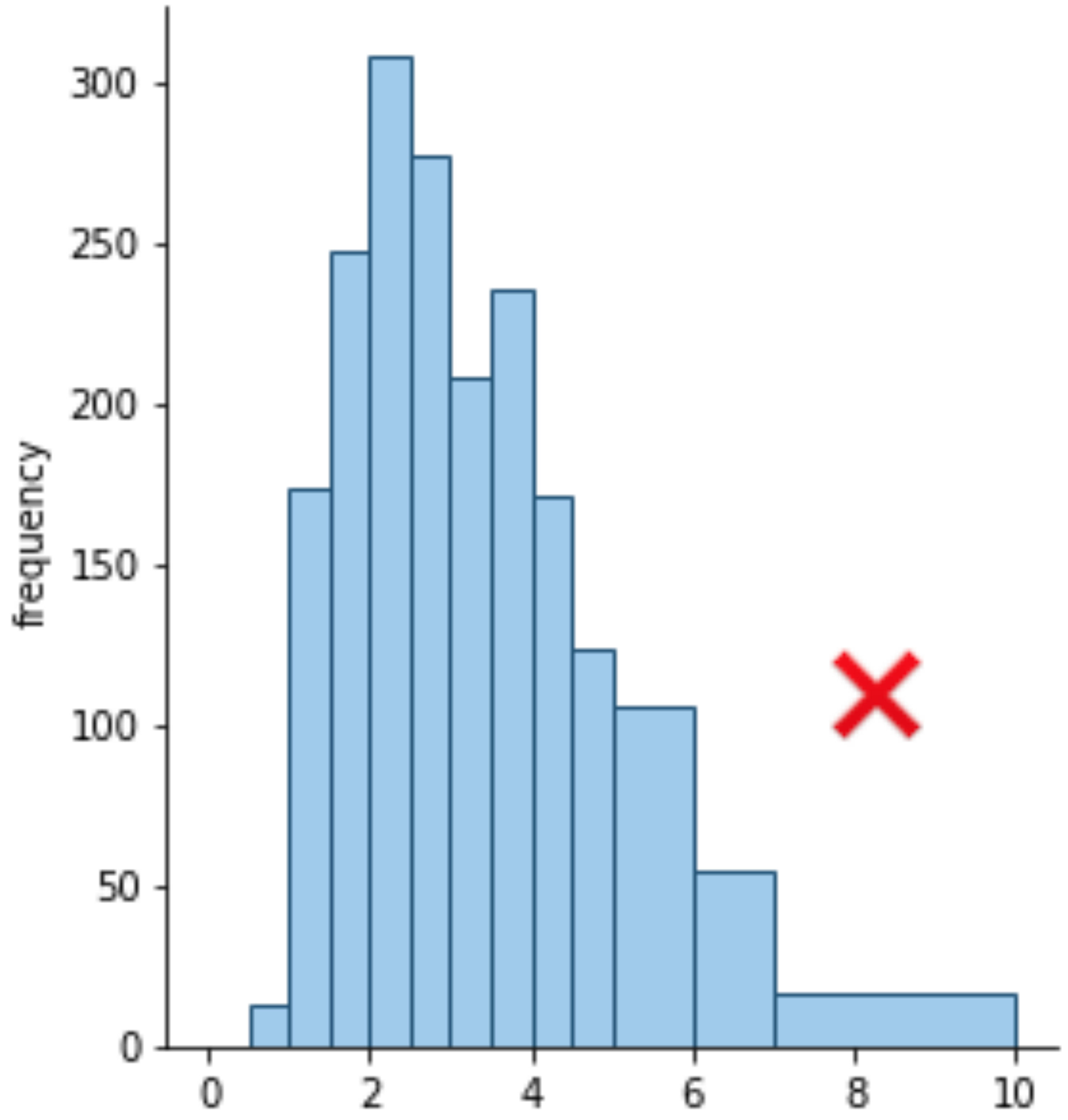
More on Histograms

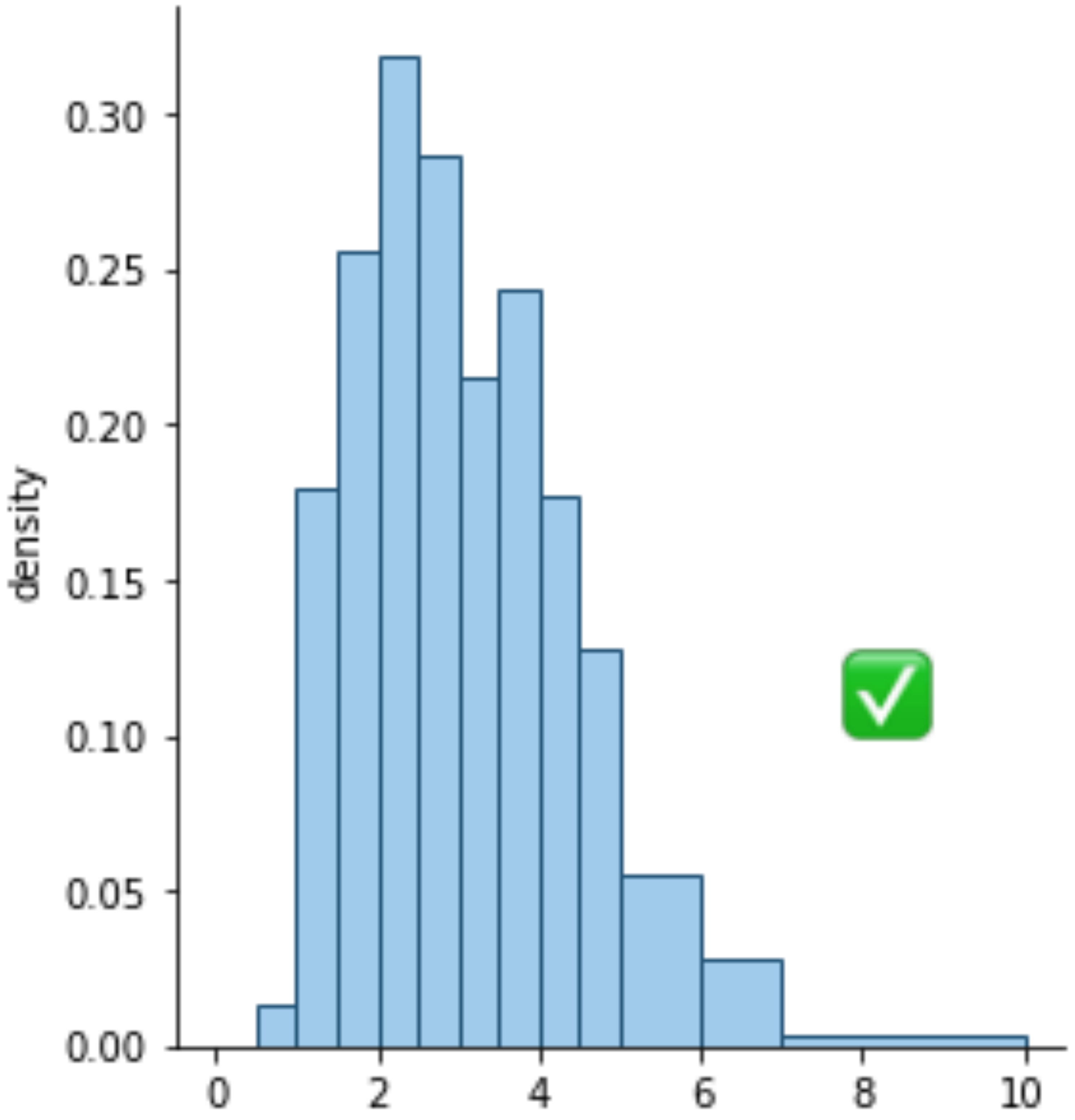
Unequal Bin Sizes

Bin sizes don’t need to be equal - unequal bin size is often used for better representing tails

For unequal bin sizes - vertical axis now represents ***density*** rather than frequency







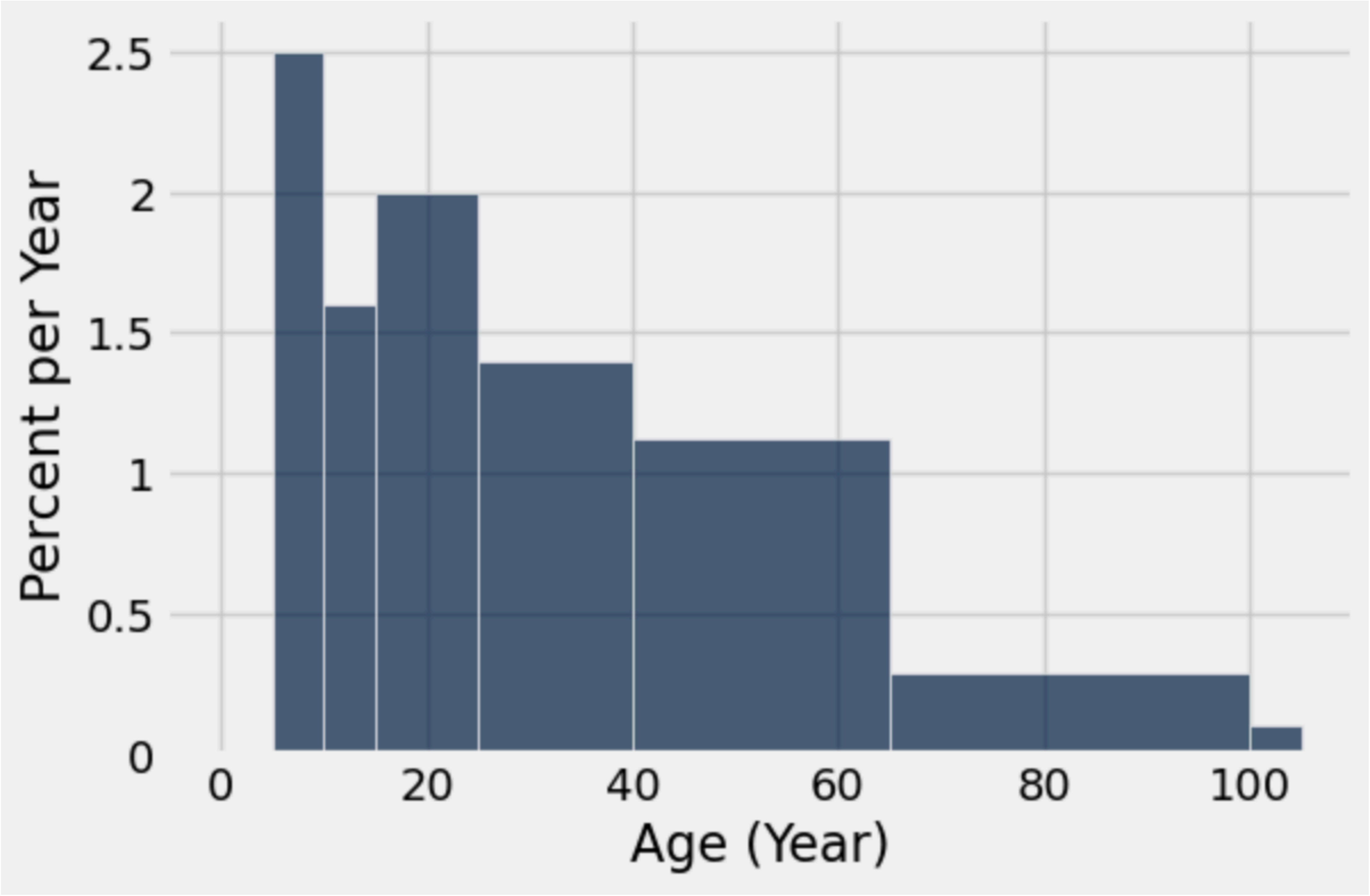
https://www.atlassian.com/data/charts/histogram-complete-guide

Histograms

The area of each bar is a percentage of the whole

The horizontal axis is a numerical distribution
the bins don’t need to be of equal size

The vertical axis is a rate
(e.g., percent/year) - density

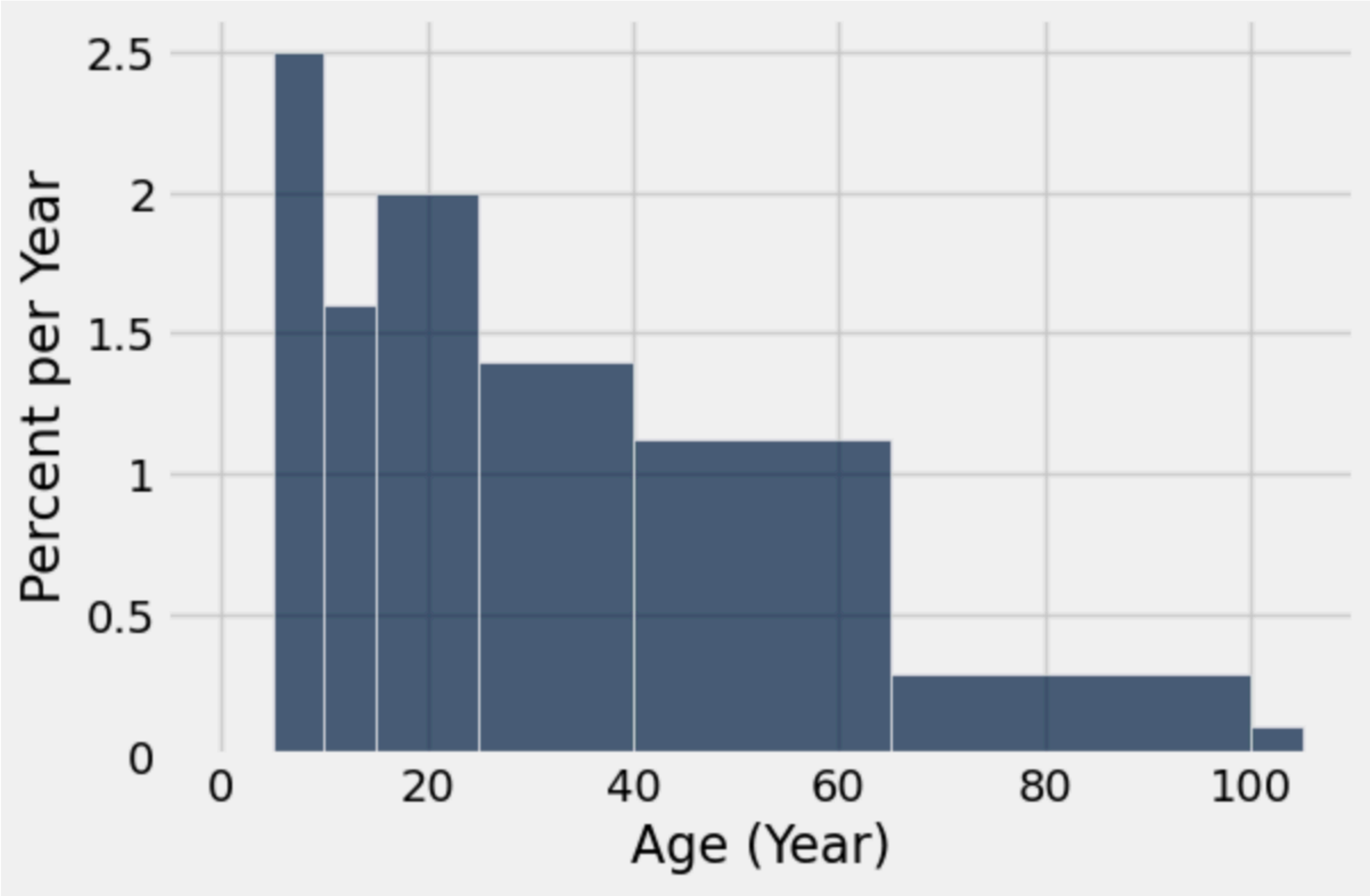


Histogram Formulas

The area of each bar is a percentage of the whole

$$\begin{aligned}\text{area of bar} &= (\text{height of bar}) \times (\text{width of bin}) \\ &= \text{percent of entries in bin}\end{aligned}$$

$$\begin{aligned}\text{height of bar} &= \frac{\text{percent of entries in bin}}{\text{width of bin}} \\ &= \frac{\text{area of bar}}{\text{width of bin}}\end{aligned}$$

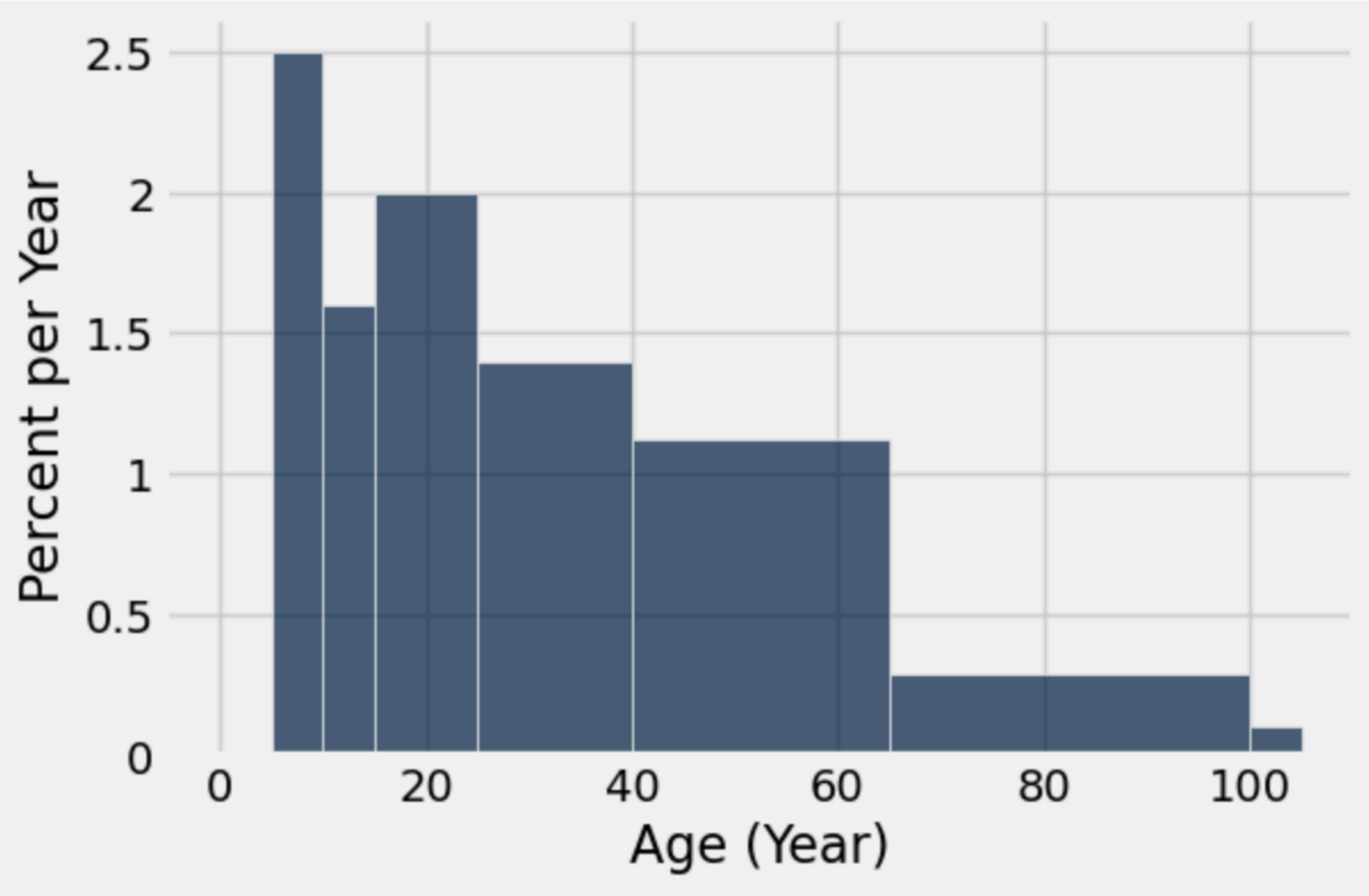


Calculating Heights

The [40, 65) bin contains 56/200 items

- The bin is 28% (56/200) of the whole
- The bin width is 65-40 = 25 years
- Height = $\dfrac{\text{28 percent}}{\text{25 years}}$

  = 1.12% per year



Area Notebook Demo

Bar Chart vs Histogram

Bar Chart

- Distribution of categorical variable

- Length of bars is proportional to the frequency / percent of individuals

Histogram

- Distribution of numerical variable

- Horizontal axis is numerical, bins can be unequal

- Area of bars is proportional of percent of individuals, height measures density

Charts Summary

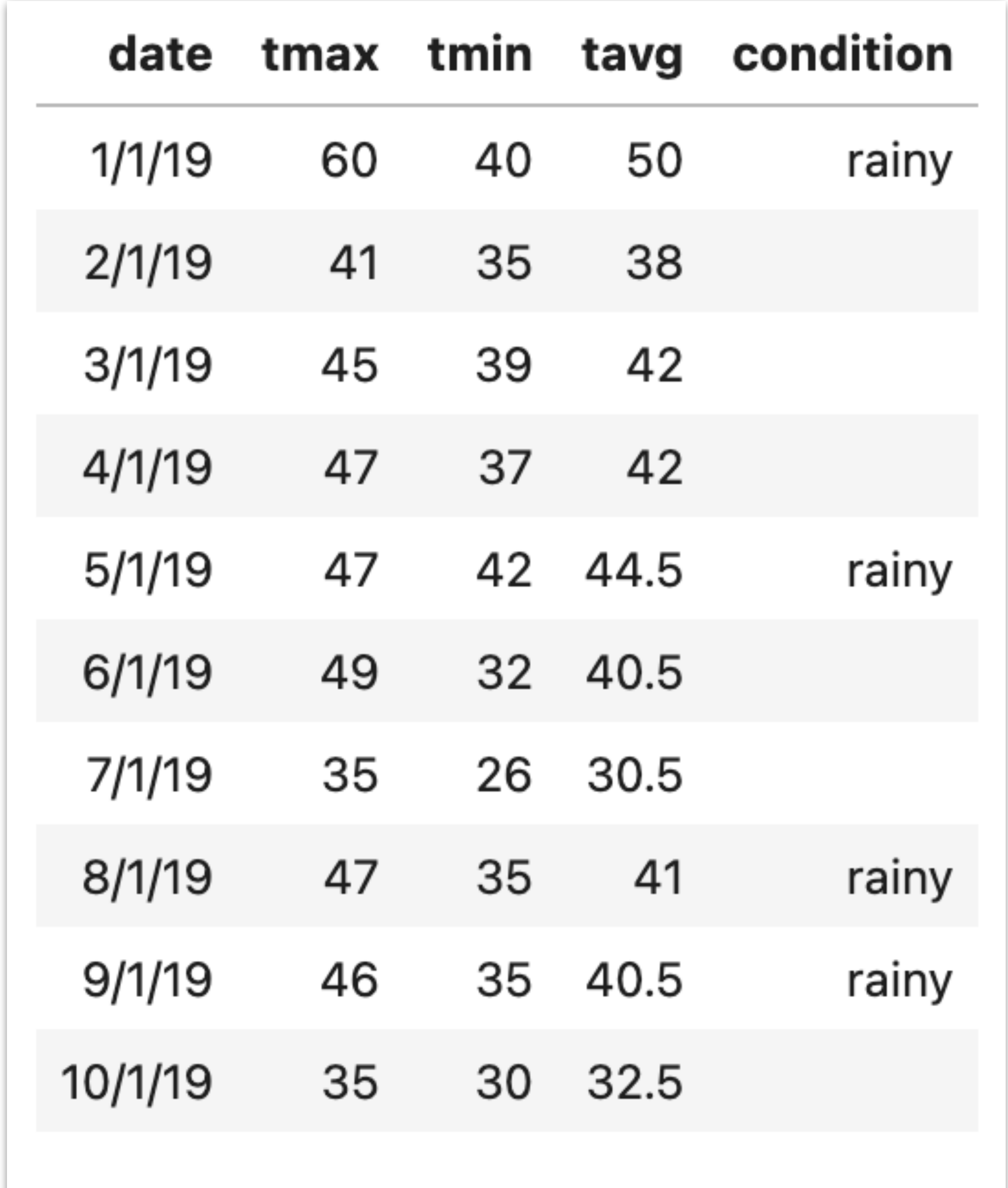
Type	Syntax	Description
Line graph	`.plot(x_axis, y_axis)`	Sequential numerical data
Scatter Plot	`.scatter(x_axis, y_axis)`	Relation between two numerical values
Bar Chart	`.barh(column_label)`	Distribution of one categorical variable (already grouped)
Histogram	`.hist(column_label, unit, bins)`	Distribution of one numerical variable

Chart Selection Exercise

We have NYC weather data from 2019 as shown below (from Kaggle)

Which type of chart (line, scatter, bar, histogram) would best help you answer to each question?

- Do days with hotter highs also tend to have hotter lows?
- How do the number of rainy days compare with the number of snowy days?
- What percent of days have a high of at least 75 degrees?

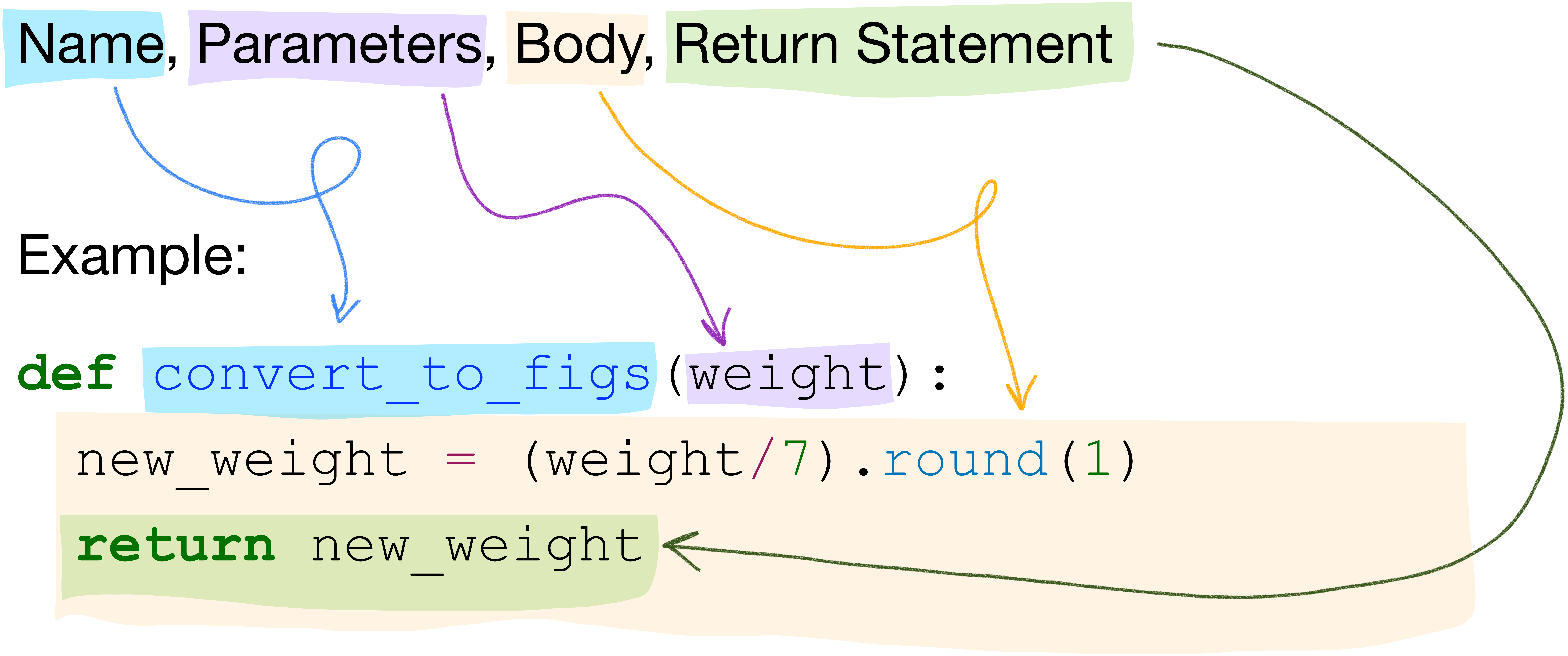


date	tmax	tmin	tavg	condition
1/1/19	60	40	50	rainy
2/1/19	41	35	38	
3/1/19	45	39	42	
4/1/19	47	37	42	
5/1/19	47	42	44.5	rainy
6/1/19	49	32	40.5	
7/1/19	35	26	30.5	
8/1/19	47	35	41	rainy
9/1/19	46	35	40.5	rainy
10/1/19	35	30	32.5	

Census Demo

Functions

Recall: Anatomy of a Function



Name, Parameters, Body, Return Statement

Example:

```
def convert_to_figs(weight):
  new_weight = (weight/7).round(1)
  return new_weight
```

Example

What does this function do?

- What type of input do you expect it takes?
- What type of output will it give?
- What’s a reasonable name for the function?

```
def f(s):
    return s / sum(s) * 100
```

Example

What does this function do?

- What type of input do you expect it takes? Array
- What type of output will it give? Array
- What's a reasonable name for the function?

```
def f(s):
    return s / sum(s) * 100
```

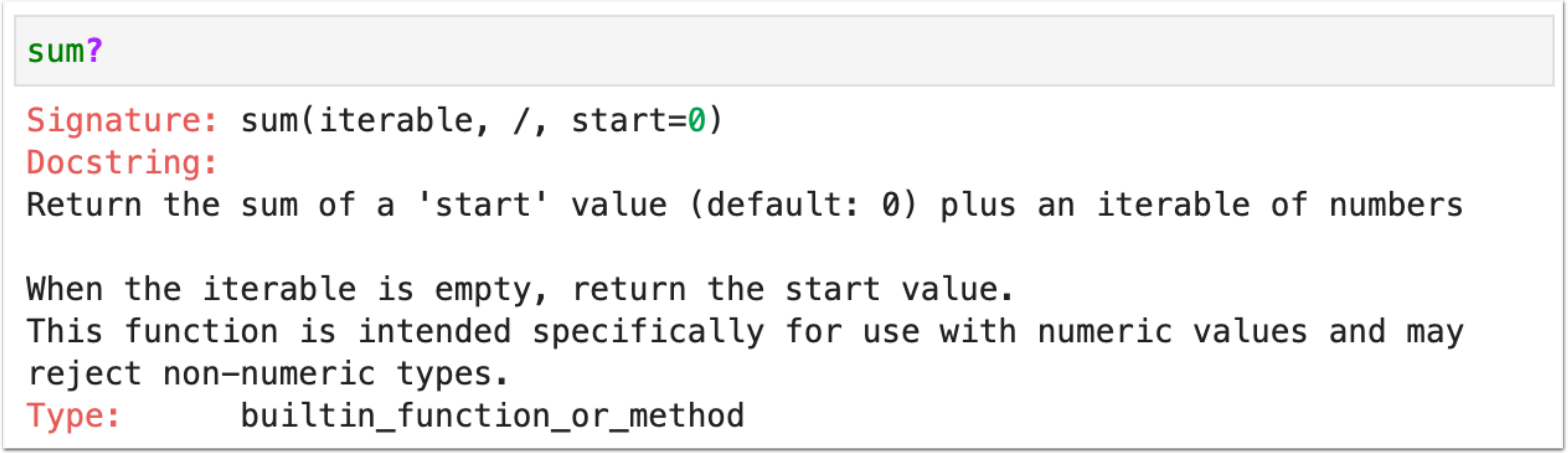
Example

What does this function do?

- What type of input do you expect it takes? Array
- What type of output will it give? Array
- What's a reasonable name for the function? Anything related to percent

```
def percent(s):
    return s / sum(s) * 100
```

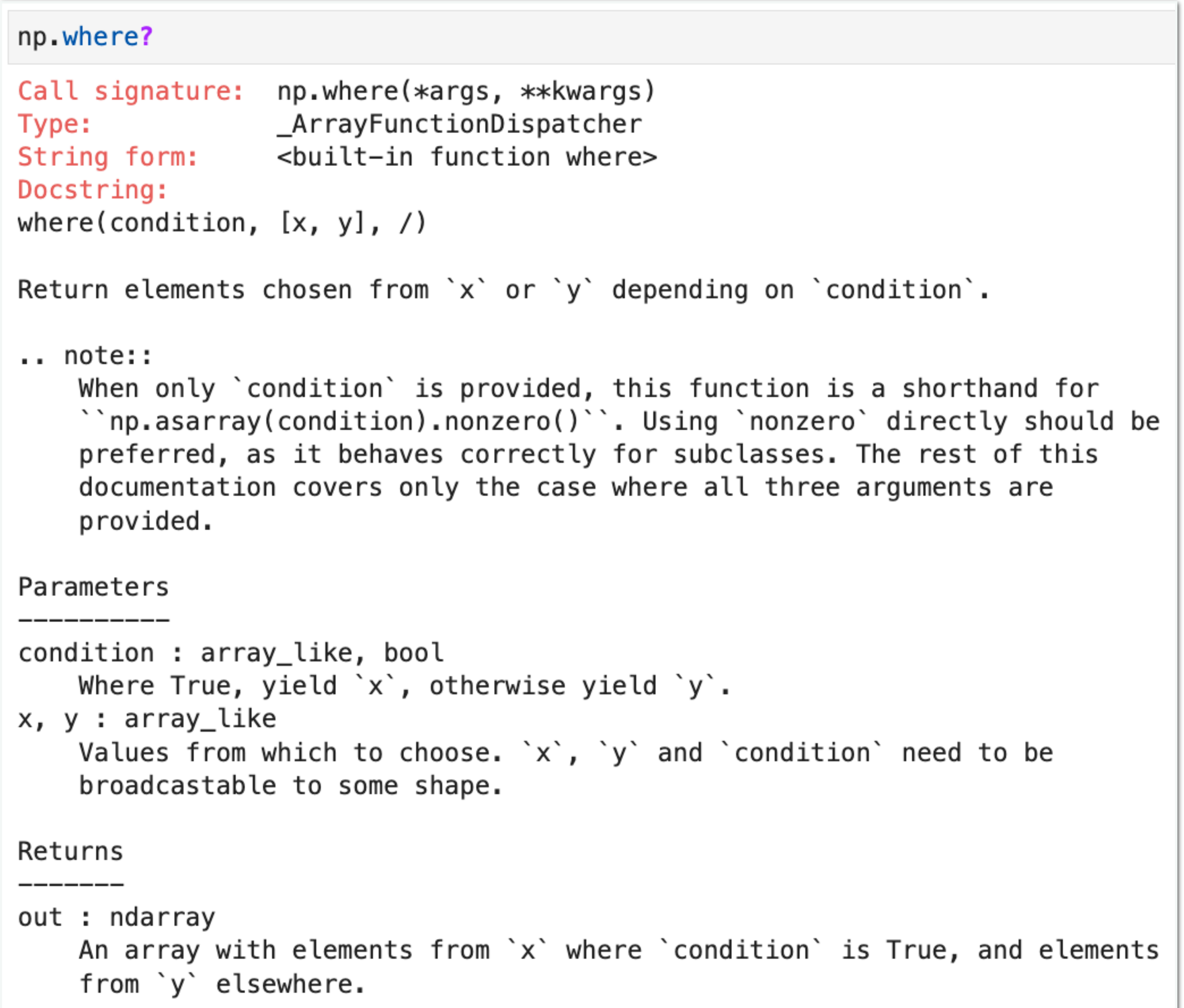
Function Documentation



```
sum?
```

```
Signature: sum(iterable, /, start=0)
Docstring:
Return the sum of a 'start' value (default: 0) plus an iterable of numbers

When the iterable is empty, return the start value.
This function is intended specifically for use with numeric values and may
reject non-numeric types.
Type:      builtin_function_or_method
```

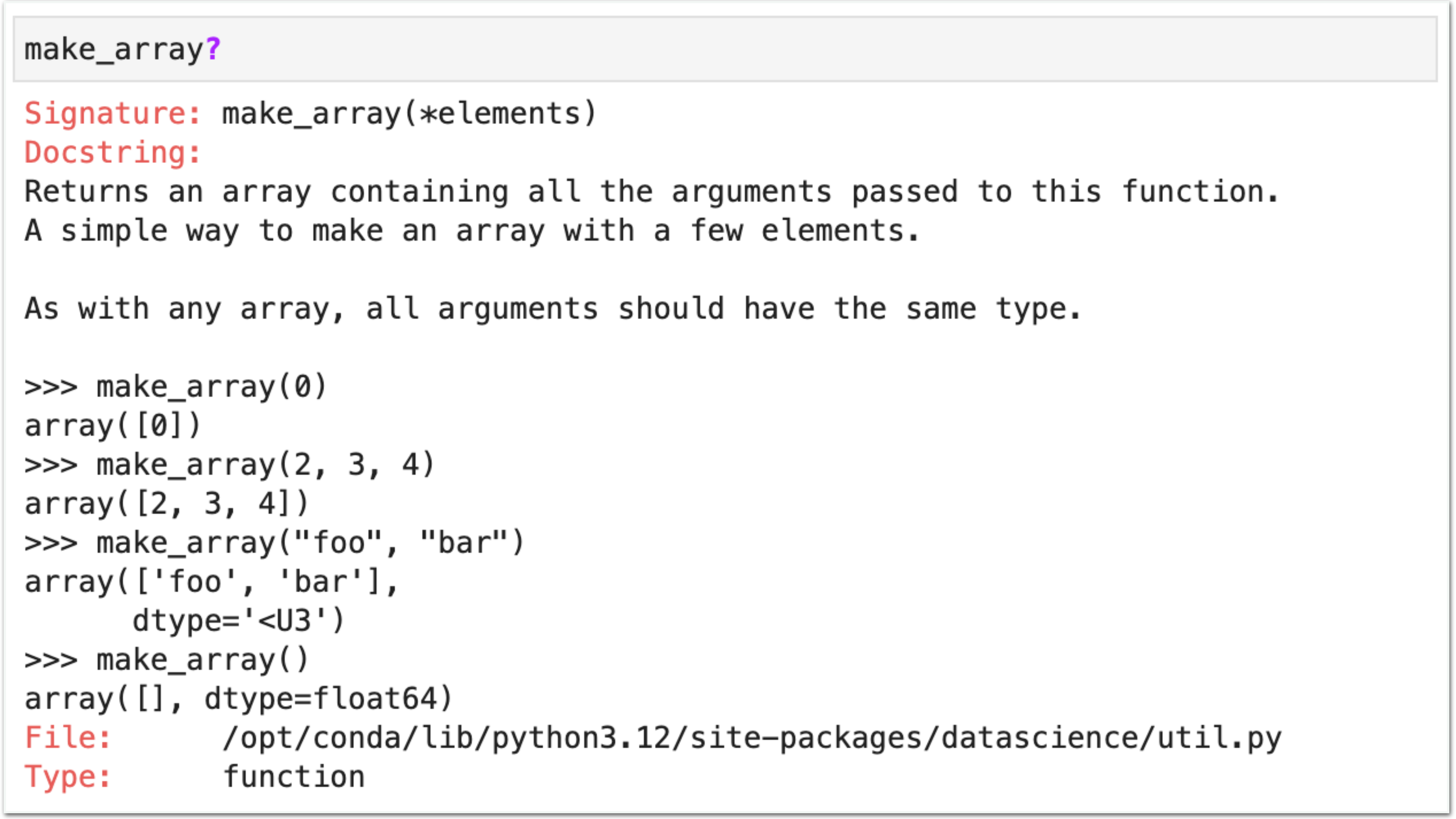
Function Documentation



```
np.where?
```

```
Call signature:  np.where(*args, **kwargs)
Type:            _ArrayFunctionDispatcher
String form:     <built-in function where>
Docstring:
where(condition, [x, y], /)

Return elements chosen from `x` or `y` depending on `condition`.

.. note::
    When only `condition` is provided, this function is a shorthand for
    ``np.asarray(condition).nonzero()``. Using `nonzero` directly should be
    preferred, as it behaves correctly for subclasses. The rest of this
    documentation covers only the case where all three arguments are
    provided.

Parameters
----------
condition : array_like, bool
    Where True, yield `x`, otherwise yield `y`.
x, y : array_like
    Values from which to choose. `x`, `y` and `condition` need to be
    broadcastable to some shape.

Returns
-------
out : ndarray
    An array with elements from `x` where `condition` is True, and elements
    from `y` elsewhere.
```

Function Documentation



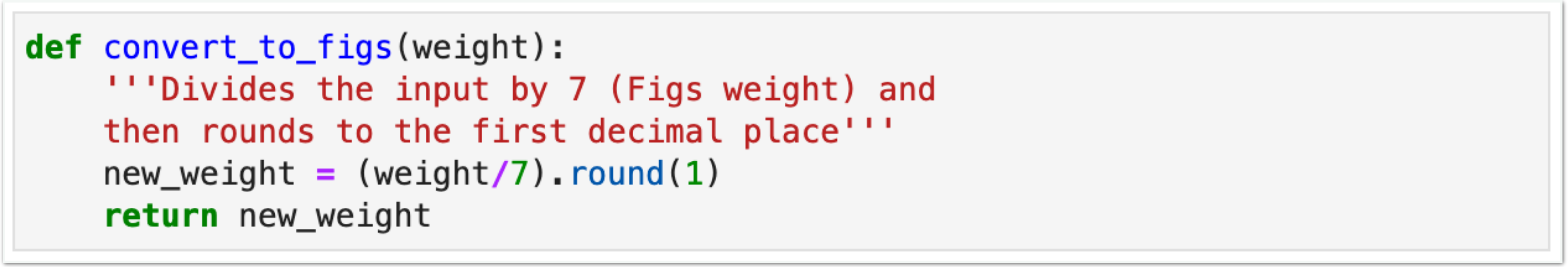
```
make_array?
```

```
Signature: make_array(*elements)
Docstring:
Returns an array containing all the arguments passed to this function.
A simple way to make an array with a few elements.

As with any array, all arguments should have the same type.

>>> make_array(0)
array([0])
>>> make_array(2, 3, 4)
array([2, 3, 4])
>>> make_array("foo", "bar")
array(['foo', 'bar'],
      dtype='<U3')
>>> make_array()
array([], dtype=float64)
File:      /opt/conda/lib/python3.12/site-packages/datascience/util.py
Type:      function
```

Adding Documentation

Putting a string in the first line of a function body defines the **Docstring**

- Typically describes behavior and expectations about its arguments

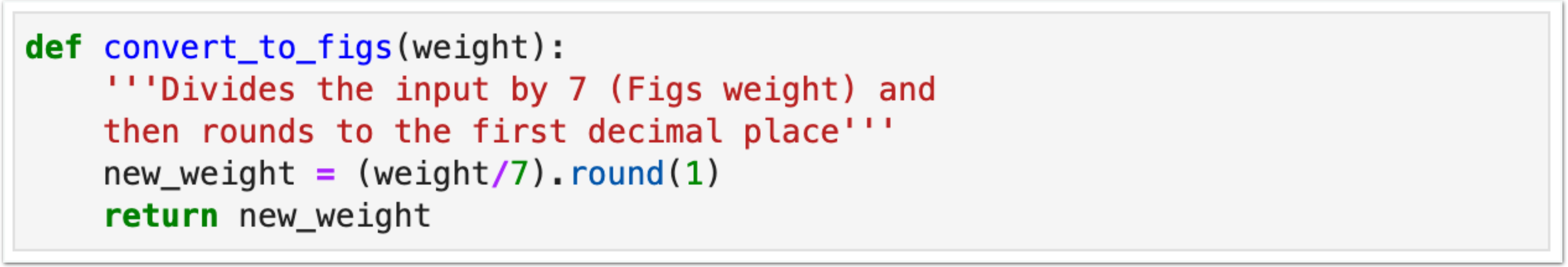


```
def convert_to_figs(weight):
    '''Divides the input by 7 (Figs weight) and
    then rounds to the first decimal place'''
    new_weight = (weight/7).round(1)
    return new_weight
```

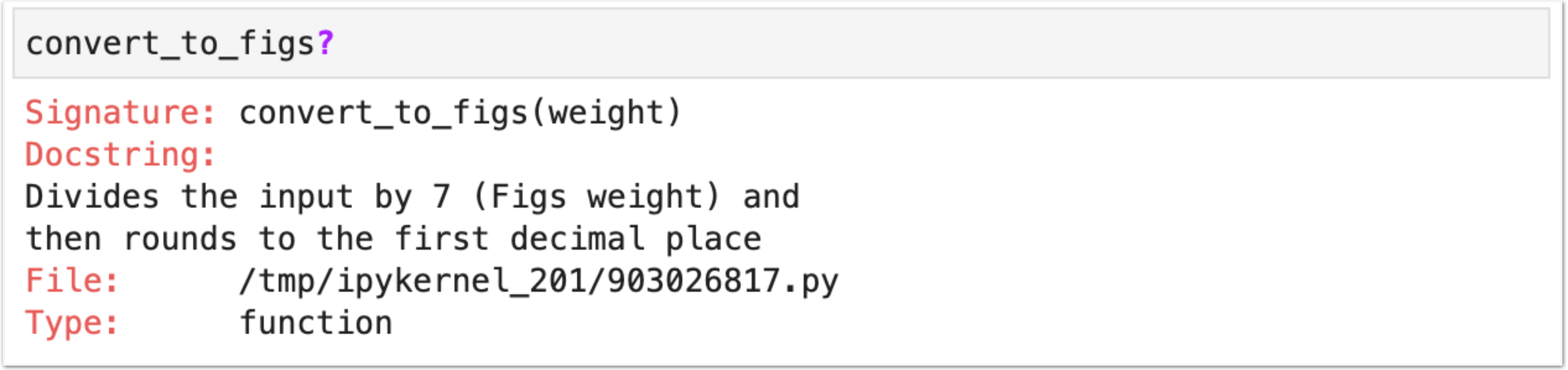
Adding Documentation

Putting a string in the first line of a function body defines the **Docstring**

- Typically describes behavior and expectations about its arguments



```python
def convert_to_figs(weight):
    '''Divides the input by 7 (Figs weight) and
    then rounds to the first decimal place'''
    new_weight = (weight/7).round(1)
    return new_weight
```



```
convert_to_figs?
```

```
Signature: convert_to_figs(weight)
Docstring:
Divides the input by 7 (Figs weight) and
then rounds to the first decimal place
File:      /tmp/ipykernel_201/903026817.py
Type:      function
```

Functions with Multiple Arguments/Parameters

Functions can take in multiple inputs

- Each argument is given a unique name and separated by commas

```
def convert_to_figs(weight, decimal_places):
  ’’’Divides the input by 7 (Figs weight) and then rounds to
  the given number of decimal places’’’
  new_weight = (weight/7).round(decimal_places)
  return new_weight
```

Functions with Multiple Arguments/Parameters

Functions can take in multiple inputs

- Each argument is given a unique name and separated by commas
- Specifying default values for particular inputs to makes them optional

```
def convert_to_figs(weight, decimal_places=1):
    ’’’Divides the input by 7 (Figs weight) and then rounds to
    the given number of decimal places’’’
    new_weight = (weight/7).round(decimal_places)
    return new_weight
```

Function Demo

Recall: apply

Use apply to call a function on each element in a column

```
def convert_to_figs(weight):
   new_weight = (weight/7).round(1)
   return new_weight
```

```
cat_tbl.apply(convert_to_figs, ‘Weight’)
```

Returns an array with convert_to_figs called on each element in the 'Weight' column

`apply` with Multiple Inputs

For functions with multiple inputs, `apply` can take multiple columns

```
def convert_to_figs(weight, decimal_places=1):
   new_weight = (weight/7).round(decimal_places)
   return new_weight


cat_tbl.apply(convert_to_figs, ‘Weight’, ‘Precision’)
```

Apply Demo

Next Class

- Today
  - Histograms
  - Functions and Apply
- Monday
  - Groups, Pivots, and Joins