From Zero to Python in 10.5 Hours

Building foundational programming skills in an introductory workshop series

Geoff Timms

Marine Resources Library
A SC Marine Resources Center Partner

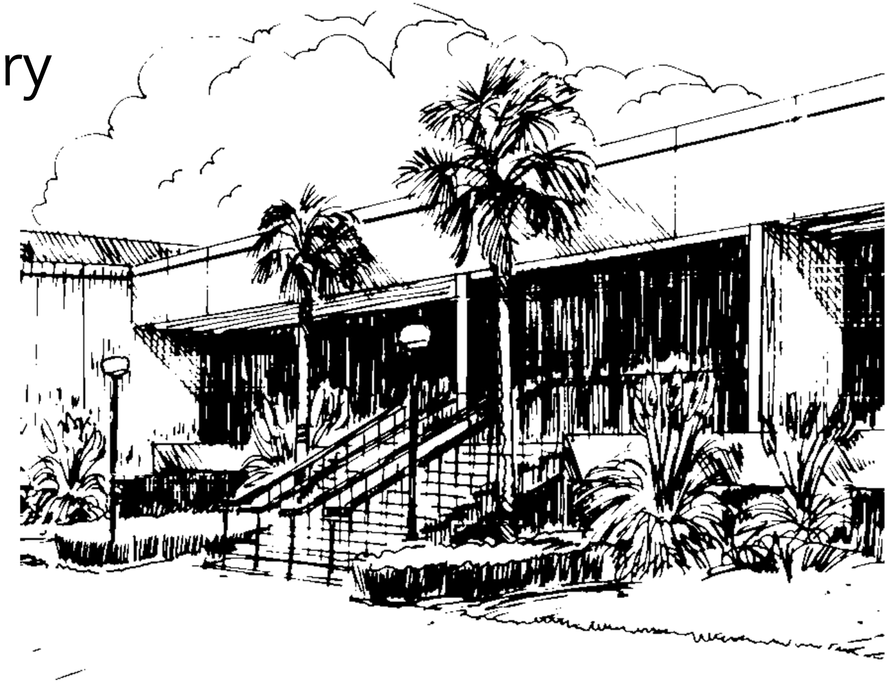
College of Charleston Libraries

Context

Marine Resources Library

A SC Marine Resources Center Partner

- Specialized Marine Science Library
- Graduate Students
- State/Federal scientists
- One librarian/one assistant



Why should scientists learn Python?

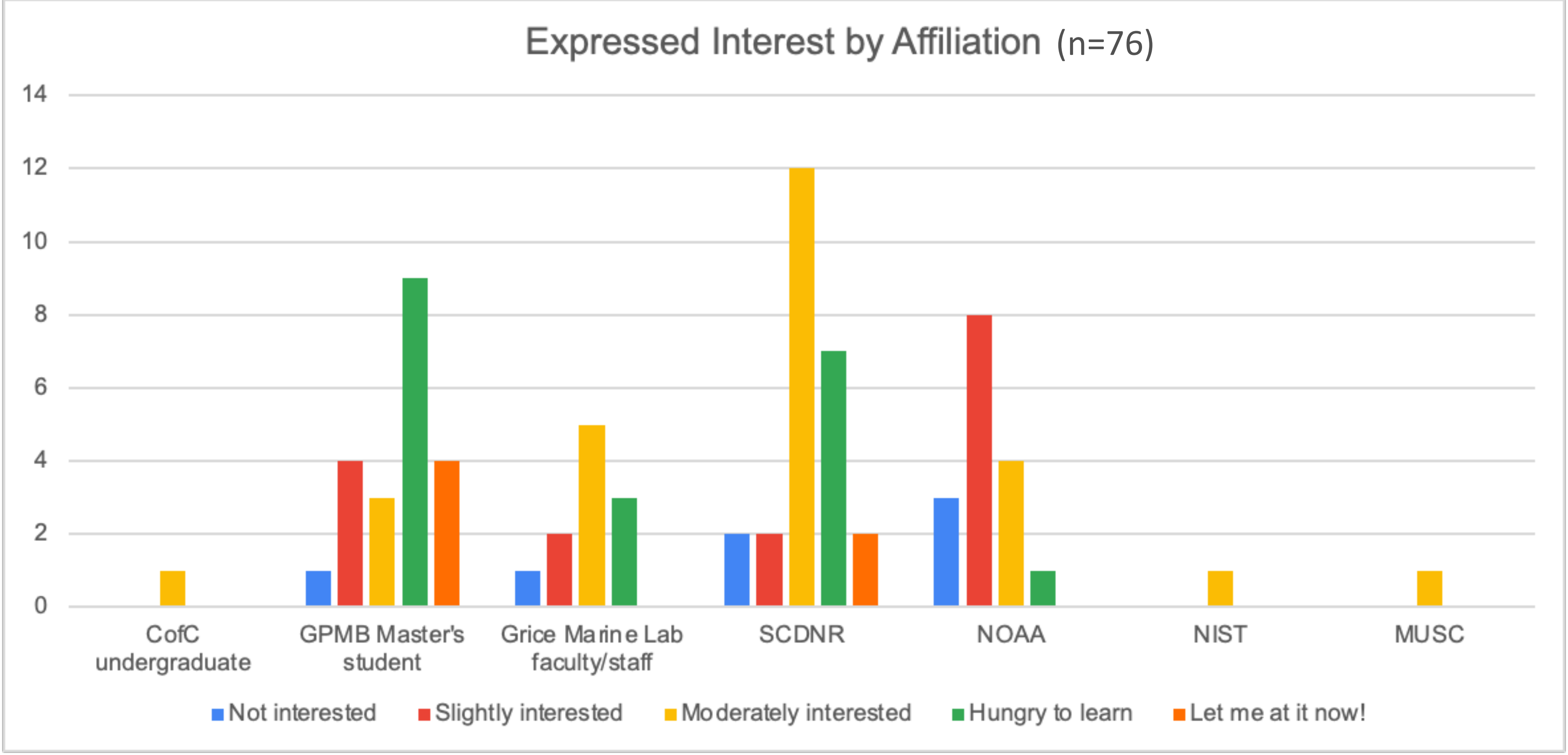
Why Python?

- Open source
- Widely used
- Highly readable
- Custom packages
- Support community
- General purpose

python™

https://www.python.org/

MRL survey of constituents



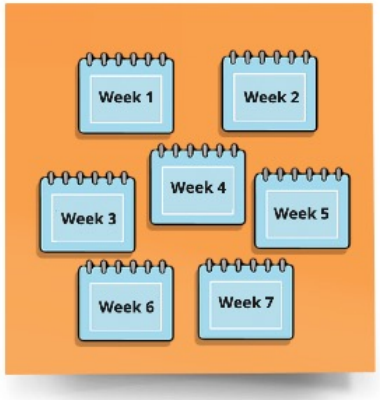
Course design

1. Small groups

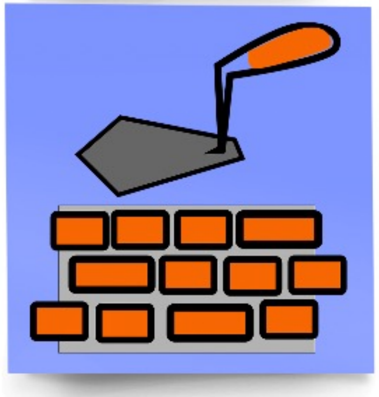
2. In person

3. Cloud-based environment

4. Concise

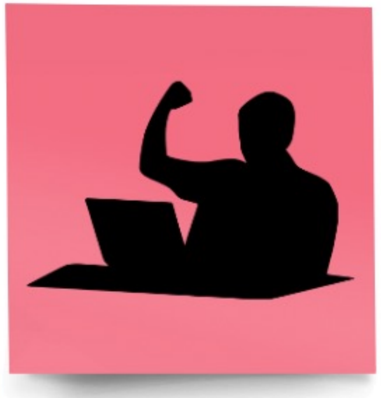


5. Scaffolded



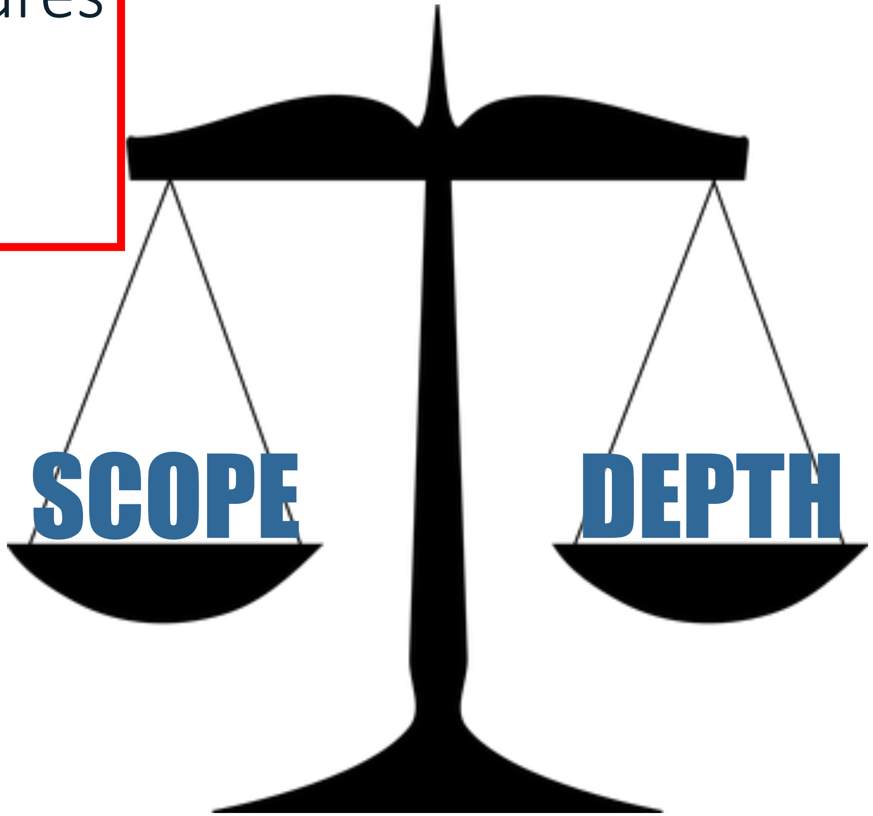
6. Foundational

7. Hands-on



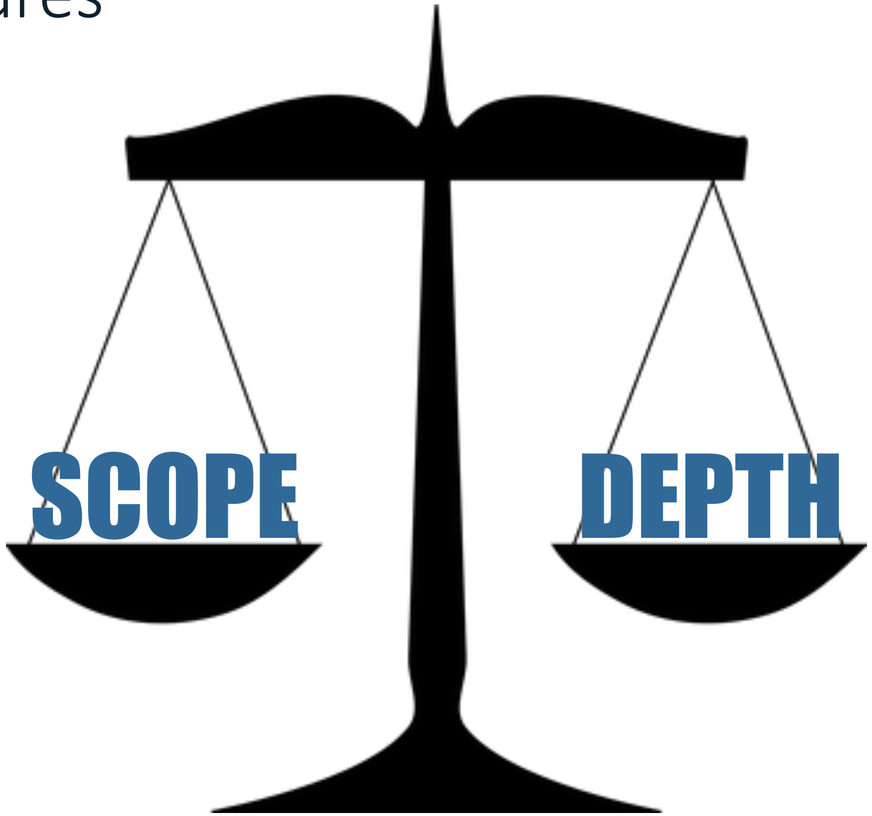
Python competencies by week

1. Orient to platform;
   Understand data types & structures
2. Read content from iterables
3. Structure code with functions
4. Read and write CSV files
5. Find data/patterns within data
6. Get data from APIs
7. Structure code with classes

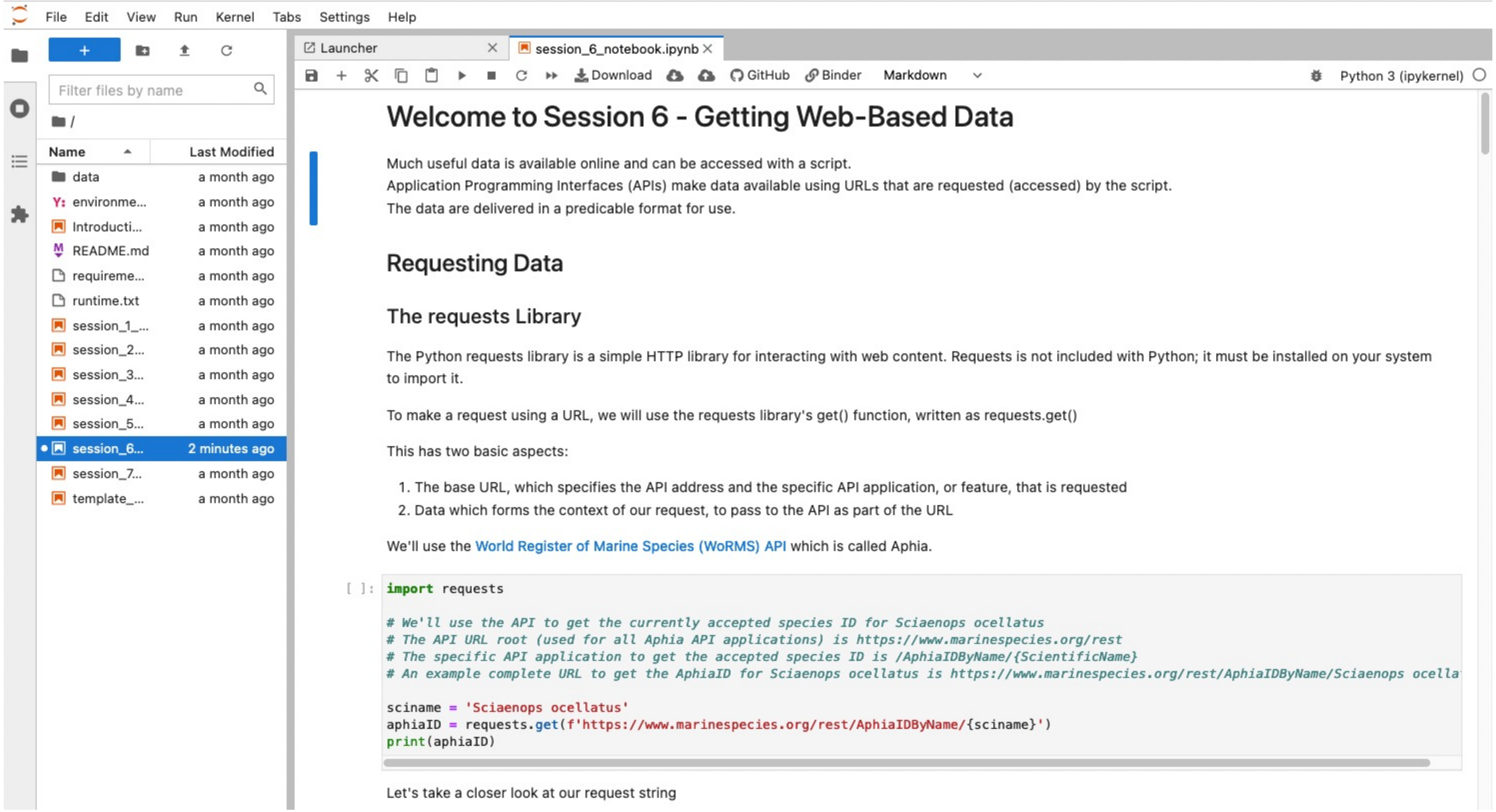


Python competencies by week

1. Orient to platform;
   Understand data types & structures
2. Read content from iterables
3. Structure code with functions
4. Read and write CSV files
5. Find data/patterns within data
6. Get data from APIs
7. Structure code with classes



The web interface: JupyterLab



Name	Last Modified
data	a month ago
environme...	a month ago
Introducti...	a month ago
README.md	a month ago
requireme...	a month ago
runtime.txt	a month ago
session_1_...	a month ago
session_2...	a month ago
session_3...	a month ago
session_4...	a month ago
session_5...	a month ago
session_6...	2 minutes ago
session_7...	a month ago
template_...	a month ago

Welcome to Session 6 - Getting Web-Based Data

Much useful data is available online and can be accessed with a script.
Application Programming Interfaces (APIs) make data available using URLs that are requested (accessed) by the script.
The data are delivered in a predicable format for use.

Requesting Data

The requests Library

The Python requests library is a simple HTTP library for interacting with web content. Requests is not included with Python; it must be installed on your system to import it.

To make a request using a URL, we will use the requests library's get() function, written as requests.get()

This has two basic aspects:

1. The base URL, which specifies the API address and the specific API application, or feature, that is requested
2. Data which forms the context of our request, to pass to the API as part of the URL

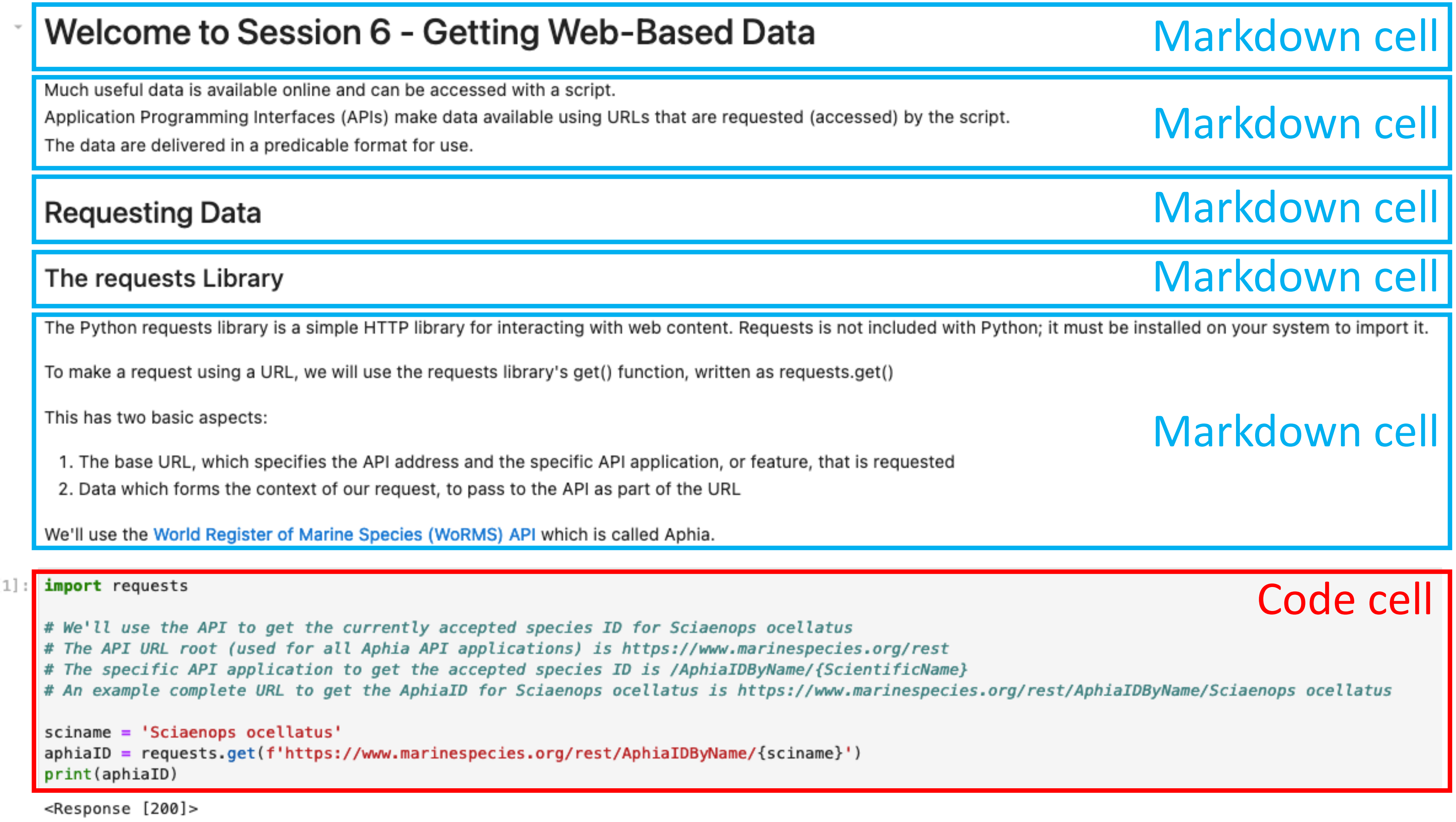
We'll use the World Register of Marine Species (WoRMS) API which is called Aphia.

[ ]:

```
import requests

# We'll use the API to get the currently accepted species ID for Sciaenops ocellatus
# The API URL root (used for all Aphia API applications) is https://www.marinespecies.org/rest
# The specific API application to get the accepted species ID is /AphiaIDByName/{ScientificName}
# An example complete URL to get the AphiaID for Sciaenops ocellatus is https://www.marinespecies.org/rest/AphiaIDByName/Sciaenops ocella

sciname = 'Sciaenops ocellatus'
aphiaID = requests.get(f'https://www.marinespecies.org/rest/AphiaIDByName/{sciname}')
print(aphiaID)
```

Let's take a closer look at our request string

What is a Jupyter Notebook?



Welcome to Session 6 - Getting Web-Based Data

Markdown cell

Much useful data is available online and can be accessed with a script.

Application Programming Interfaces (APIs) make data available using URLs that are requested (accessed) by the script.

The data are delivered in a predicable format for use.

Markdown cell

Requesting Data

Markdown cell

The requests Library

Markdown cell

The Python requests library is a simple HTTP library for interacting with web content. Requests is not included with Python; it must be installed on your system to import it.

To make a request using a URL, we will use the requests library's get() function, written as requests.get()

This has two basic aspects:

1. The base URL, which specifies the API address and the specific API application, or feature, that is requested
2. Data which forms the context of our request, to pass to the API as part of the URL

We'll use the World Register of Marine Species (WoRMS) API which is called Aphia.

Markdown cell

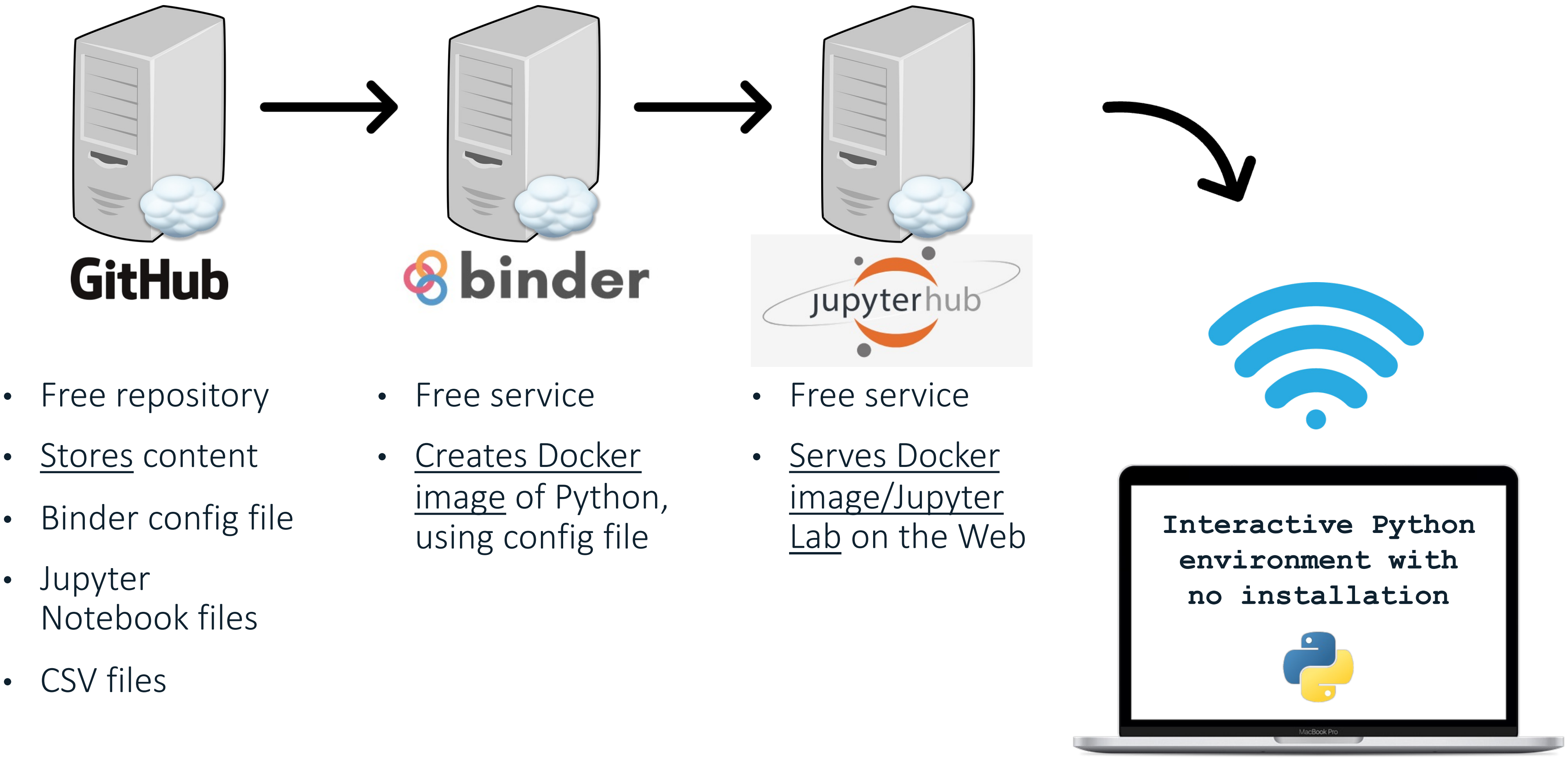
1]:

```python
import requests

# We'll use the API to get the currently accepted species ID for Sciaenops ocellatus
# The API URL root (used for all Aphia API applications) is https://www.marinespecies.org/rest
# The specific API application to get the accepted species ID is /AphiaIDByName/{ScientificName}
# An example complete URL to get the AphiaID for Sciaenops ocellatus is https://www.marinespecies.org/rest/AphiaIDByName/Sciaenops ocellatus

sciname = 'Sciaenops ocellatus'
aphiaID = requests.get(f'https://www.marinespecies.org/rest/AphiaIDByName/{sciname}')
print(aphiaID)
```

Code cell

```
<Response [200]>
```

Web-based environment



- Free repository
- Stores content
- Binder config file
- Jupyter Notebook files
- CSV files

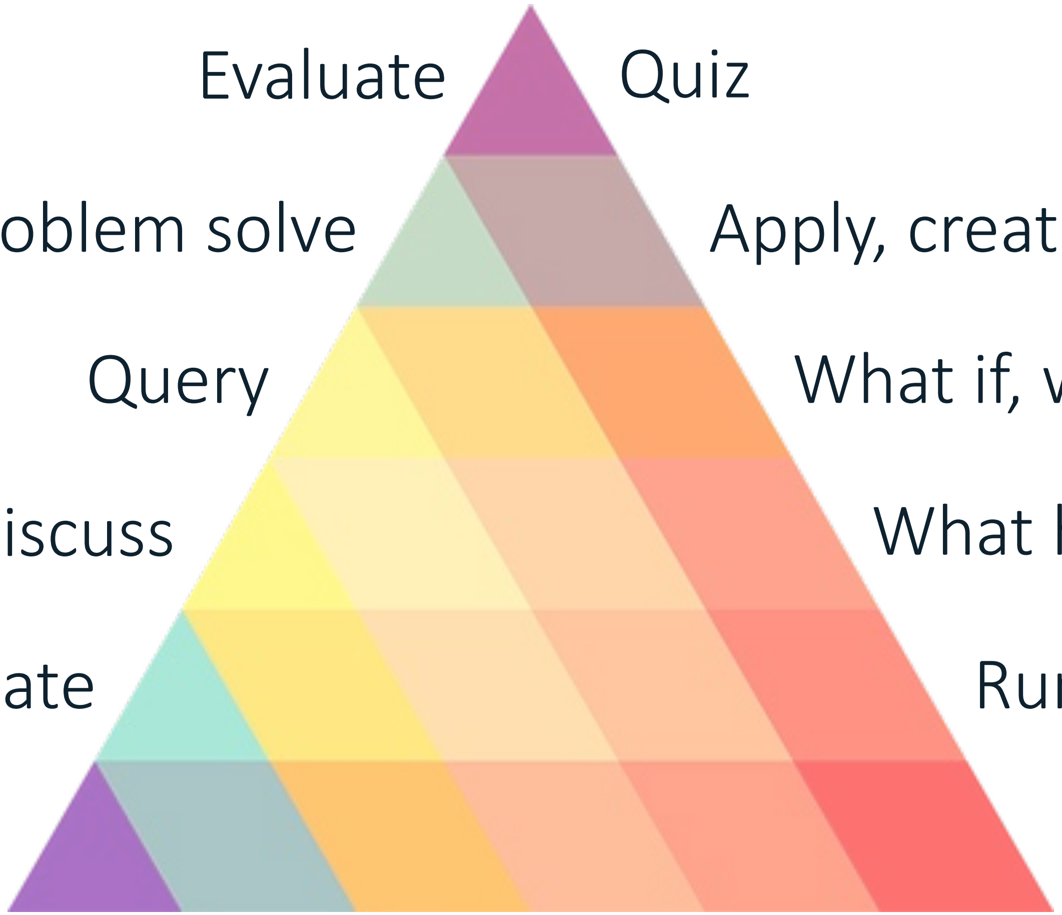
- Free service
- Creates Docker image of Python, using config file

- Free service
- Serves Docker image/Jupyter Lab on the Web

Interactive Python environment with no installation

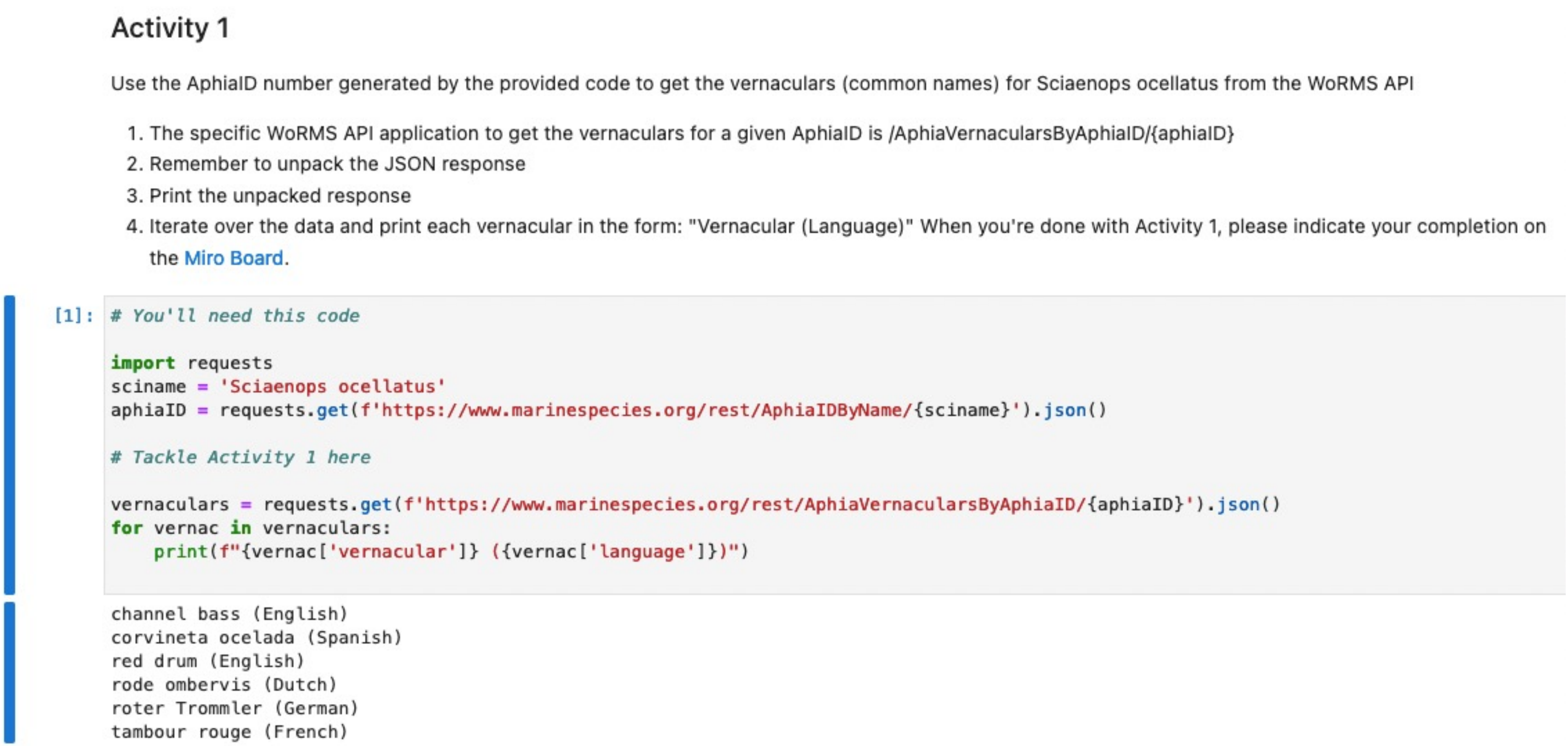
See https://the-turing-way.netlify.app/communication/binder/zero-to-binder.html

Scaffolded learning



Evaluate	Quiz
Problem solve	Apply, create, troubleshoot
Query	What if, what would happen...?
Discuss	What happened? Why?
Replicate	Run/modify code
Watch	Code demonstration

Active Learning: Consolidate knowledge



Activity 1

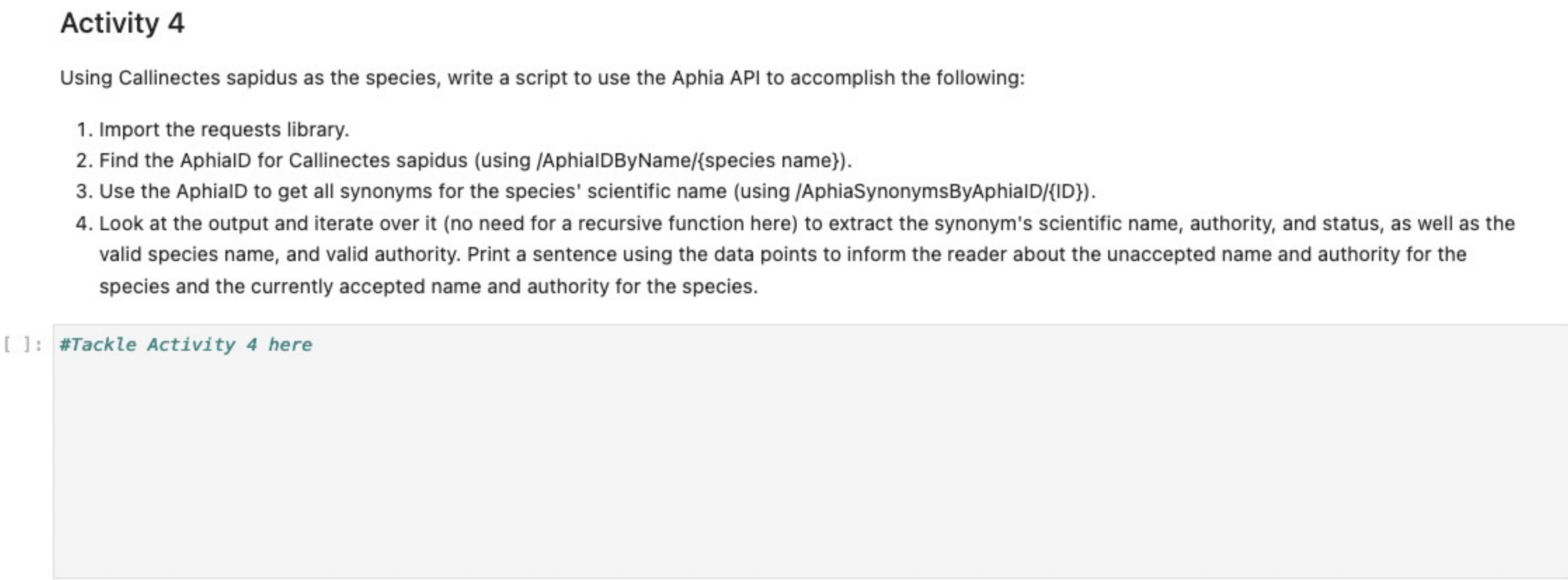
Use the AphiaID number generated by the provided code to get the vernaculars (common names) for Sciaenops ocellatus from the WoRMS API

1. The specific WoRMS API application to get the vernaculars for a given AphiaID is /AphiaVernacularsByAphiaID/{aphiaID}
2. Remember to unpack the JSON response
3. Print the unpacked response
4. Iterate over the data and print each vernacular in the form: "Vernacular (Language)" When you're done with Activity 1, please indicate your completion on the Miro Board.

```
# You'll need this code

import requests
sciname = 'Sciaenops ocellatus'
aphiaID = requests.get(f'https://www.marinespecies.org/rest/AphiaIDByName/{sciname}').json()

# Tackle Activity 1 here

vernaculars = requests.get(f'https://www.marinespecies.org/rest/AphiaVernacularsByAphiaID/{aphiaID}').json()
for vernac in vernaculars:
    print(f"{vernac['vernacular']} ({vernac['language']})")
```

```
channel bass (English)
corvineta ocelada (Spanish)
red drum (English)
rode ombervis (Dutch)
roter Trommler (German)
tambour rouge (French)
```

Higher order thinking: Problem solving



Activity 4

Using Callinectes sapidus as the species, write a script to use the Aphia API to accomplish the following:

1. Import the requests library.
2. Find the AphiaID for Callinectes sapidus (using /AphiaIDByName/{species name}).
3. Use the AphiaID to get all synonyms for the species' scientific name (using /AphiaSynonymsByAphiaID/{ID}).
4. Look at the output and iterate over it (no need for a recursive function here) to extract the synonym's scientific name, authority, and status, as well as the valid species name, and valid authority. Print a sentence using the data points to inform the reader about the unaccepted name and authority for the species and the currently accepted name and authority for the species.

```
[ ]: #Tackle Activity 4 here
```

Registration

- Two cohorts of 9 initially offered
- Demand for third cohort
- Three cohorts launched
- Waitlist of 6

Affiliation	Registration Count	Full Participation Count
College of Charleston (Graduate student)	6	6
SC Department of Natural Resources	16	16
NOAA	5	3
Totals	**27**	**25**

Expressed Expectations

How do you hope to use Python in the future?

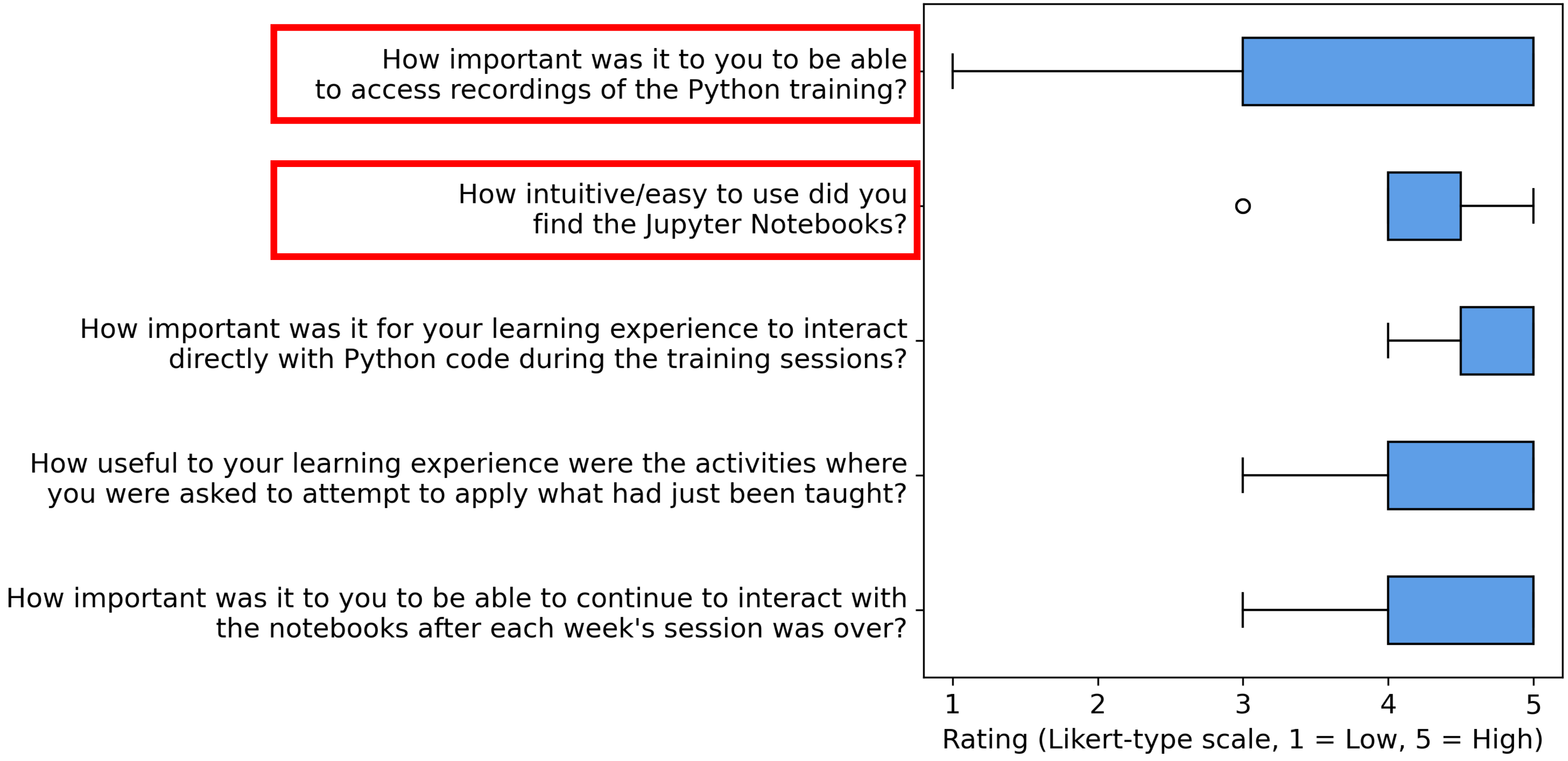
Learning Aspiration	Count (n=22 stickies)
GIS	11
Data analysis	7
Data collection automation	2
Genetics/Genomics	2
Develop applications/advanced scripting	2
Not sure	2
Modeling	1

Analyze images of oyster reefs and develop GIS application about oyster recruitment

Best laid plans

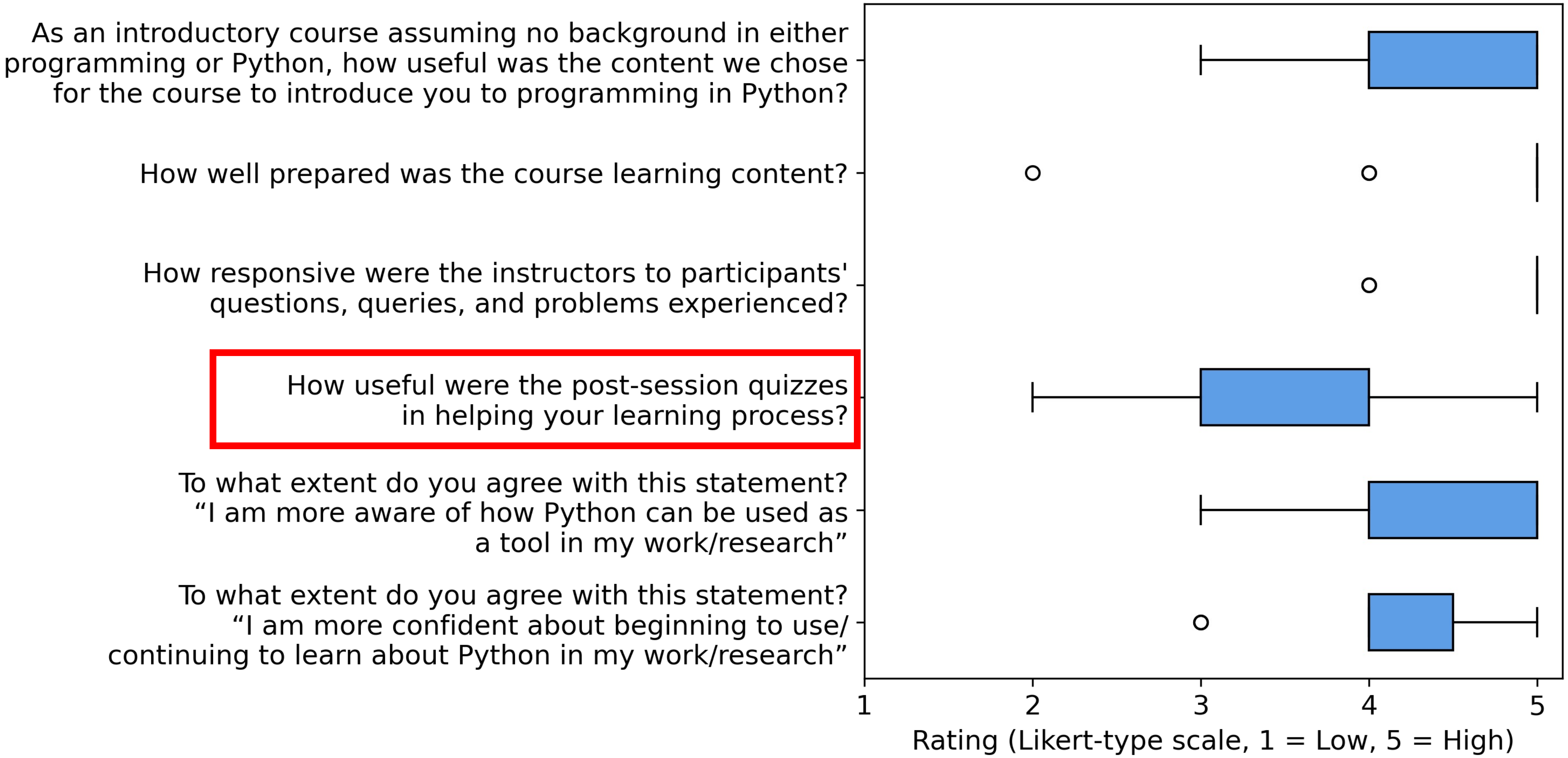
- Demand to go hybrid (Zoom + in person)
- Recordings requested
- Schedule conflicts = ~~cohorts~~
- Mixed skill levels/expectations
- Roll with the punches!

Outcomes



N=15

Outcomes



N=15

Could we offer this?
(But I don’t know Python)

- Library as facilitator
  - Coordinate/host training
  - Curate learning resources/online guide
  - Python Users Group
  - Showcase Python-based projects
- Great opportunity for partnerships!
- Learn Python!

What next?

- Update intro course
  - New session: Python installation (Anaconda package)
  - Use participant’s datasets
- Prepare short higher-level courses in:
  - Data analysis (Pandas library)
  - Data visualization (Matplotlib library)

From Zero to Python in 10.5 Hours

Thank you!

Geoff Timms, College of Charleston Libraries
timmsgp@cofc.edu

Marine Resources Library
A SC Marine Resources Center Partner

Thanks to Jeff Guyon, Branch Chief, Key Species and Bioinformatics Branch
NOAA/NCCOS Charleston