

COMS BC1016

Introduction to Computational Thinking and Data Science

Lecture 3: Arrays and Tables

BARNARD COLLEGE OF COLUMBIA UNIVERSITY

Sep 30, 2025

Lab Reminders

- **Reminder: You must be enrolled in a 1017 section!**
 - As of Tuesday, there are still more people enrolled in 1016 than 1017
- Labs begin this week (today and tomorrow)
 - **Email your TA if you'll be late or missing!**
 - **Wednesday:** Nami Jain nbj2115@columbia.edu
 - **Thursday:** Sathya Raman sr4213@columbia.edu
 - 50% of your lab grade is attendance!
 - One unexcused absence + lowest lab dropped

Course Website

Slides, emails, and helpful links are on the course website:

<https://www.eysalee.com/courses/s26/bc1016.html>

Course Links

Jupyter Hub: [Link](#) (login required)

Class Discussion Forum: [EdStem](#) (login required)

Courseworks: [Link](#)

Syllabus: [Link](#)

Resources

Python Resources:

Data8 Python Reference: <https://www.data8.org/fa24/reference/>

DataScience Python Library Developer Documentation: <https://www.data8.org/datascience/>

Data8 Textbook: <https://inferentialthinking.com/chapters/intro.html>

Lecture Schedule

The schedule below will be updated as the course progresses.

Week	Date	Topic	Lab	Assignment
1	1/21	1 - Introduction [Slides]	<i>No Lab</i>	
2	1/26	2 - Introduction to Python [Slides] (Remote - Snow Day)		

Last time: Data Types

Types

Type	Example
Int	3
Float	3.78
String	'Gertrude'
Boolean	True
Lists/Arrays	["Hamby", "Fig", "Ruby"]
Functions	abs(-5)

What's the type?

Suppose we download information about the class.

What data type would each of the following fields be?

- Course title:
- Enrollment count:
- Names of students enrolled in the class:
- Average GPA of students in the class:

What's the type?

Suppose we download information about the class.

What data type would each of the following fields be?

- Course title: `String`
- Enrollment count: `Integer`
- Names of students enrolled in the class: `Array (or list) of strings`
- Average GPA of students in the class: `Float`

Variables and Assignments

- **Assignments** change the meaning of the name to the left the = symbol
- **Variables** are values you can assign values to
 - “Variable” because they can change
- You can assign outputs of functions and operations to variables

```
max_of_list = max(4, 200, 7)  
max_of_list + 5
```

205

Python quirks

- Python is case sensitive
 - `Apple` is not the same as `apple`
- Indentation and new lines matter
 - Be careful not to add extra spaces or indentations at the beginning of the line!
- Python runs line-by-line
 - It'll stop as soon as it runs into an issue and tell you what's wrong
- Lines starting with `#` are comments and are ignored

```
[1]: "hello"
      "meow"
      "I have evaded notice!"

Cell In[1], line 2
      "meow"
      ^
IndentationError: unexpected indent
```

Type Exercise

Open a new notebook and define these three variables:

```
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

Open a new notebook and define these three variables:

```
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

Open a new notebook and define these three variables:

```
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

Open a new notebook and define these three variables:

```
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

Open a new notebook and define these three variables:

```
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?

Arrays

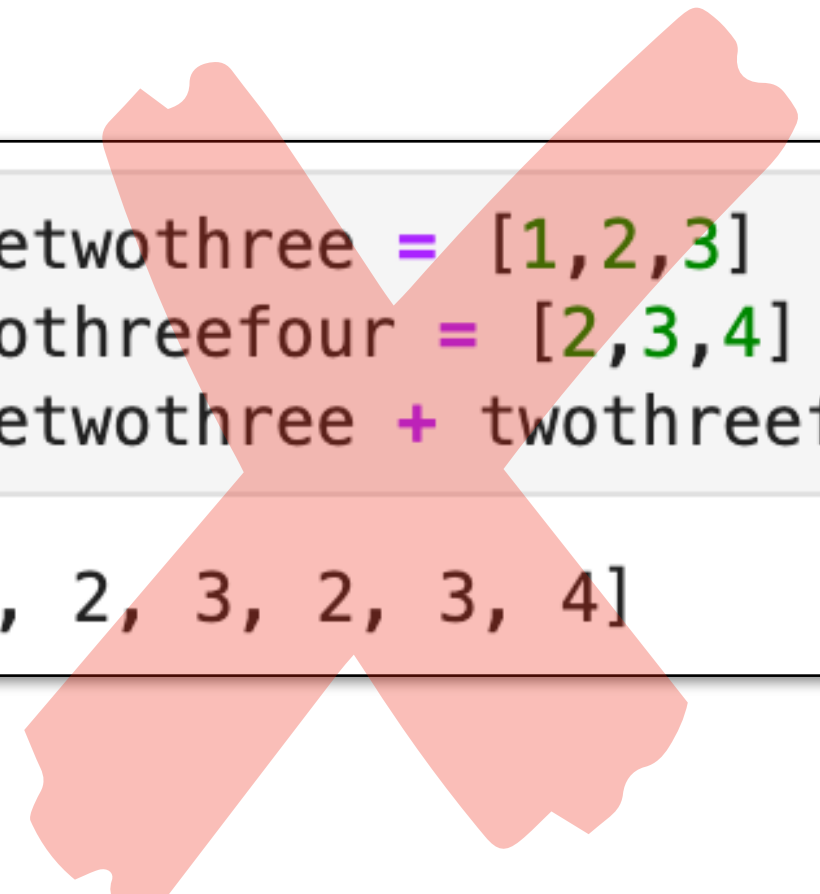
Arrays

- 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
- 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

- Can access the `i`th element either using `.item(i)` or `[i]`
 - Example (for an array `x`): `x.item(i)` or `x[i]`
- `len(x)` and `x.size` give the number of elements in the array `x`
- Counting starts at 0 (sometimes referred to as being 0-indexed)
 - `x[0]` gives the first element
 - `x[len(x) - 1]` gives the last element

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"))
```

Creating datascience Tables

Table() creates an empty table

.with_columns() adds a column

The first argument to .with_columns is the name of column

Each column of a table is an array and with_columns creates a new column

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

... Followed by an array with the column values

Creating datascience Tables

Table() creates an empty table

.with_columns() adds a column

The first argument to .with_columns is the name of column

Each column of a table is an array and with_columns creates a new column

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

... Followed by an array with the column values

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"))
```

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))
```

We can add more columns with a comma and following this same pattern

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>

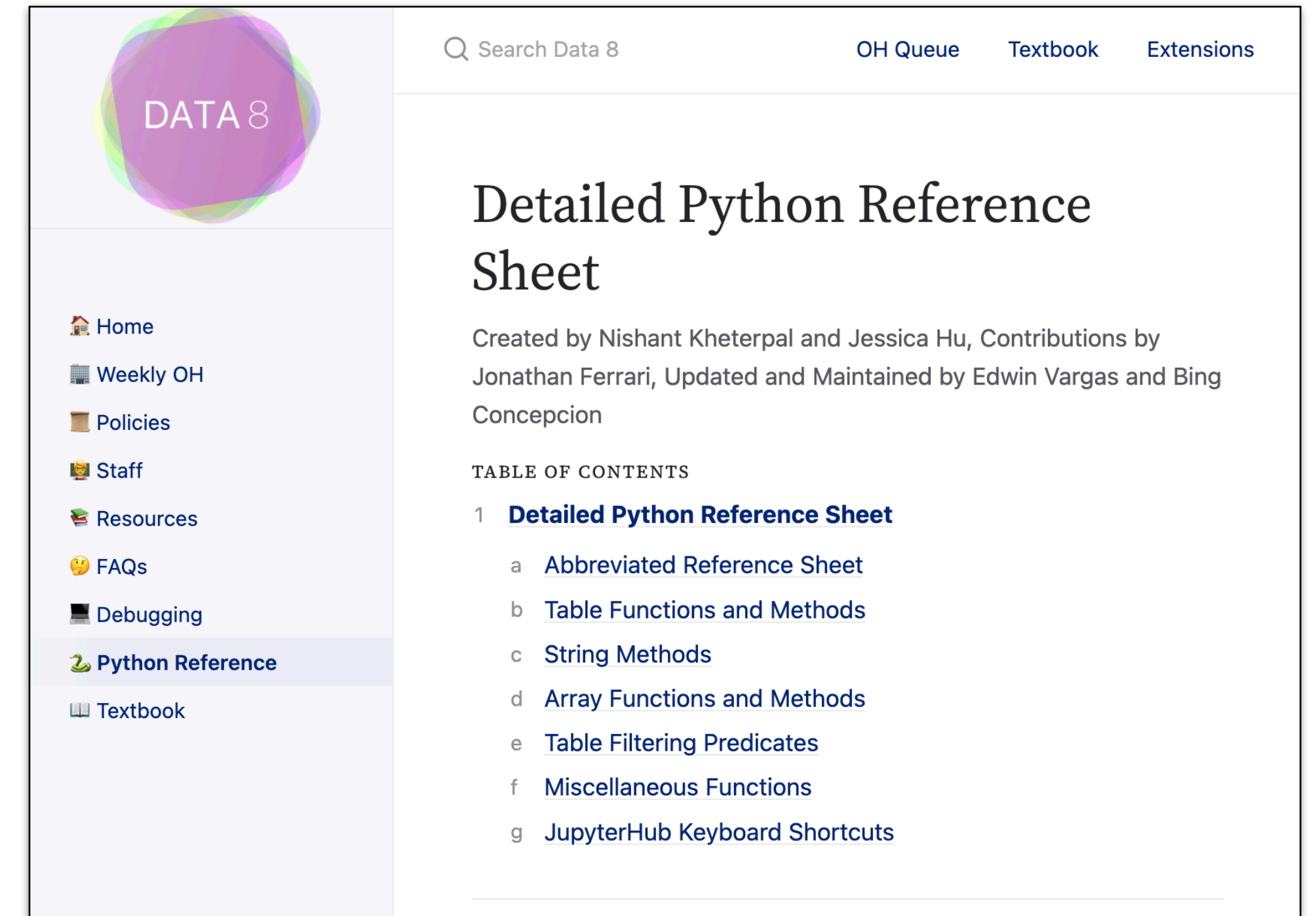
Table Methods

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/sp25/reference/>



The screenshot shows the Data 8 website's Python Reference page. The header includes a search bar, "OH Queue", "Textbook", and "Extensions" links. A sidebar on the left contains links to Home, Weekly OH, Policies, Staff, Resources, FAQs, Debugging, Python Reference (highlighted), and Textbook. The main content area is titled "Detailed Python Reference Sheet" and credits Nishant Kheterpal, Jessica Hu, Jonathan Ferrari, Edwin Vargas, and Bing Concepcion. It features a "TABLE OF CONTENTS" with links to an abbreviated reference sheet, table functions, string methods, array functions, table filtering predicates, miscellaneous functions, and JupyterHub keyboard shortcuts.

DATA 8

Search Data 8 OH Queue Textbook Extensions

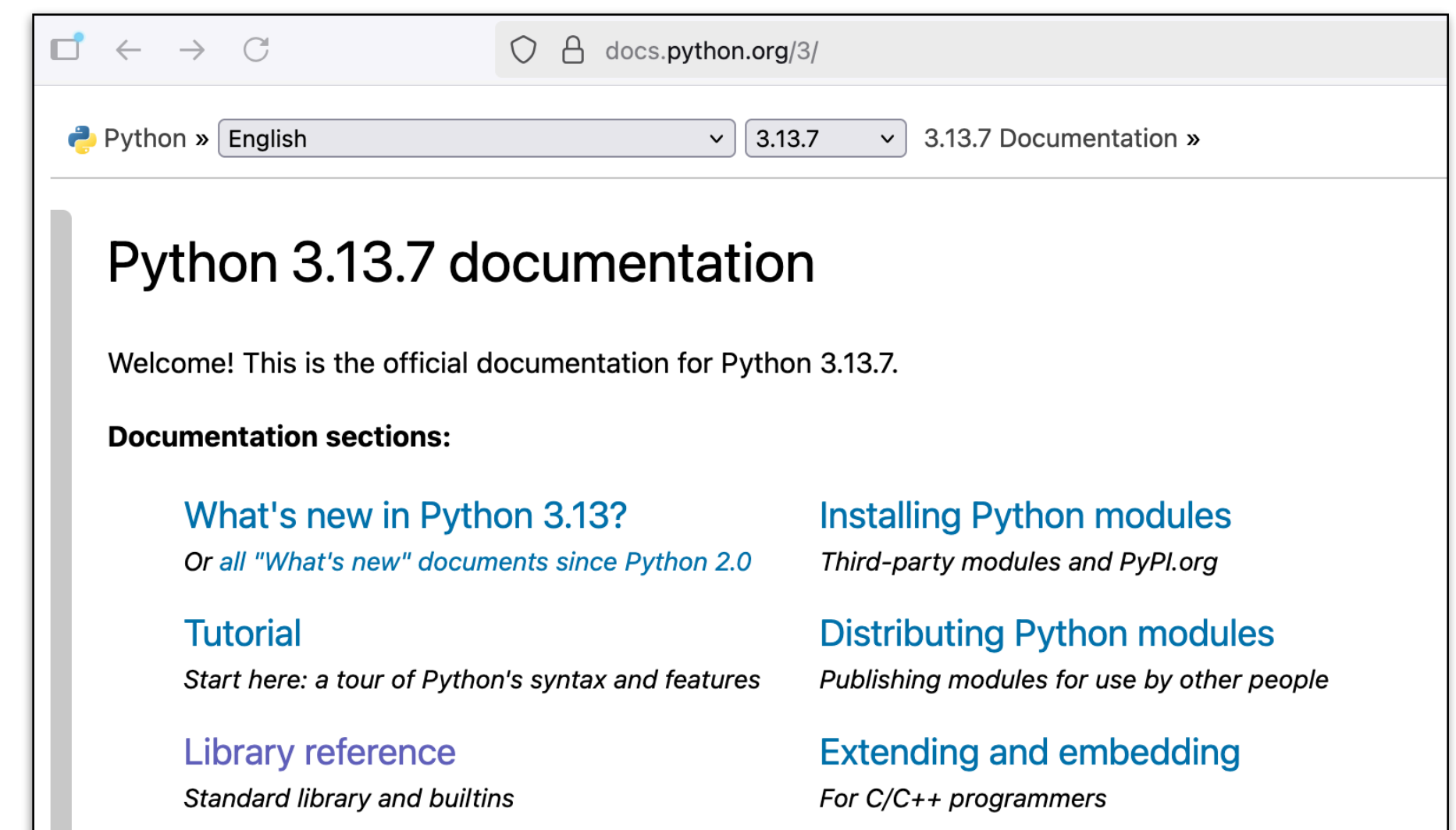
Detailed Python Reference Sheet

Created by Nishant Kheterpal and Jessica Hu, Contributions by Jonathan Ferrari, Updated and Maintained by Edwin Vargas and Bing Concepcion

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](#)

<https://docs.python.org/3/>



The screenshot shows the official Python 3.13.7 documentation website. The browser address bar shows "docs.python.org/3/". The page has a header with "Python »", a language dropdown set to "English", a version dropdown set to "3.13.7", and a link to "3.13.7 Documentation ». The main heading is "Python 3.13.7 documentation". Below it, a welcome message states: "Welcome! This is the official documentation for Python 3.13.7." A section titled "Documentation sections:" lists several links: "What's new in Python 3.13?" (with a sub-link for "What's new" documents since Python 2.0), "Installing Python modules" (with a sub-link for third-party modules and PyPI.org), "Tutorial" (with a sub-link for a tour of Python's syntax and features), "Distributing Python modules" (with a sub-link for publishing modules for use by other people), "Library reference" (with a sub-link for standard library and builtins), and "Extending and embedding" (with a sub-link for C/C++ programmers).

Python » English 3.13.7 3.13.7 Documentation »

Python 3.13.7 documentation

Welcome! This is the official documentation for Python 3.13.7.

Documentation sections:

- [What's new in Python 3.13?](#)
Or all "What's new" documents since Python 2.0
- [Installing Python modules](#)
Third-party modules and PyPI.org
- [Tutorial](#)
Start here: a tour of Python's syntax and features
- [Distributing Python modules](#)
Publishing modules for use by other people
- [Library reference](#)
Standard library and builtins
- [Extending and embedding](#)
For C/C++ programmers

Next Class

- Today
 - Tables
- Monday (HW 1 is released)
 - Charts & Visualization