

Introduction

SURP 2022 Python Bootcamp

Ohio State Astronomy

Slides by: James W. Johnson

Hello!

We'd like to introduce ourselves

- Amanda Ash (2nd year)
- Anusha Pai (4th year)
- Deb Pathak (1st year)
- James Johnson (6th year)
- Joy Bhattacharyya (3rd year)
- Kaz Gary (1st year)
- Liam Dubay (2nd year)
- Paarmita Pandey (1st year)

Format

Slides typically take 60-90 minutes

- ~5-minute break partway through
- Strongly advised to *not* copy example codes – the slides are all online anyway
- Spend your mental energy on *understanding* rather than *copying*

Each session has a set of exercises

- We won't be collecting/grading anything – this isn't a course

Material is very condensed to fit into six sessions – mastery of these skills will come over time, even beyond this summer

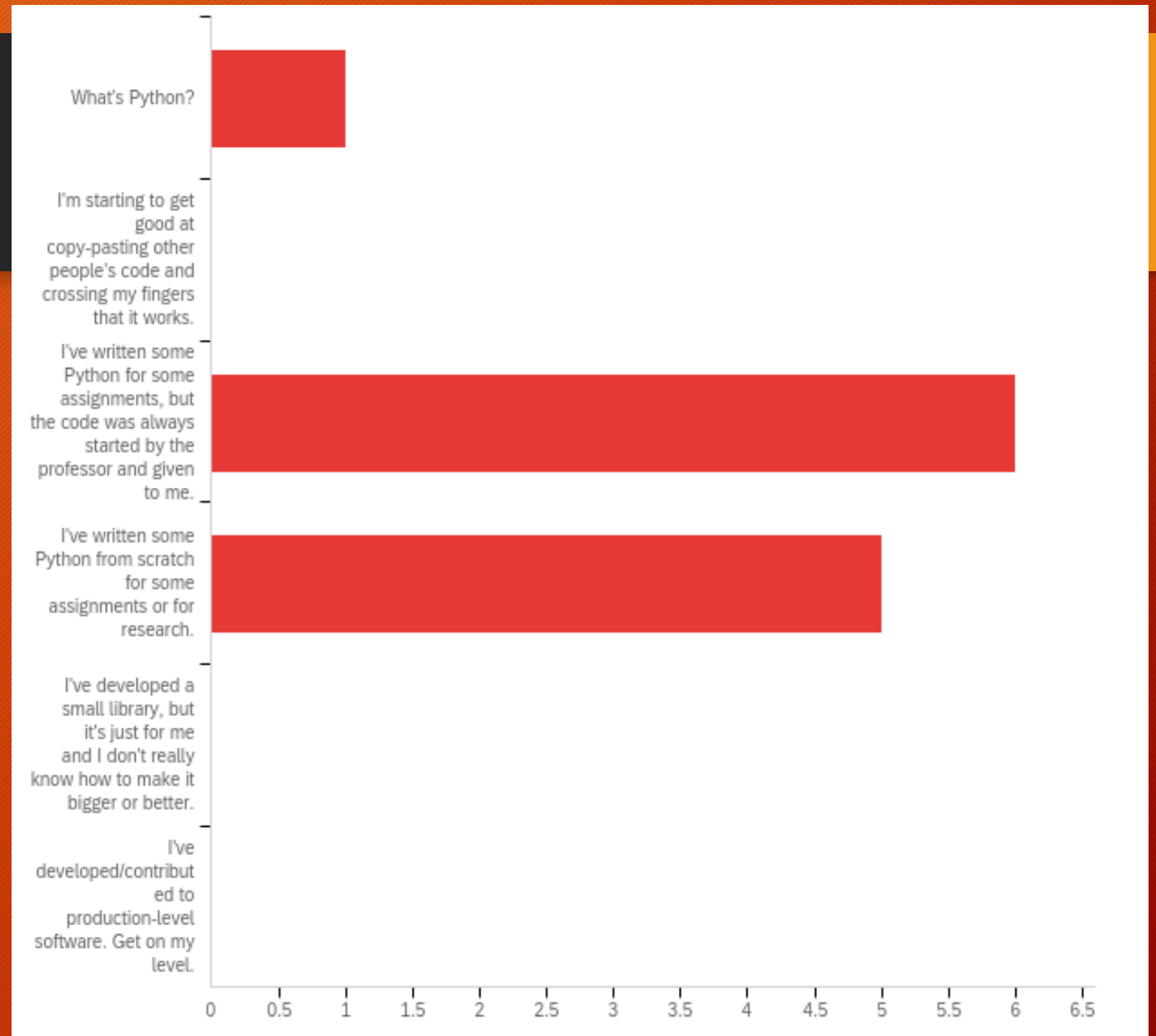
In the long run, what you get out of this will reflect what you put into it

Survey Responses

Q1: How would you describe your skill level in Python?

A wide range of skills, but mostly new to Python

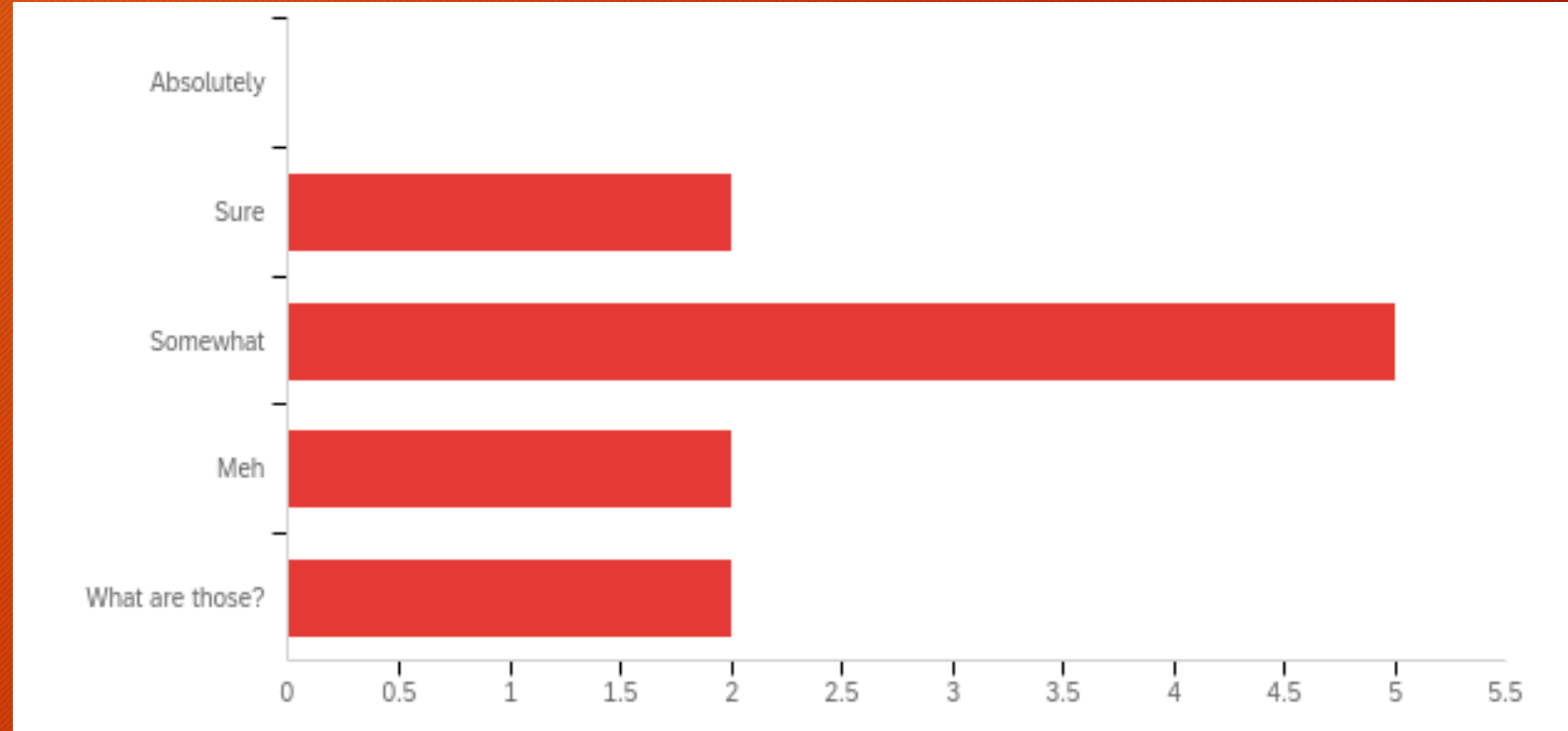
We're aiming to get you up to the second-to-last option



Survey Responses

Q2: Are you comfortable with if/else conditions, for- and while-loops, and def statements?

A wide range of skills in functional programming

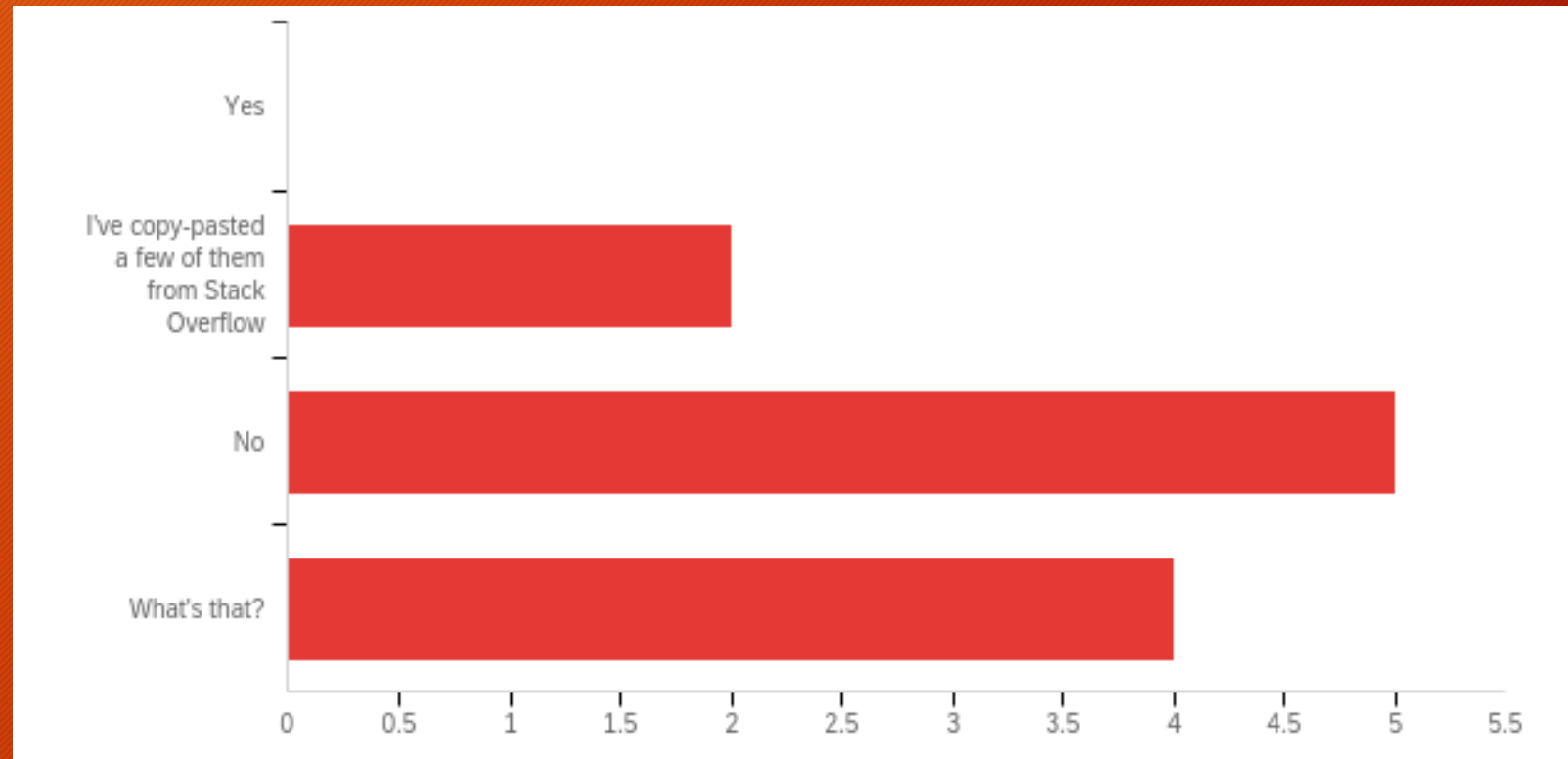


Survey Responses

Q3: Have you ever written a class?

Most of you are new to classes

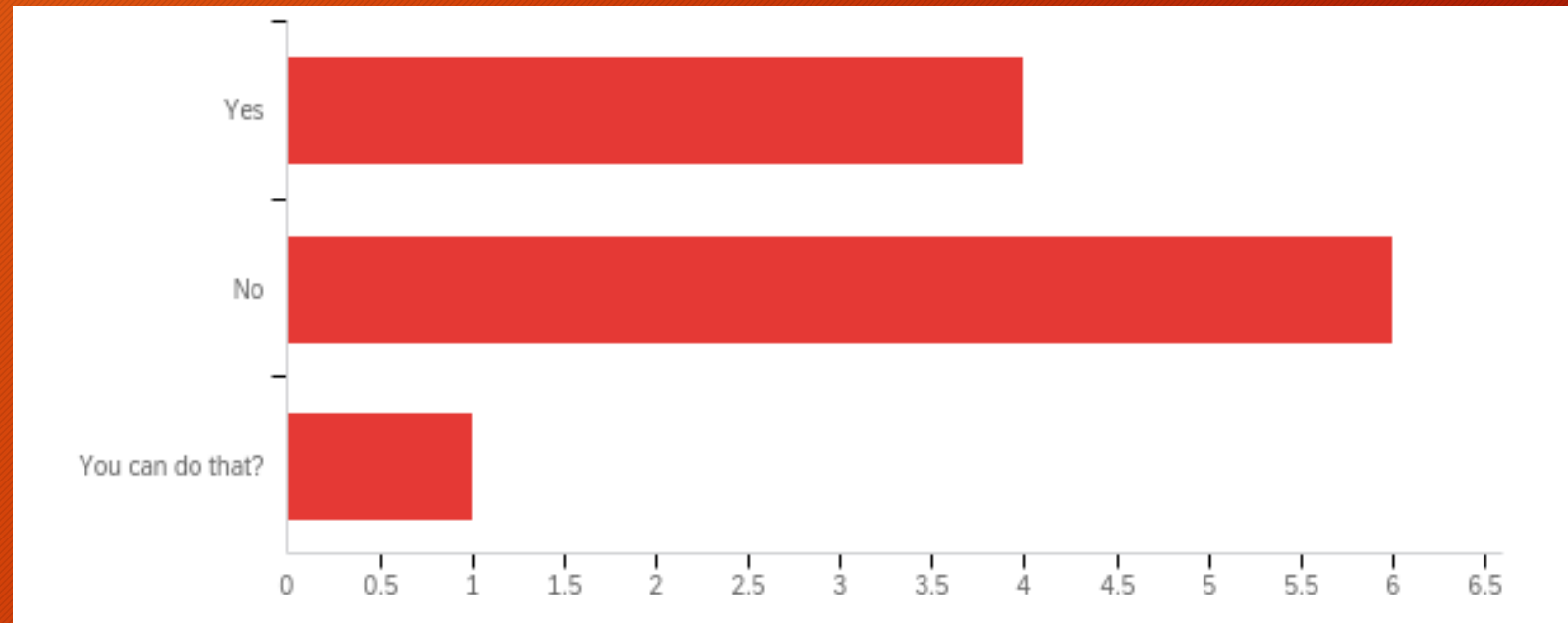
This will be the focus of sessions 4 and 5



Survey Responses

Q4: Do you know how to, or have you already imported your own code from another file?

Some of you have, some of you haven't

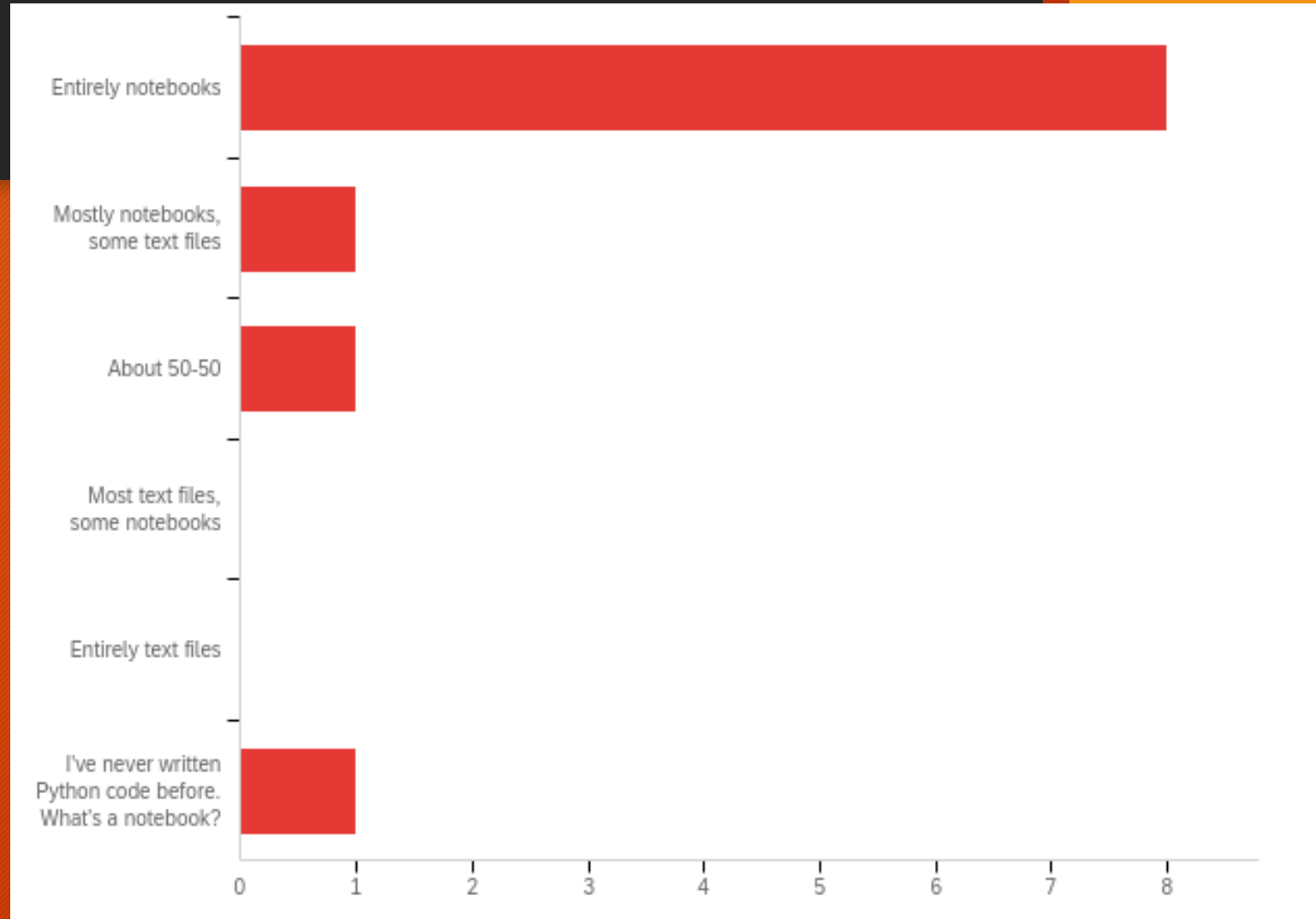


Survey Responses

Q5: Do you write code in text files or jupyter notebooks (or some equivalent)?

Mostly notebooks, but a few of you have worked in text files

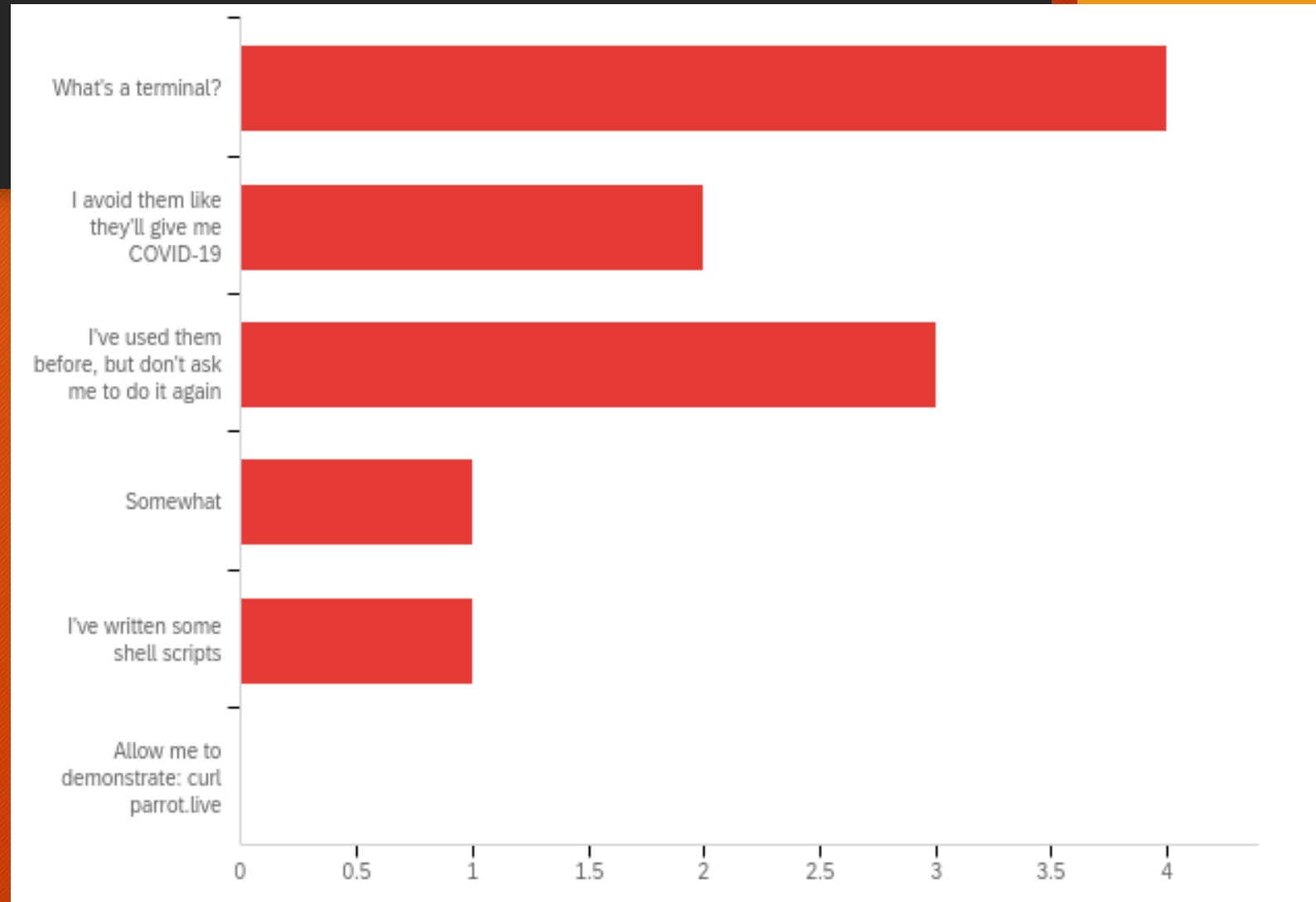
When you've built up a large code base for a project, they serve different purposes



Survey Responses

Q6: How comfortable are you using a terminal/command-line?

Some of you have never used them, others have



Survey Responses

Q6: How comfortable are you using a terminal/command-line?

Some of you have never used them, others have



Survey Responses

New material:

- For many of you: the terminal, object-oriented programming
- For about half of you:
 - How to import your own code from elsewhere in your computer (packaging may be new to some)

Some of the material is either impossible or quite difficult when coding in a notebook. I advise all of you to use this bootcamp as practice for working in text files.

Additional Learning Material

Those of you new to Python – we strongly advise going through some of this ahead of Wednesday's session, which will be a crash course

Python Foundation's Beginner's Guide: <https://www.python.org/about/gettingstarted/>

<https://www.learnpython.org/> - There is also an iOS app for this

Codecademy: <https://www.codecademy.com/learn/learn-python-3>

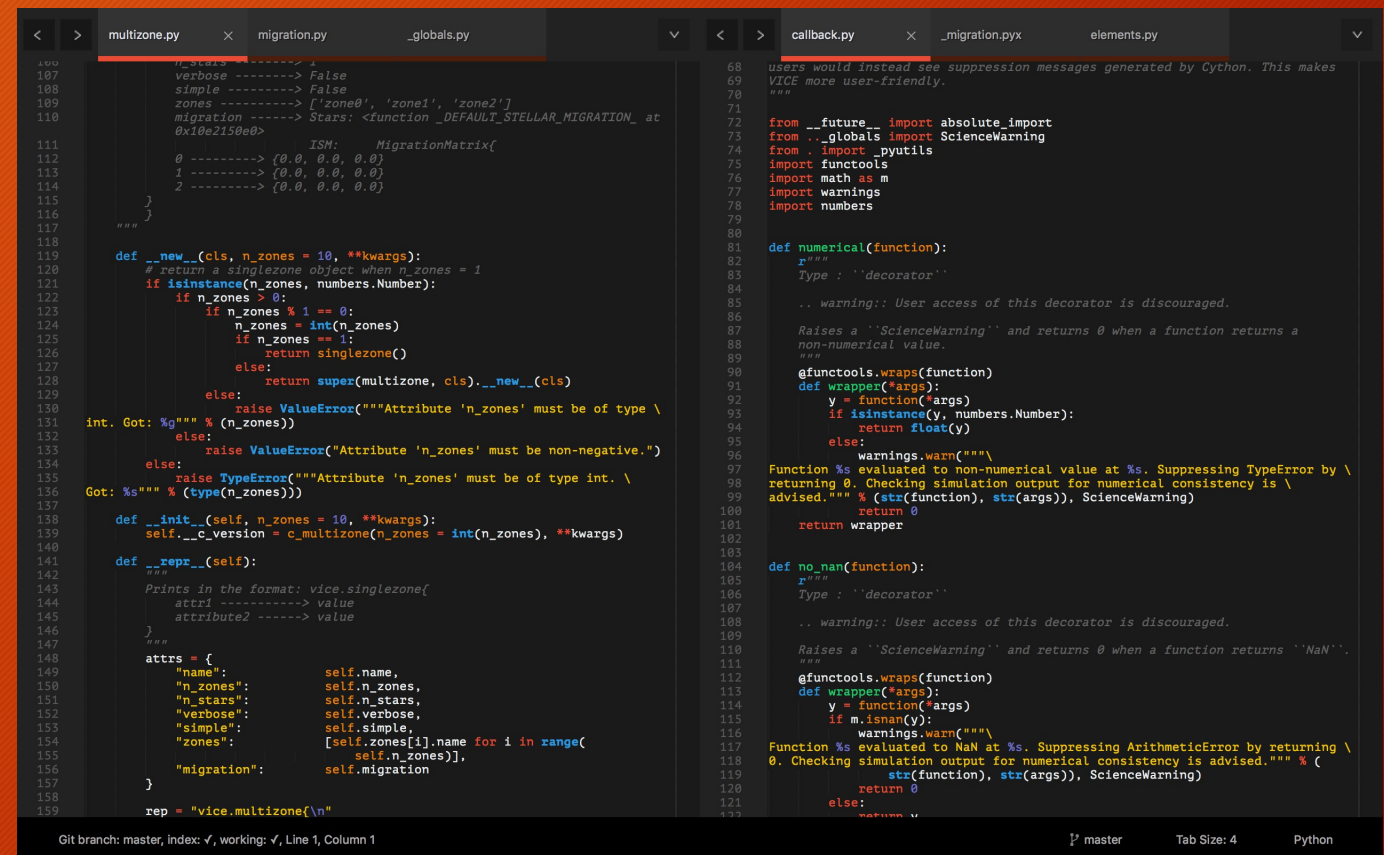
Tools: A Text Editor

Differs from an *Integrated Development Environment* (IDE) in that IDEs will *run* the code – all they do is open, create, edit, etc. plain text files

- pycharm, spyder

I recommend Sublime Text

<https://www.sublimetext.com/>



```
107 verbose -----> False
108 simple -----> False
109 zones -----> ['zone0', 'zone1', 'zone2']
110 migration -----> Stars: <function _DEFAULT_STELLAR_MIGRATION_ at
0x10e2150e0>
111
112         ISM: MigrationMatrix{
113         0 -----> {0.0, 0.0, 0.0}
114         1 -----> {0.0, 0.0, 0.0}
115         2 -----> {0.0, 0.0, 0.0}
116     }
117 }
118 """
119
120 def __new__(cls, n_zones = 10, **kwargs):
121     # return a singlezone object when n_zones = 1
122     if isinstance(n_zones, numbers.Number):
123         if n_zones > 0:
124             if n_zones % 1 == 0:
125                 n_zones = int(n_zones)
126                 if n_zones == 1:
127                     return singlezone()
128             else:
129                 return super(multizone, cls).__new__(cls)
130         else:
131             raise ValueError("Attribute 'n_zones' must be of type \
132 int. Got: %g" % (n_zones))
133     else:
134         raise ValueError("Attribute 'n_zones' must be non-negative.")
135     else:
136         raise TypeError("Attribute 'n_zones' must be of type int. \
137 Got: %s" % (type(n_zones)))
138
139 def __init__(self, n_zones = 10, **kwargs):
140     self.__c_version = c_multizone(n_zones = int(n_zones), **kwargs)
141
142 def __repr__(self):
143     """
144     Prints in the format: vice.singlezone{
145     attr1 -----> value
146     attribute2 -----> value
147     }
148     """
149     attrs = {
150         "name": self.name,
151         "n_zones": self.n_zones,
152         "n_stars": self.n_stars,
153         "verbose": self.verbose,
154         "simple": self.simple,
155         "zones": [self.zones[i].name for i in range(
156 self.n_zones)],
157         "migration": self.migration
158     }
159     rep = "vice.multizone{\n"
68 users would instead see suppression messages generated by Cython. This makes
69 VICE more user-friendly.
70 """
71
72 from __future__ import absolute_import
73 from ..globals import ScienceWarning
74 from .. import pyutils
75 import functools
76 import math as m
77 import warnings
78 import numbers
79
80
81 def numerical(function):
82     r"""
83     Type : 'decorator'
84
85     .. warning:: User access of this decorator is discouraged.
86
87     Raises a 'ScienceWarning' and returns 0 when a function returns a
88     non-numerical value.
89     """
90     @functools.wraps(function)
91     def wrapper(*args):
92         y = function(*args)
93         if isinstance(y, numbers.Number):
94             return float(y)
95         else:
96             warnings.warn("""\
97 Function %s evaluated to non-numerical value at %. Suppressing TypeError by \
98 returning 0. Checking simulation output for numerical consistency is \
99 advised.""", % (str(function), str(args)), ScienceWarning)
100         return 0
101     return wrapper
102
103
104 def no_nan(function):
105     r"""
106     Type : 'decorator'
107
108     .. warning:: User access of this decorator is discouraged.
109
110     Raises a 'ScienceWarning' and returns 0 when a function returns 'NaN'.
111     """
112     @functools.wraps(function)
113     def wrapper(*args):
114         y = function(*args)
115         if m.isnan(y):
116             warnings.warn("""\
117 Function %s evaluated to NaN at %. Suppressing ArithmeticError by returning \
118 0. Checking simulation output for numerical consistency is advised.""", % (
119 str(function), str(args)), ScienceWarning)
120         return 0
121     else:
122         return y
123 """
```

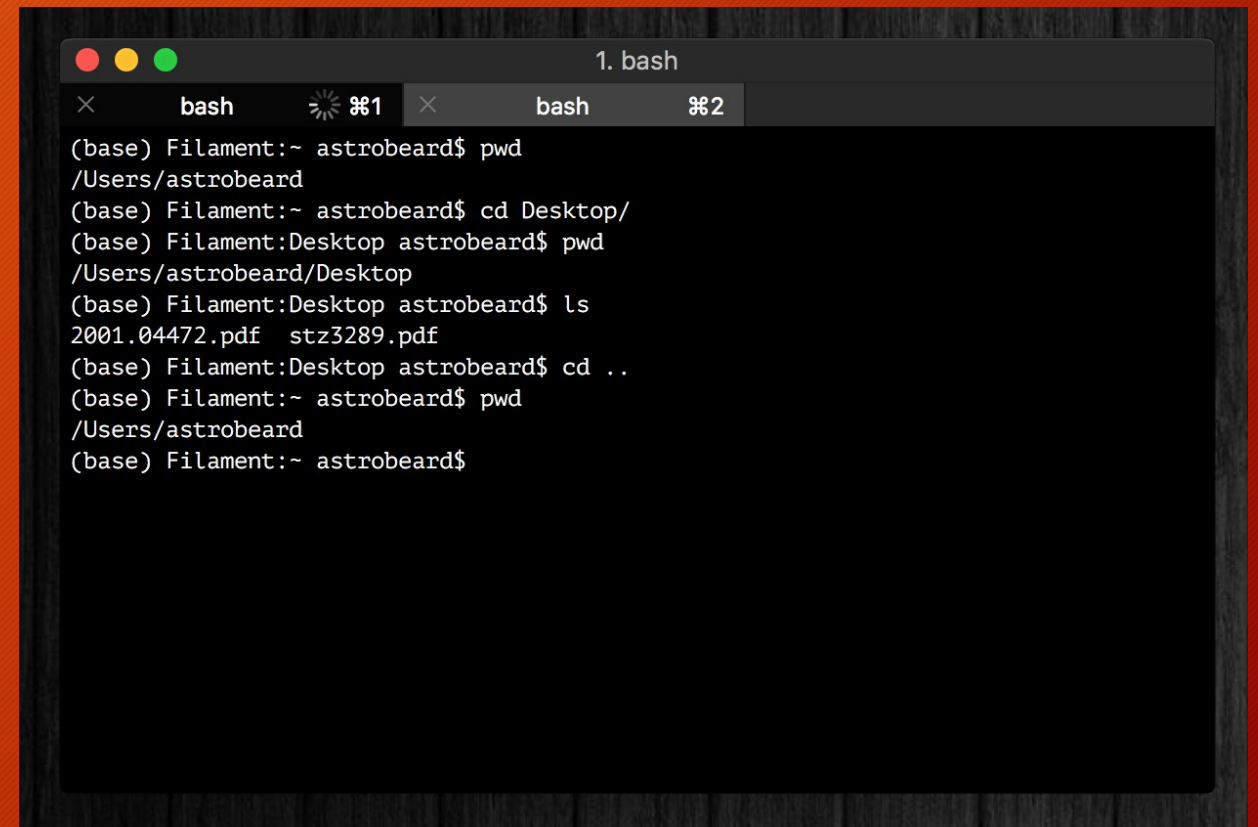

Tools: A Terminal

I recommend iTerm2

- Terminal *replacement*
- <https://www.iterm2.com/>

You should think of a terminal as just a different interface on a Finder window with some extra programs built-in

Tip: you can run python line-by-line in a terminal (*python* or *ipython*)

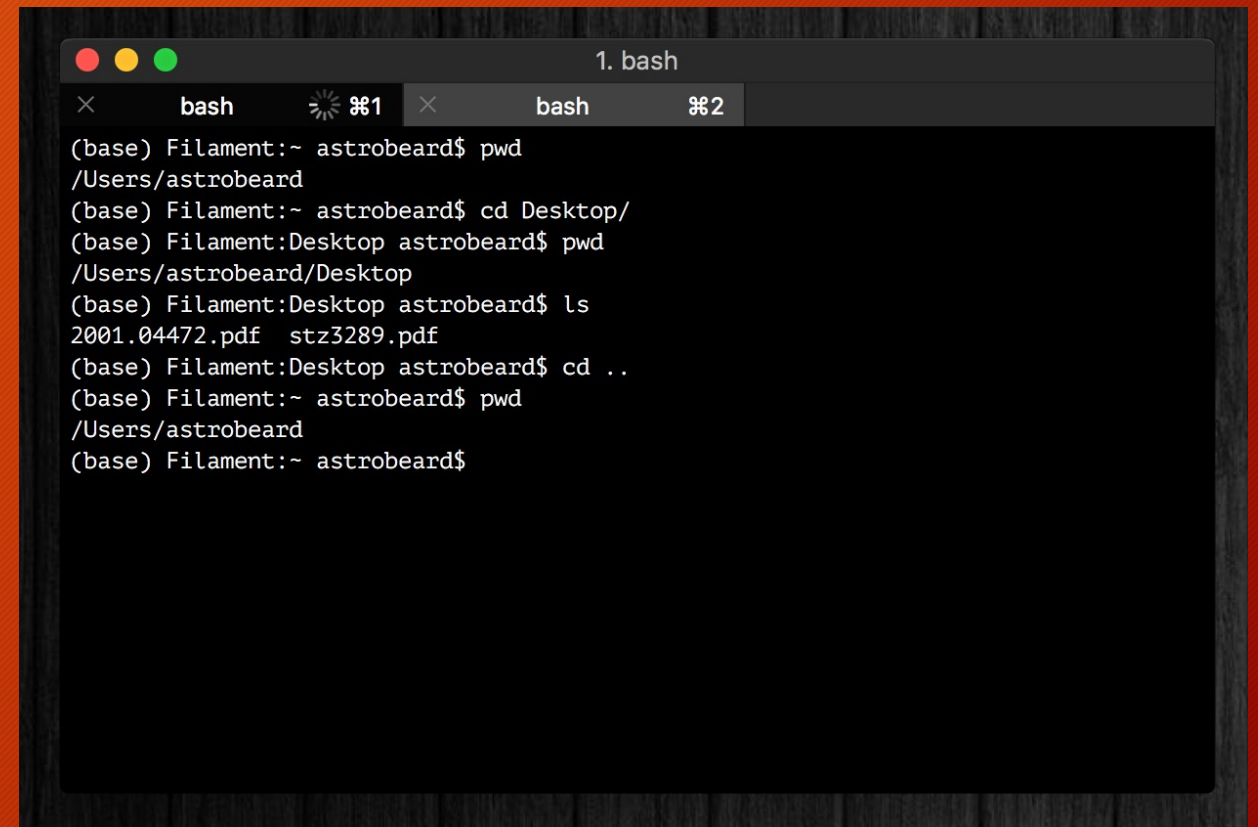


```
1. bash
× bash ⓘ1 × bash ⓘ2
(base) Filament:~ astrobeard$ pwd
/Users/astrobeard
(base) Filament:~ astrobeard$ cd Desktop/
(base) Filament:Desktop astrobeard$ pwd
/Users/astrobeard/Desktop
(base) Filament:Desktop astrobeard$ ls
2001.04472.pdf stz3289.pdf
(base) Filament:Desktop astrobeard$ cd ..
(base) Filament:~ astrobeard$ pwd
/Users/astrobeard
(base) Filament:~ astrobeard$
```


Tools: A Terminal

If you're running Windows, your terminal will be different than some of the notes and exercises here, unless you take some extra steps at the beginning to set up a bash interpreter.

Talk to us if you need help with this!



```
1. bash
× bash ⓘ1 × bash ⓘ2
(base) Filament:~ astrobeard$ pwd
/Users/astrobeard
(base) Filament:~ astrobeard$ cd Desktop/
(base) Filament:Desktop astrobeard$ pwd
/Users/astrobeard/Desktop
(base) Filament:Desktop astrobeard$ ls
2001.04472.pdf stz3289.pdf
(base) Filament:Desktop astrobeard$ cd ..
(base) Filament:~ astrobeard$ pwd
/Users/astrobeard
(base) Filament:~ astrobeard$
```


Tools: Cloud Computing

Allows you to run python on a remote server

SciServer is a popular platform across many STEM fields:

<https://www.sciserver.org/>



In the long run you should choose the tools that you're most comfortable with

If You Haven't Already

<https://www.anaconda.com/products/individual>

- This will install Python, Anaconda, and Jupyter Notebooks

Latest version of python: 3.11.3

- Most libraries now require ≥ 3.8
- Python 3.7 will be deprecated June 27, 2023

Python 2.7 is *deprecated*

- Talk to us if you need to use it, for now just don't install that version



Goals

What we'll aim to cover:

- How to use a terminal
- Review of the basics: control structures, data types, functions, import, etc.
- How to read documentation
- Common uses of Anaconda in astronomy
- How to import your own code, and how to set up a directory tree to organize it
- Classes: how to make new objects
 - Inheritance and Composition
- Some basic software engineering principles (i.e. good habits)

Monday Motivation

Every expert coder was once a novice.

You can't improve your coding practices without first criticizing what you once thought was great code.

Since we're not professional developers, scientists have to *actively* create and foster good coding habits if they want them. Being early career researchers, you have the option to make this decision now. It will only become more difficult to do this.