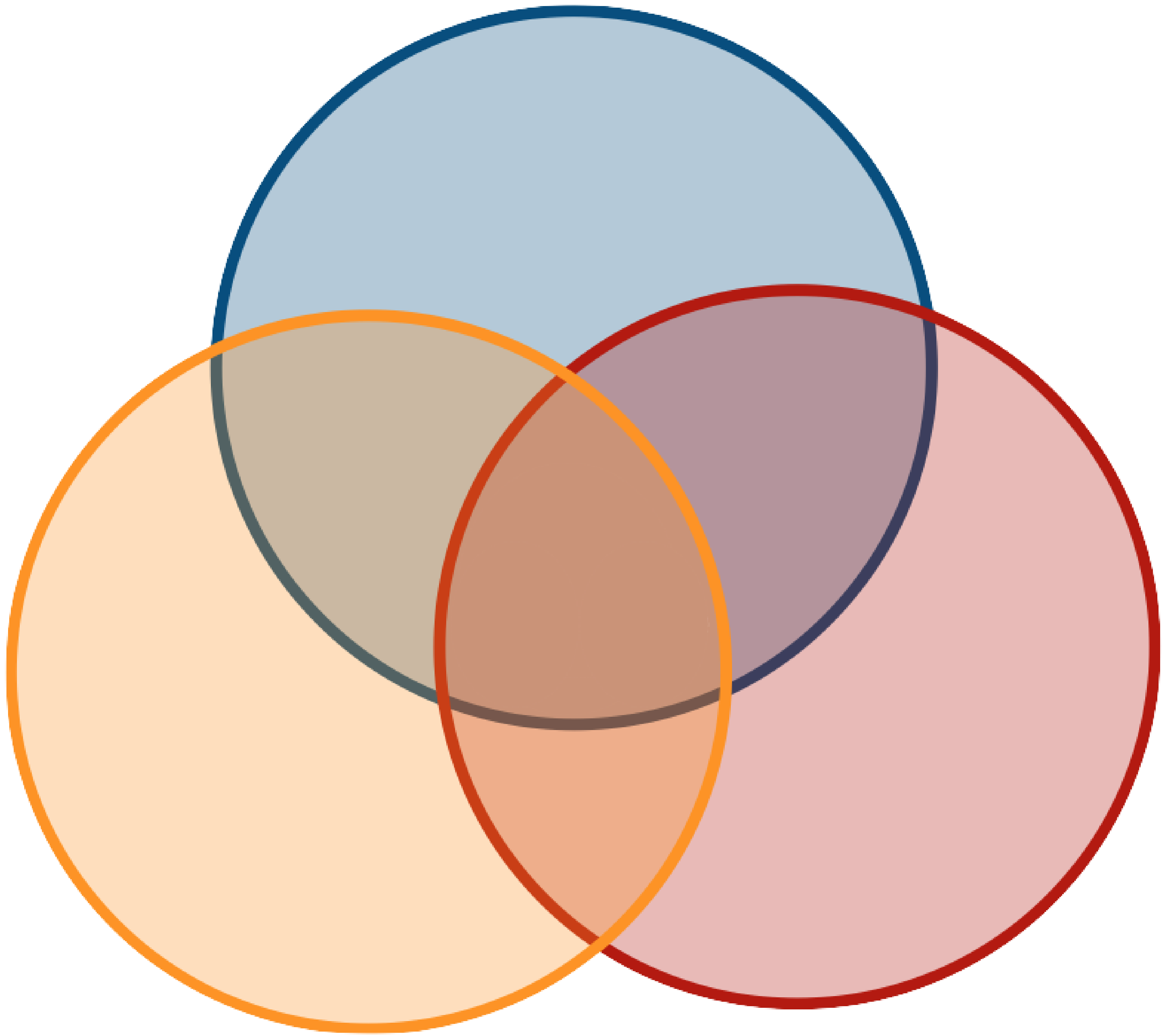
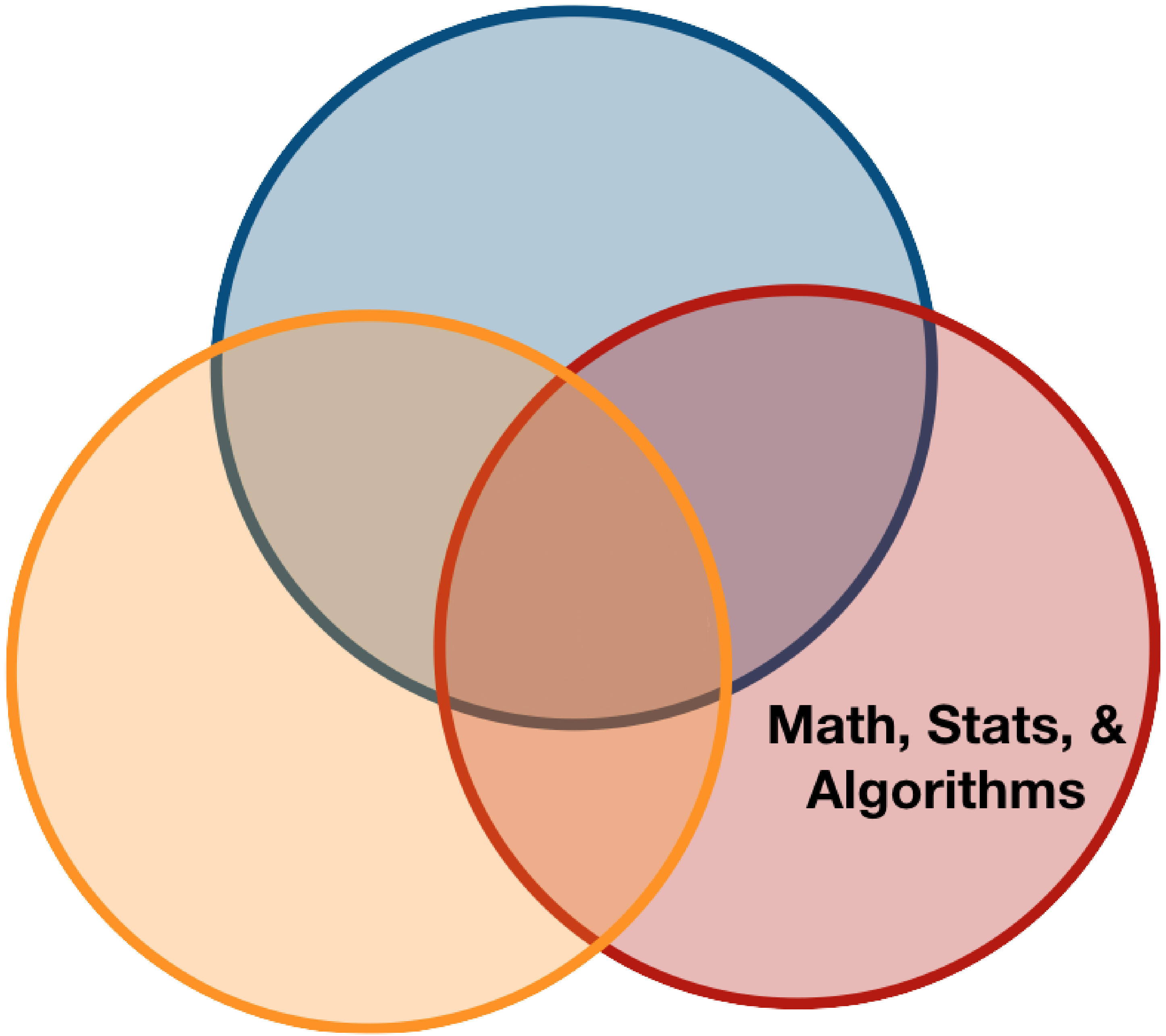
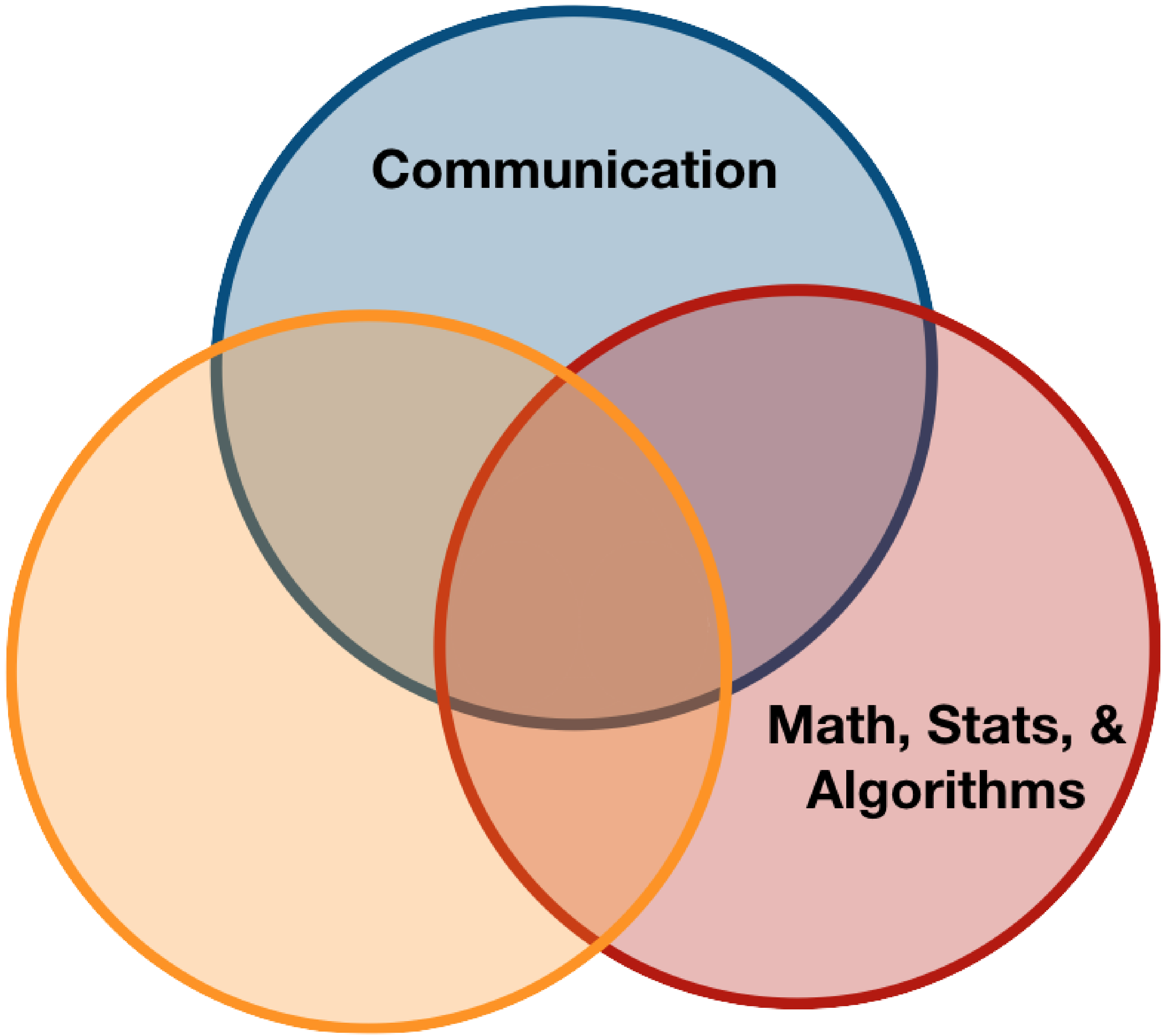
Python, data science, & software engineering

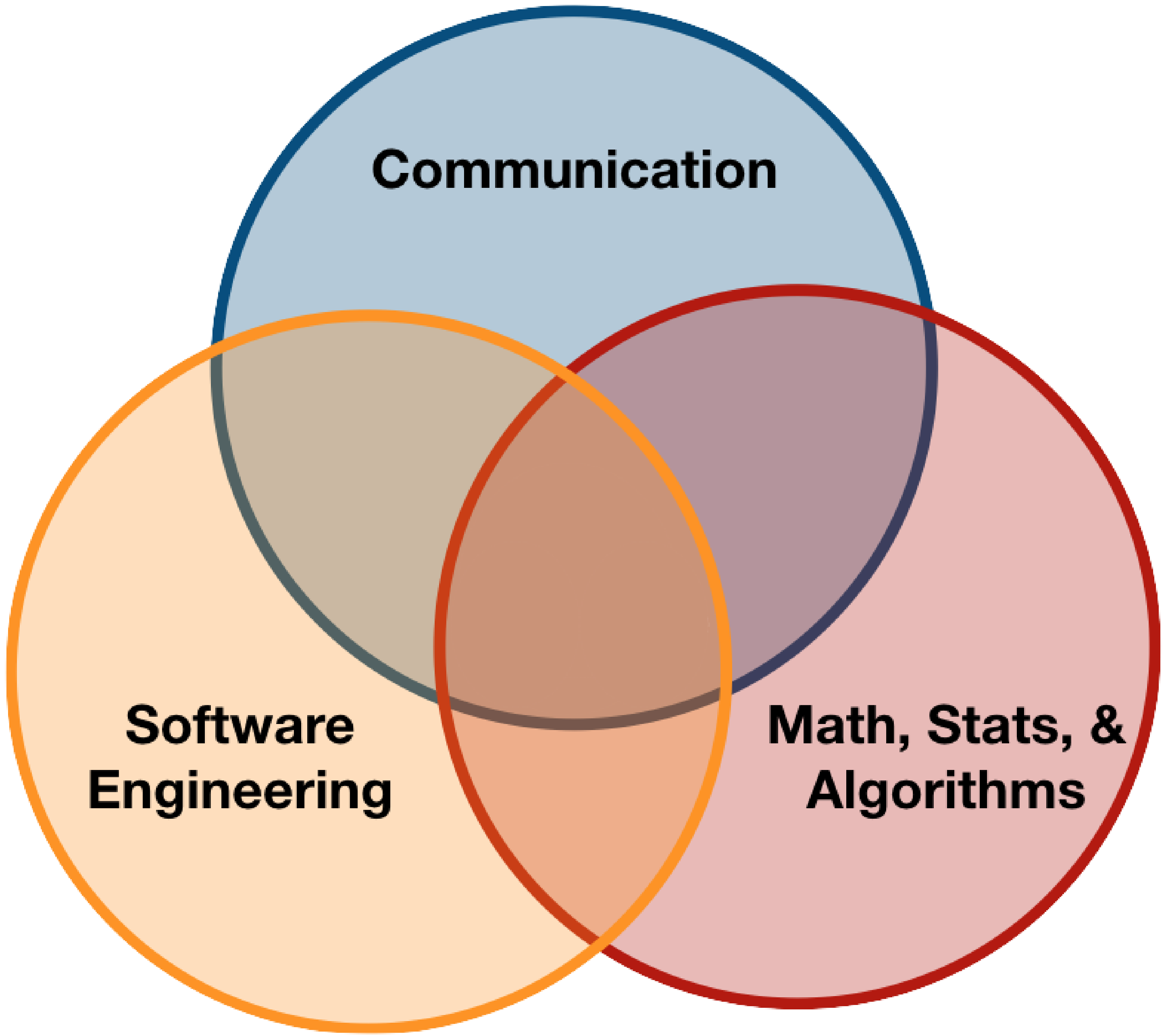
SOFTWARE ENGINEERING FOR DATA SCIENTISTS IN PYTHON

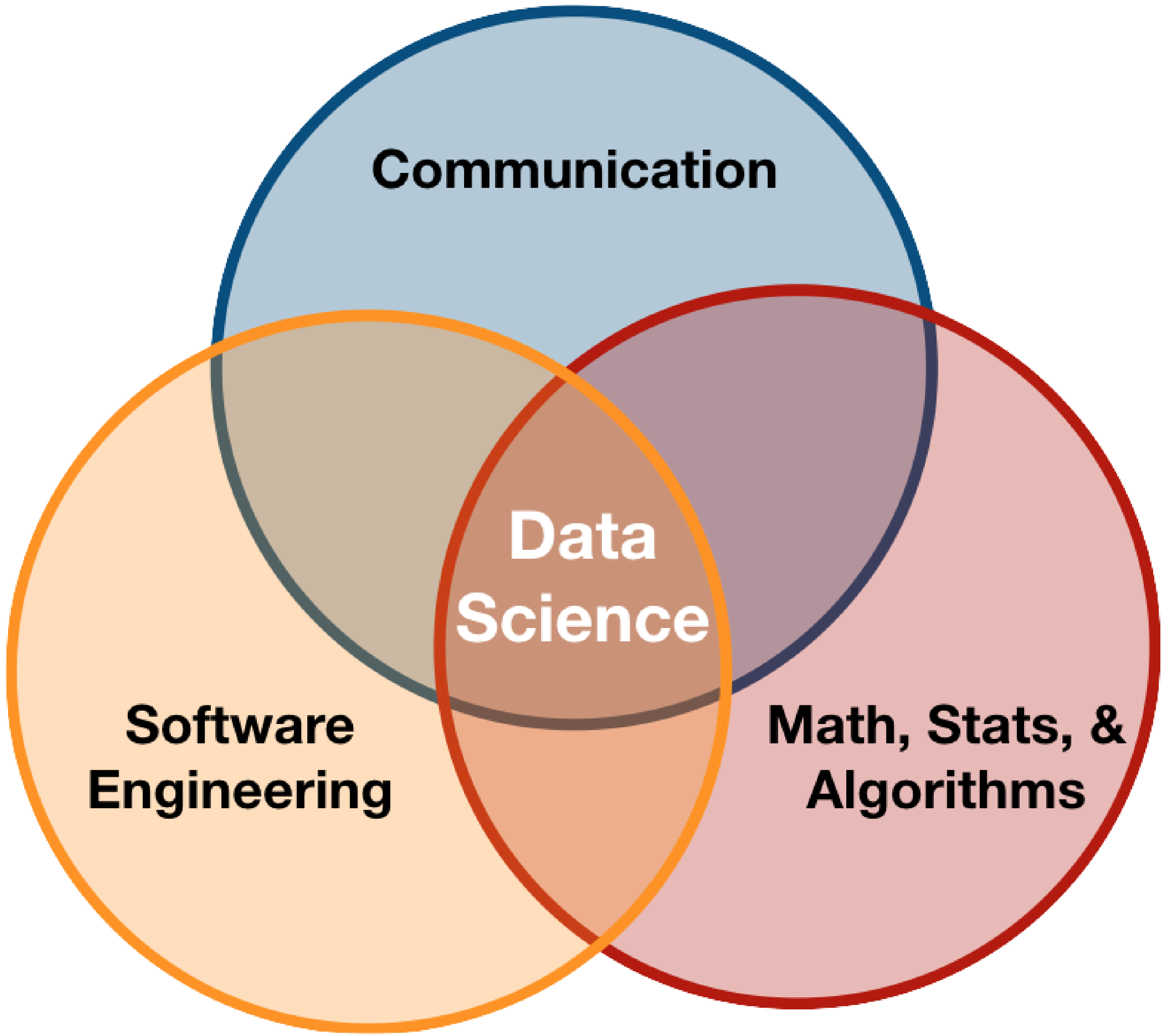
Adam Spannbauer
Machine Learning Engineer at Eastman




Math, Stats, &
Algorithms


Communication
Math, Stats, &
Algorithms


Communication
Software Engineering
Math, Stats, & Algorithms

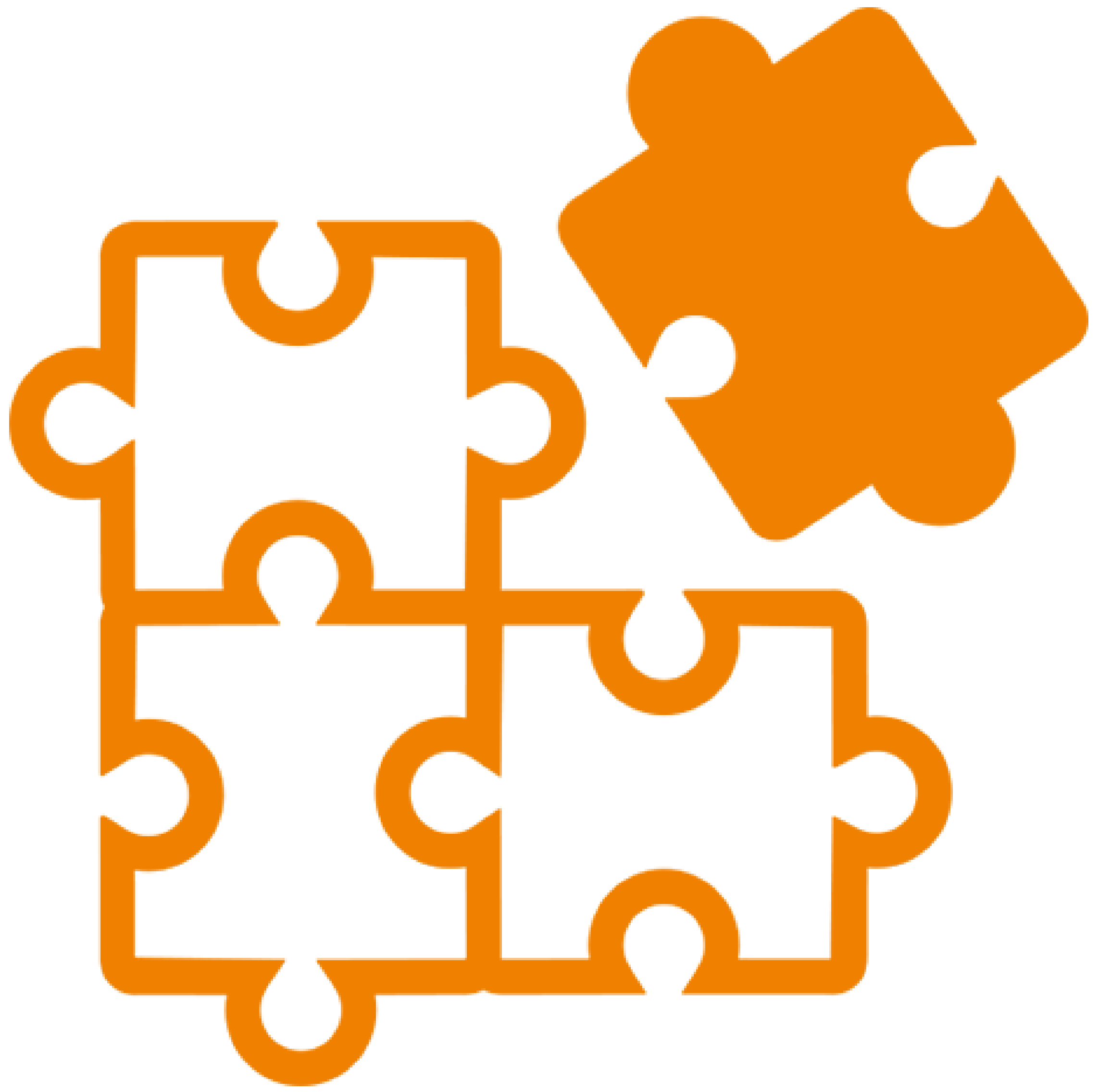

Communication
Data Science
Software Engineering
Math, Stats, & Algorithms

Software engineering concepts

- Modularity
- Documentation
- Testing
- Version Control & Git

Benefits of modularity

- Improve readability
- Improve maintainability
- Solve problems only once

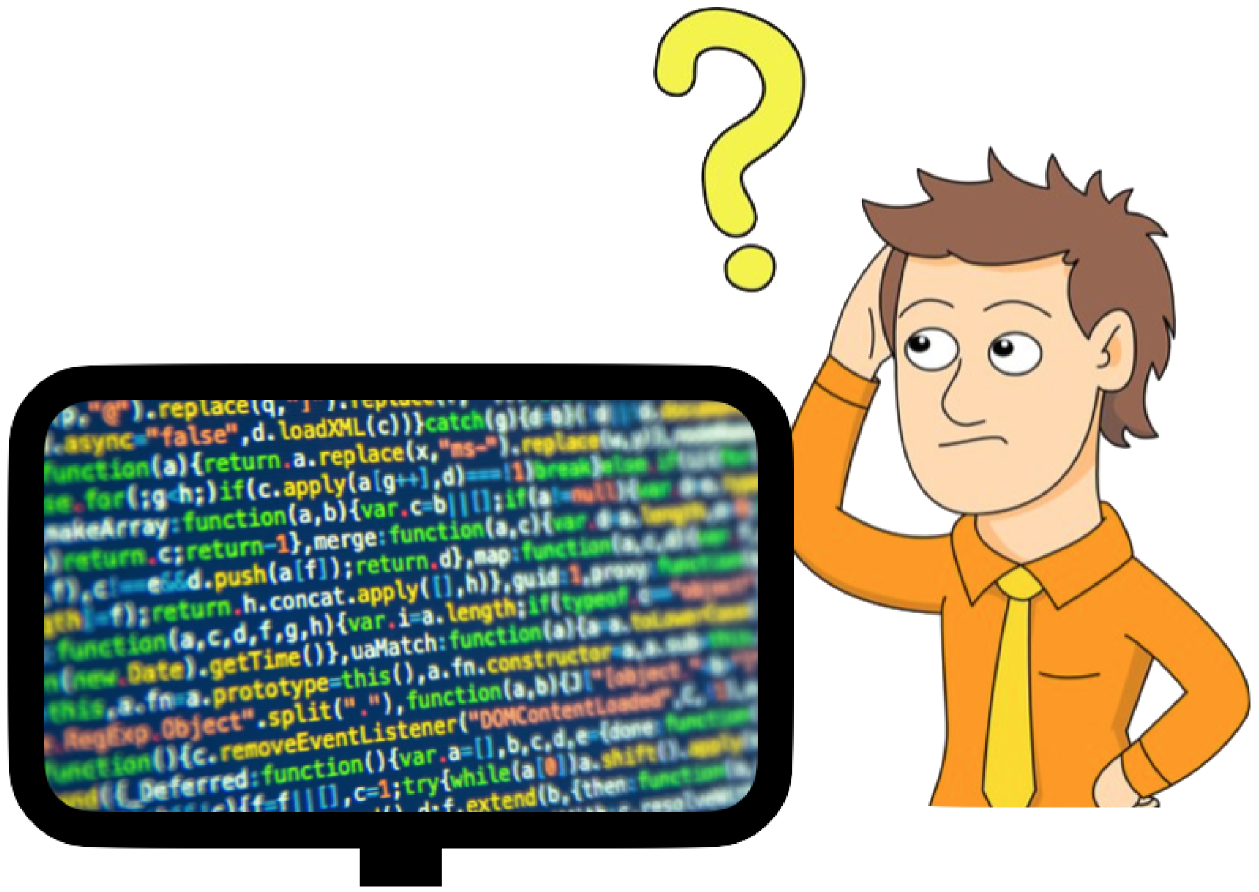


Modularity in python

```python
# Import the pandas PACKAGE
import pandas as pd

# Create some example data
data = {'x': [1, 2, 3, 4],
        'y': [20.1, 62.5, 34.8, 42.7]}

# Create a dataframe CLASS object
df = pd.DataFrame(data)

# Use the plot METHOD
df.plot('x', 'y')
```

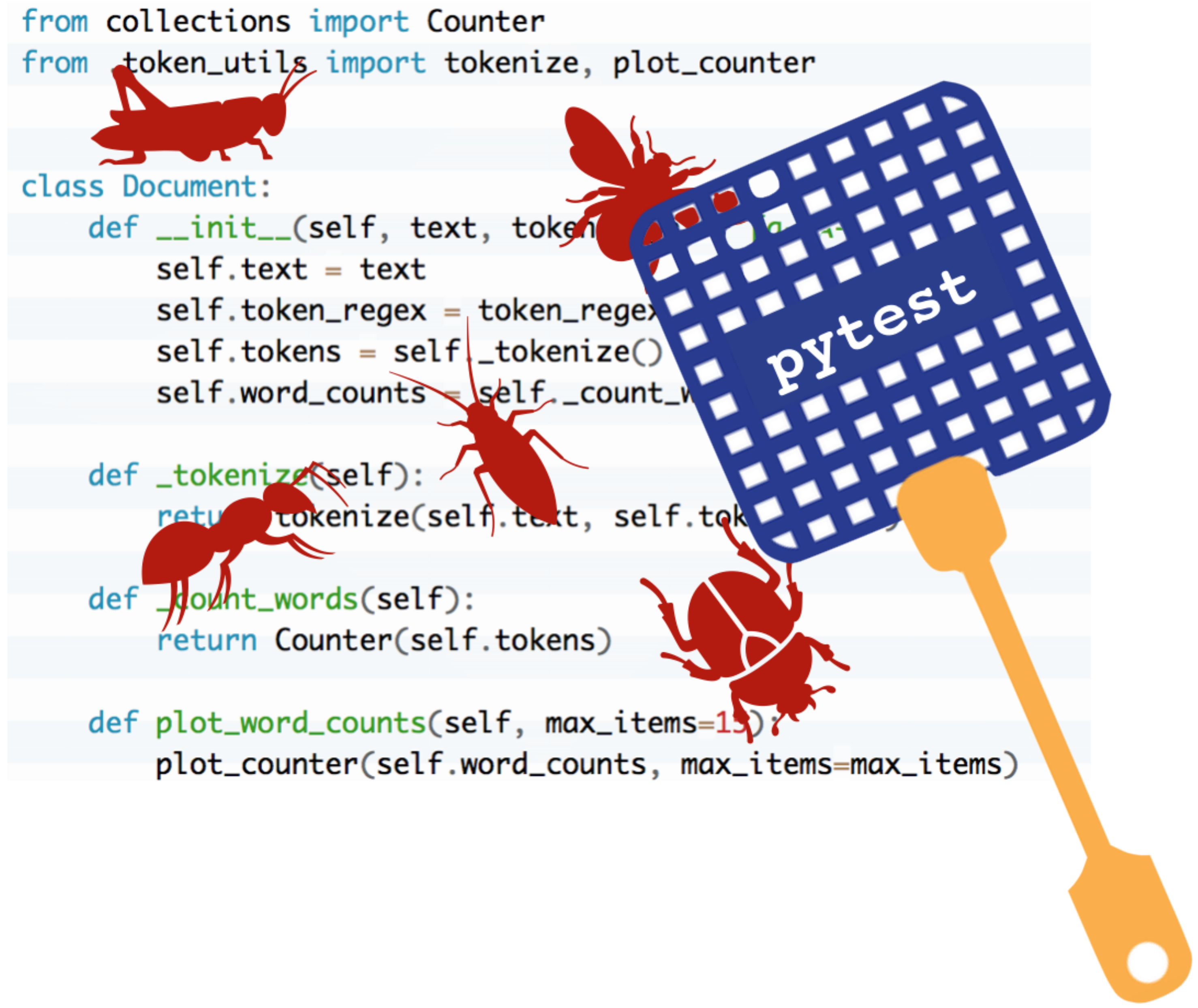
Benefits of documentation

- Show users how to use your project
- Prevent confusion from your collaborators
- Prevent frustration from future you



Benefits of automated testing

- Save time over manual testing
- Find & fix more bugs
- Run tests anytime/anywhere



```
from collections import Counter
from token_utils import tokenize, plot_counter


class Document:
    def __init__(self, text, token
        self.text = text
        self.token_regex = token_rege
        self.tokens = self._tokenize()
        self.word_counts = self._count_w

    def _tokenize(self):
        return tokenize(self.text, self.tok

    def _count_words(self):
        return Counter(self.tokens)

    def plot_word_counts(self, max_items=15):
        plot_counter(self.word_counts, max_items=max_items)
```

Let's Review

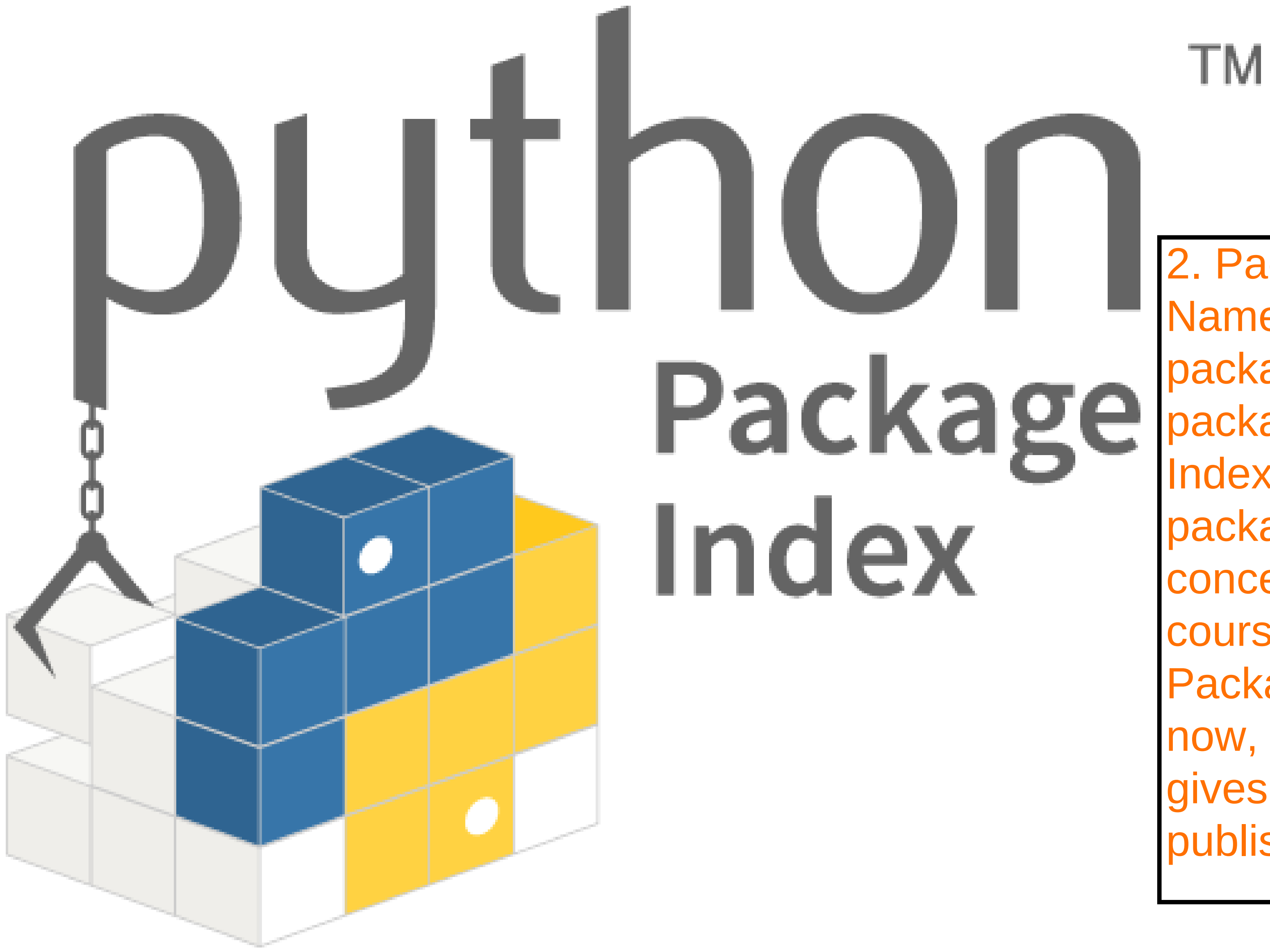
SOFTWARE ENGINEERING FOR DATA SCIENTISTS IN PYTHON

Introduction to Packages & Documentation

SOFTWARE ENGINEERING FOR DATA SCIENTISTS IN PYTHON

Adam Spannbauer
Machine Learning Engineer at Eastman

Packages and PyPi



2. Packages and PyPi
Namely, we're going to investigate packages. We'll cover how to install packages from the 'Python Package Index', and we'll see how these packages leverage software engineering concepts we've covered. Later in the course, we'll talk more about the 'Python Package Index', or PyPi for short, but for now, all you need to know is that PyPi gives us an easy platform to leverage published packages.

Intro to pip

```
$ pip install 📦
```

Intro to pip

python™
Package
Index

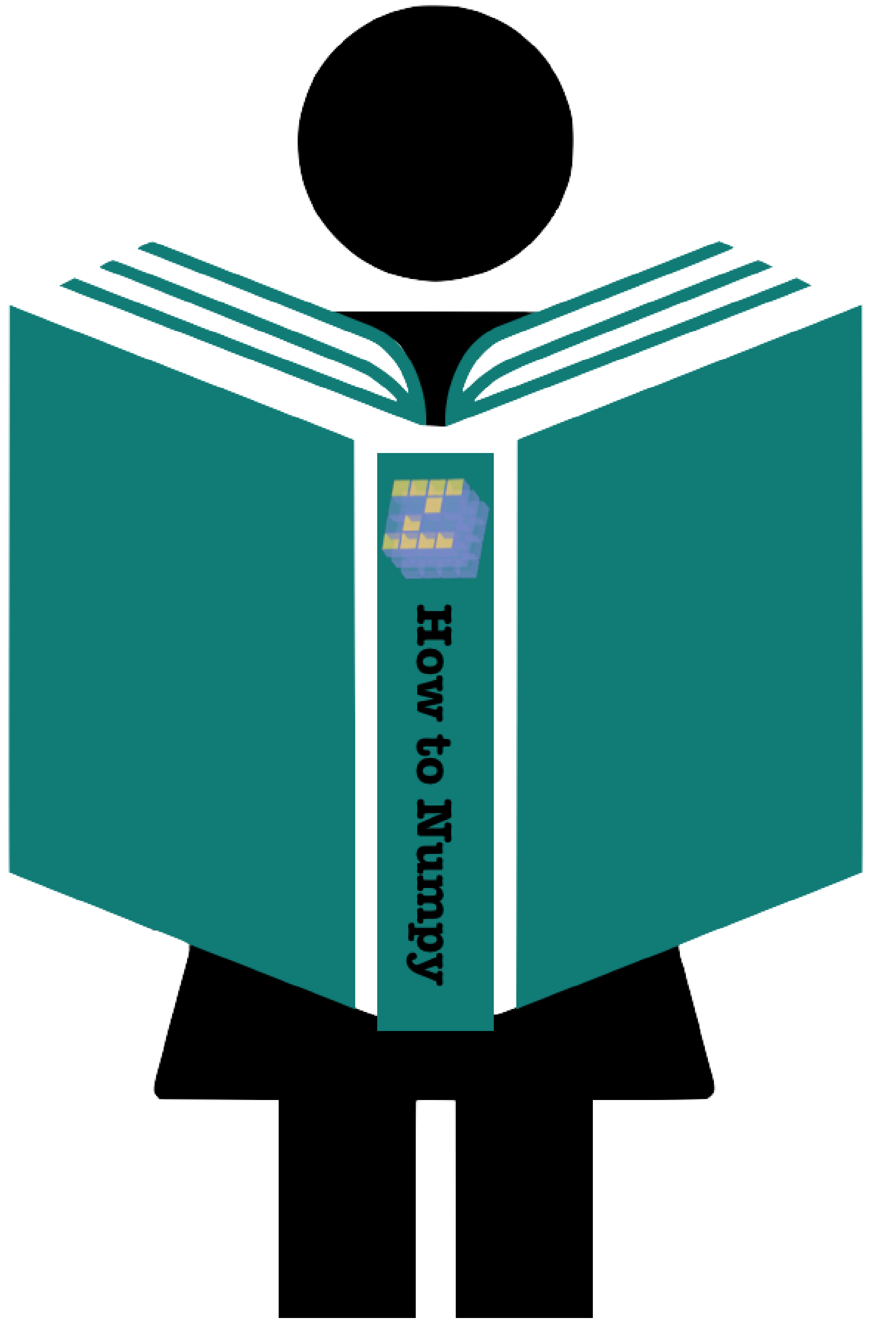
```
$ pip install 📦
```

Using pip to install numpy

```
datacamp@server:~$ pip install numpy
```

```
Collecting numpy
    100% |????????????????????????????????| 24.5MB 44kB/s
Installing collected packages: numpy
Successfully installed numpy-1.15.4
```

How do we use numpy?



Reading documentation with help()

```
help(numpy.busday_count)
```

```
busday_count(begindates, enddates)
    Counts the number of valid days between `begindates` and
    `enddates`, not including the day of `enddates`.

    Parameters
    ----------
    begindates : the first dates for counting.
    enddates : the end dates for counting (excluded from the count)

    Returns
    -------
    out : the number of valid days between the begin and end dates.

    Examples
    --------
    >>> # Number of weekdays in 2011
    ...  np.busday_count('2011', '2012')
    260
```

Reading documentation with help()

```python
import numpy as np
help(np)
```

```
    Provides
      1. An array object of arbitrary homogeneous items
      2. Fast mathematical operations over arrays
      3. Linear Algebra, Fourier Transforms, Random Number Generation
```

```python
help(42)
```

```
class int(object)
 |  int(x=0) -> integer
 |  int(x, base=10) -> integer
 |
 |  Convert a number or string to an integer, or return 0 if no arguments
 |  are given.  If x is a number, return x.__int__().  For floating point
 |  numbers, this truncates towards zero.
```

Let's Practice

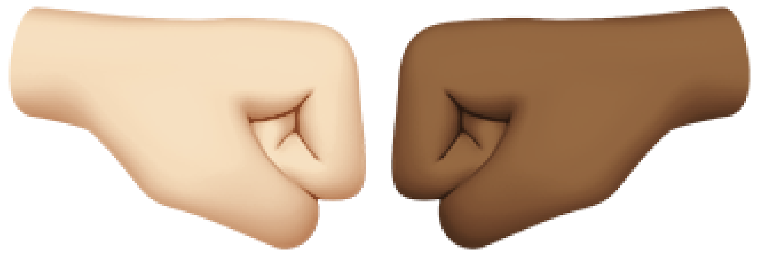
SOFTWARE ENGINEERING FOR DATA SCIENTISTS IN PYTHON

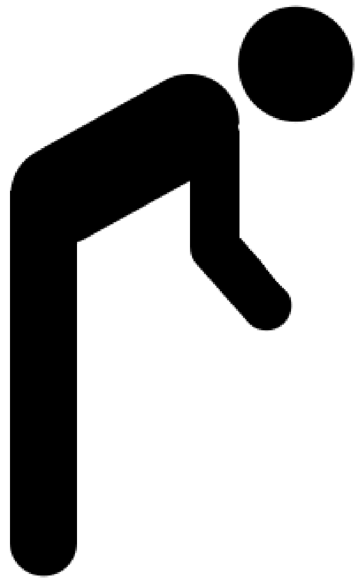
Conventions and PEP 8

SOFTWARE ENGINEERING FOR DATA SCIENTISTS IN PYTHON

Adam Spannbauer
Machine Learning Engineer at Eastman

What are conventions?





Introducing PEP 8

"Code is read much more often than it is written"

3. Introducing PEP 8
Luckily for Pythonistas, these conventions aren't unwritten. We can turn to Python Enhancement Protocol 8, or PEP 8. PEP 8 is the defacto Style Guide for Python Code. It lets us know how to format our code to be as readable as possible, and to quote PEP 8, 'code is read much more often than it is written'. So readability is not something that should be overlooked.

Violating PEP 8

```
#define our data
my_dict ={
    'a'  : 10,
'b': 3,
    'c'  :   4,
             'd': 7}
#import needed package
import numpy as np
#helper function
def DictToArray(d):
    """Convert dictionary values to numpy array"""
    #extract values and convert
              x=np.array(d.values())
              return x
print(DictToArray(my_dict))
```

Even without knowing the specific PEP 8 rules being broken, you can probably tell this isn't the most readable chunk of code.

```
array([10,  4,  3,  7])
```

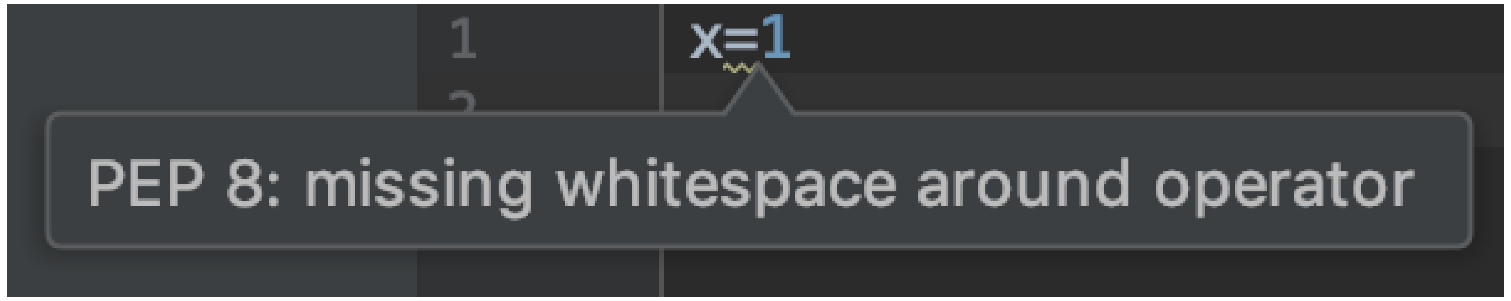
Following PEP 8

```python
# Import needed package
import numpy as np

# Define our data
my_dict = {'a': 10, 'b': 3, 'c': 4, 'd': 7}


# Helper function
def dict_to_array(d):
    """Convert dictionary values to numpy array"""
    # Extract values and convert
    x = np.array(d.values())
    return x


print(dict_to_array(my_dict))
```

```
array([10,  4,  3,  7])
```

5. Following PEP 8
Let's see what the same chunk of code looks like after being rewritten to conform to PEP 8. Much better. By following the agreed-upon rules in PEP 8 and using whitespace appropriately, this code became much more readable despite accomplishing the same exact task.

PEP 8 Tools



```
x=1
```

PEP 8: missing whitespace around operator

6. PEP 8 Tools
So how is it possible to keep up with the many rules defined in PEP 8? Thankfully for you and me, there are tools that can check your code for you just like a spellchecker. Personally, I use an IDE that flags violations as soon as I write a bad line of code, but there are other options. One in particular that you'll be exposed to in this course is the **pycodestyle package.** pycodestyle can check code in multiple files at once and it outputs descriptions of the violations along with information to let you know exactly where you need to go to fix the issue.

Using pycodestyle

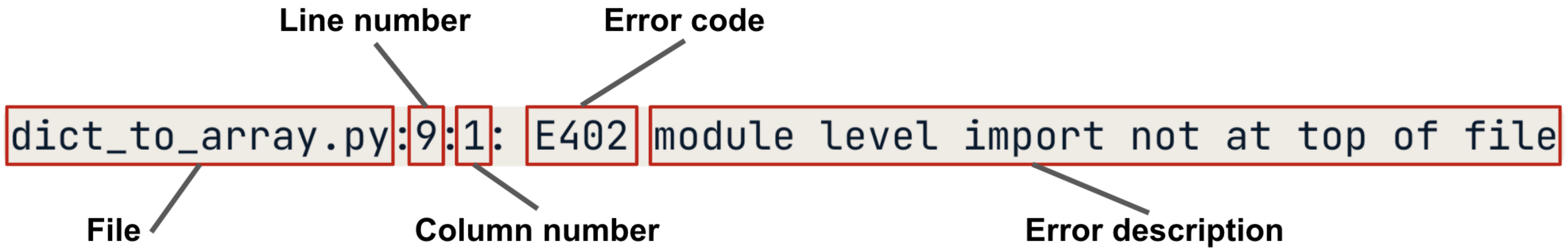
```
datacamp@server:~$ pip install pycodestyle
datacamp@server:~$ pycodestyle dict_to_array.py
```

```
dict_to_array.py:5:9: E203 whitespace before ':'
dict_to_array.py:6:14: E131 continuation line unaligned for hanging indent
dict_to_array.py:8:1: E265 block comment should start with '# '
dict_to_array.py:9:1: E402 module level import not at top of file
dict_to_array.py:11:1: E302 expected 2 blank lines, found 0
dict_to_array.py:13:15: E111 indentation is not a multiple of four
```

7. Using pycodestyle

Here is some example code on how to install and use pycodestyle from the shell. As you'll see in the exercises later you can also run pycodestyle from a python script. Here we use the command line interface to check the contents of the file dict to array dot py that contains our poorly formatted code from before. Note that the output we see on this slide has been truncated.

Output from pycodestyle



```
dict_to_array.py:9:1: E402 module level import not at top of file
```

8. Output from pycodestyle

The output shows us the exact location of any violations by showing the file name, line number, and column number where the problem occurred. Note that the output does not use zero based indexing. Additionally, pycodestyle outputs a human readable description of the PEP 8 violation and an error code. A complete list of pycodestyle's possible error codes can be seen in the package's documentation.

Let's Practice

SOFTWARE ENGINEERING FOR DATA SCIENTISTS IN PYTHON