

# Module 1: Exploratory Data Analysis in One Dimension

# Overview

- This module will introduce you to vectorized data structures and the Pandas library.
- You will learn how to explore single dimensional data sets.
- Pandas 2.x

**Vectorized Data Structures let you  
perform operations on your data all at once**

# Advantages of Vectorized Data Structures

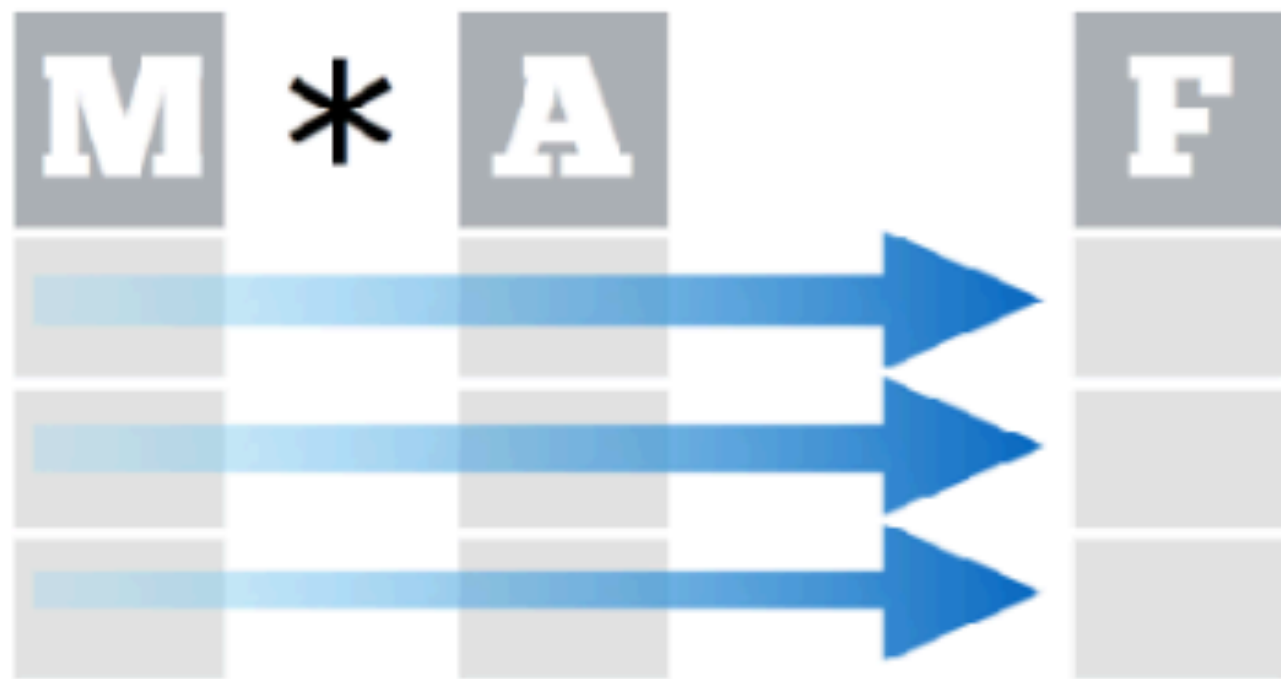
- More readable and simpler code
- Allows the module to optimize for CPU and Memory usage
- Pass serialized objects from one system to another. (More on this later)

```
for x in range(0, len( data )):  
    data[x] = data[x] + 1
```

```
for x in range(0, len( data )):  
    data[x] = data[x] + 1
```

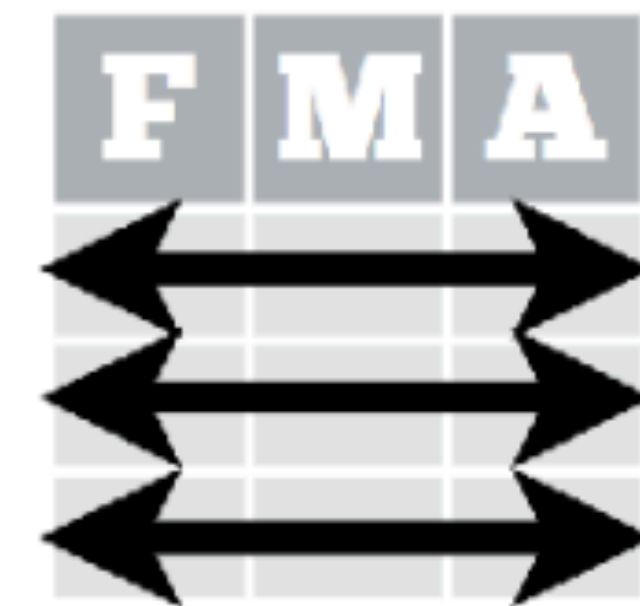
```
odds = evens + 1
```

$$F = M * A$$



Each **variable** is saved  
in its own **column**

&



Each **observation** is  
saved in its own **row**

# No loops









| Dimensions | Name       | Description                          |
|------------|------------|--------------------------------------|
| 1          | Series     | Indexed 1 dimensional data structure |
| 1          | Timeseries | Series using timestamps as an index  |
| 2          | DataFrame  | A two dimensional table              |

Columns

rows

| Regd. No | Name   | Marks% |
|----------|--------|--------|
| 1000     | Steve  | 86.29  |
| 1001     | Mathew | 91.63  |
| 1002     | Jose   | 72.90  |
| 1003     | Patty  | 69.23  |
| 1004     | Vin    | 88.30  |

The diagram illustrates a table structure. The word 'Columns' is positioned above the table with two arrows pointing to the first two columns. The word 'rows' is positioned to the left of the table with three arrows pointing to the first three rows.



# Libraries in Python

# What is a Python library?

- Python libraries are also called modules
- Python has large set of built-in libraries
- Can contain classes, functions or variables
- Imported into a script or session

```
import ipaddress
```

# Python library: full name

**Functions** defined in a **library** are called using **dot notation** when importing the full library.

`library name + period + function name`

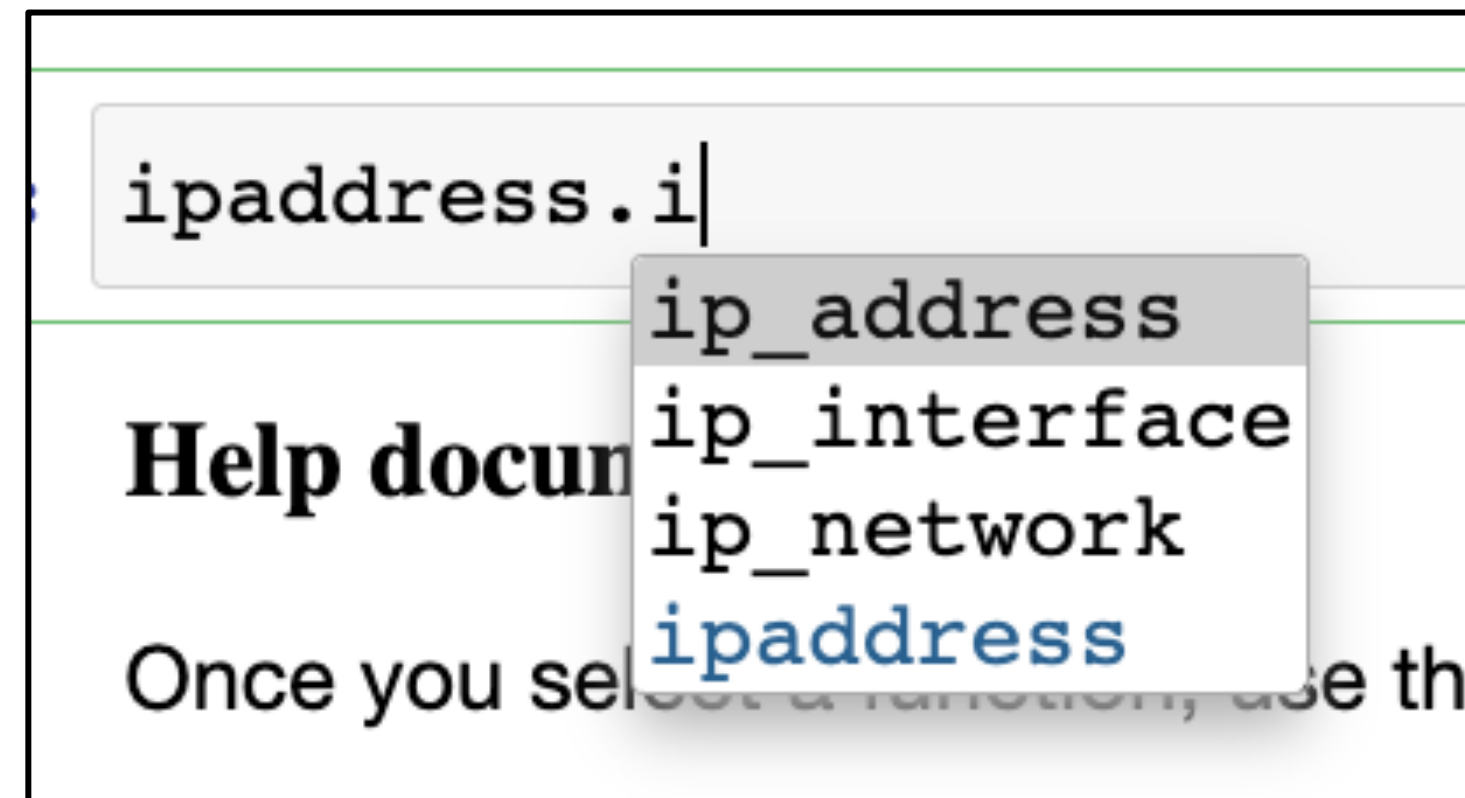
```
import ipaddress  
user_address = ipaddress.ip_address( "0:0:0:0:0:FFFF:185.15.58.226" )  
user_address.is_private
```

False

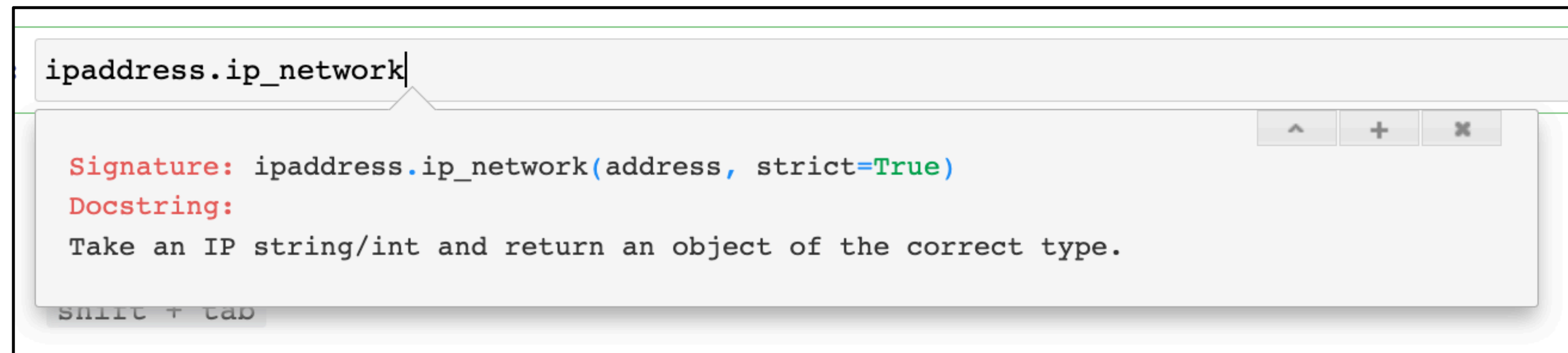
**Note:** try using use tab completion after typing the dot and the letter 'i' to get a tool tip of possible functions that start with 'i' from the ipaddress library.

# Python library: Jupyter shortcuts

Tab completion



Shift + Tab =  
documentation



# Python library: alias

Use an **alias** to avoid typing the full name of the **library** each time a **function** is called.

```
import ipaddress as ip
user_address = ip.ip_address("185.15.58.226")
```

```
import pandas as pd
raw_empty_data_frame = pd.DataFrame()
```



# Python library: specific functions

When importing specific **functions** from a **library**, the function name can be called without the dot notation if we import them specifically using

```
from library import function1, function2
```

```
from ipaddress import ip_address, ip_network, ip_interface
```

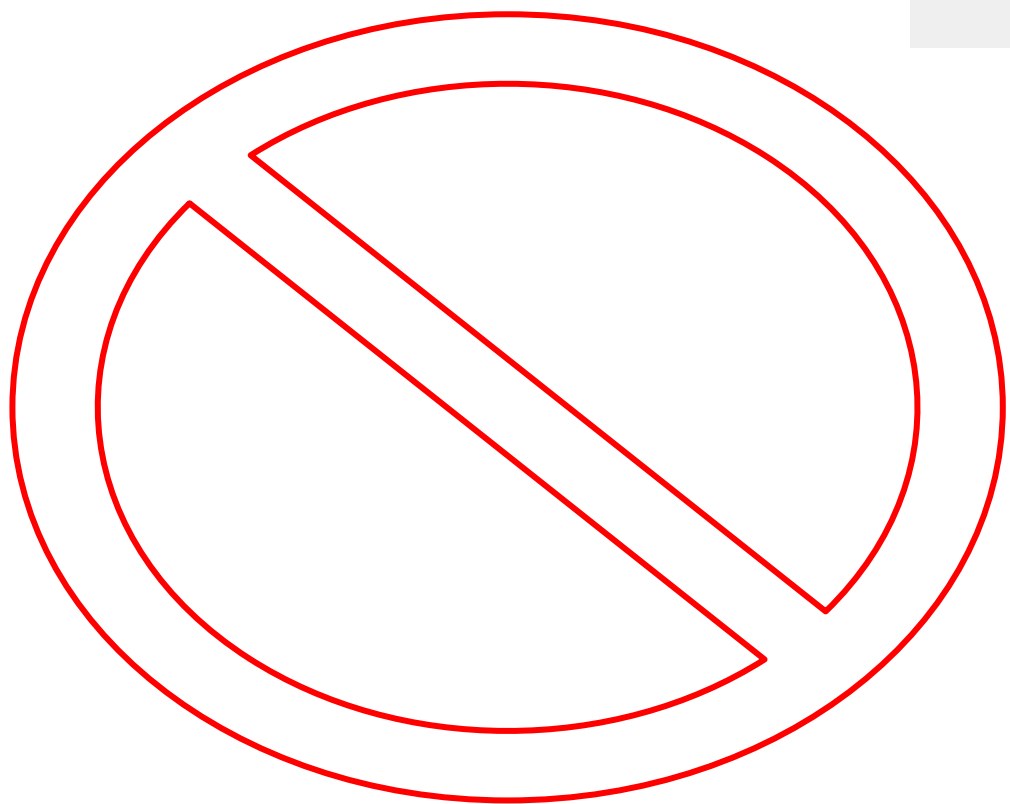
```
ip_interface("0:0:0:0:0:FFFF:185.15.58.226")
```

# Python library: wild card

## DON'T DO THIS!!!

When importing specific **functions** from a **library**, all the functions, variables, etc can be imported using the **wildcard**.

```
from ipaddress import *  
ip_network("192.0.2.0/24")
```





Note: this is bad practice because you don't always know what you have imported and this can cause naming conflicts, make code harder to read, and clutter the namespace with things you won't use. Despite this, you will see it used all the time.

# Third party Python library

- Third party library = written by someone other than Python developers
- Many useful third party libraries for data science tasks
- Not installed with Python
- Can be installed into the Python environment (pip, conda) or directly in a Jupyter Notebook (pip)
- Download from PyPI (Python Package Index) ([pypi.org](https://pypi.org))
- Imported/used the same way the built-in libraries are imported

# Third party Python library

pypi.org/search/?q=security&o=&c=Intended+Audience+%3A%3A+System+Administrators

 security 

Help Sponsors Log in Register






Filter by classifier

1,245 projects for "security" with the selected classifier

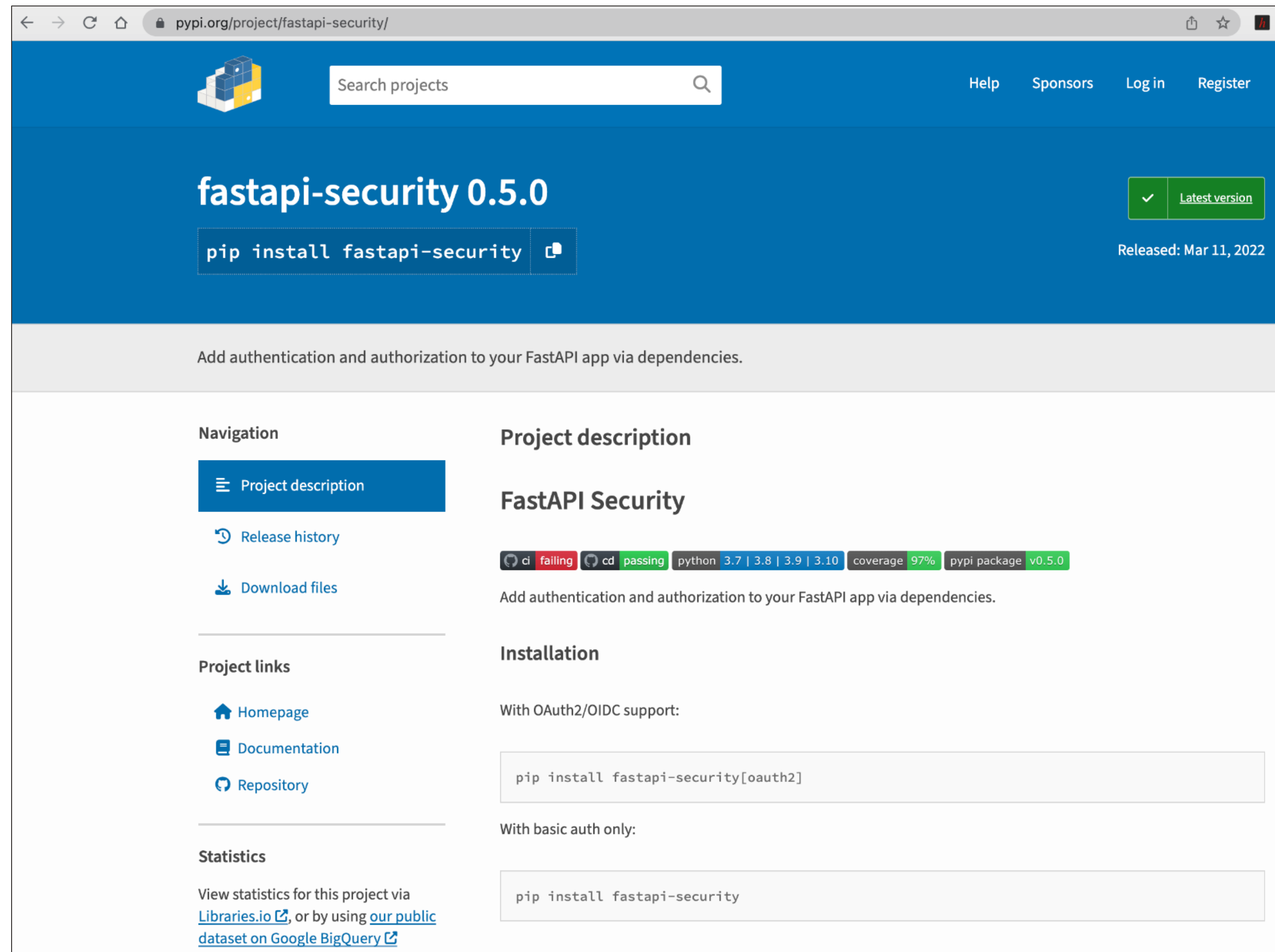
Order by Relevance

INTENDED AUDIENCE :: SYSTEM ADMINISTRATORS

- Framework
- Topic
- Development Status
- License
- Programming Language
- Operating System
- Environment
- Intended Audience
  - ☐ Customer Service
  - ☐ Developers
  - ☐ Education
  - ☐ End Users/Desktop
  - ☐ Financial and Insurance Industry
  - ☐ Healthcare Industry
  - ☐ Information Technology
  - ☐ Legal Industry
  - ☐ Manufacturing

|   |   |              |
|---|---|--------------|
|    | <b>fastapi-security 0.5.0</b><br>Add authentication and authorization to your FastAPI app via dependencies.                     | Mar 11, 2022 |
|  | <b>security_monkey 0.4.0</b><br>Security Monkey monitors policy changes and alerts on insecure configurations in an AWS account | Dec 15, 2015 |
|  | <b>aura-security 2.1</b><br>Security auditing and static analysis for python  | Jun 22, 2021 |
|  | <b>security-cam 0.1</b><br>Attachment for Motion to set up a security system  | Apr 13, 2013 |
|  | <b>aws-security-test 0.1.0</b><br>Test automation to determine adherence to pre-defined set of security recommendations         | Oct 3, 2017  |

# Third party Python library



The screenshot shows the PyPI project page for **fastapi-security 0.5.0**. The page has a blue header with the PyPI logo, a search bar, and links for Help, Sponsors, Log in, and Register. Below the header, the project name and version are displayed, along with a green checkmark and a "Latest version" button. A code block shows the installation command: `pip install fastapi-security`. The release date is listed as "Released: Mar 11, 2022".

A light gray banner below the header states: "Add authentication and authorization to your FastAPI app via dependencies."

The main content area is divided into two columns. The left column contains a "Navigation" section with links for "Project description" (highlighted), "Release history", and "Download files". Below this is a "Project links" section with links for "Homepage", "Documentation", and "Repository". At the bottom of the left column is a "Statistics" section with a link to "View statistics for this project via Libraries.io" and a link to "our public dataset on Google BigQuery".

The right column contains a "Project description" section with the title "FastAPI Security" and a row of status badges: "ci failing", "cd passing", "python 3.7 | 3.8 | 3.9 | 3.10", "coverage 97%", and "pypi package v0.5.0". Below the badges is the same light gray banner: "Add authentication and authorization to your FastAPI app via dependencies." Underneath is an "Installation" section with the text "With OAuth2/OIDC support:" followed by a code block showing `pip install fastapi-security[oauth2]`. Below that is the text "With basic auth only:" followed by a code block showing `pip install fastapi-security`.

# Pandas: The Series Object

A **Series** is like a list or a  
dictionary **but BETTER!!**

**The **Series** is the **primary building block** for  
all the other data structures we will discuss**



# Python libraries: Common for Data Sci

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt
```

# Creating a Series

```
import pandas as pd  
myData = pd.Series( <data> )
```

# Creating a Series

```
import pandas as pd
```

```
series1 = pd.Series( ['a', 'b', 'c', 'd', 'e'] )
```

```
series2 = pd.Series( { 'firstName' : 'Charles',  
                       'lastName'  : 'Givre' } )
```

# Pandas Series

```
series1 = pd.Series( ['a', 'b', 'c', 'd', 'e'] )  
print(series1)
```

```
0    a  
1    b  
2    c  
3    d  
4    e  
dtype: object
```

# Pandas Series: single elements

```
series1 = pd.Series( ['a', 'b', 'c', 'd', 'e'] )  
print(series1[3])
```

d

# Pandas Series: slicing

```
series1 = pd.Series( ['a', 'b', 'c', 'd', 'e'] )  
print(series1[1:3])
```

```
1      b  
2      c  
dtype: object
```

# Accessing a Series

```
series2 = pd.Series( {'firstName': 'Charles',  
                      'lastName' : 'Givre'} )  
  
print(series2[ 'firstName' ])
```

Charles

# Accessing a Series

```
series2 = pd.Series( {'firstName': 'Charles',  
                      'lastName' : 'Givre'} )  
  
print(series2[ ['firstName', 'lastName'] ])
```

```
firstName    Charles  
lastName     Givre  
dtype: object
```



# Accessing a Series

```
series2 = pd.Series( {'firstName': 'Charles',  
                      'lastName'  : 'Givre'} )  
  
print(series2[0])
```

Charles

# Accessing a Series

```
randomNumbers = pd.Series(  
    np.random.random_integers(1, 100, 50))
```

```
randomNumbers.head()
```

|   |    |
|---|----|
| 0 | 48 |
| 1 | 34 |
| 2 | 84 |
| 3 | 85 |
| 4 | 58 |

**dtype: int64**

# Accessing a Series

```
randomNumbers = pd.Series(  
    np.random.random_integers(1, 100, 50) )  
randomNumbers.tail( 7 )
```

```
43      66  
44      66  
45      43  
46      55  
47      99  
48      82  
49      19  
dtype: int64
```

# Pause

# Filtering Data in a Series

```
randomNumbers [ <boolean condition> ]  
randomNumbers [ randomNumbers < 10 ]
```

```
12      5  
21      2  
24      1  
27      1  
29      1  
dtype: int64
```

# Filtering Data in a Series

```
record.str.contains( 'Cha' )
```

```
firstname      True
```

```
lastname       False
```

```
middle         False
```

```
dtype: bool
```

```
record[ record.str.contains( 'Cha' ) ]
```

```
firstname      Charles
```

```
dtype: object
```

# String Functions

| Function                             | Explanation  |
|--------------------------------------|--|
| Series.str.contains(<pattern>)       | Returns true/false if text matches a pattern                           |
| Series.str.count(<pattern>)          | Returns number of occurrences of a pattern in a string                 |
| Series.str.extract(<pattern>)        | Returns matching groups from a string                                  |
| Series.str.find(<string>)            | Returns index of first occurrences of a substring<br>(Note: not regex) |
| Series.str.findall(<pattern>)        | Returns all occurrences of a regex                                     |
| Series.str.len()                     | Returns the length of text   |
| Series.str.replace(<pat>, <replace>) | Replaces matches with a replacement string                             |

# Date/Time Operations

```
pd.to_datetime(<times>, format='<format string>')
```



# Selected Date/Time Operations

| Function                              | Explanation   |
|---------------------------------------|---|
| <code>series.dt.year</code>           | Returns the year of the date time   |
| <code>series.dt.month</code>          | Returns the month of the date time  |
| <code>series.dt.day</code>            | Returns the days of the date time   |
| <code>series.dt.weekday</code>        | Returns the day of the week   |
| <code>series.dt.is_month_start</code> | Returns true if it is the first day of the month                              |
| <code>series.dt.strftime( )</code>    | Returns an array of formatted strings specified by a <code>date_format</code> |

```
combinedSeries = series1.add( series2 )
```

```
def addTwo(n):  
    return n+2  
  
odds.apply(addTwo)
```

```
odds.apply(lambda x: x + 1)
```

# Questions?

# **In Class Exercise**

## **Worksheet 1.1: Working with One Dimensional Data**