

COMS BC1016

Introduction to Computational Thinking and Data Science

Lecture 4: Arrays and Tables

BARNARD COLLEGE OF COLUMBIA UNIVERSITY

Sept 3, 2025

Copyright © 2025 Barnard College

Sept 15, 2025

Misc Updates

- Updated office hours
- Don't forget lab attendance is required!!
 - Please email your TAs in advance if you'll be late or can't make it!
- HW 1 out today, due next week Wednesday
 - Known issue with the autograder for Q4.2, it's fine
 - **Remember to run the last cell in the notebook**

Office Hours:

- M 2:30pm-4:00pm, Milstein Center 503 (Erin Ma)
- Tu 2:00pm-3:30pm, Milstein Center 503 (Amaya Kejriwal)
- W 3:00pm-4:00pm, Milstein Center 512 (Prof. Eysa Lee)
- Th 10:00am-11:30am, Virtual (via [Google Meet](#)) (Ken Mah)
- Th 1:00pm-2:30pm, Milstein Center 503 (Justin Zeng)
- MW 2:45pm-3:45pm, Milstein Center 511 ([Prof. Murad Megjhani](#))

Last Class Recap

- Built-in Data Types
 - Numbers (integers and floats)
 - Strings
 - Booleans
- Programming basics
 - Functions
 - Assignments

Type Exercise

Let's say you have defined the following variables in your notebook

```
x = 3
```

```
y = '4'
```

```
z = '5.6'
```

What would the source of the error in these examples?

How could you fix it?

```
1. x + y
```

```
2. x + int(y + z)
```

```
3. str(x) + int(y)
```

```
4. y + float(z)
```

Type Exercise

Take 3 min to work on your own
or with a neighbor!

Let's say you have
defined the following
variables in your notebook

```
x = 3
```

```
y = '4'
```

```
z = '5.6'
```

What would the source of
the error in this example:

```
x + y
```

How could you fix it?

Type Exercise

Take 3 min to work on your own
or with a neighbor!

Let's say you have
defined the following
variables in your notebook

```
x = 3
```

```
y = '4'
```

```
z = '5.6'
```

What would the source of
the error in this example:

```
x + int(y + z)
```

How could you fix it?

Type Exercise

Take 3 min to work on your own
or with a neighbor!

Let's say you have
defined the following
variables in your notebook

```
x = 3
```

```
y = '4'
```

```
z = '5.6'
```

What would the source of
the error in this example:

```
str(x) + int(y)
```

How could you fix it?

Type Exercise

Take 3 min to work on your own
or with a neighbor!

Let's say you have
defined the following
variables in your notebook

```
x = 3
```

```
y = '4'
```

```
z = '5.6'
```

What would the source of
the error in this example:

```
y + float(z)
```

How could you fix it?

Importing Libraries

- Recall we can **import** libraries
- Can also do **import** library **as** shorter name
 - e.g., **import** numpy **as** np
- Can also specific functions to import from a specific library
 - e.g., **from** math **import** sqrt
 - Use * to import all functions
 - e.g., **from** datascience **import** *
- Be careful doing this with too many libraries at once!

```
import numpy  
numpy.sqrt(4)
```

2.0

Arrays

Arrays

- Arrays are a sequence of values
 - e.g., ["Mystery", "Abby", "Jinu", "Baby", "Romance"] or [1, 2, 3, 5]
 - Elements of an array should have the same type
- We'll mostly use NumPy arrays
 - Can make arrays using `datascience.make_array` or `numpy.array`
- Can perform component-wise arithmetic
 - Note this only works for numpy arrays but not built-in Python lists!

```
from datascience import *  
onetwothree = make_array(1, 2, 3)  
onetwothree * 2  
  
array([2, 4, 6])
```

```
from datascience import *  
onetwothree = make_array(1, 2, 3)  
twothreefour = make_array(2, 3, 4)  
onetwothree + twothreefour  
  
array([3, 5, 7])
```

```
onetwothree = [1, 2, 3]  
twothreefour = [2, 3, 4]  
onetwothree + twothreefour  
  
[1, 2, 3, 2, 3, 4]
```


Arrays

- Many useful functions for operating on arrays
 - Helpful to be aware of, but you do not need to memorize them!

Function	Description
<code>np.prod</code>	Multiply all elements together
<code>np.sum</code>	Add all elements together
<code>np.all</code>	Test whether all elements are true values (non-zero numbers are true)
<code>np.any</code>	Test whether any elements are true values (non-zero numbers are true)
<code>np.count_nonzero</code>	Count the number of non-zero elements

Function	Description
<code>np.char.lower</code>	Lowercase each element
<code>np.char.upper</code>	Uppercase each element
<code>np.char.strip</code>	Remove spaces at the beginning or end of each element
<code>np.char.isalpha</code>	Whether each element is only letters (no numbers or symbols)
<code>np.char.isnumeric</code>	Whether each element is only numeric (no letters)

Function	Description
<code>np.diff</code>	Difference between adjacent elements
<code>np.round</code>	Round each number to the nearest integer (whole number)
<code>np.cumprod</code>	A cumulative product: for each element, multiply all elements so far
<code>np.cumsum</code>	A cumulative sum: for each element, add all elements so far
<code>np.exp</code>	Exponentiate each element
<code>np.log</code>	Take the natural logarithm of each element
<code>np.sqrt</code>	Take the square root of each element
<code>np.sort</code>	Sort the elements

Function	Description
<code>np.char.count</code>	Count the number of times a search string appears among the elements of an array
<code>np.char.find</code>	The position within each element that a search string is found first
<code>np.char.rfind</code>	The position within each element that a search string is found last
<code>np.char.startswith</code>	Whether each element starts with the search string

Ranges

A range is an array of consecutive numbers

- `np.arange(end)`

Create an array of increasing integers from 0 up to `end`

- `np.arange(start, end)`

Create an array of increasing integers from `start` up to `end`

- `np.arange(start, end, step)`

A range where `step` is added between consecutive values

The range always includes `start` but *excludes* `end`

Tables

Tables

A table is a way of representing data sets

- Each row is an individual
- Each column is an attribute of the individual

Name	Age	Coloring	Favorite Food
Gertrude	15 yrs	Tuxedo	Milk
Ruby	14 yrs	Tuxedo	Potato chips
Corina	6 yrs	Dilute Tortoiseshell	Kibble
Frito	1 yr	Tabby	Cheese

Creating datascience Tables

Create an empty table using `Table()`

Each column of a table is an array and `with_columns` creates a table with the array of values as a new column

Name	Description	Input	Output
<code>Table()</code>	Create an empty table, usually to extend with data (Ch 6)	None	An empty Table
<code>Table().read_table(filename)</code>	Create a table from a data file (Ch 6)	string : the name of the file	Table with the contents of the data file
<code>tbl.with_columns(name, values)</code> <code>tbl.with_columns(n1, v1, n2, v2, ...)</code>	A table with an additional or replaced column or columns. name is a string for the name of a column, values is an array (Ch 6)	1. string : the name of the new column; 2. array : the values in that column	Table : a copy of the original Table with the new columns added

Creating datascience Tables

Create an empty table using `Table()`

Each column of a table is an array and `with_columns` creates a table with the array of values as a new column

```
Table().with_columns("Name", make_array("Gertrude",  
"Ruby", "Corina", "Frito"))
```

Name
Gertrude
Ruby
Corina
Frito

Creating datascience Tables

Create an empty table using `Table()`

Each column of a table is an array and `with_columns` creates a table with the array of values as a new column

```
Table().with_columns("Name", make_array("Gertrude",  
"Ruby", "Corina", "Frito"),  
"Age", make_array(15, 14, 6, 1))
```

Name	Age
Gertrude	15
Ruby	14
Corina	6
Frito	1

More Ways to Create Tables

- Read from a CSV file

- `Table.read_table(filename)`

- Create a new table from an existing table. Let `tbl` be a table and `c, c1, c2` be column names or indices

- `tbl.select(c1, c2, ...)`

Table with only columns `c1, c2, ...`

- `tbl.drop(c1, c2, ...)`

Table without columns `c1, c2, ...`

- `tbl.sort(c[, descending=False])`

Table sorted by elements in column `c`

- `tbl.where(c, predicate)`

Only rows in the table where the value in column `c` satisfies the predicate

- `tbl.take(row_indices)`

only the specified rows

Filtering

<https://www.data8.org/sp22/python-reference.html>

Table.where Predicates

Any of these predicates can be negated by adding `not_` in front of them, e.g. `are.not_equal_to(Z)` or `are.not_containing(S)`.

Predicate	Description
<code>are.equal_to(Z)</code>	Equal to <code>Z</code>
<code>are.not_equal_to(Z)</code>	Not equal to <code>Z</code>
<code>are.above(x)</code>	Greater than <code>x</code>
<code>are.above_or_equal_to(x)</code>	Greater than or equal to <code>x</code>
<code>are.below(x)</code>	Less than <code>x</code>
<code>are.below_or_equal_to(x)</code>	Less than or equal to <code>x</code>
<code>are.between(x,y)</code>	Greater than or equal to <code>x</code> and less than <code>y</code>
<code>are.between_or_equal_to(x,y)</code>	Greater than or equal to <code>x</code> , and less than or equal to <code>y</code>
<code>are.contained_in(A)</code>	Is a substring of <code>A</code> (if <code>A</code> is a string) or an element of <code>A</code> (if <code>A</code> is a list/array)
<code>are.containing(S)</code>	Contains the string <code>S</code>
<code>are.strictly_between(x,y)</code>	Greater than <code>x</code> and less than <code>y</code>

Table Methods

Recall each column in a Table is an array

- `column` takes a label or index and returns an array
- Array methods work on data in the columns
 - e.g., `sum`, `min`, `max`, `average`

Python Reference

<https://www.data8.org/fa25/reference/>



🔍 Search Data 8

Detailed Python Reference Sheet

Created by Nishant Kheterpal and Jessica Hu; contributions by Jonathan Ferrari, Edwin Vargas and Bing Concepcion
Updated and maintained by Marissa Lumpkin and Isaac Chung

TABLE OF CONTENTS

- 1 **[Detailed Python Reference Sheet](#)**
 - a [Abbreviated Reference Sheet](#)
 - b [Table Functions and Methods](#)
 - c [String Methods](#)
 - d [Array Functions and Methods](#)
 - e [Table Filtering Predicates](#)
 - f [Miscellaneous Functions](#)
 - g [JupyterHub Keyboard Shortcuts](#)

🏠 Home

📅 Weekly Calendar & OH

📖 Syllabus

👤 Staff

📚 Resources

🗨️ FAQs & Debugging

🐍 Python Reference

📖 Textbook

Class Activity

Next Class

- Last Wednesday
 - Jupyter Notebooks
 - Expressions
 - Data Types
- Today
 - Tables (and arrays)
- Wednesday
 - Charts & Visualization