DS3500

Advanced Programming with Data

John Rachlin
Northeastern University

Principles of Chess

The Top 35 Chess Principles

Control the Center	Doubled rooks on an open file are very strong
Develop pieces quickly	Bishops are better in open positions, knights are better in closed positions
Knights before bishops	The best way to deal with a flank attack, is with a counter attack in the center
Don't move the same piece twice in the opening	When 2 pawns can capture the same piece, capture towards the center
Don't bring your queen out too early	The king should be activated in the endgame
Castle before move 10	Rooks go behind passed pawns
Connect your rooks	2 connected passed pawns on the 6th rank will beat a rook
Rooks should go to open or half open files	Attack the base of a pawn chain
Knights on the rim are grim	Knights are usually the best piece to use to blockade a pawn
Try to avoid doubled pawns	If your position is cramped, trading pieces can help
Try to avoid isolated pawns	Trade your passive pieces for your opponent's active pieces
Try to avoid backward pawns	When ahead material, trade pieces, not pawns
Don't trade a bishop for a knight without a good reason	When behind material, trade pawns, not pieces
Avoid moving pawns in front of your castled king	Games with opposite colored bishops are dangerous in the middlegame, and drawish in the endgame
Don't open the center if your king is still there	Don't play "hope" chess
2 minor pieces are usually better than a rook and a pawn	When you see a good move, look for a better move
3 minor pieces are usually better than a queen	A really good chess player knows the right time to ignore a chess principle
Rooks are good on the 7th rank	

DS 2000: Programming with Data

What is DS 2000 about?

DS2000: Programming with Data is a first-course introduction to computer programming using the Python programming language. The course will introduce using various Python toolkits for manipulating and analyzing data in a broad range of applications including science, finance, and healthcare. The course assumes no prior programming experience.

Who should take DS 2000?

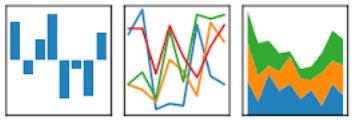
DS 2000 is aimed at students majoring in subjects outside of Computer Science or Data Science who wish to use computer programming as an essential *tool* for working with data in their respective fields. The course teaches students how to work with data algorithmically. *DS 2000 is the one computer programming course to take if you are only going to take one!*

Take DS 2000 if:	Alternatively, take CS 2500 if:
You are a DS major aiming to become a professional data scientists or you are a non-CS / non-DS major looking to gain some practical experience with computer programming as a tool for solving problems in various technical fields. Students with some prior programming skills should consider DS 2500 instead.	You are a CS major aiming to become a professional computer programmer or computer scientist. CS minors interested in the foundations of Computer Science and hoping to eventually tackle larger-scale software projects will also find CS 2500 to be a better fit.

python™

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Northeastern University
Khoury College of Computer Sciences

DS 2500: Intermediate Programming with Data

What is DS 2500 about?

DS2500: Intermediate Programming with Data is a follow-on course to *DS 2000: Programming with Data*. It provides students with the opportunity to advance their computer programming skills using the Python programming language. The course will explore more advanced programming methodologies including object-oriented design and test-driven development – techniques that are useful for tackling larger-scale programming projects.

Who should take DS 2500?

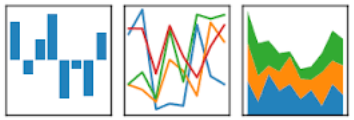
DS 2500 is aimed at DS majors, DS minors, and non-CS / non-DS majors who have taken DS 2000 or have equivalent programming experience. *DS 2500 is the course to take if you are interested in taking your programming skills to the next level and your focus is practical problem solving, data analysis, and discovery in areas such as business, the sciences, and healthcare.*

Take DS 2500 if:	Alternatively, take CS 2510 if:
You are a DS major or DS minor, or you are a non-CS / non-DS major looking to gain greater practical experience with computer programming and data-centric problem solving. DS 2500 requires DS 2000 or equivalent programming experience.	You are a CS major or a DS major interested in the fundamentals of computer science, data structures, and algorithms. CS minors interested in the foundations of Computer Science will also find CS 2500 to be a better fit.

python™

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Northeastern University
Khoury College of Computer Sciences

DS 3500: Advanced Programming with Data

What is DS 3500 about?

DS3500: Advanced Programming with Data is a follow-on course to *DS 2500: Programming with Data*. It presents a deep-dive into the design and implementation of enterprise-grade software systems with an emphasis on software architectures for more complex data-driven applications. Topics will include tools and techniques for constructing service-oriented, scalable systems for data analytics, and event-based data processing.

Who should take DS 3500?

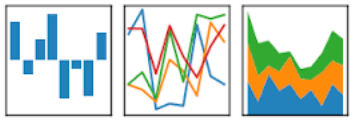
DS 3500 is aimed at DS majors, DS minors, and non-CS / non-DS majors who have taken DS 2500 or have equivalent programming experience. *DS 3500 is the course to take if you want to become rock solid in programming for data science.*

Take DS 3500 if:	Alternatively, take CS 3500 if:
You are a DS major or DS minor, or you are a non-CS / non-DS major looking to achieve a solid foundation in computer programming and data-centric problem solving. DS 3500 requires DS 2500 or equivalent programming experience.	You are a CS major or a DS major interested in the fundamentals of computer science, data structures, and algorithms. CS minors interested in the foundations of Computer Science will also find CS 3500 to be a better fit.

python™

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Northeastern University
Khoury College of Computer Sciences

The *Programming with Data* Track

- Northeastern's Data Science curriculum includes three courses in Python programming:
  - DS 2000: Programming with data
  - DS 2500: Intermediate programming with data
  - DS 3500: Advanced programming with data

- All three courses leverage the python software development ecosystem.

- The three-course programming track follows a logical progression from learning first principles of software development in DS2000 (with no programming experience required) to building object-oriented libraries and data applications in DS2500, to developing more complex *enterprise-grade systems* for data engineering and data science in DS3500.

- The DS Programming with Data track provides an alternative to the traditional CS-oriented *fundamentals* sequence (Fundamentals 1 and 2 and Object-Oriented Design) with special emphasis on data-centric problem solving, analytics, and engineering.

python™ pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$ matplotlib

Northeastern University
Khoury College of Computer Sciences

DS3500 Topics

Tooling

- Managing environments
- Anaconda library management
- The *git* software control system
- The JetBrains PyCharm IDE

Language Features

- Exception handling
- Object-oriented design patterns
- Test-Driven Development
- Functional programming

Modules

- Advanced visualization with plot.ly
- Modeling and simulation
- Web-development / User-interfaces with *flask*
- Building simple dashboards
- Working with relational and/or non-relational databases
- Optimization and evolutionary computing
- Working with streaming data
- Time-series analysis
- Multiple regression
- Data structures: Property graphs for Network Analysis
- Machine Learning: decision trees
- Machine Learning: deep learning
- Animation
- Image processing
- Concurrency with *asyncio*
- Other?

DS3500 –Programming with Data

Setting up your Python development environment

Three critical elements

- **Anaconda**: A python distribution and environment manager (or "package manager") that comes pre-installed with a *base* environment containing hundreds of packages. (Mine currently has 278!)

- An **IDE** (Integrated Development Environment): For creating projects and writing, running, and testing code. We'll be mostly using **Jetbrains PyCharm** in class but I might sometimes revert to spyder or Jupyter Notebooks. PyCharm readily integrates with Anaconda environments.

- **GIT**: A software control system – for working together on shared development projects. PyCharm integrates with GIT.

Python Distributions

A python *distribution* typically consists of:

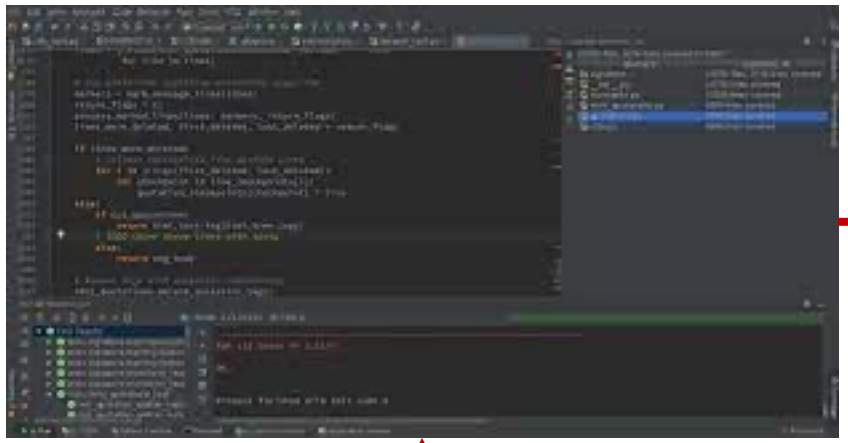
- The python programming language
- Development tools
- Pre-installed python packages (libraries) that make python more powerful
- A package manager (allows you to customize your distribution by adding more libraries or updating existing libraries)

The three pillars of your development environment

ANACONDA

environment
(available packages)

PyCharm



add / commit / push

GIT

clone / pull

» PythonDistributions

FrontPage
RecentChanges
FindPage
HelpContents
PythonDistributions

Page
» Immutable Page
» Info
» Attachments
» More Actions:

User
» Login

Python Distributions

Aside from the official CPython distribution available from python.org, other distributions based on CPython include the following:

» ActivePython from ActiveState
» Anaconda from Continuum Analytics
» ChinesePython Project: Translation of Python's keywords, internal types and classes into Chinese. Eventually allows a progra
» Enthought's Canopy
» Win9xPython: Backport of mainline CPython 2.6/2.7 to old versions of Windows 9x/NT.
» IPython and its IPyKit variant
» PocketPython
» Portable Python: Run Python from USB device - no installation needed.
» PyIMSL Studio
» PyPy: a Python implementation in Python.
» Python(x,y): Python(x,y) is a scientific-oriented Python Distribution based on Qt, Eclipse and Spyder
» PythonForArmLinux
» PythonLabsPython: an old name for the python.org distribution
» PythonwarePython
» StacklessPython
» Tiny Python (archived link) - not to be confused with tinypy
» WinPython: Another scientific-focused Python distribution, based around Spyder

www.anaconda.com/products/individual

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform

Download

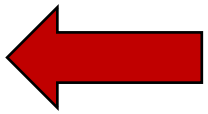
The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews

Anaconda installers

Windows	MacOS	Linux
Python 3.7	Python 3.7	Python 3.7
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (442 MB)	64-Bit (x86) Installer (522 MB)
32-Bit Graphical Installer (423 MB)	64-Bit Command Line Installer (430 MB)	64-Bit (Power8 and Power9) Installer (276 MB)
Python 2.7	Python 2.7	
64-Bit Graphical Installer (413 MB)	64-Bit Graphical Installer (637 MB)	Python 2.7
32-Bit Graphical Installer (356 MB)	64-Bit Command Line Installer (409 MB)	64-Bit (x86) Installer (477 MB)
		64-Bit (Power8 and Power9) Installer (295 MB)

Make sure you get Python 3.7
You will not need Python 2.7



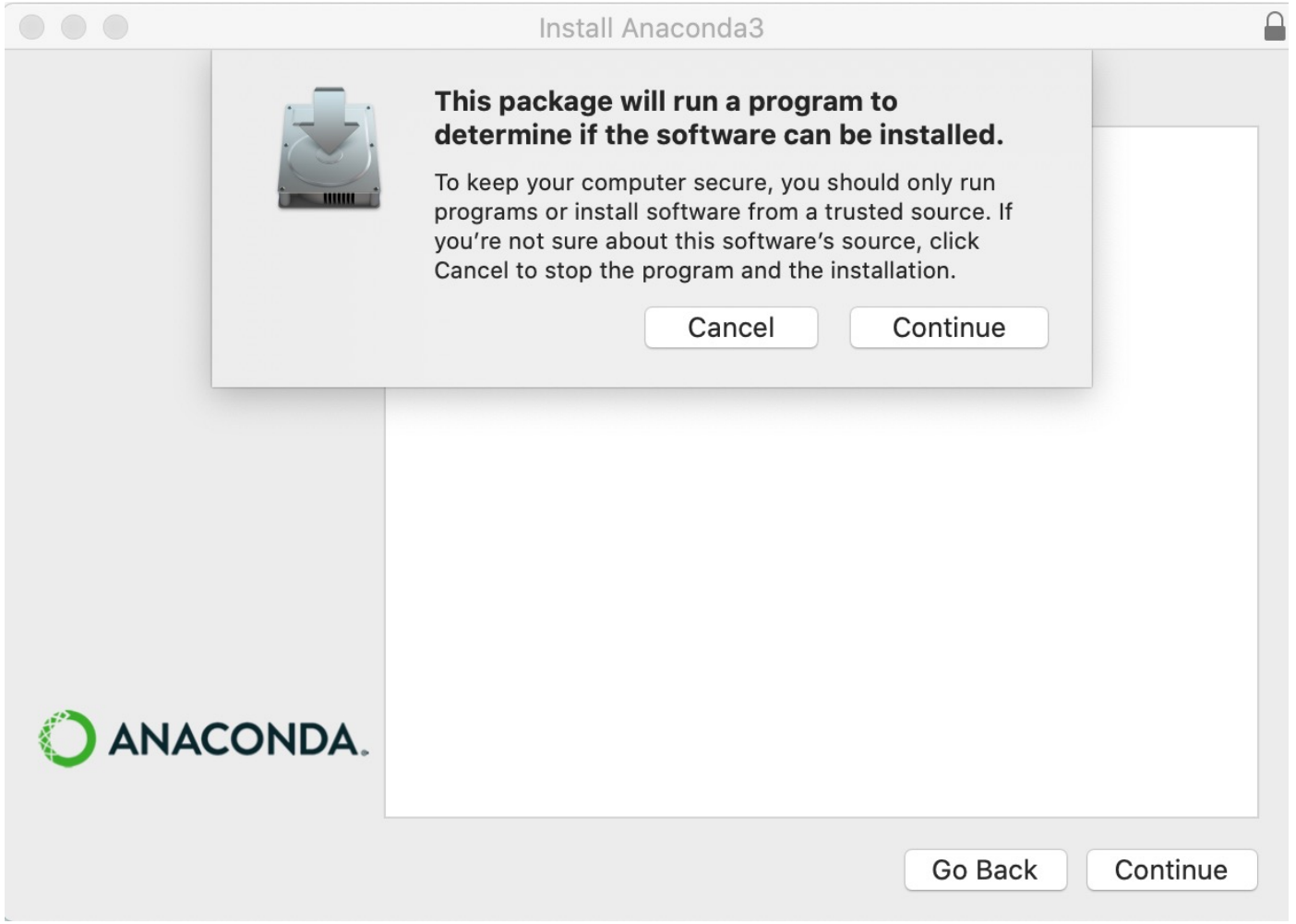
Installing Anaconda on a Mac

Double-click on the downloaded .pkg file.

Mine is called:

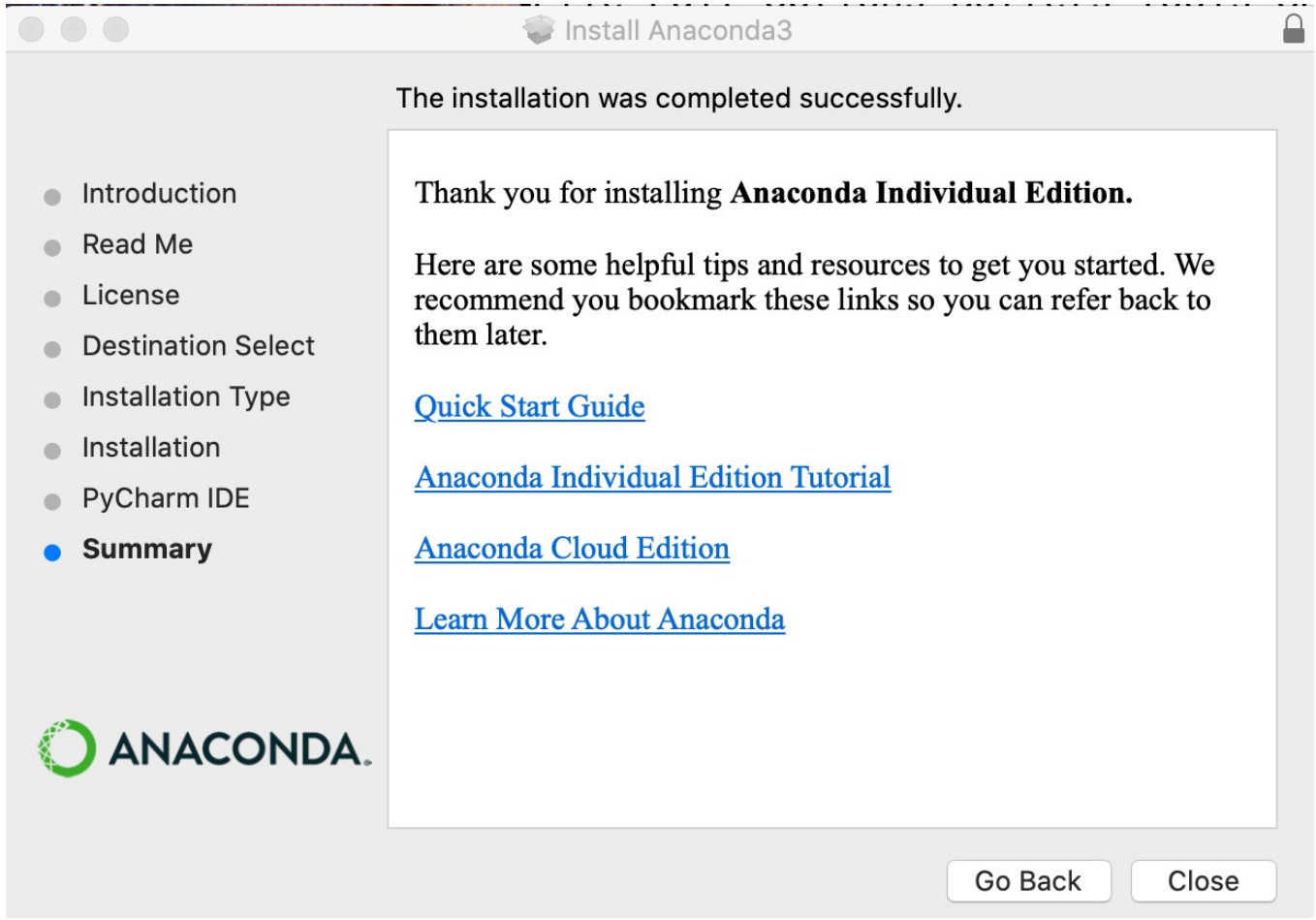
Anaconda3-2020.02-MacOSX-x86_64.pkg

Optional: specify where you want Anaconda to be installed. I installed mine in **/Users/rachlin/apps**

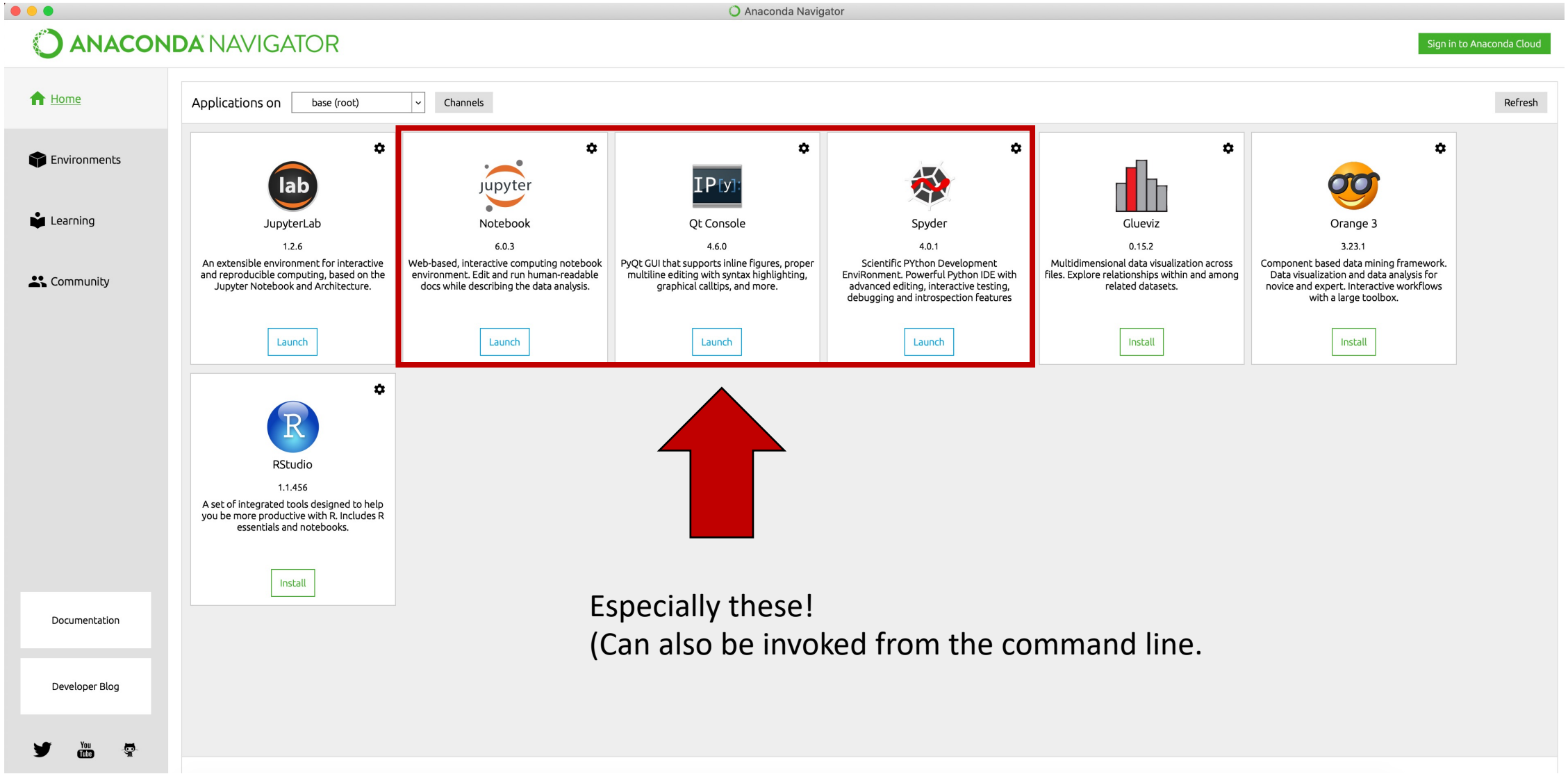


Installing Anaconda on a Mac

You can basically accept the defaults and keep hitting the “continue” button until you get….

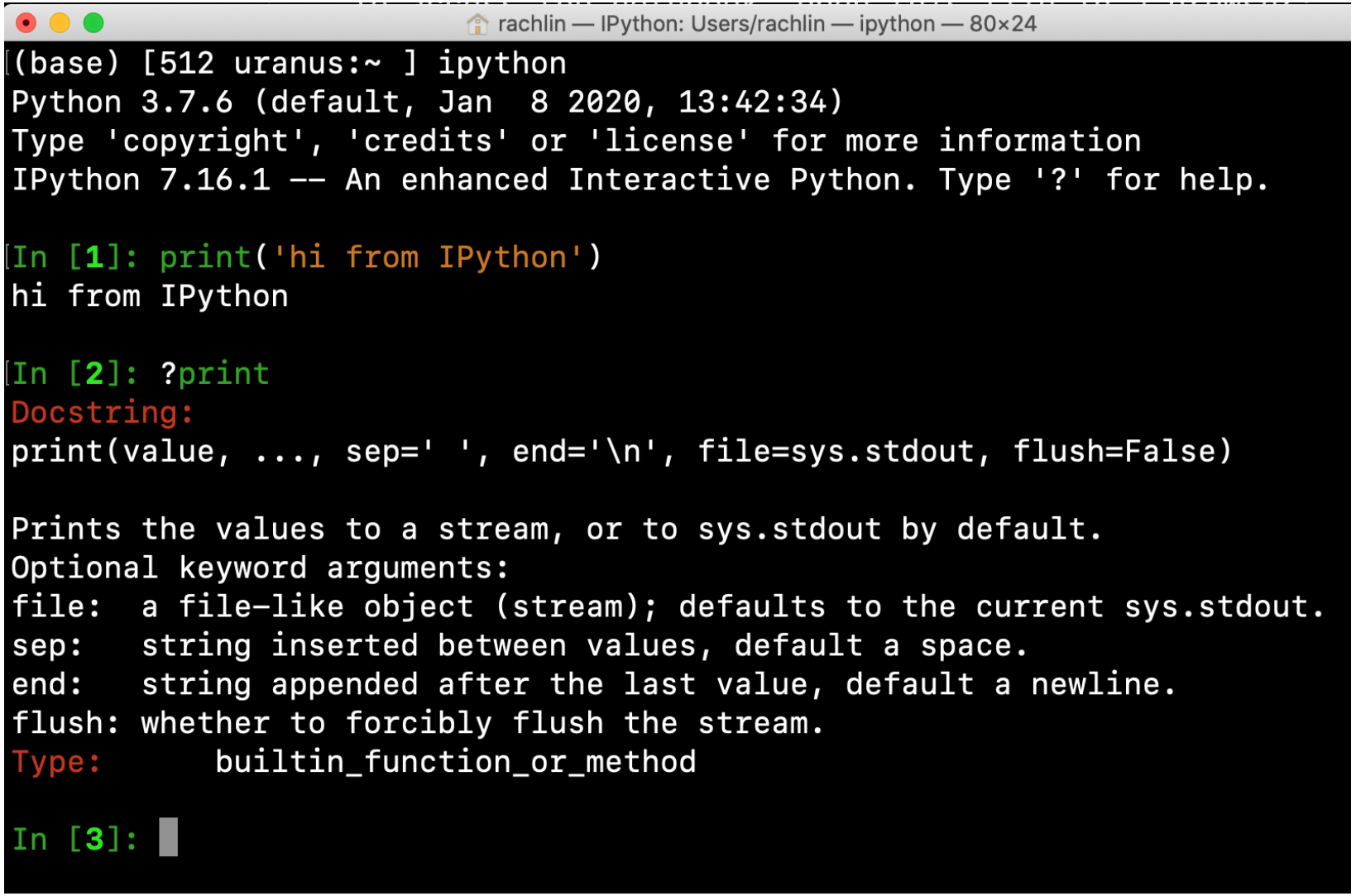


Anaconda Navigator: Access to Applications



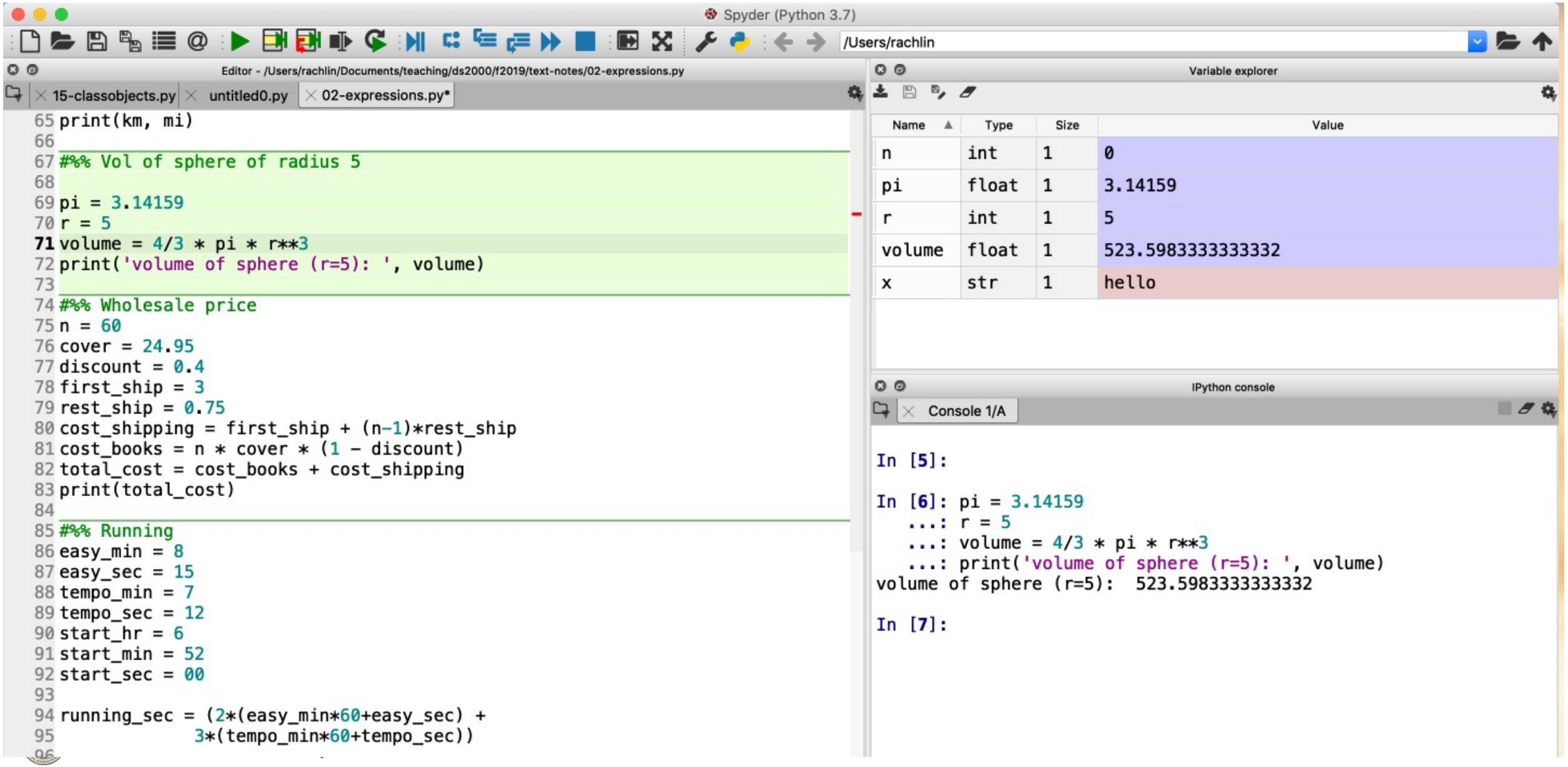
Especially these!
(Can also be invoked from the command line.

IPython: An enhanced REPL for Python

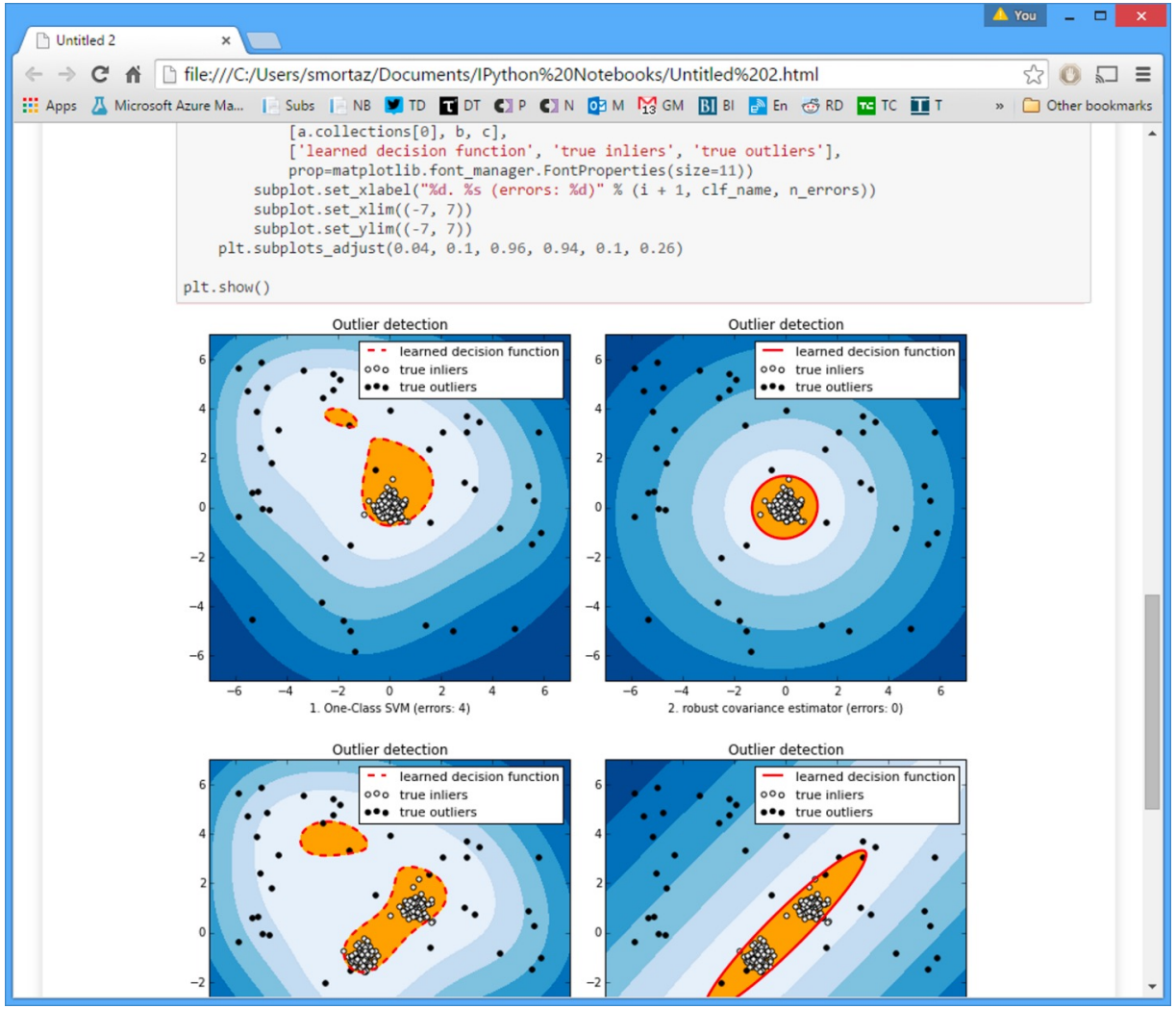


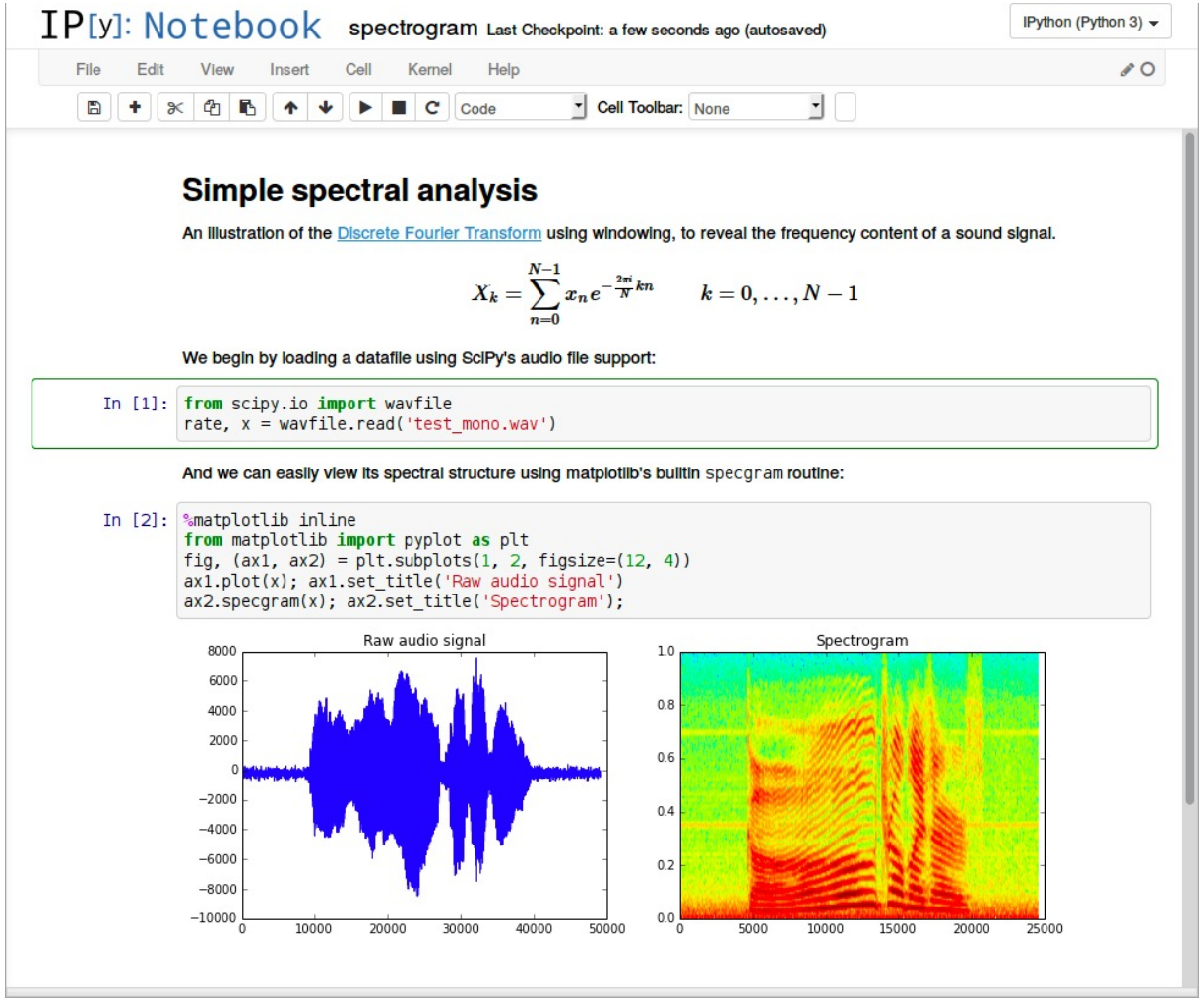
- Syntax highlighting
- Code completion
- Documentation

Spyder: A full-fledged Integrated Development Environment (IDE)

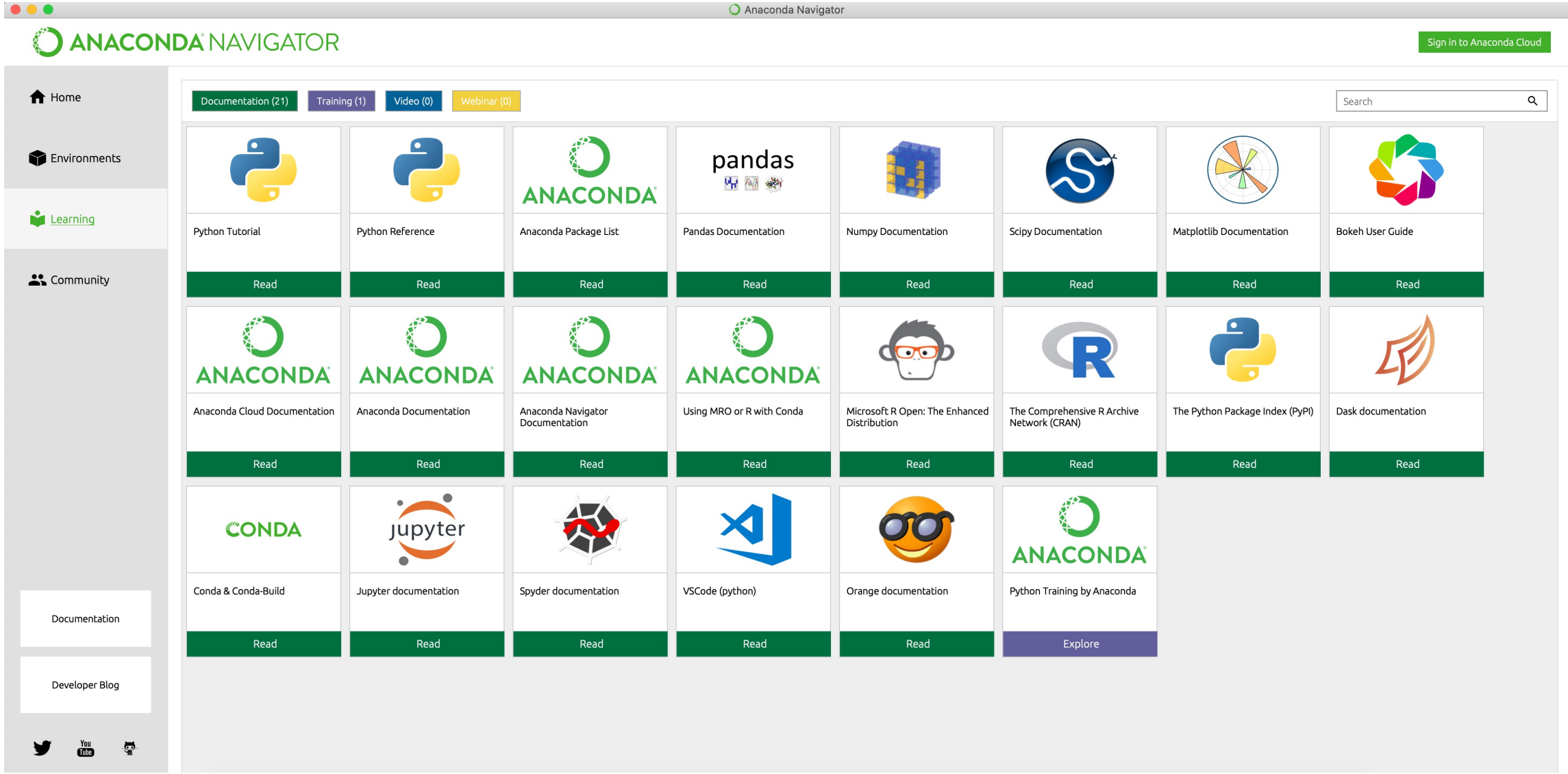


Jupyter Notebooks: Interactive Python for Data Science

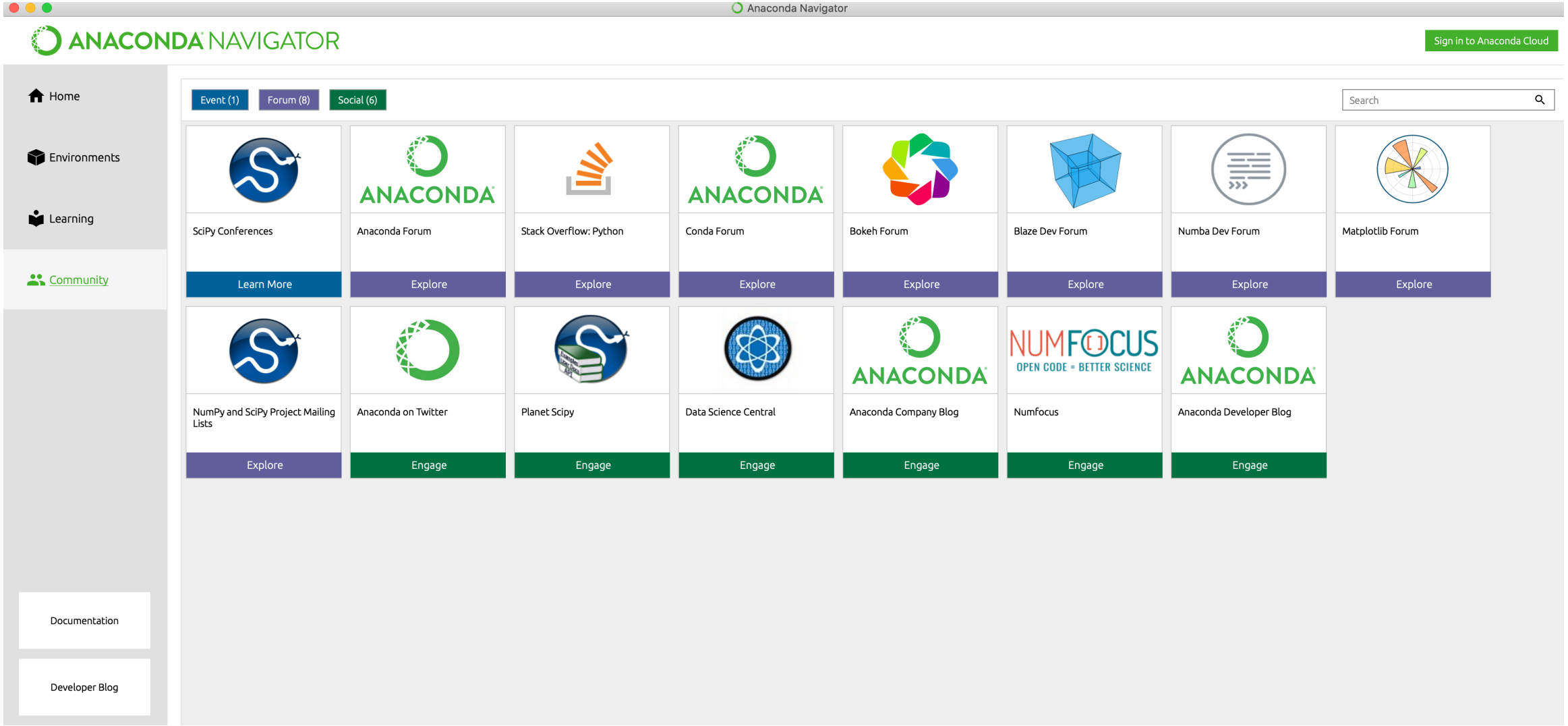




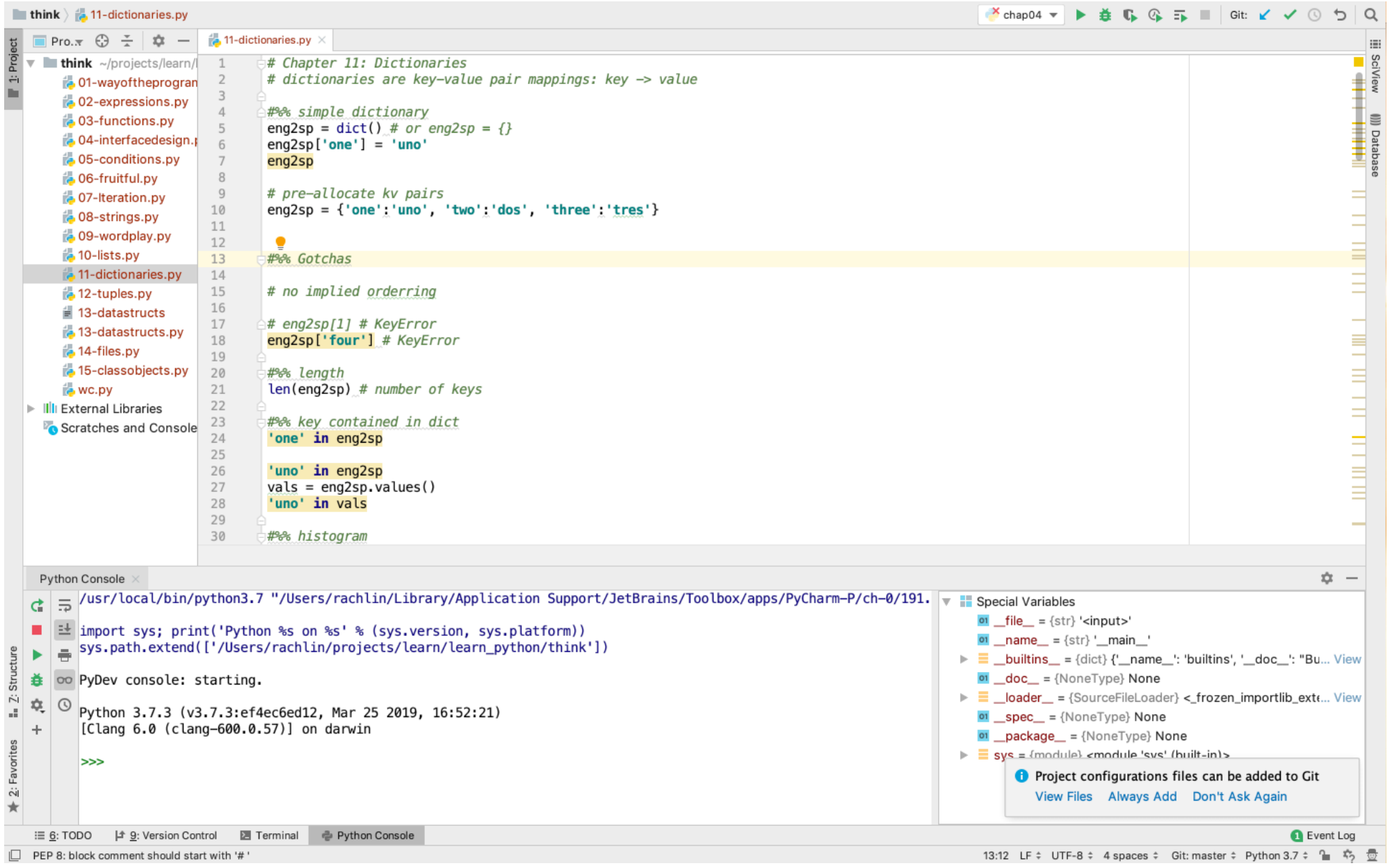
Anaconda Navigator: Access to Learning resources



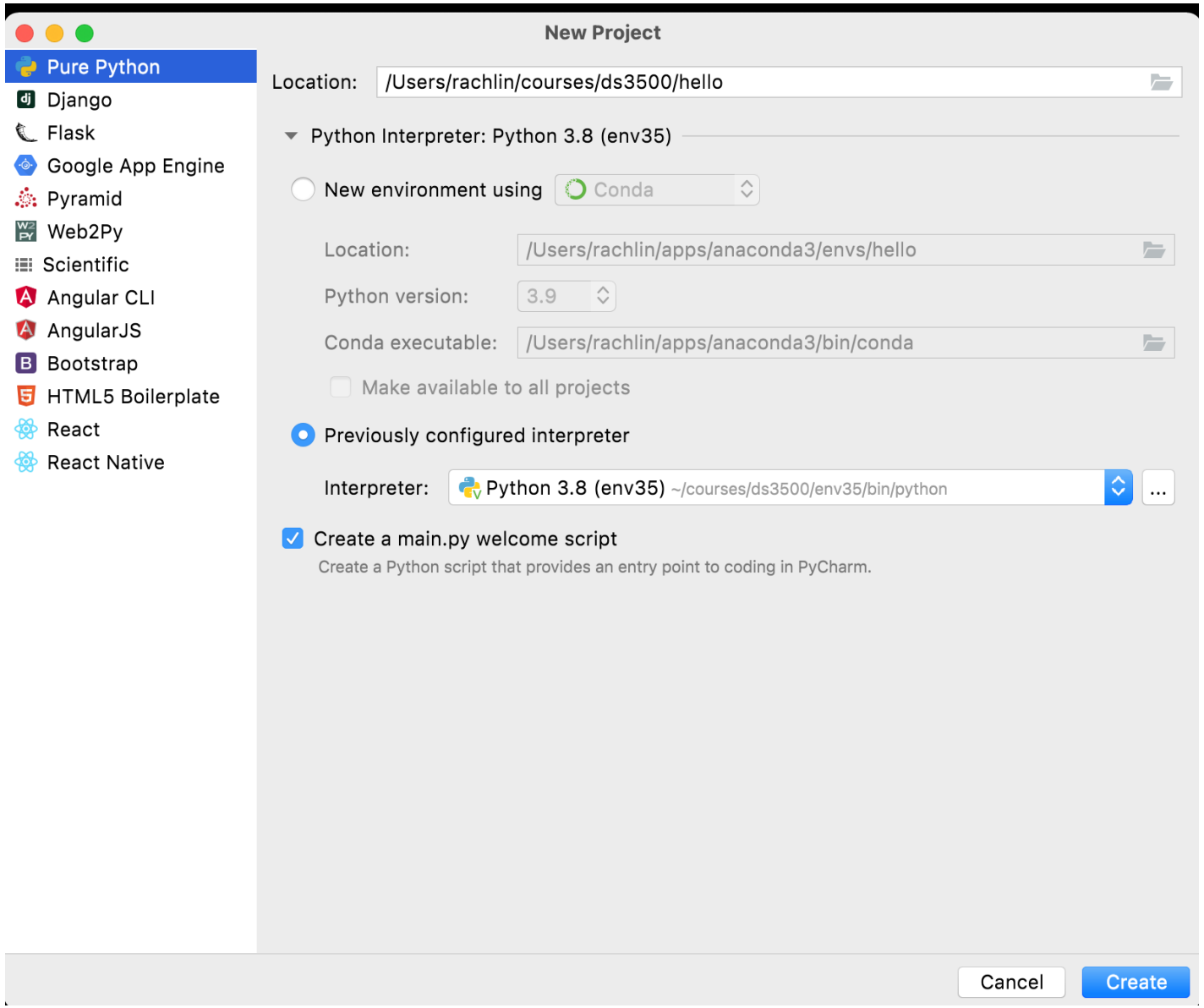
Anaconda Navigator: Access to Community Forums



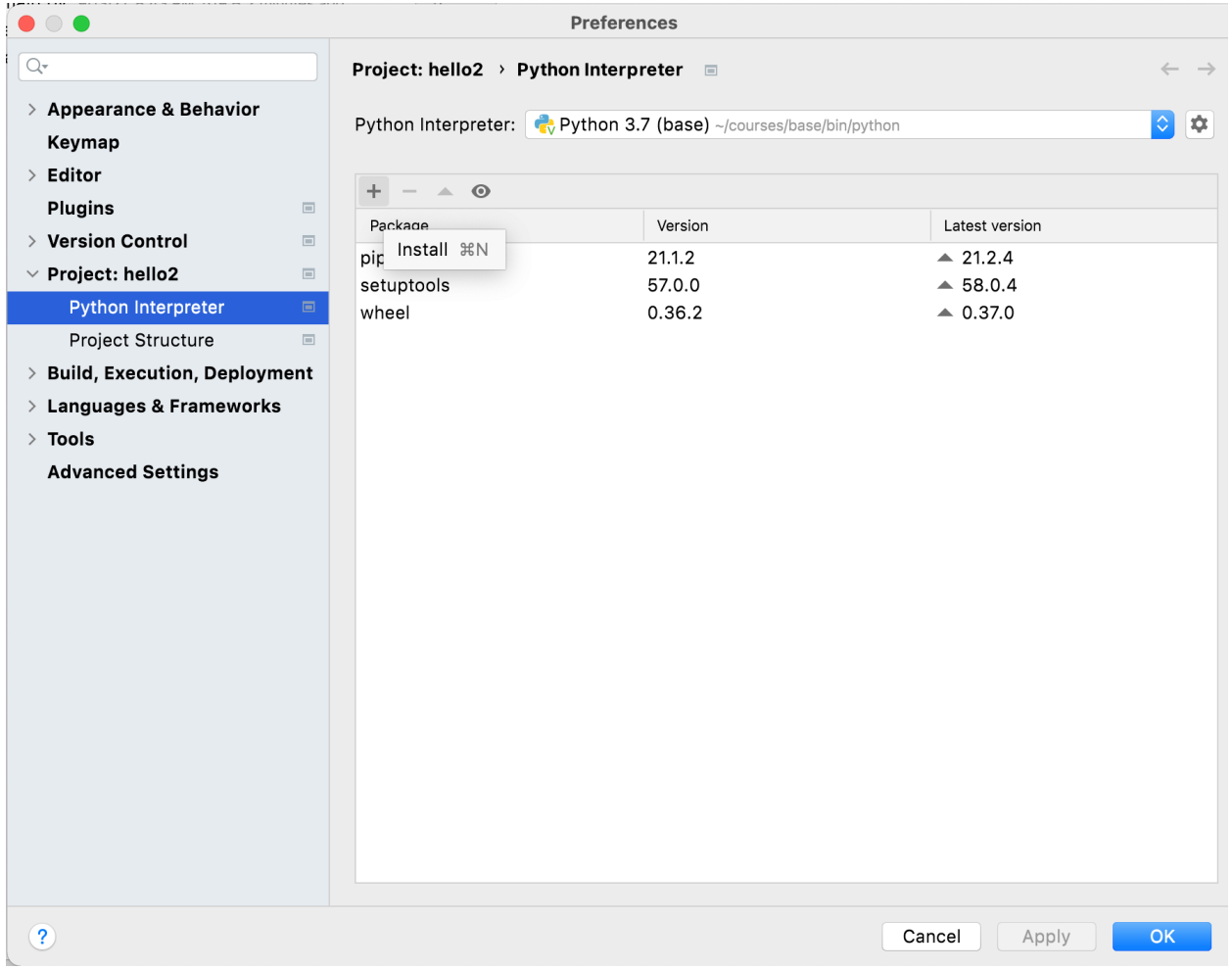
PyCharm (JetBrains): Another popular Python IDE

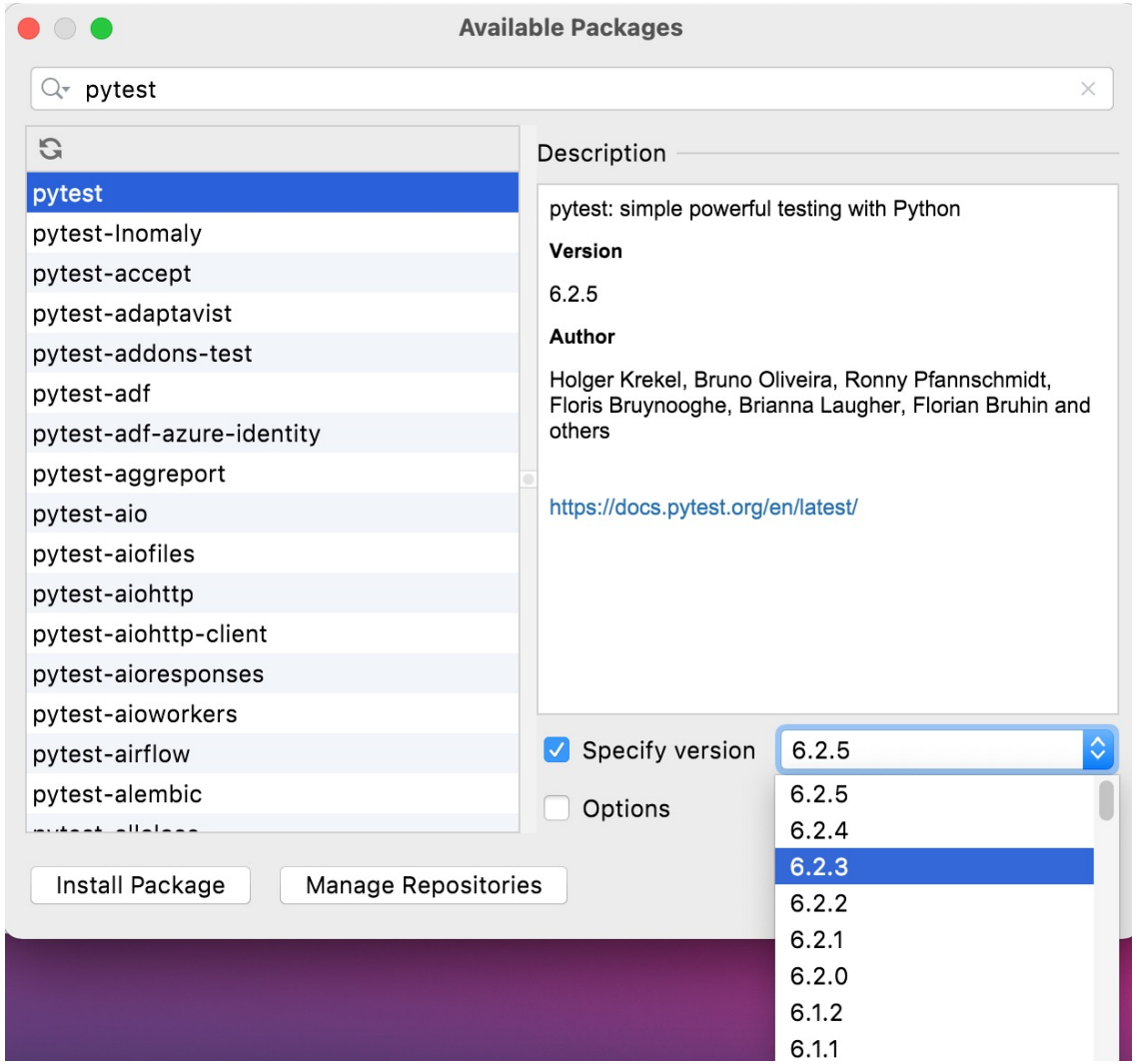


Connecting PyCharm project to an environment

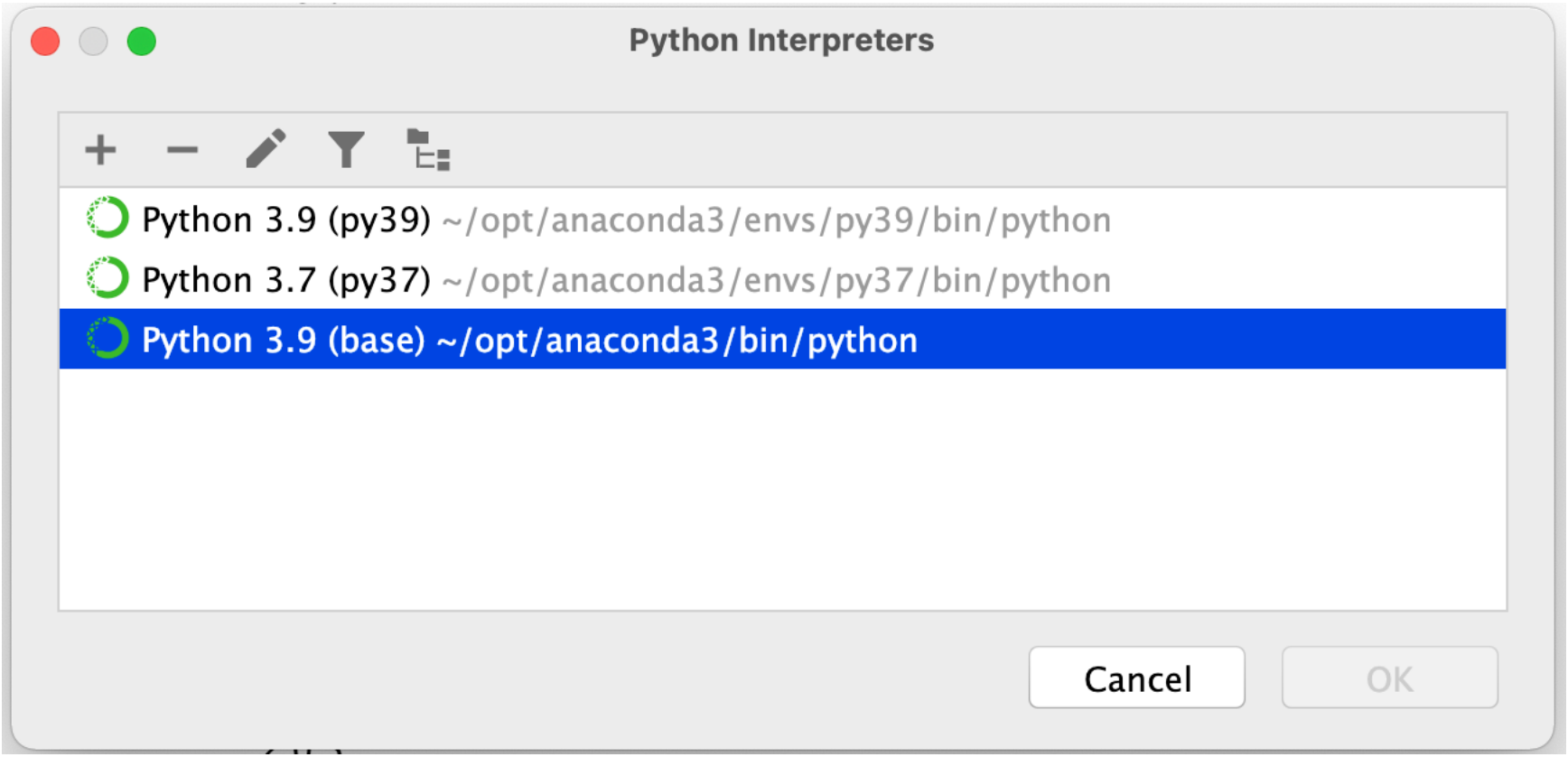


Or we could create a brand-new environment

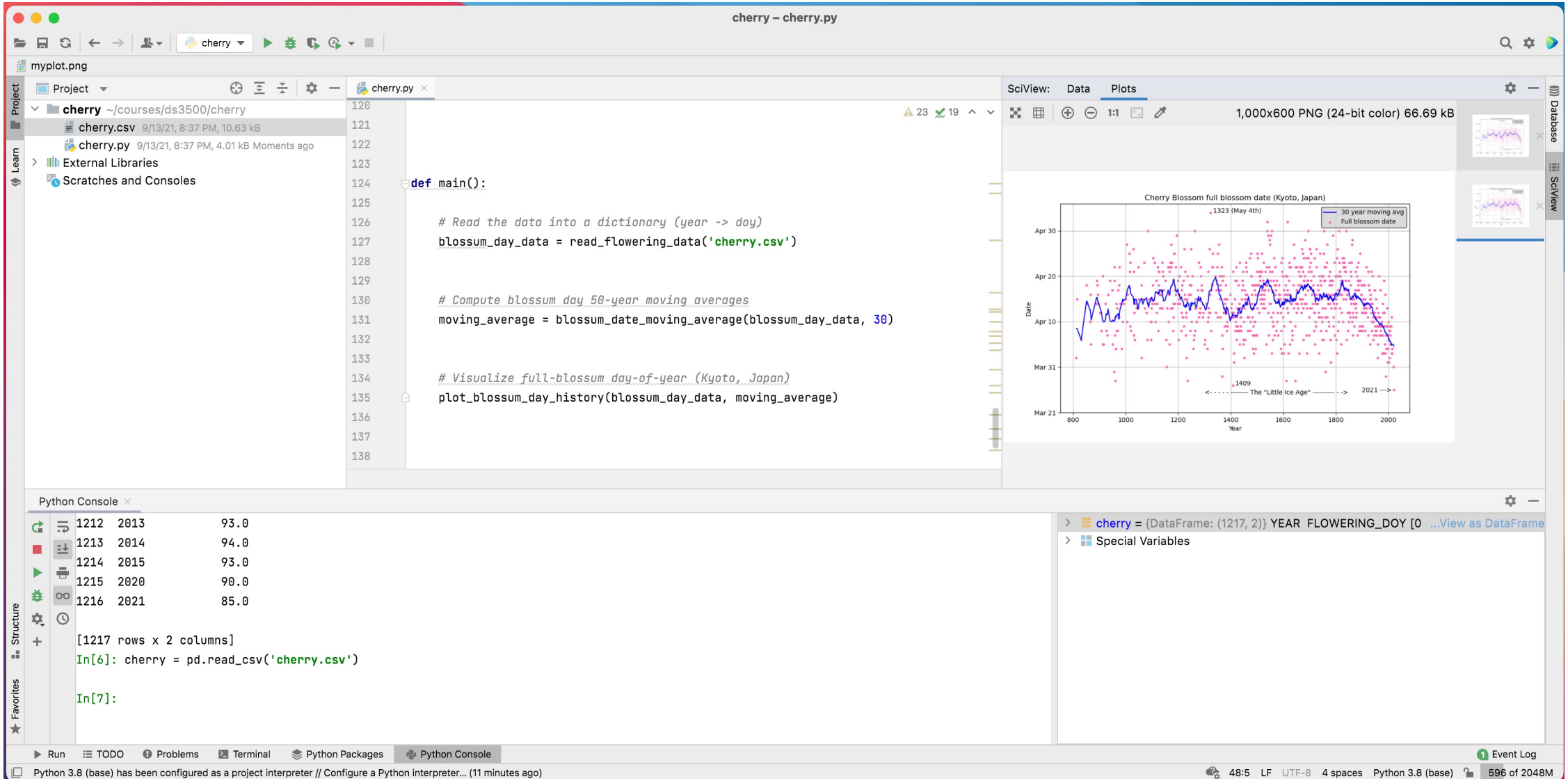




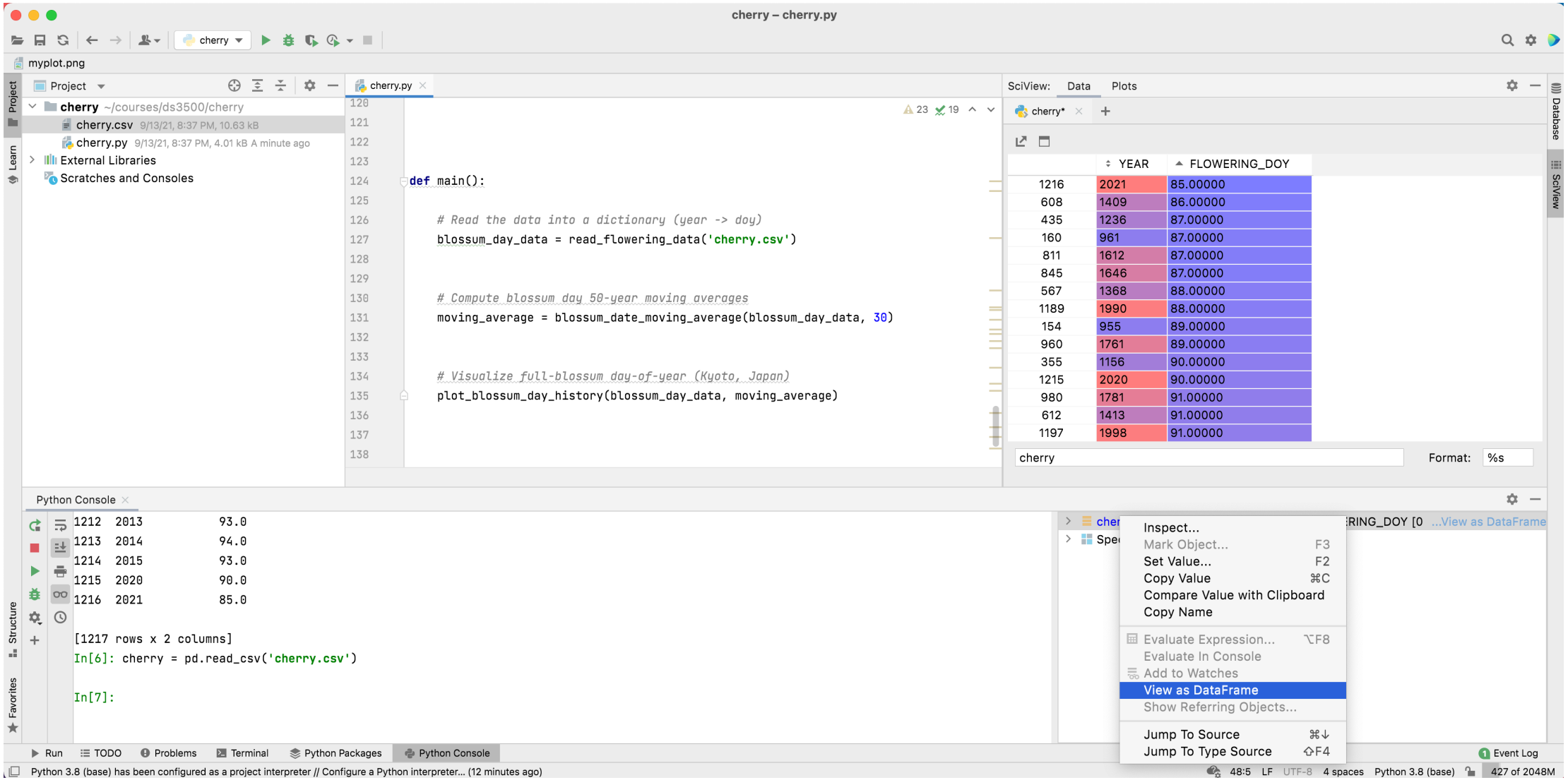
Or we can select a conda environment



PyCharm Plots



Viewing data as a DataFrame



Python Development Tools - Summary

Tool / Platform	Strengths and Weaknesses
REPL	Great for testing small snippets of Python and for learning Python syntax. Not intended for application development. IPython is an enhanced REPL for Python that comes packaged with the Anaconda distribution.
IDLE	Start coding today! Comes with Python distribution. Fine for small, single-file, stand-alone scripts or applications. Not recommended for more complex projects.
Atom text editor	An extendible text-editor with many plugins for Python programming. See Prof. Rachlin's handout on recommended plugins.
Spyder	A full-fledged IDE installed with the Anaconda distributions. Manage multiple files. Monitor variables. Run isolated blocks of code.
PyCharm (JetBrains)	Another popular IDE for python. Has more functionality than what you will need for this class. I won't be making use of PyCharm in this class, but you are welcome to explore its capabilities and use it for your homework.
Jupyter notebooks	A *notebook-style* development environment popular with data scientists. Integrate code, text, tables, visualizations in one file. Not suited for software *applications* requiring multiple python files.
Anaconda	A Python distribution that comes with Python, popular pre-installed libraries, and a variety of development tools including Spyder, Ipython, and Jupyter notebooks. The Anaconda Navigator is a jumping-off point for applications, learning resources, and community forums. **Recommended.**

Git commits

H1 **Git + GitHub lecture**

H2 **Git**

H3 **Basics**

1. `git add --all`

2. `gti commit -m "Fix ..."`

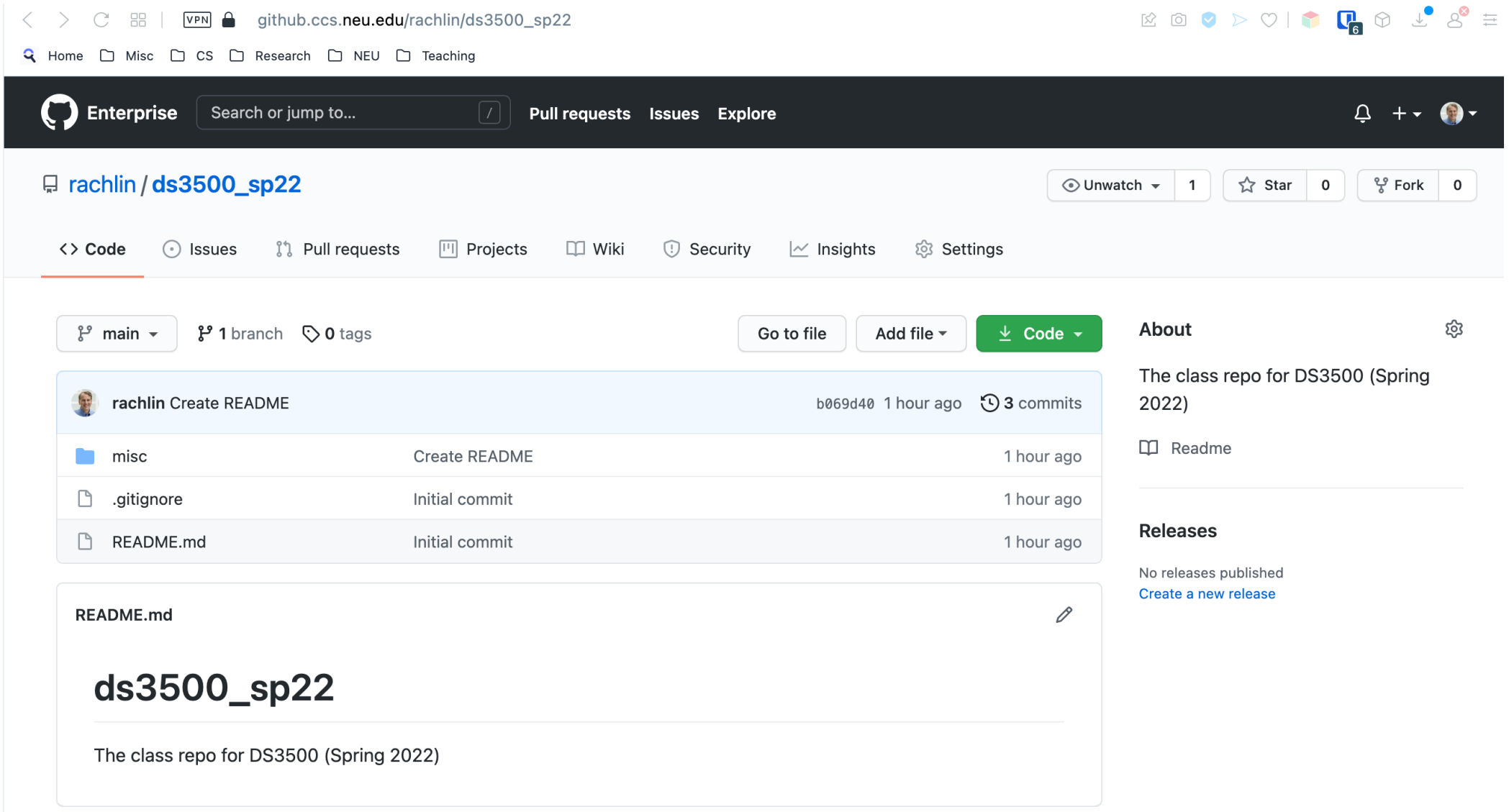
[How to Write a Git Commit Message](🔗)

If applied, this commit will *~*your subject line here*~*

3. `git pull`

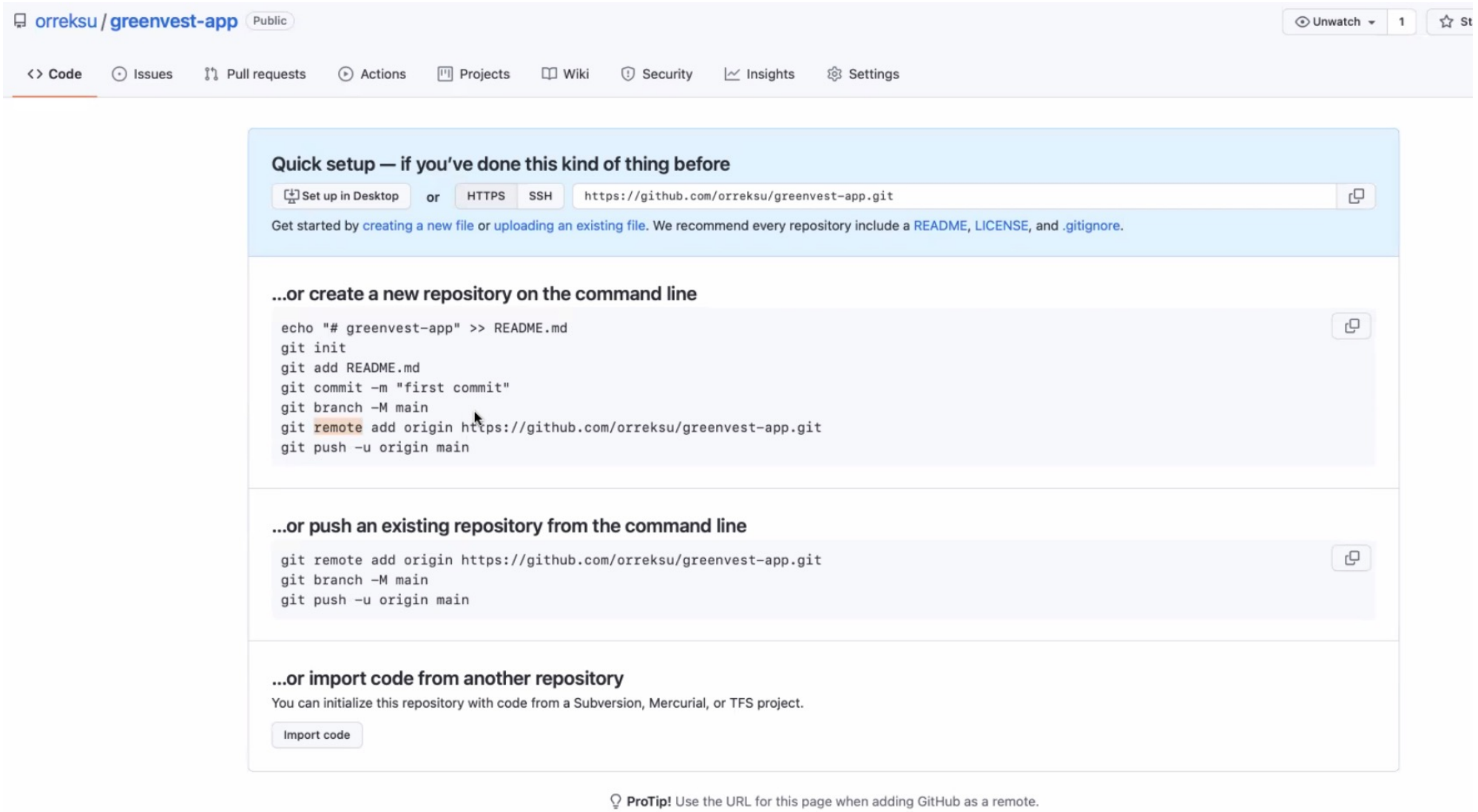
4. `git push`

Create git repo: https://github.ccs.neu.edu

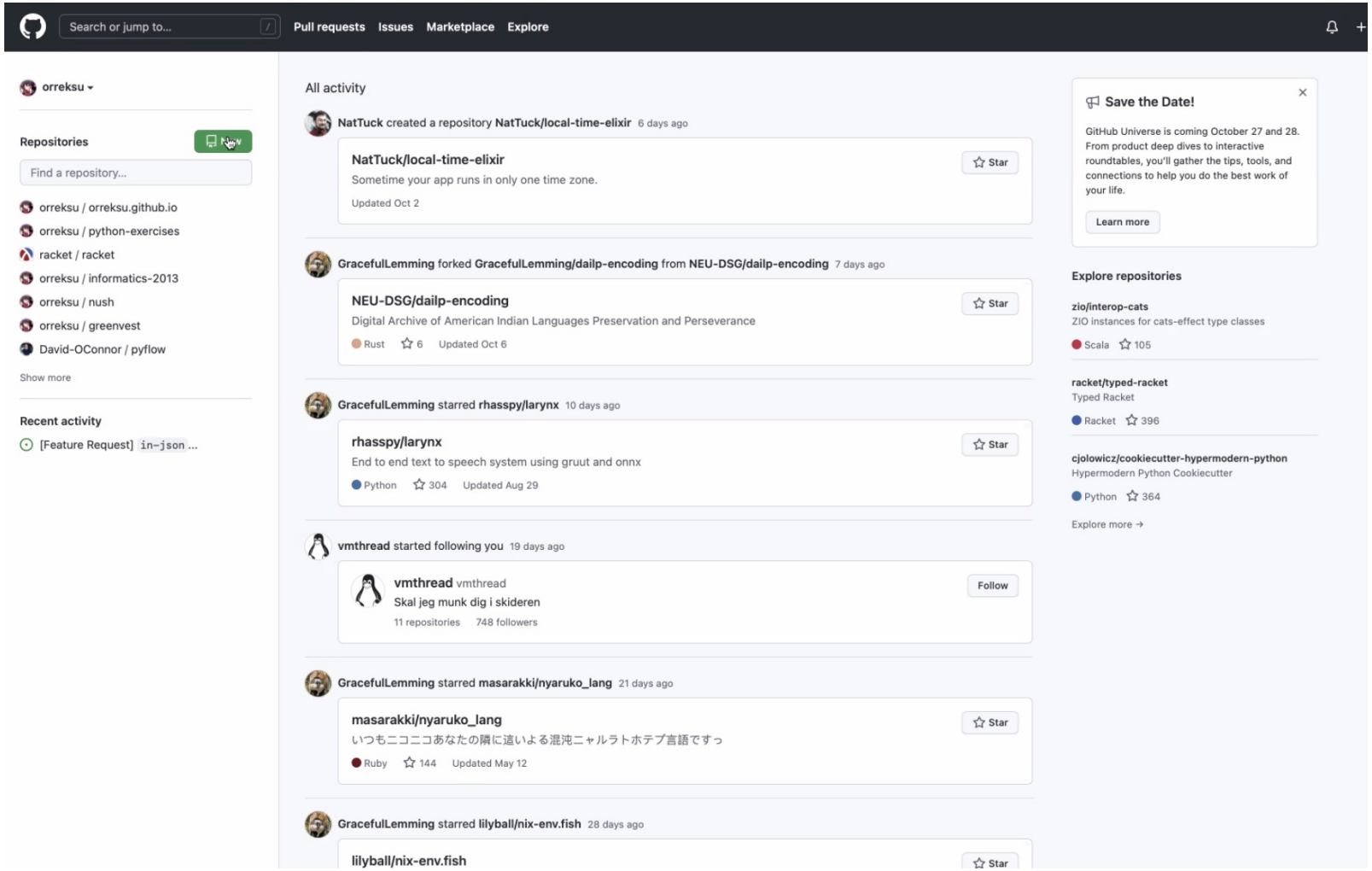


https://github.ccs.neu.edu/rachlin/ds3500_fa21.git

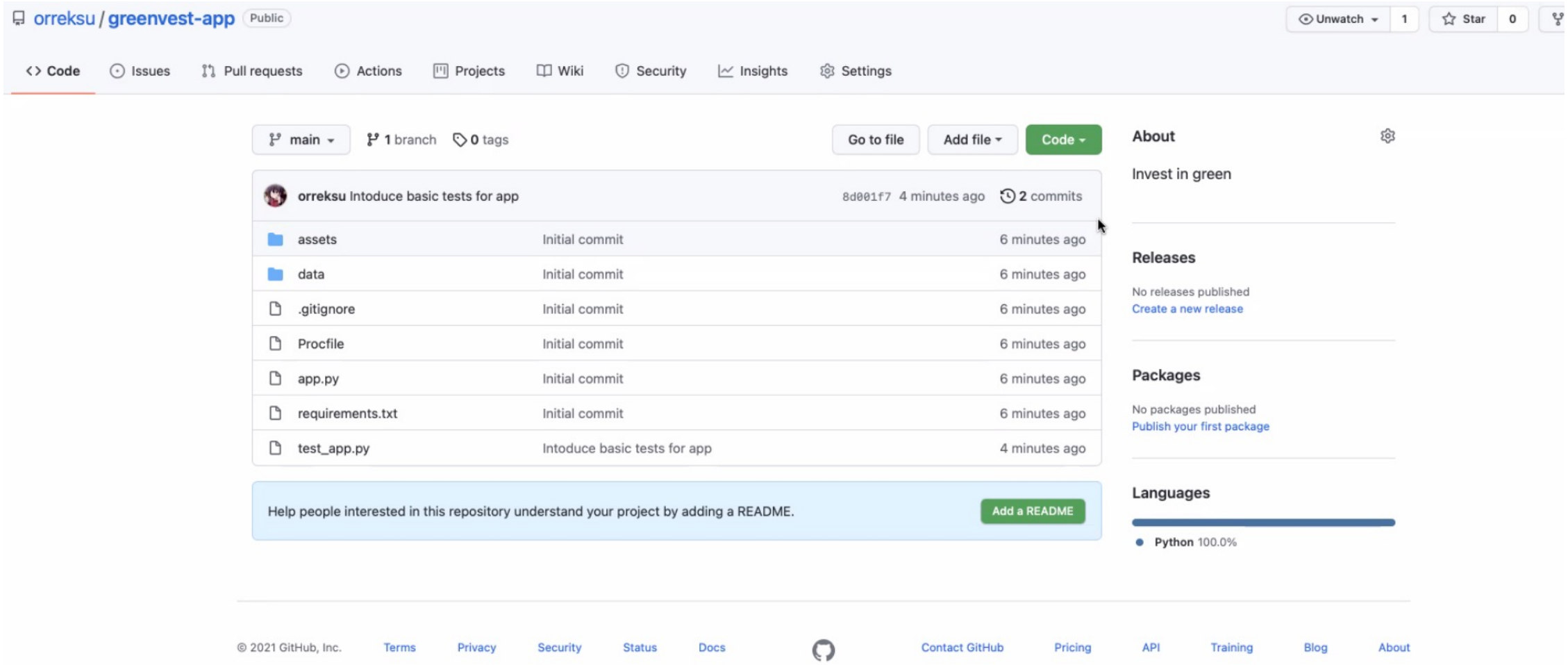
Setting up Git Repositories



Viewing Git Repositories



Your new repo!



Cloning

```
~/duc-home
❯ ls

~/duc-home
❯ git clone https://github.com/orreksu/greenvest-app.git
Cloning into 'greenvest-app'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 22 (delta 1), reused 22 (delta 1), pack-reused 0
Receiving objects: 100% (22/22), 2.60 MiB | 14.60 MiB/s, done.
Resolving deltas: 100% (1/1), done.

~/duc-home
❯
```

Failed commits / Auto-merging

```
greenvest_app on  main
❯ ls
.git  assets  data  .gitignore  app.py  Procfile  requirements.txt  test_app.py

greenvest_app on  main
❯ code .

greenvest_app on  main
❯ git add --all

greenvest_app on  main [+]
❯ git commit -m "Increse size of the logo"
[main 899ded4] Increse size of the logo
 1 file changed, 1 insertion(+), 1 deletion(-)

greenvest_app on  main [↑]
❯ git push
To https://github.com/orreksu/greenvest-app.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/orreksu/greenvest-app.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

greenvest_app on  main [↑]
❯
```

Github has changes that I don’t yet have.

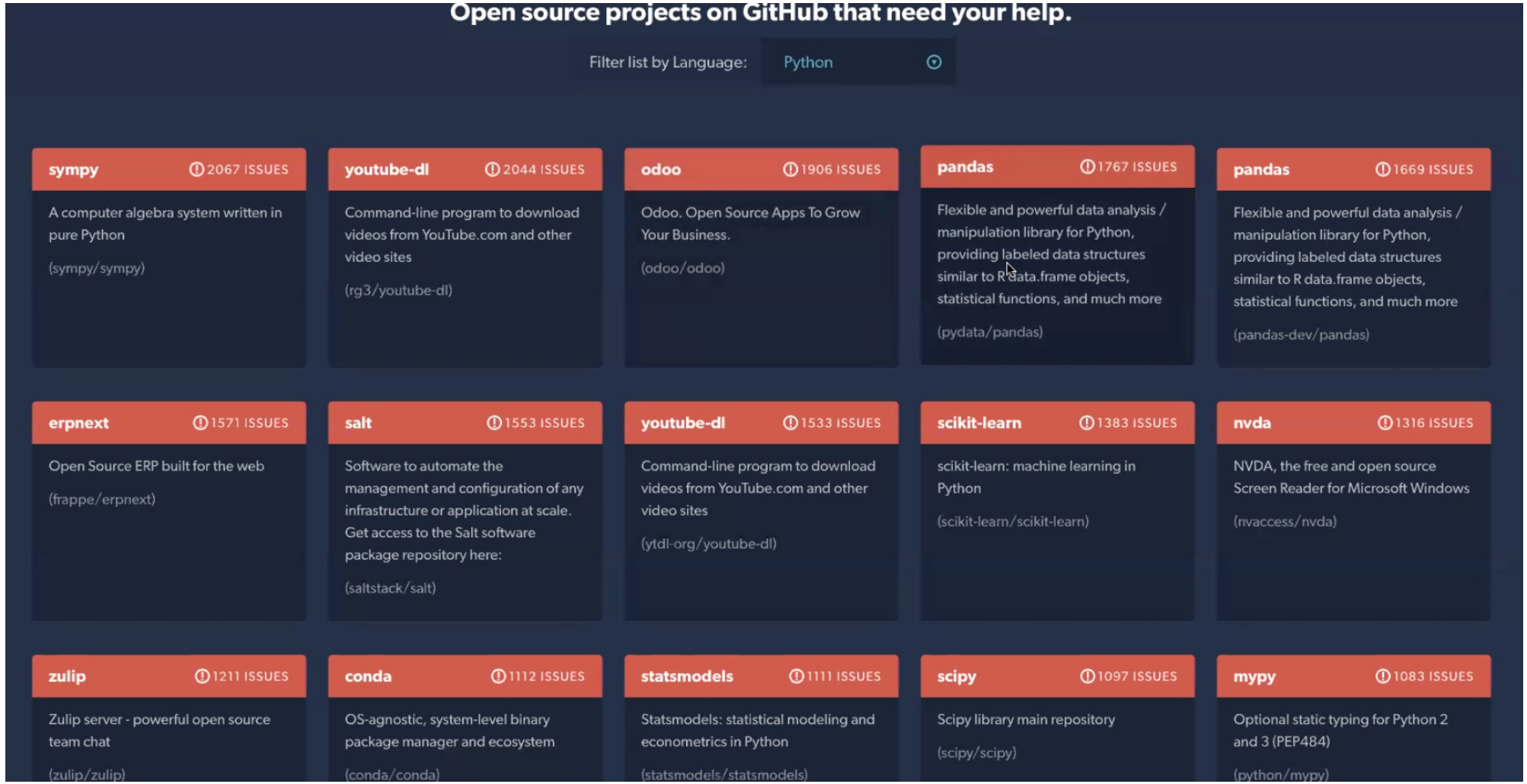
When automerge fails….

e.g., two developers changing the same line of code.

```
greenvest_app on  main [↑]
❯ git push
To https://github.com/orreksu/greenvest-app.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/orreksu/greenvest-app.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

greenvest_app on  main [↑]
❯ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), 309 bytes | 103.00 KiB/s, done.
From https://github.com/orreksu/greenvest-app
   839aec8..e40affa  main       -> origin/main
Auto-merging app.py
CONFLICT (content): Merge conflict in app.
Automatic merge failed; fix conflicts and

greenvest_app on  main (MERGING) [⇕]
❯
```

```
        html.H2('GreenVest', style={'color': '#1A9968', 'font-weight': 'bold'}),
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes | Start Live Share Session
<<<<<<< HEAD (Current Change)
            html.H5('Invest in future', style={"marginTop": "-10px"})
=======
            html.H5('Be Green!', style={"marginTop": "-10px"})
>>>>>>> e40affa4f48ca8a653676e18a295662fd2d4f491 (Incoming Change)
        ], className="two columns"),
```

CodeTriage: Tracking projects with lots of issues



Continuous Integration / Continuous Deployment

Actions

CI vs CD

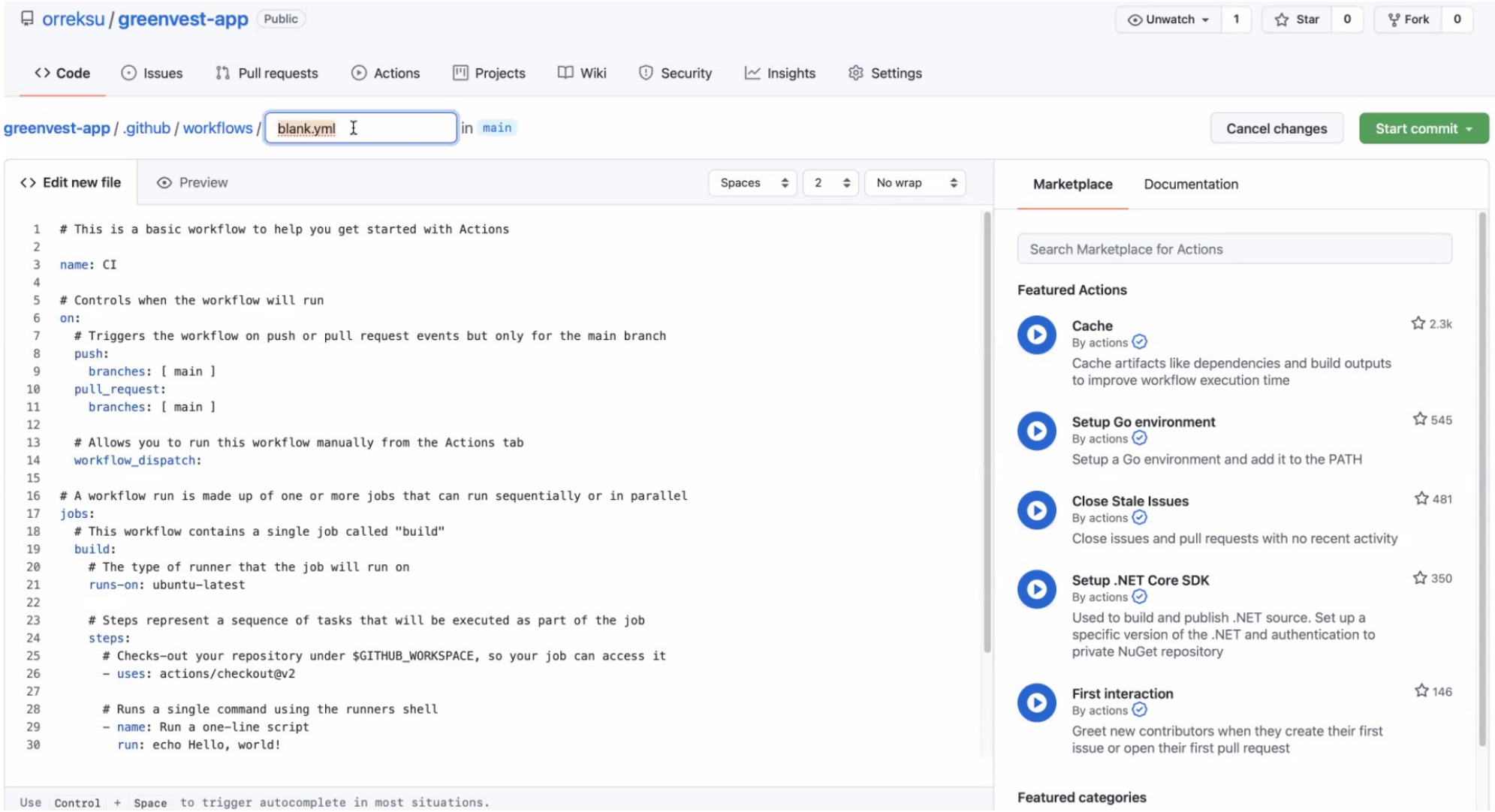
Deliver fast in small chunk to incrementally improve product.
Commit often, test every time.

CI (Continuous Integration) helps to build, test, and merge branches.
CD (Continuous Deployment) helps automate the process of deployment to the server.

Simple Actions

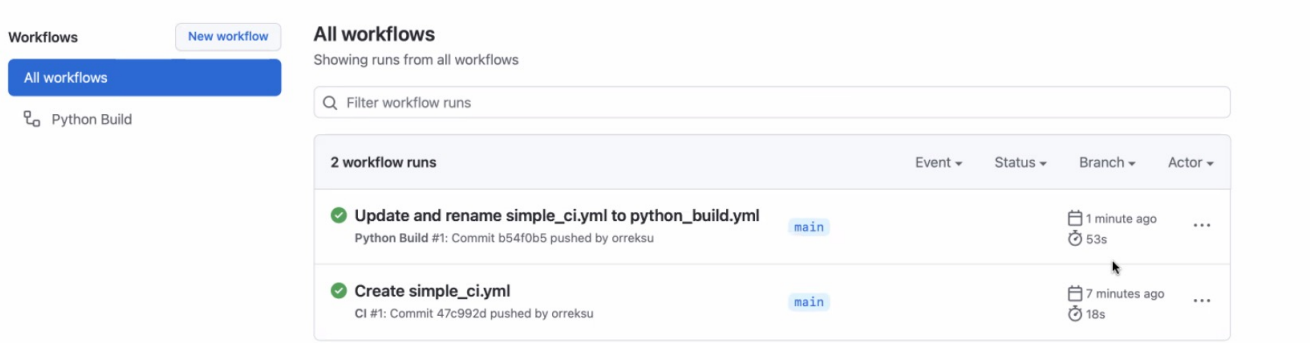
```yaml
# This is a basic workflow to help you get started with Actions

name: CI

# Controls when the workflow will run

on:
  # Triggers the workflow on push or pull request events but only for the main branch
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:
```

Workflows: “On push, run the following jobs”



Modified workflow for python

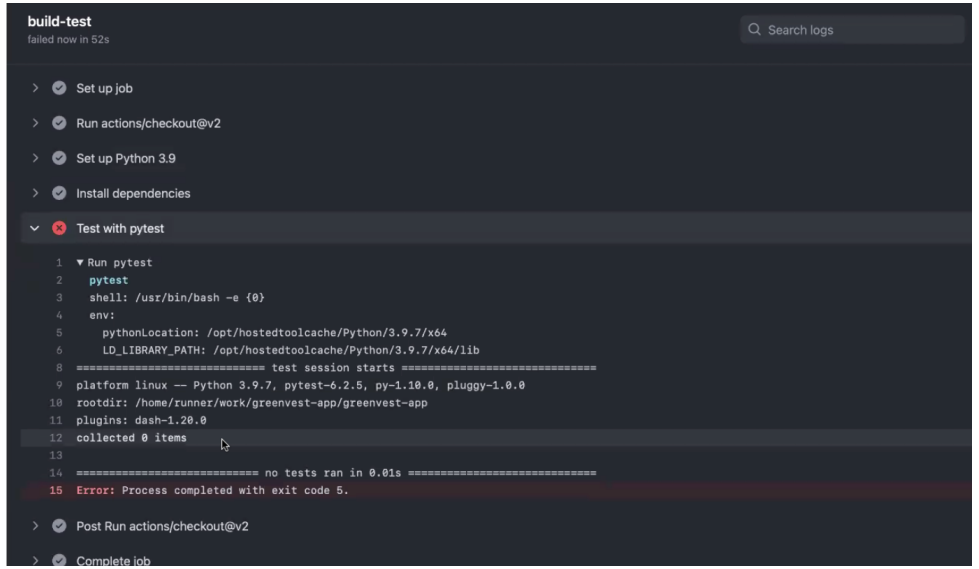
```yaml
name: Python Build

on:
  push:
    branches: [ main ]

  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Set up Python 3.9
        uses: actions/setup-python@v2
        with:
          python-version: 3.9

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install flake8 pytest
          if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
```



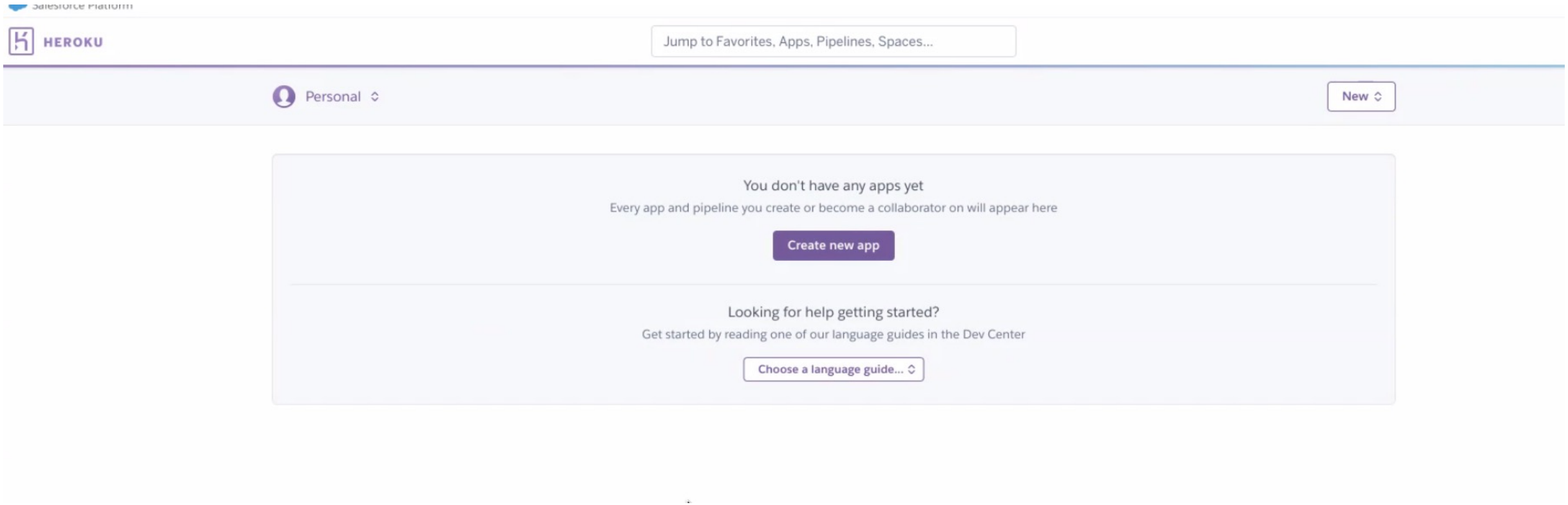
A workflow that uses pytest

```yaml
jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Set up Python 3.9
        uses: actions/setup-python@v2
        with:
          python-version: 3.9

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip

          if [ -f requirements.txt ]; then pip install -r requirements.txt; fi

  test:
    needs: build
    runs-on: ubuntu-latest

    steps:
      - name: Install test tools
          pip install flake8 pytest
      - name: Test with pytest
        run: |
          pytest
```

```python
# content of test_sample.py
def inc(x):
    return x + 1

def test_answer_1():
    assert inc(3) == 5

def test_answer_2():
    assert inc(3) == 4
```



Heroku


HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Personal
New
You don't have any apps yet
Every app and pipeline you create or become a collaborator on will appear here
Create new app
Looking for help getting started?
Get started by reading one of our language guides in the Dev Center
Choose a language guide...

Heroku Deployment

```
$ git init
Initialized empty Git repository in .git/
$ git add .
$ git commit -m "My first commit"
Created initial commit 5df2d09: My first commit
 44 files changed, 8393 insertions(+), 0 deletions(-)
 create mode 100644 README
 create mode 100644 Procfile
 create mode 100644 app/controllers/source_file
...
```

Be sure to initialize the Git repository in your app's root directory. If your app is in a subdirectory of your repository, it won't run when it is pushed to Heroku.

Your app's code is now tracked in a local Git repository. It has not yet been pushed to any remote servers.

Creating a Heroku remote

Git remotes are versions of your repository that live on other servers. You deploy your app by pushing its code to a special Heroku-hosted remote that's associated with your app.

For a new Heroku app

The `heroku create` CLI command creates a new empty application on Heroku, along with an associated empty Git repository. If you run this command from your app's root directory, the empty Heroku Git repository is automatically set as a remote for your local repository

with the `heroku git:remote` command. All you need is your Heroku app's name.

```
$ heroku git:remote -a thawing-inlet-61413
set git remote heroku to https://git.heroku.com/thawing-inlet-61413.git
```

Renaming remotes

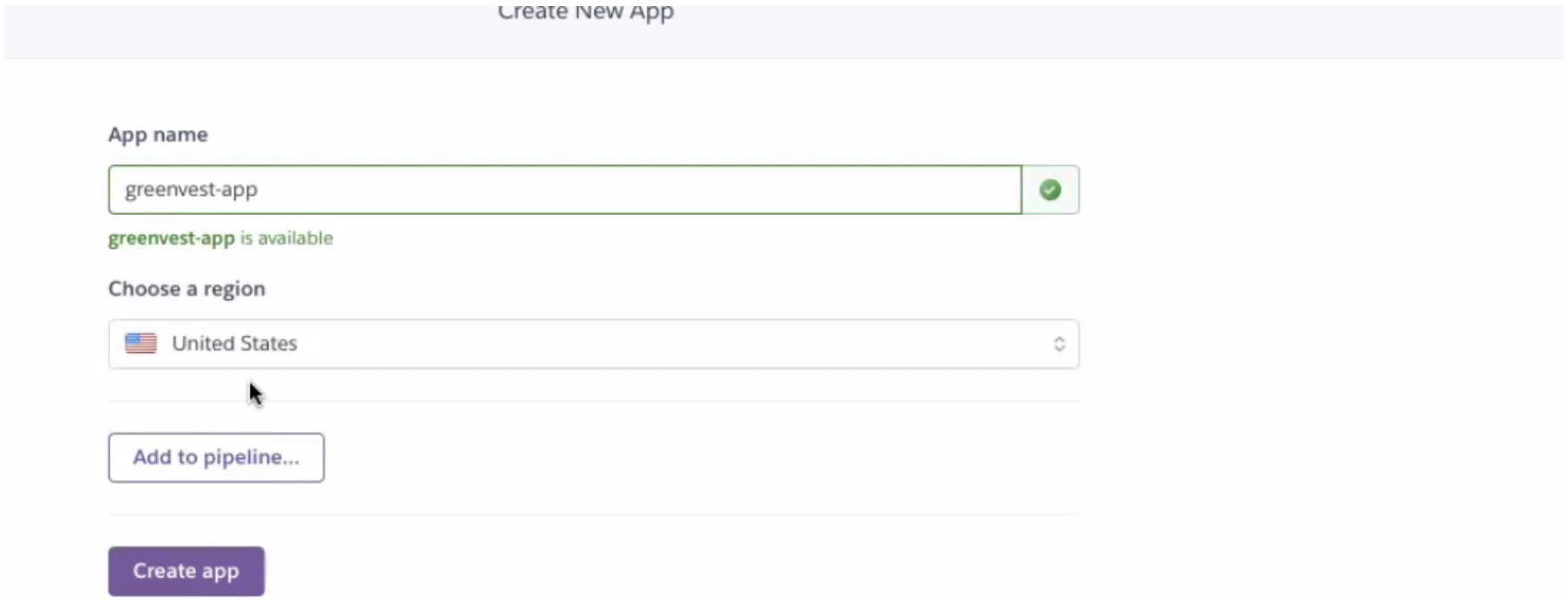
By default, the Heroku CLI names all of the Heroku remotes it creates for your app `heroku`. You can rename your remotes with the `git remote rename` command:

```
$ git remote rename heroku heroku-staging
```

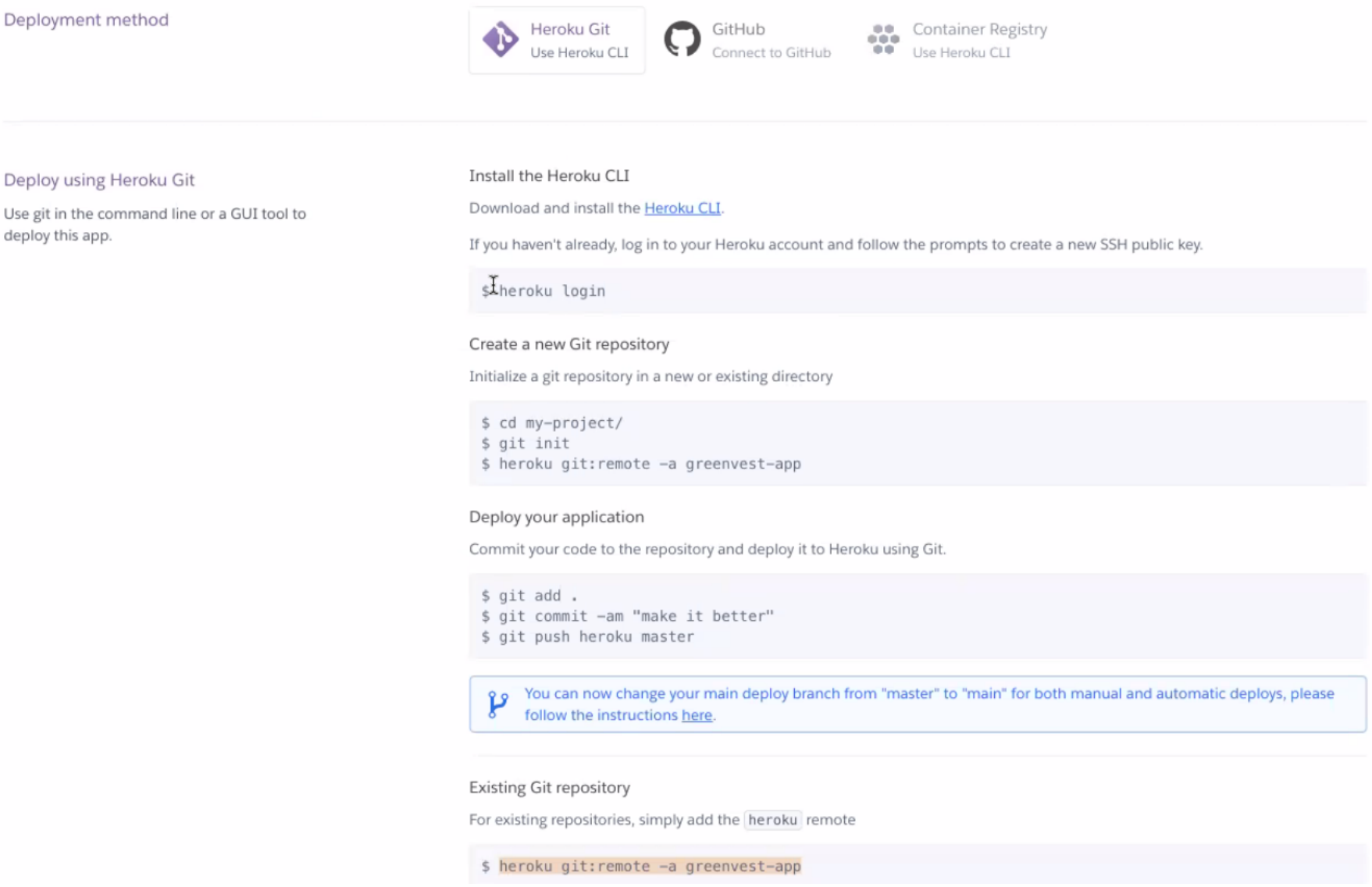
Renaming your Heroku remote can be handy if you have multiple Heroku apps that use the same codebase (for example, the staging and production versions of an app). In this case, each Heroku app has its own remote in your local repository.

The remainder of this article assumes your app has a single Heroku remote that is named `heroku`.

Creating the Heroku App

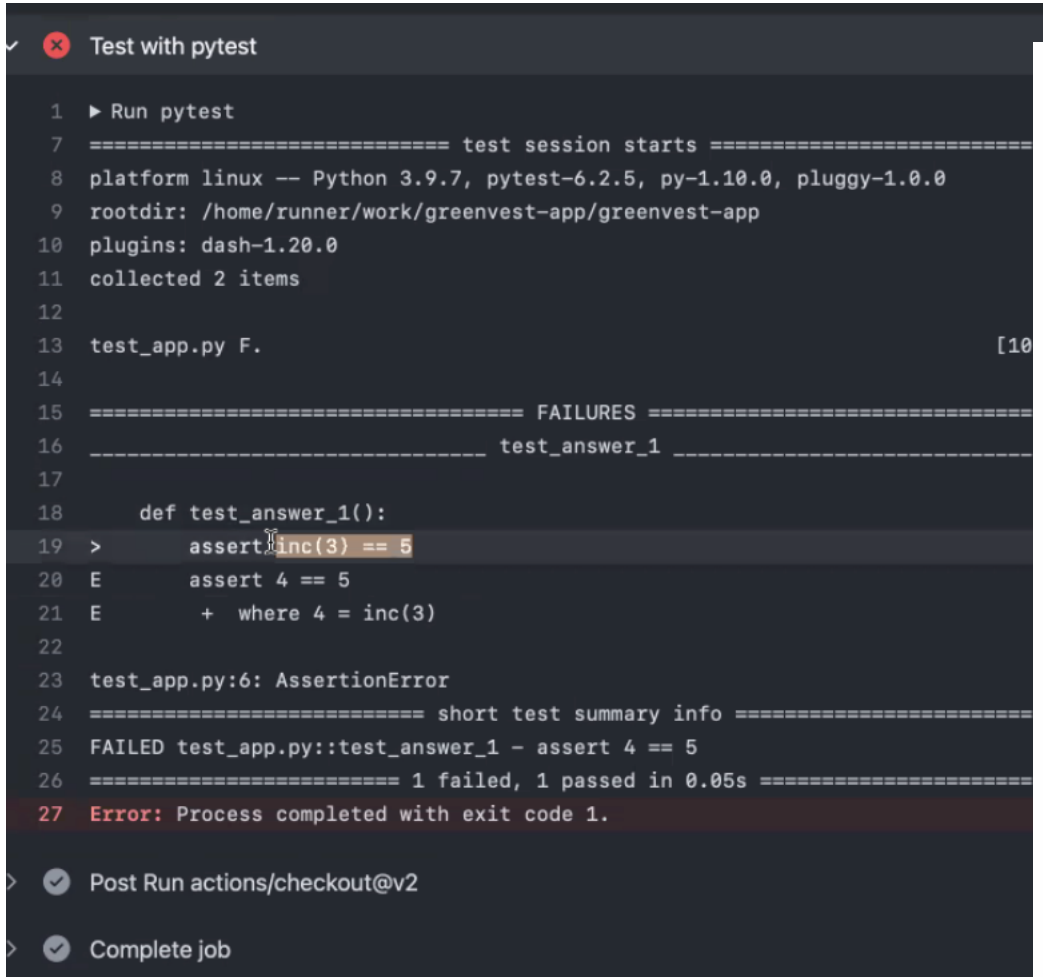


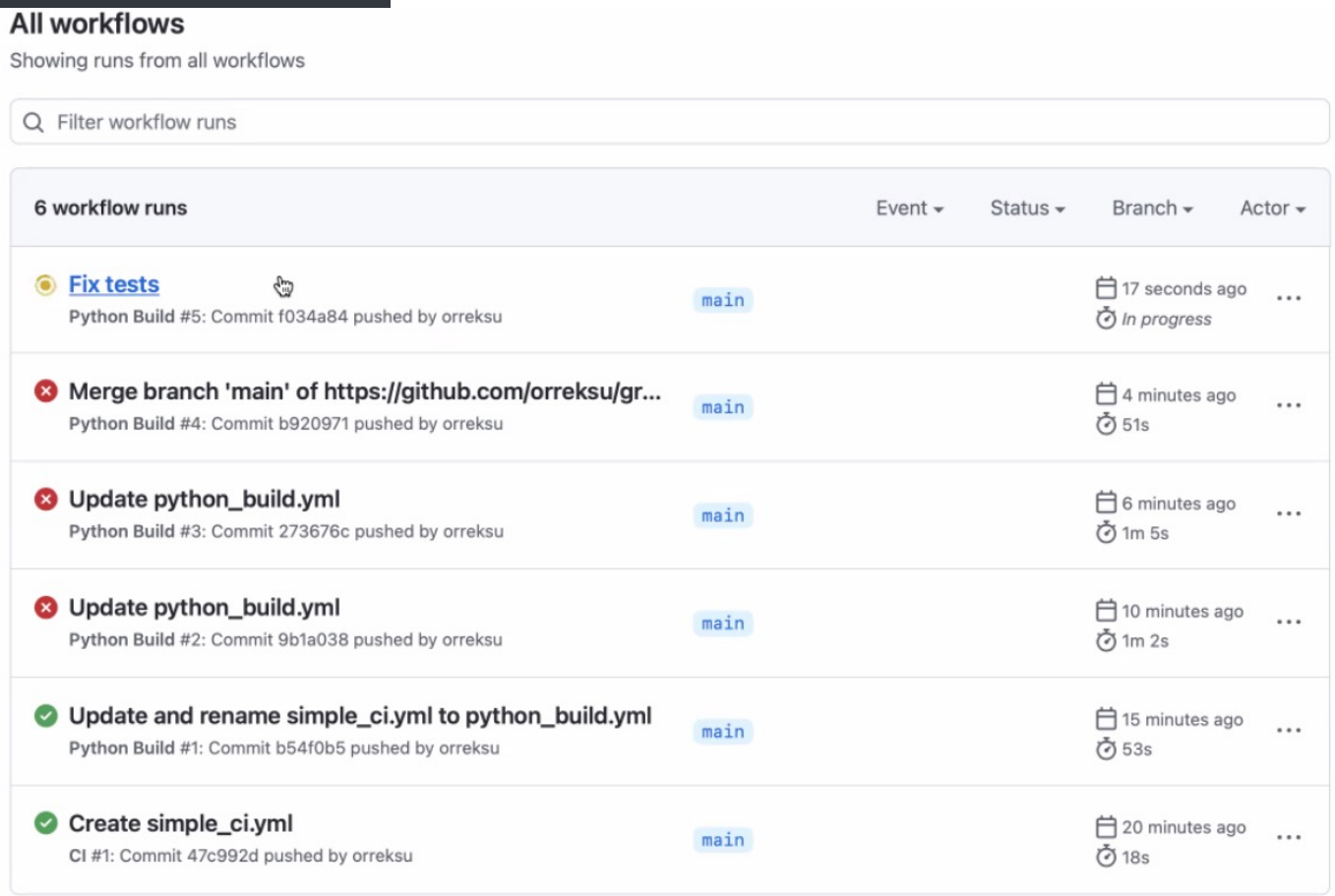
Install the Heroku Command-Line Interface (CLI)



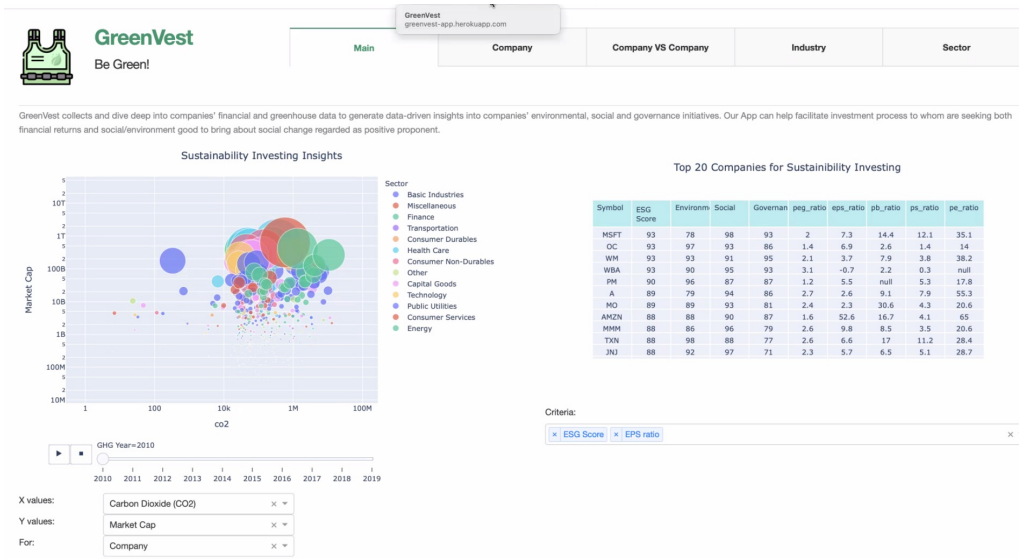
Our previous test is failing….

retriggering tests after fixing:

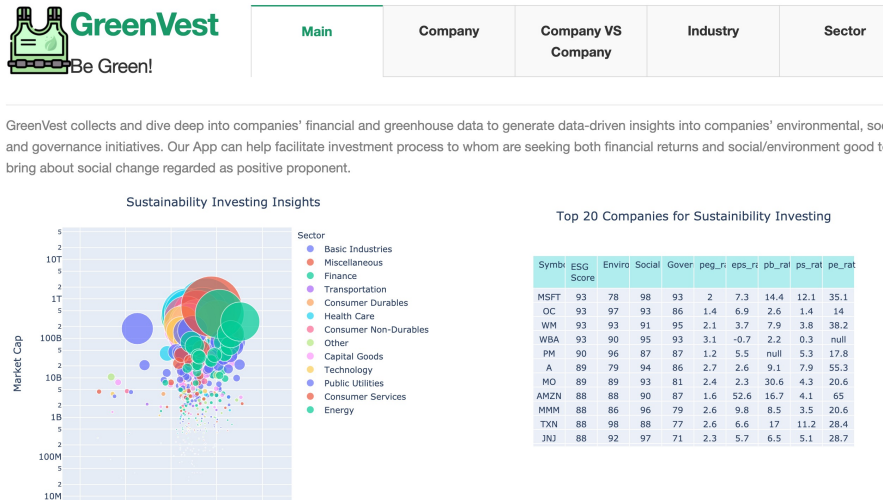


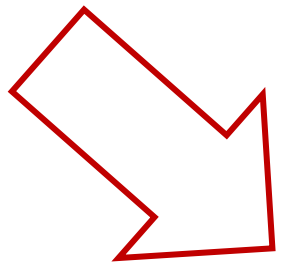


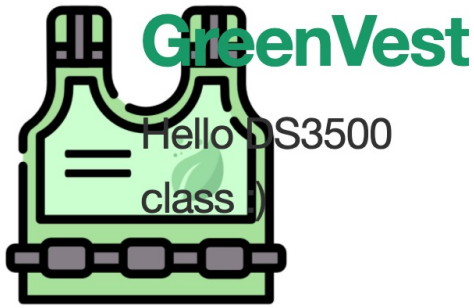
```
greenvest_app on  main
❯ git push heroku main
Enumerating objects: 66, done.
Counting objects: 100% (66/66), done.
Delta compression using up to 16 threads
Compressing objects: 100% (55/55), done.
Writing objects: 100% (66/66), 2.60 MiB | 2.74 MiB/s, done.
Total 66 (delta 23), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Determining which buildpack to use for this app
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the buildpack default: python-
3.9.7
remote:        To use a different version, see: https://devcenter.heroku.com/article
s/python-runtimes
remote: -----> Installing python-3.9.7
remote: -----> Installing pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote:        Collecting Brotli==1.0.9
remote:          Downloading Brotli-1.0.9-cp39-cp39-manylinux1_x86_64.whl (357 kB)
```

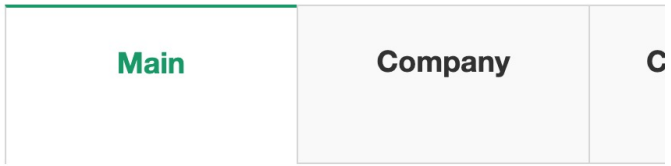


Changing the app. Everything automated





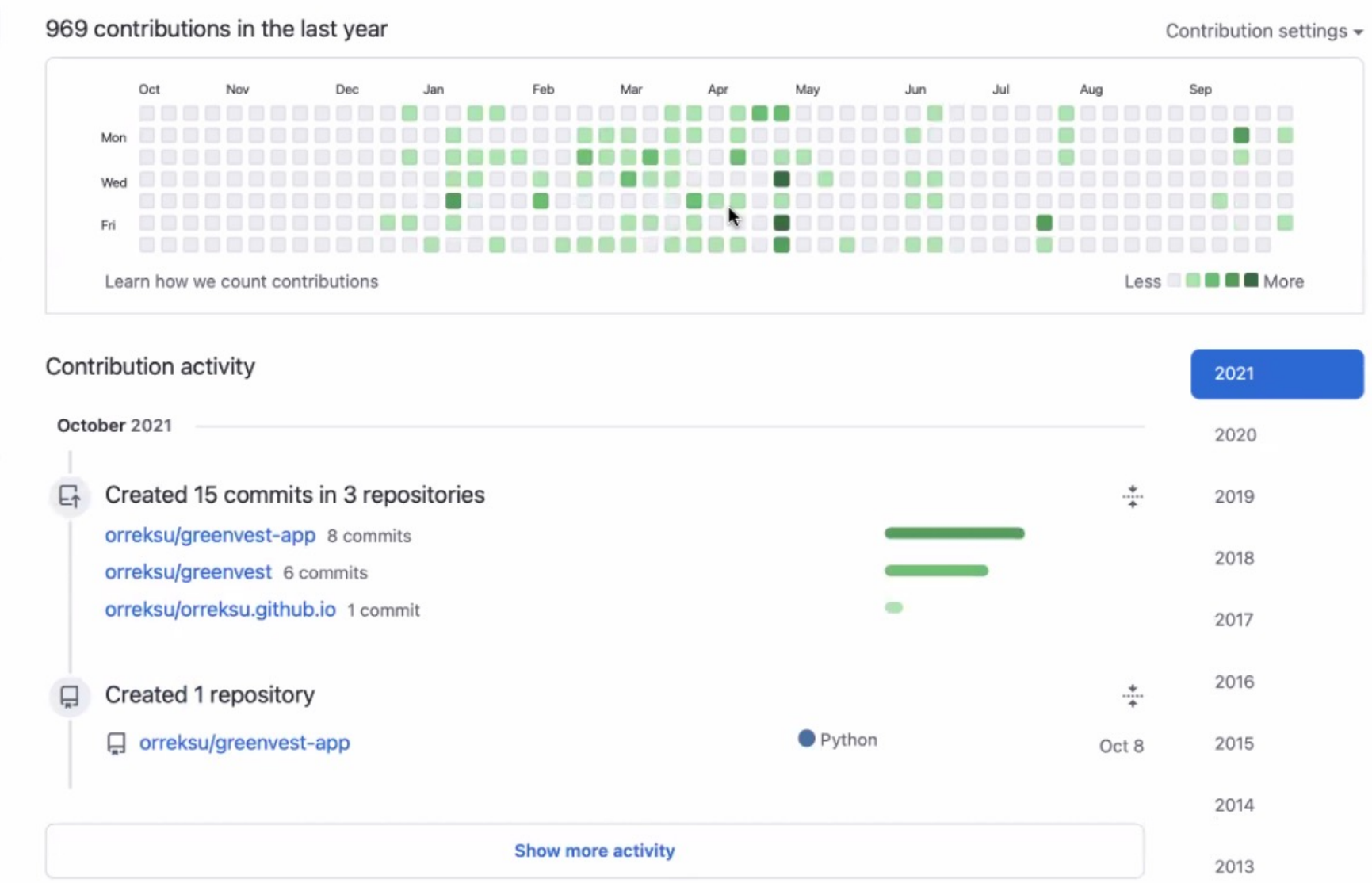




GreenVest collects and dive deep into companies' financial and greenhouse data to generat
and governance initiatives. Our App can help facilitate investment process to whom are seek
bring about social change regarded as positive proponent.

Sustainability Investing Insights

Tracking your changes.



Reliability, Scalability, and Maintainability

Challenges

- **Reliability**: The system should continue to work *correctly* in response to hardware, software, or user errors

- **Scalability**: As the system grows (data volume, traffic, or complexity) there should be reasonable ways of dealing with that growth. The design should be *elastic.*

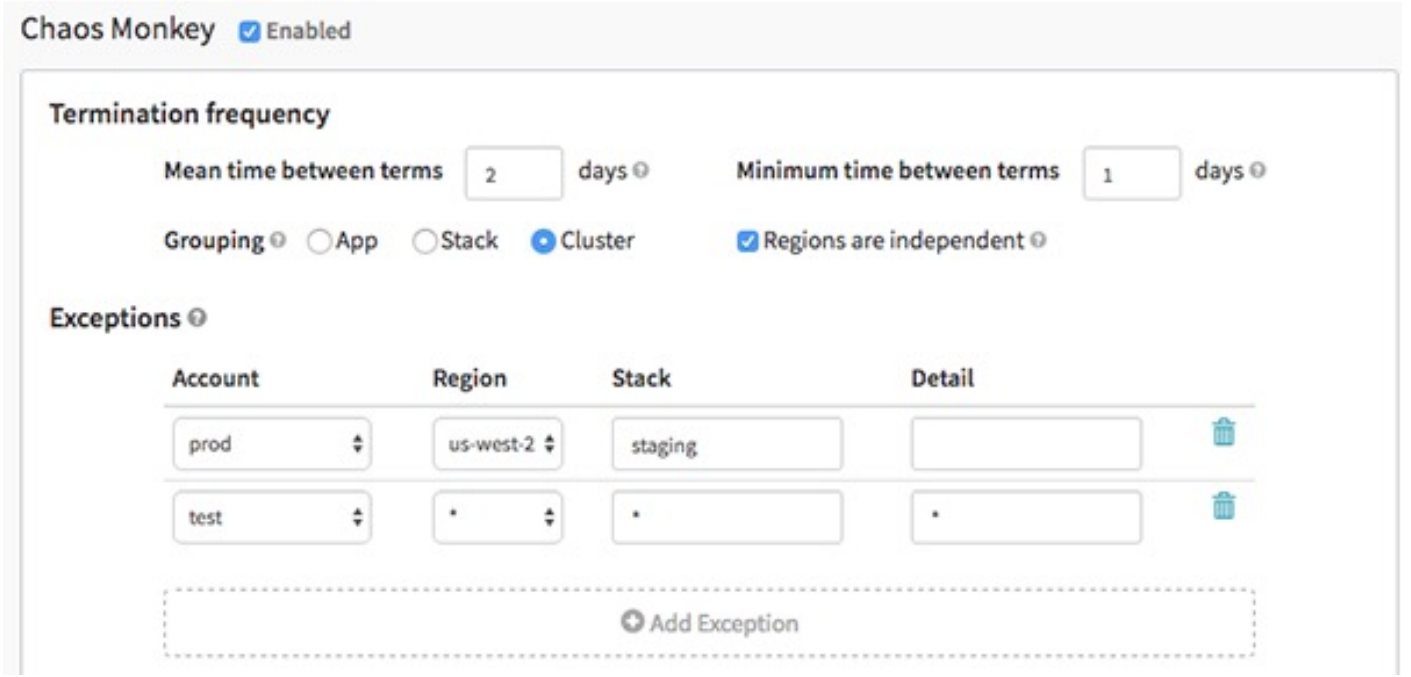
- **Maintainability**: Productively work on the system as people come and go. Add new features to adapt to changing requirements.

What is reliability?

- **Predictable**: Application performs as expected
- **Fault-tolerant:** Application copes with human error & hardware failures.
- **Performant**: Performance is adequate under expected load and data volume
- **Secure**: System prevents unauthorized access

Reliability

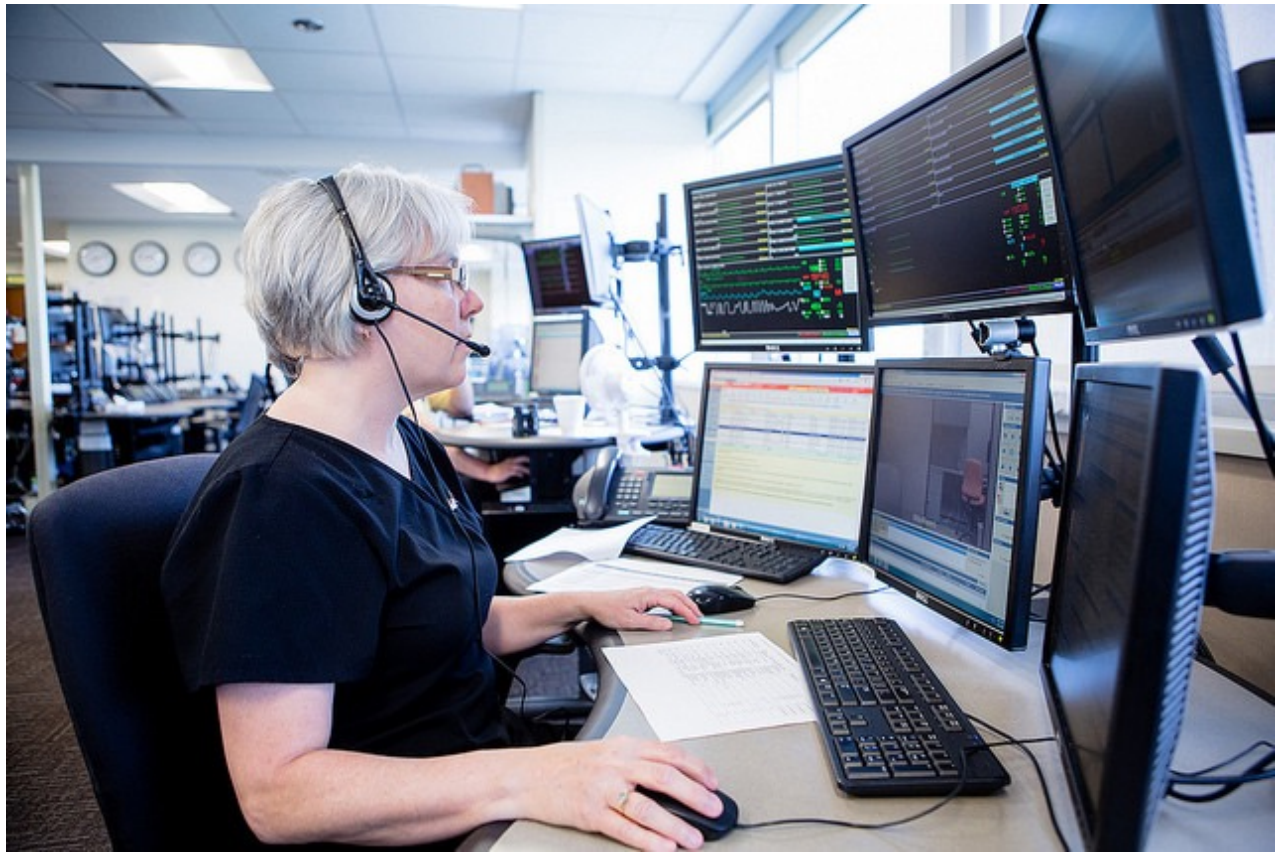
- **Faults** are a deviation in the spec of one particular component of the system. **Failures** cause the whole system to stop, requiring user intervention.
- A system is *fault-tolerant* if it anticipates and copes with certain kind of faults.
- One way to design fault-tolerant systems is to induce them deliberately! (e.g., submit bad input by the user, shut down a process, etc.)



Netflix Chaos Monkey

Reliability: Human Errors

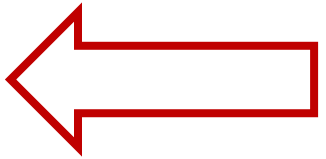
- Application / service / network configuration errors
- Balance flexibility with restrictions aimed at encouraging users to do the right thing
- Sandbox testing/ Automated unit testing to identify edge cases
- Automated rollout / rollback of code changes (DevOps)
- Monitoring / Dashboards / Telemetry for error detection, performance tracking



Maintainability

Software cost breakdown:

Initial development: ~10-20%
Error correction: ~10-20%
On-going Maintenance: ~60-80%



Operability: Make it easy for operations teams to keep system running

Simplicity: Make it easy for engineers to understand

Evolvability: Make it easy for engineers to modify in response to changing requirements *(extensibility / plasticity)*

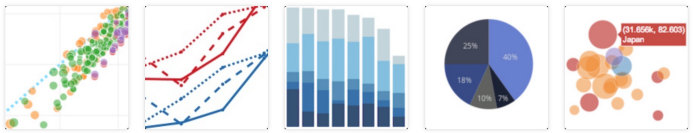
Python Exception Hierarchy

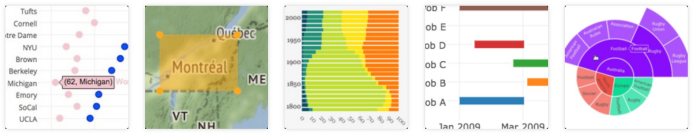
```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
      |    +-- ModuleNotFoundError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
      +-- OSError
      |    +-- BlockingIOError
      |    +-- ChildProcessError
      |    +-- ConnectionError
      |    |    +-- BrokenPipeError
      |    |    +-- ConnectionAbortedError
      |    |    +-- ConnectionRefusedError
      |    |    +-- ConnectionResetError
      |    +-- FileExistsError
      |    +-- FileNotFoundError
      |    +-- InterruptedError
      |    +-- IsADirectoryError
      |    +-- NotADirectoryError
      |    +-- PermissionError
      |    +-- ProcessLookupError
      |    +-- TimeoutError
      +-- ReferenceError
      +-- RuntimeError
      |    +-- NotImplementedError
      |    +-- RecursionError
      +-- SyntaxError
      |    +-- IndentationError
      |         +-- TabError
      +-- SystemError
      +-- TypeError
      +-- ValueError
      |    +-- UnicodeError
      |         +-- UnicodeDecodeError
      |         +-- UnicodeEncodeError
      |         +-- UnicodeTranslateError
      +-- Warning
           +-- DeprecationWarning
           +-- PendingDeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
           +-- UnicodeWarning
           +-- BytesWarning
           +-- ResourceWarning
```

In Python, all exceptions must be instances of a class that derives from BaseException. In a try statement with an except clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which *it* is derived).

Advanced Visualization with Plotly

A gallery of possibilities: https://plotly.com/python/

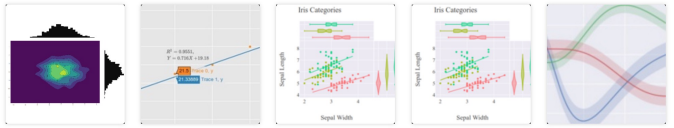

Scatter Plots | Line Charts | Bar Charts | Pie Charts | Bubble Charts


Dot Plots | Filled Area Plots | Horizontal Bar Charts | Gantt Charts | Sunburst Charts


Tables | Sankey Diagram | Treemap Charts | WebGL vs SVG | Figure Factory Tables


Error Bars | Box Plots | Histograms | Distplots | 2D Histograms

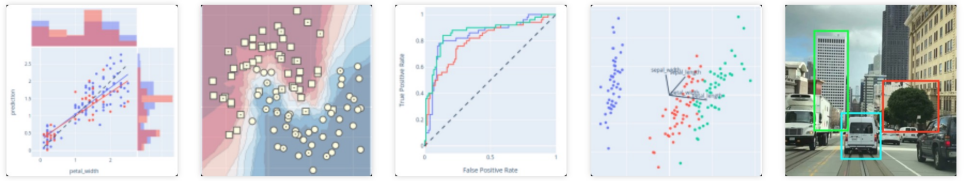

Scatterplot Matrix | Facet and Trellis Plots | Parallel Categories Diagram | Tree-plots | Violin Plots

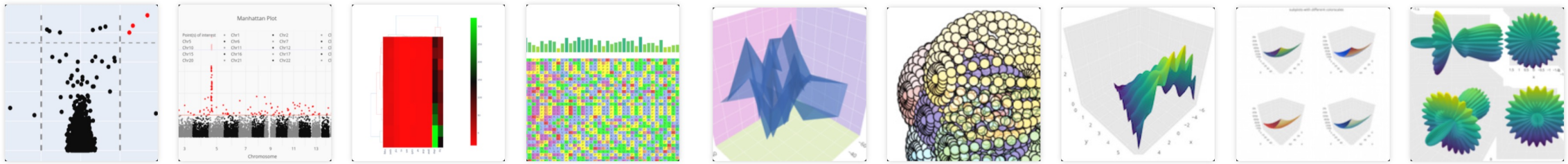

2D Histogram Contour | Linear and Non-Linear Trendlines | Marginal Distribution Plots | Strip Charts | Continuous Error Bands


Mapbox Choropleth Maps | Lines on Mapbox | Filled Area on Maps | Bubble Maps | Mapbox Density Heatmap

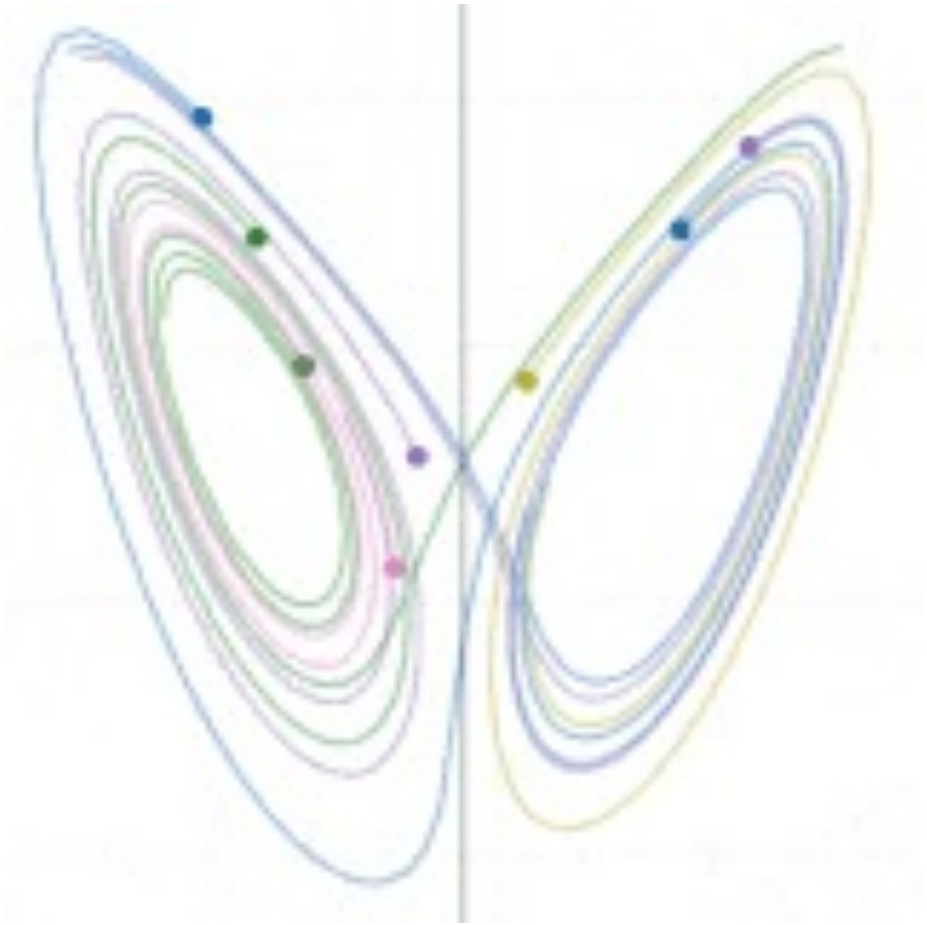
Artificial Intelligence and Machine Learning

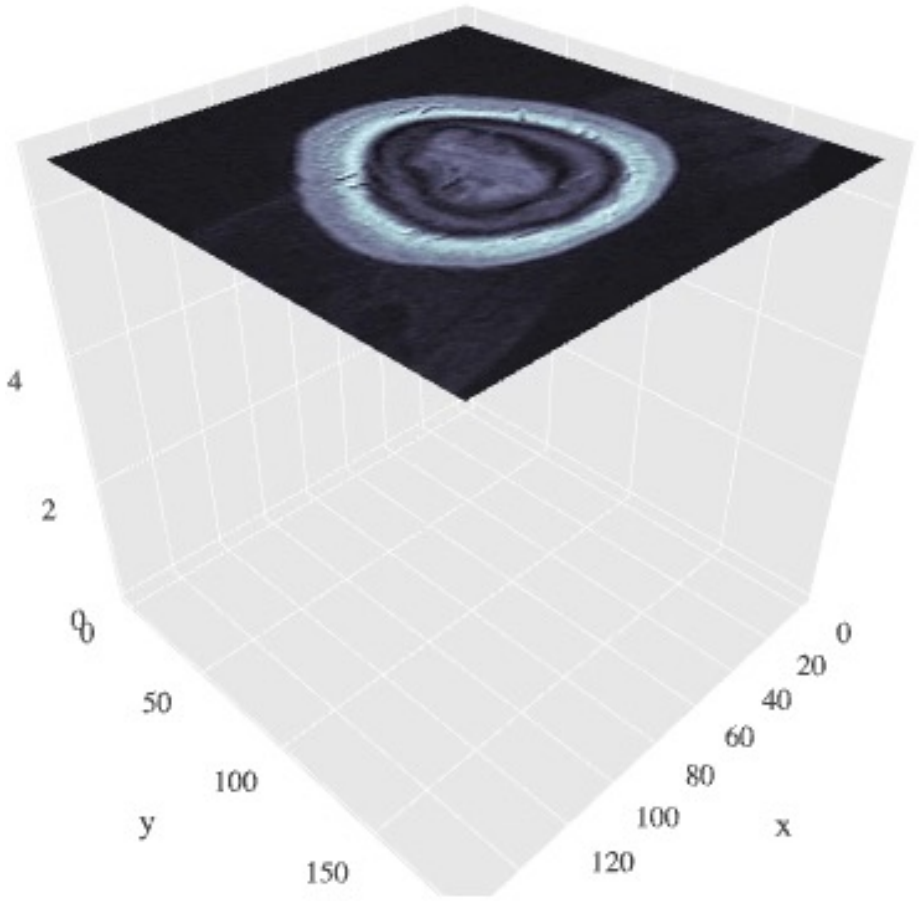
More AI and ML »


ML Regression | kNN Classification | ROC and PR Curves | PCA Visualization | AI/ML Apps with Dash


Volcano Plot | Manhattan Plot | Clustergram | Alignment Chart | 3D Axes | 3D Scatter Plots | 3D Surface Plots | 3D Subplots | 3D Camera Controls

Animations!





Is Plotly for Python Free?

Yes. Plotly for Python is free and open-source software, licensed under the **MIT license**. It costs nothing to install and use. You can view the source, report issues or contribute using our Github repository.

Can I use Plotly for Python without signing up to any service?

Yes. You can use Plotly for Python to make, view, and distribute charts and maps without registering for any service, obtaining any token, or creating any account. The one exception is that to view tile maps which use tiles from the Mapbox service (which is optional, as you can use other tile servers), you will need to have a Mapbox token.

Can I use Plotly for Python offline, without being connected to the internet?

Yes. You can use Plotly for Python to make, view, and distribute graphics totally offline. The one exception is that to view tile maps which use tiles from a cloud-hosted service, such as Open Street Maps or Mapbox, you will need a connection to that service. You can view tile maps totally offline if you run your own local tile server and use its tiles.

Free Software: Free as in beer / Free as in speech

Free as in Beer

- The software doesn't cost you anything (monetarily) to use.
- There are still restrictions on **how** you use the software – for example you can't necessarily view, modify, or redistribute the source code.

Free as in Speech

- You are at liberty to use the software however you like.
- You can inspect the source code.
- You can redistribute or repackage the code.
- You can create your own improved versions.

Installation

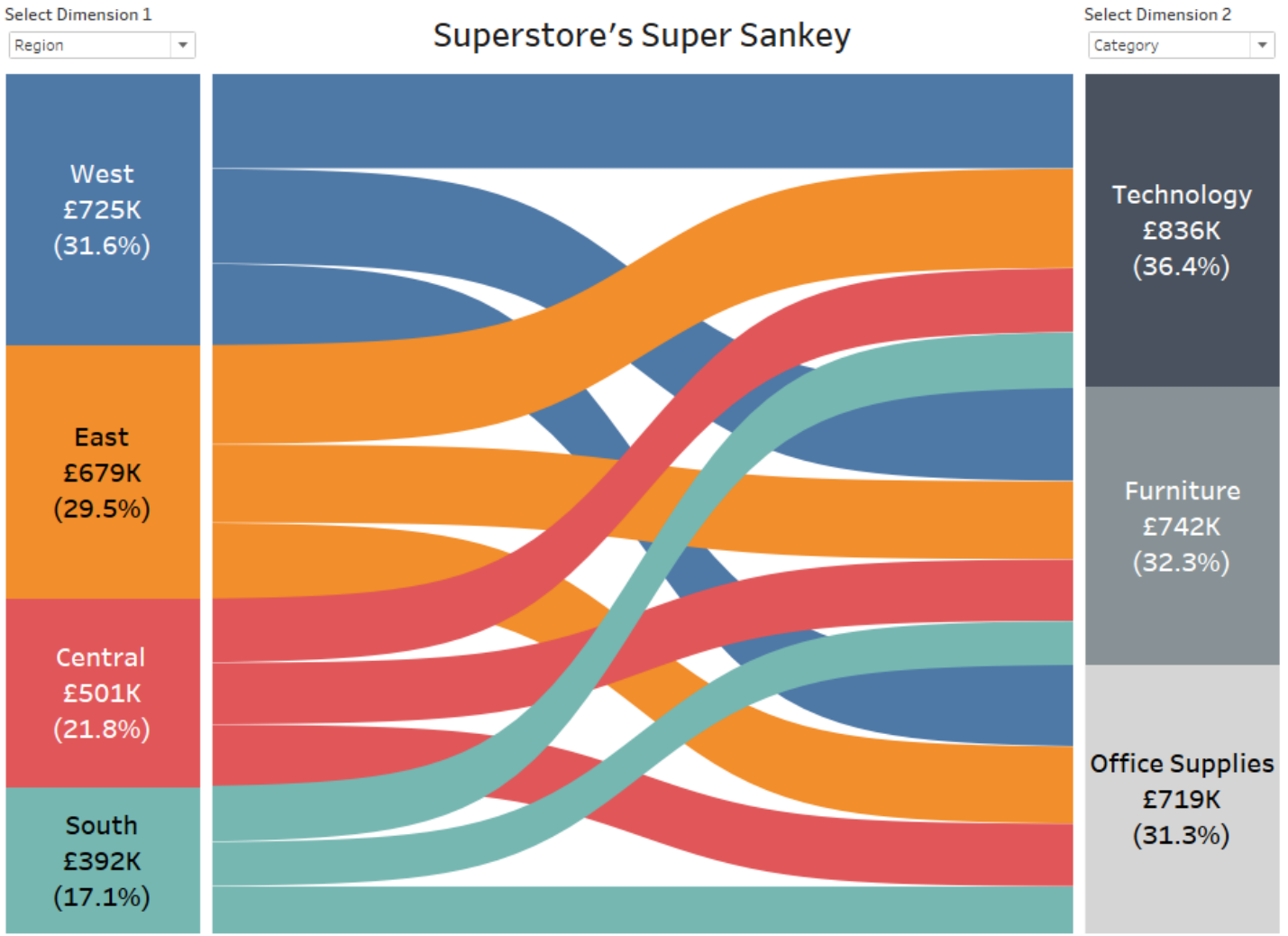
$ conda install –c plotly plotly=5.5.0
Or create a new environment
and install it there

$ conda create –name py37 python=3.7
$ conda activate py37
$ conda install –c plotly plotly=5.5.0

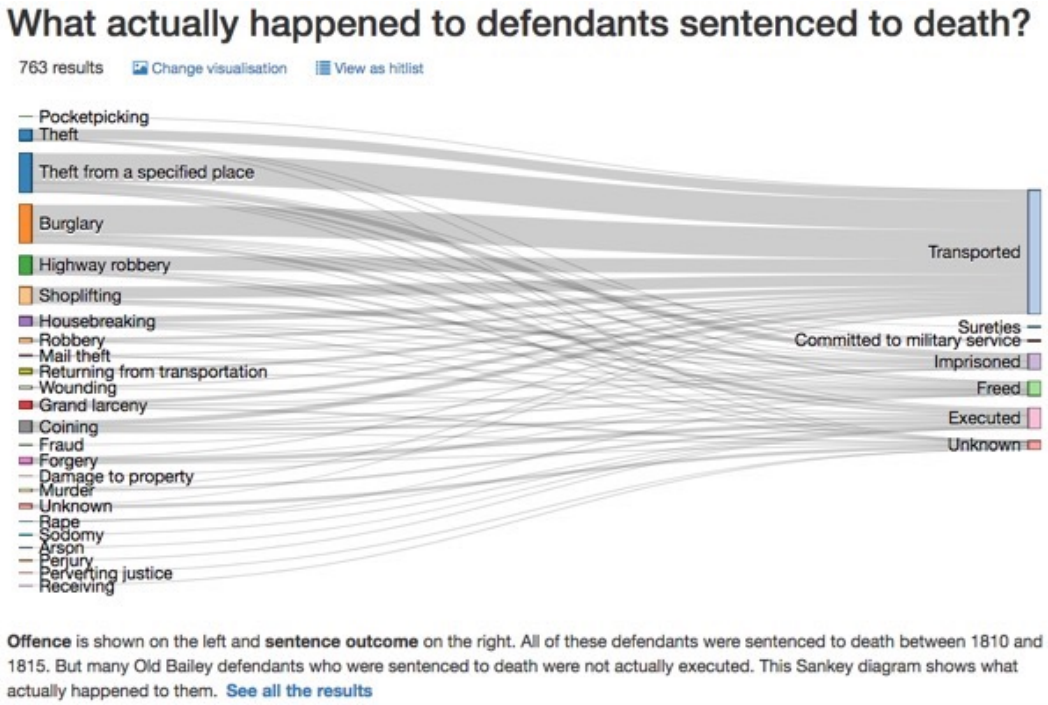
(Took about 1 minute)

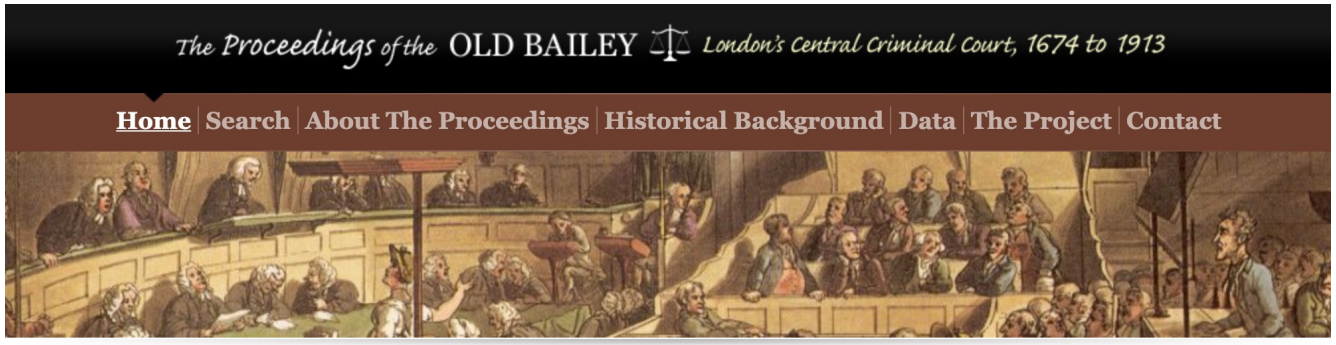
```
(py37) [508 uranus:~ ] conda install -c plotly plotly=5.5.0
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/rachlin/opt/anaconda3/envs/py37

  added / updated specs:
    - plotly=5.5.0


The following NEW packages will be INSTALLED:

  plotly             plotly/noarch::plotly-5.5.0-py_0
  tenacity           conda-forge/noarch::tenacity-8.0.1-pyhd8ed1ab_0


Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Sankey Diagrams: Market Analysis



Crime and Punishment in the Old Bailey





Home Page
Search
About the Proceedings
Historical Background
API
The Project
Copyright & Citation Guide
Contact

ON THIS DAY IN... 1722

James Lanman, (aged 11) and Samuel Armstrong, (aged 13) allegedly stole a silver snuffbox from Simon Hansel's shop. A month later they were sentenced to hang. read more

The Proceedings of the Old Bailey, 1674-1913

A fully searchable edition of the largest body of texts detailing the lives of non-elite people ever published, containing 197,745 criminal trials held at London's central criminal court. If you are new to this site, you may find the Getting Started and Guide to Searching videos and tutorials helpful.

To search the Proceedings use the boxes on the right or go to the Search Pages.

This site uses cookies. See our privacy policy.

February 2018 Update

Several bugs have been fixed and tagging errors corrected. In addition, the following new features have been added: '

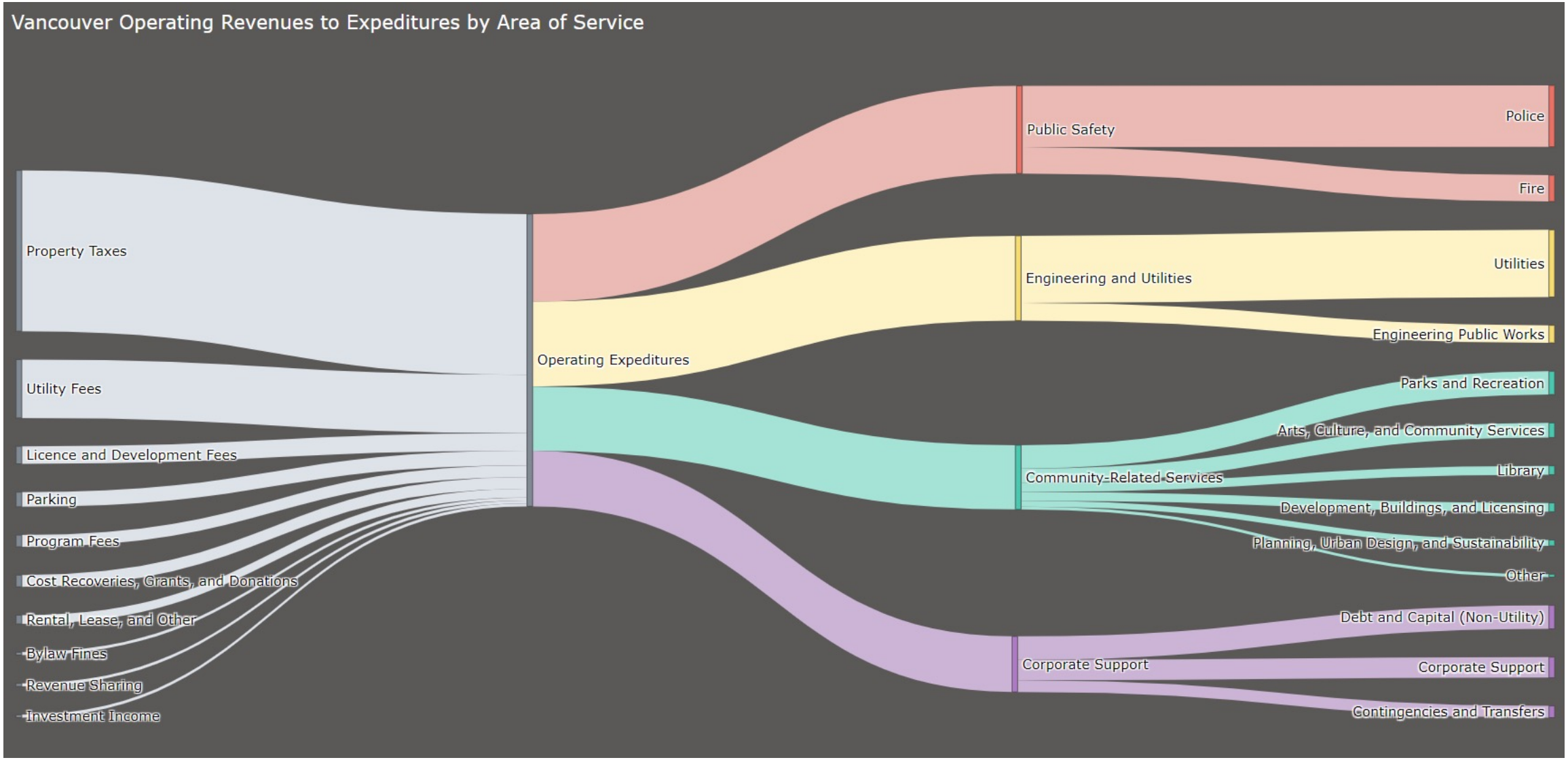
SEARCH
the Proceedings
Keyword(s)
Reference No.
Search In
<All Text>
SEARCH

The Proceedings can also be searched in:

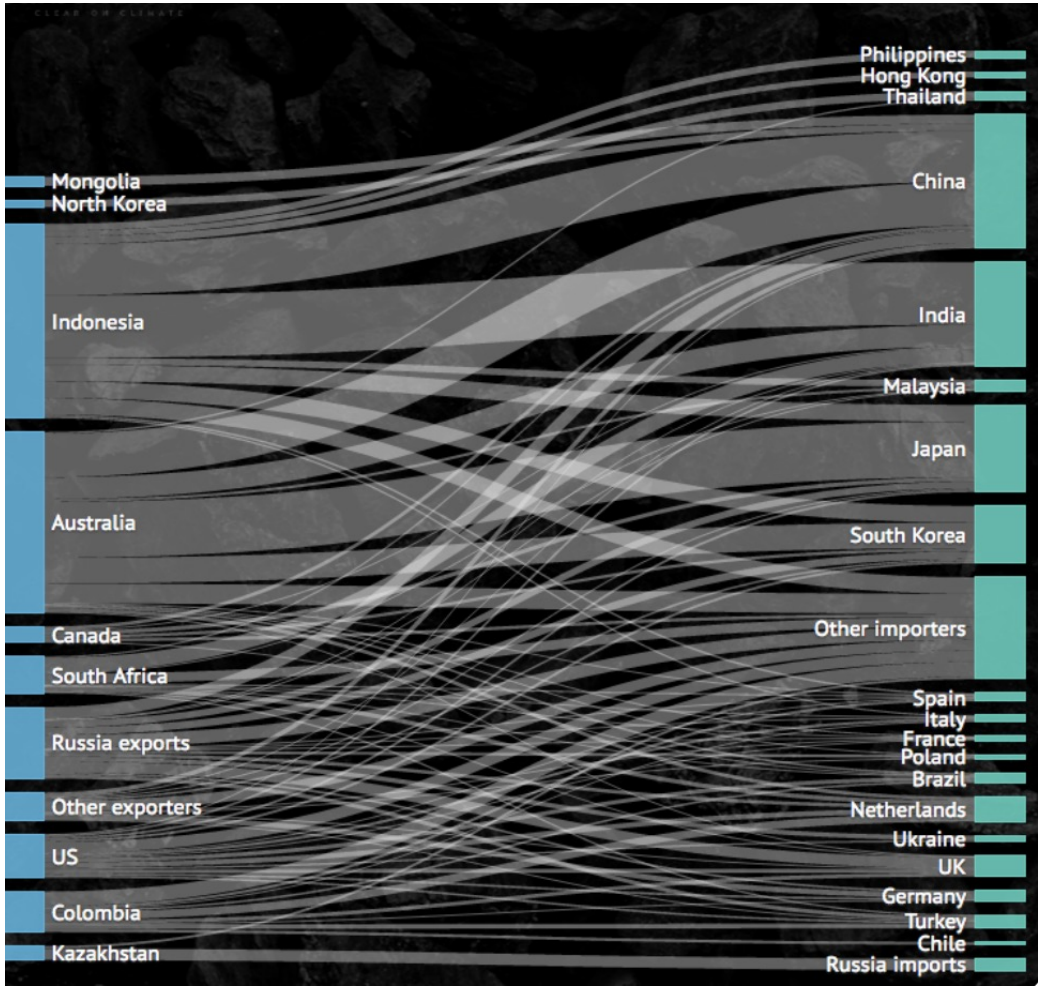
digital panopticon

Northeastern University

Sankey Diagrams: Analyzing Expenses

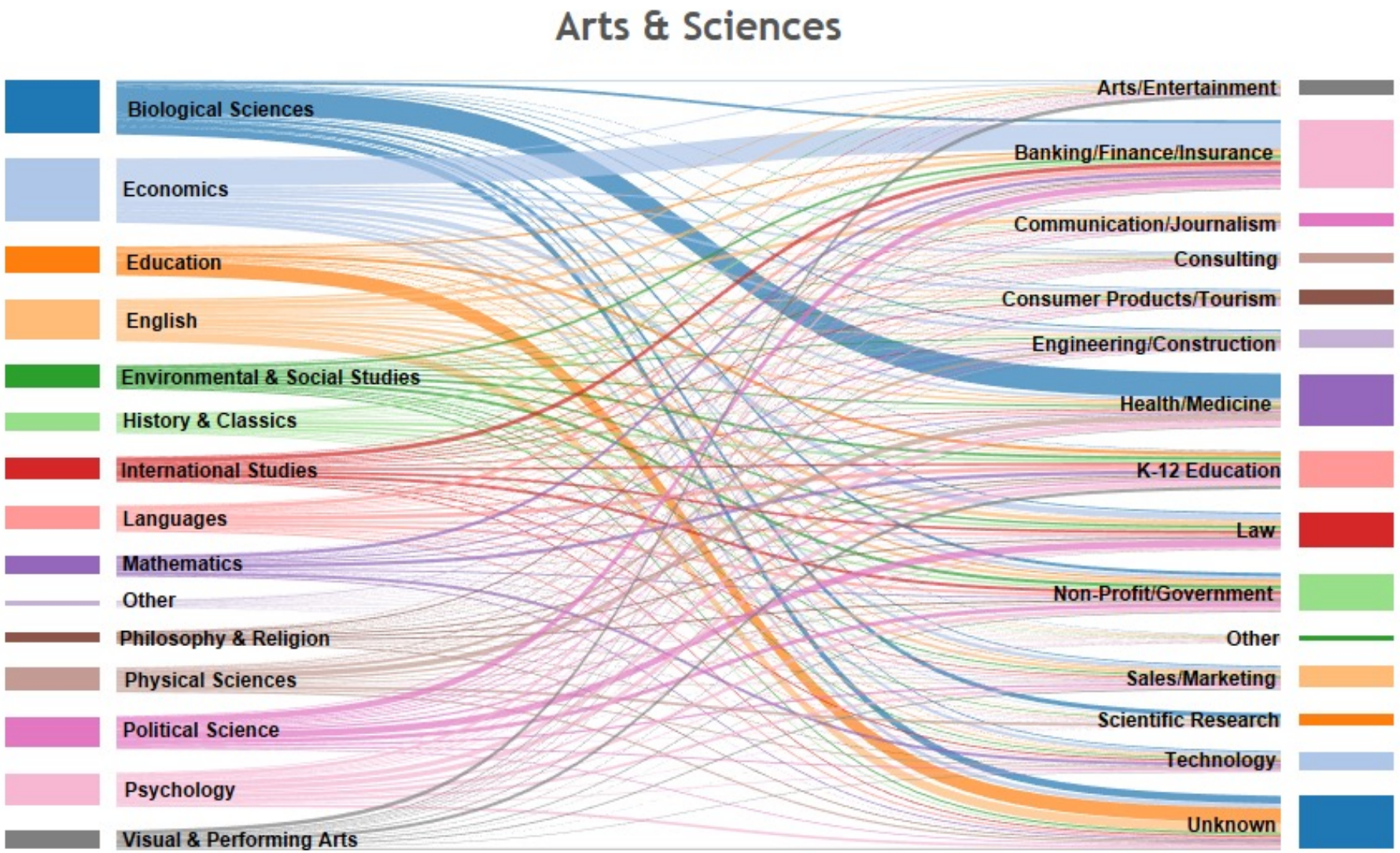


Sankey Diagrams: Trade

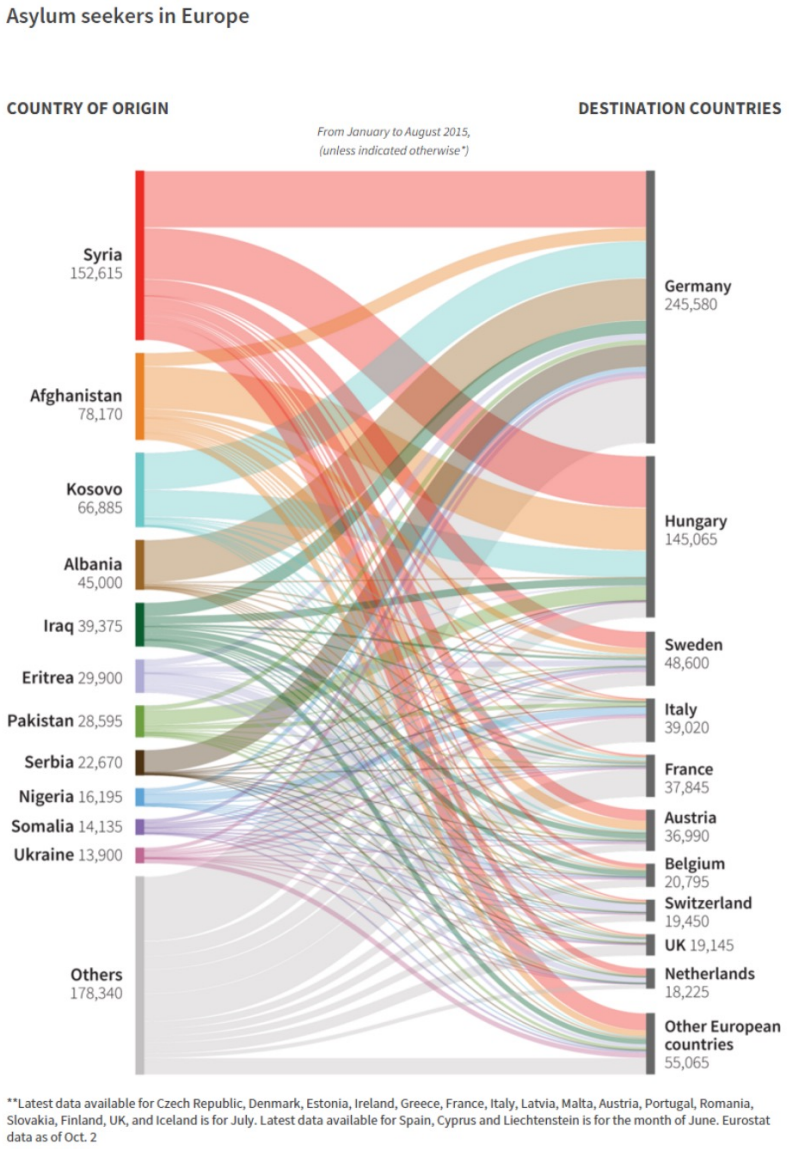


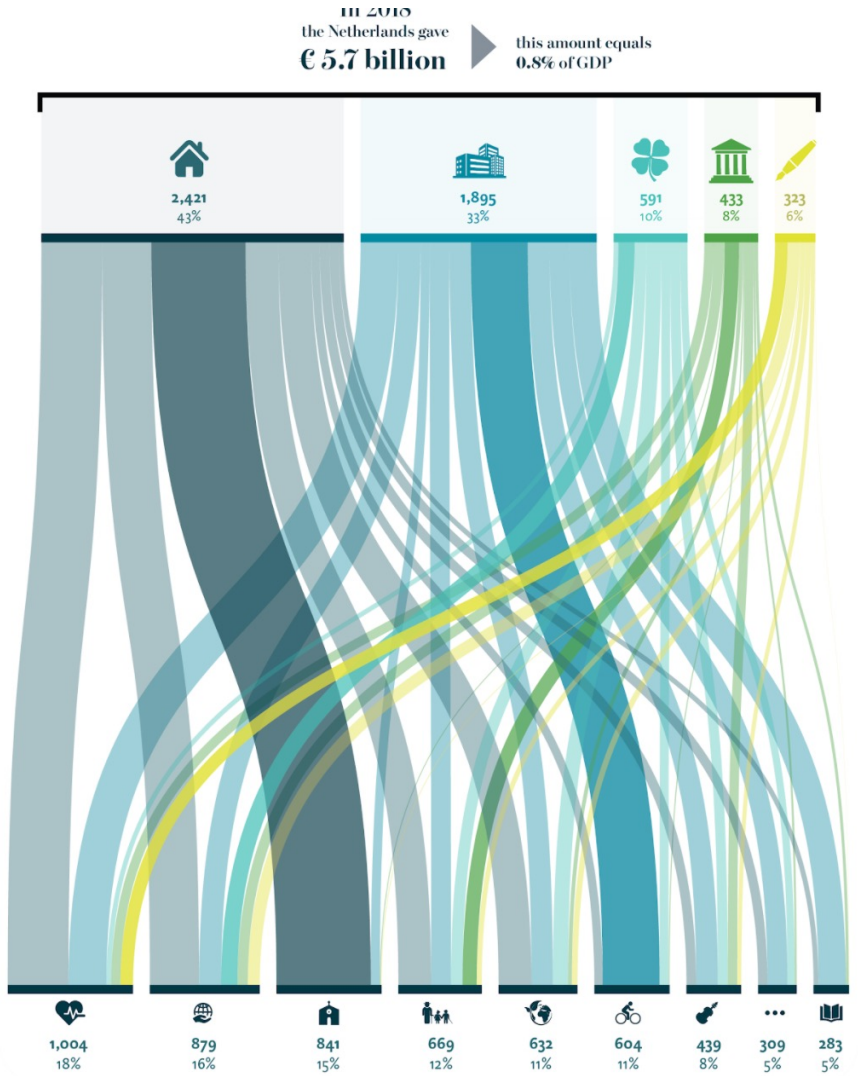
2014 Coal Exports and Imports

Sankey Diagrams: The value of a liberal arts education



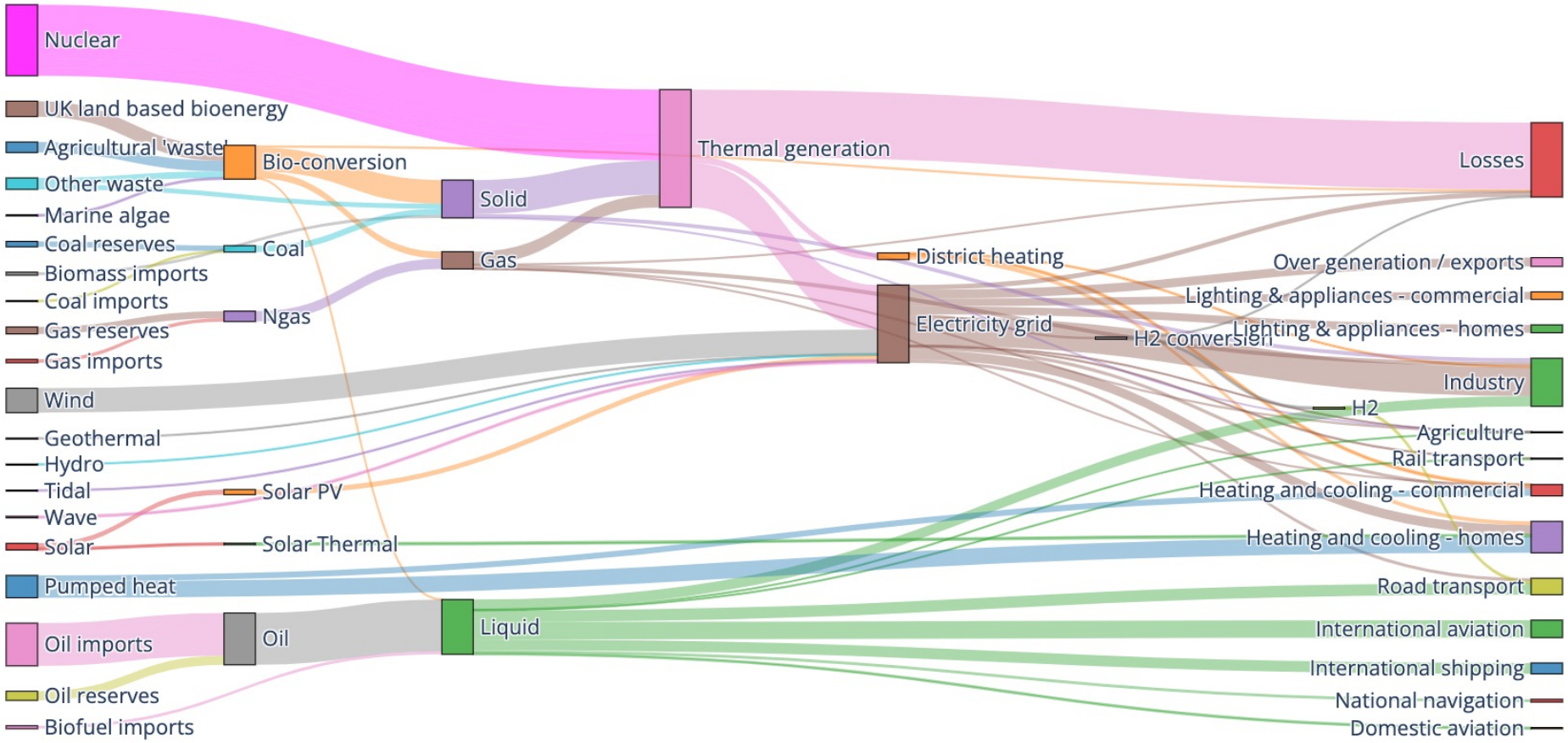
Sankey Diagrams: Migration Patterns



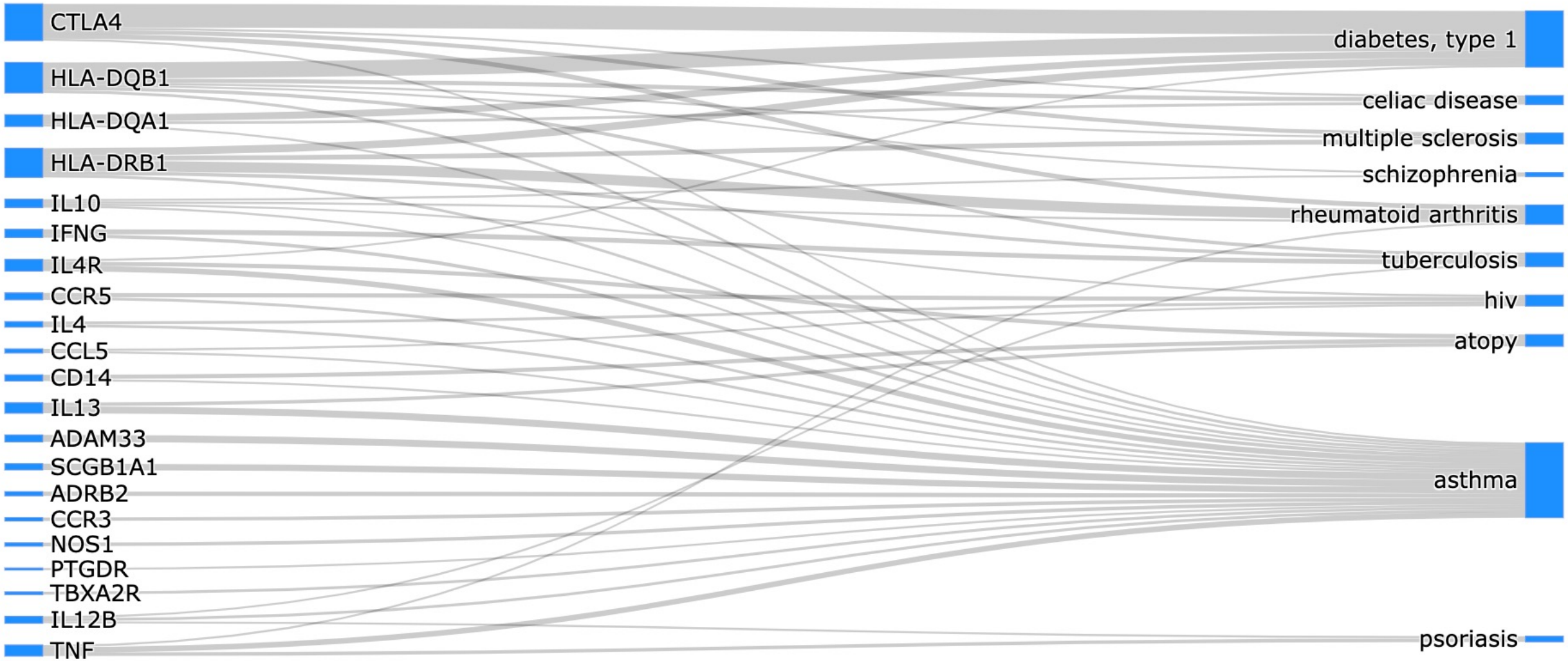


Energy Forecasting

Energy forecast for 2050
Source: Department of Energy & Climate Change, Tom Counsell via Mike Bostock



Sankey: The Genetic Association Database



Links between Asthma and Type 1 Diabetes

This Issue | Views 3,824 | Citations 5 | Altmetric 40 | Comments 1

PDF | More | Cite | Permissions

Original Investigation | Pediatrics

March 12, 2020

Familial Coaggregation of Asthma and Type 1 Diabetes in Children

Awad I. Smew, MD[1]; Cecilia Lundholm, MSc[1]; Lars Sävendahl, MD, PhD[2]; et al

Author Affiliations | Article Information

JAMA Netw Open. 2020;3(3):e200834. doi:10.1001/jamanetworkopen.2020.0834

Key Points | Español | 中文 (Chinese)

Question Is there an association between childhood asthma and type 1 diabetes and do shared familial factors contribute to the comorbidity?

Wolters Kluwer Medicine

Home Search Submit a Manuscript

Medicine (Baltimore). 2015 Sep; 94(36): e1466. PMCID: PMC46166
Published online 2015 Sep 11. PMID: 263567
doi: 10.1097/MD.0000000000001466

Type 1 Diabetes and Increased Risk of Subsequent Asthma

A Nationwide Population-Based Cohort Study

Yung-Tsung Hsiao, MD, Wen-Chien Cheng, MD, Wei-Chih Liao, MD, Cheng-Li Lin, MS
Te-Chun Shen, MD, Wei-Chun Chen, MD, Chia-Hung Chen, MD, and Chia-Hung Kao, MD

Monitoring Editor: Zhu. Xiaolin

Author information | Article notes | Copyright and License information | Disclaimer

This article has been cited by other articles in PMC.

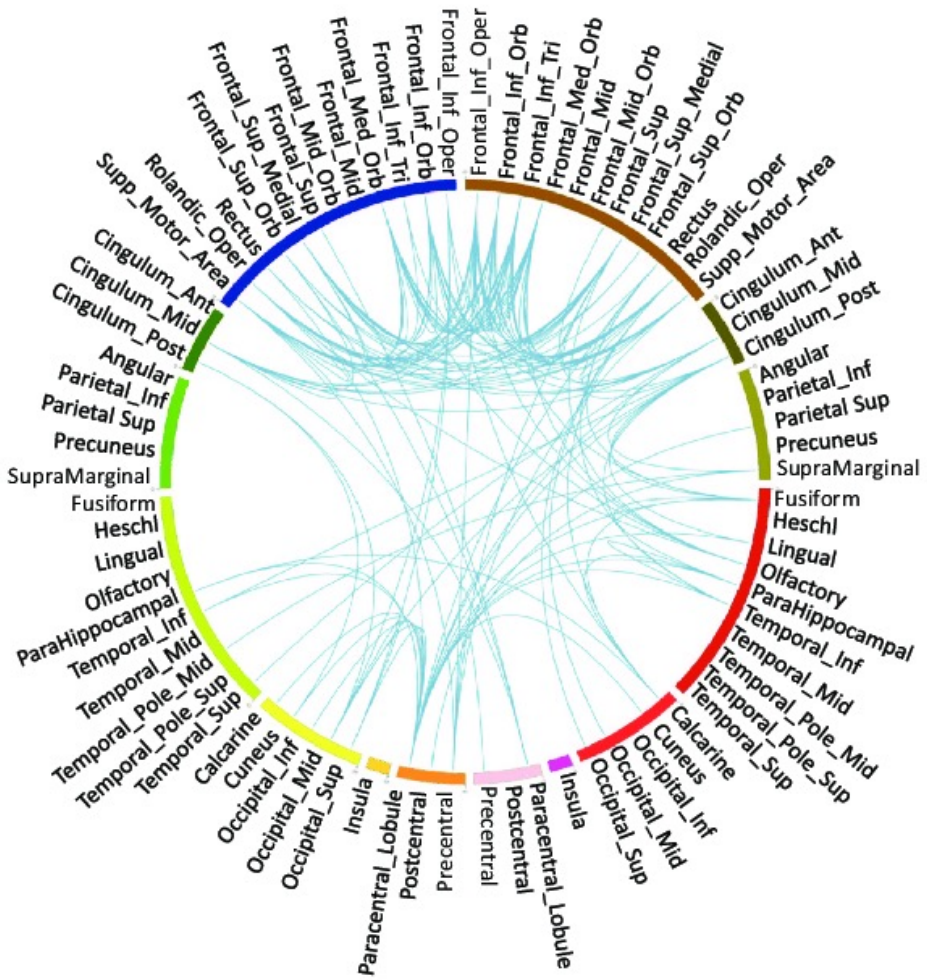
Abstract

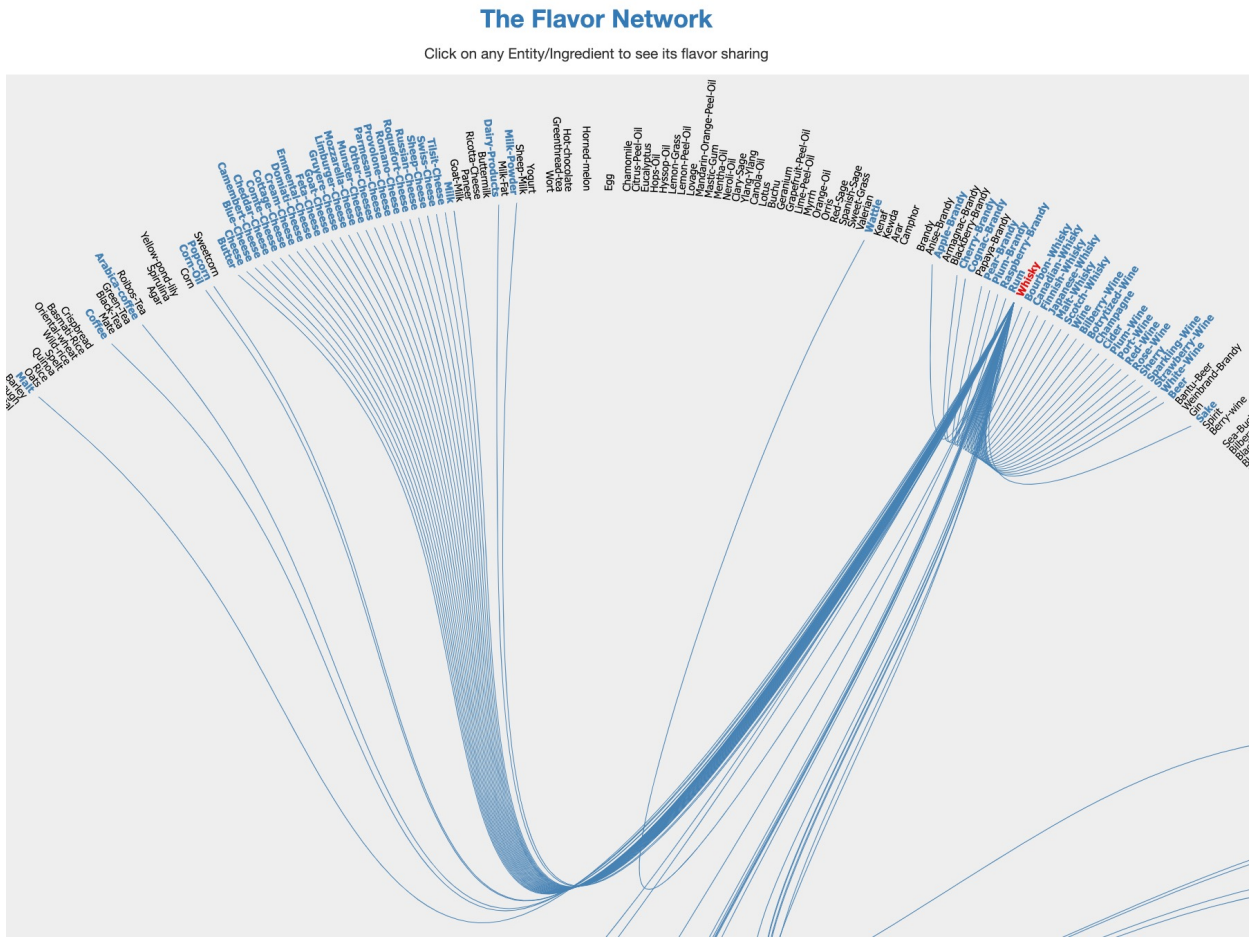
Search | Copy

Go to:

The association between type 1 diabetes mellitus (T1DM) and asthma remains controversial and has led to new interest in these 2 disorders. The purpose of this study was to examine the associations among young people with T1DM and asthma and offer a clinical demonstration of the balance between Th1 and Th2 responses.

Circle Graphs: Single-Set Sankey





Dashboarding for Interactive Visualization

Minimum citation(s):

2

Journal(s) published:

- PLoS One (961 publication(s))
- Viruses (436 publication(s))
- Sci Rep (261 publication(s))
- PLoS Pathog (251 publication(s))

Citation network:

Show citation connections

Dimensionality reduction algorithm

UMAP t-SNE

t-SNE parameters (not applicable to UMAP):

Current t-SNE perplexity: 40 (min: 10, max:100)

10 100

Title: Distinct Patterns Of Ifitm-Mediated Restriction Of Filoviruses, Sars Coronavirus, And Influenza A Virus

Journal: Plos Pathog, Published: 2011-01-06

Author(s): Huang, I-Chueh; Bailey, Charles C.; Weyer, Jessica L.; Radoshitzky, Sheli R.; Becker, Michelle M.; Chiang, Jessica J.; Brass, Abraham L.; Ahmed, Asim A.; Chi, Xiaoli; Dong, Lian; Longobardi, Lindsay E.; Boltz, Dutch; Kuhn, Jens H.; Elledge, Stephen J.; Bavari, Sina; Denison, Mark R.; Choe, Hyeryun; Farzan, Michael, Citations: 76

CORD-19 LDA Analysis

NLP AI ML Cytoscape t-SNE LDA COVID Healthcare Pharma

This app demonstrates the use of Dash Cytoscape and t-SNE to display the output of Latent Dirichlet Allocation (LDA) analysis on the CORD-19 dataset of literature related to COVID-19.
Get the source code for this app in Python!

Privacy

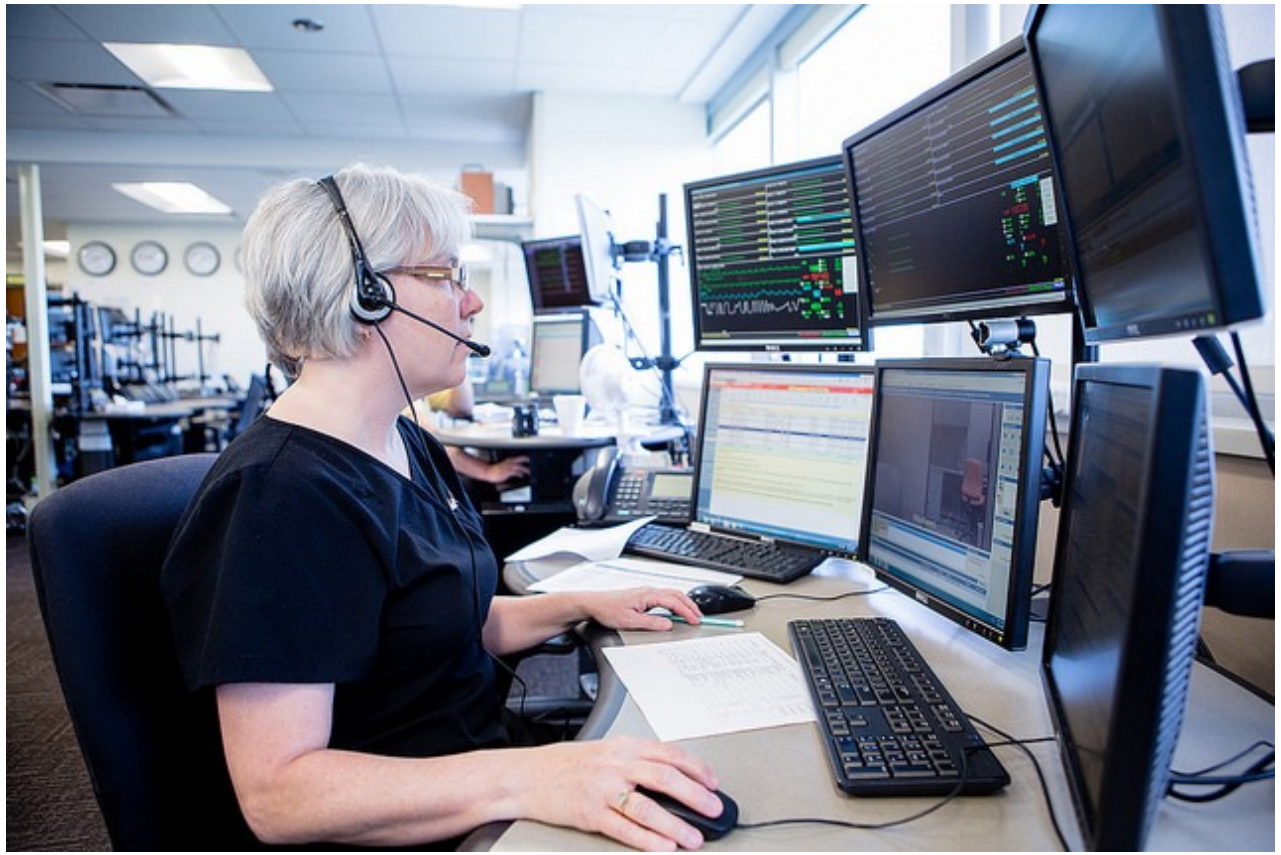
Unauthenticated

Anyone can access this app

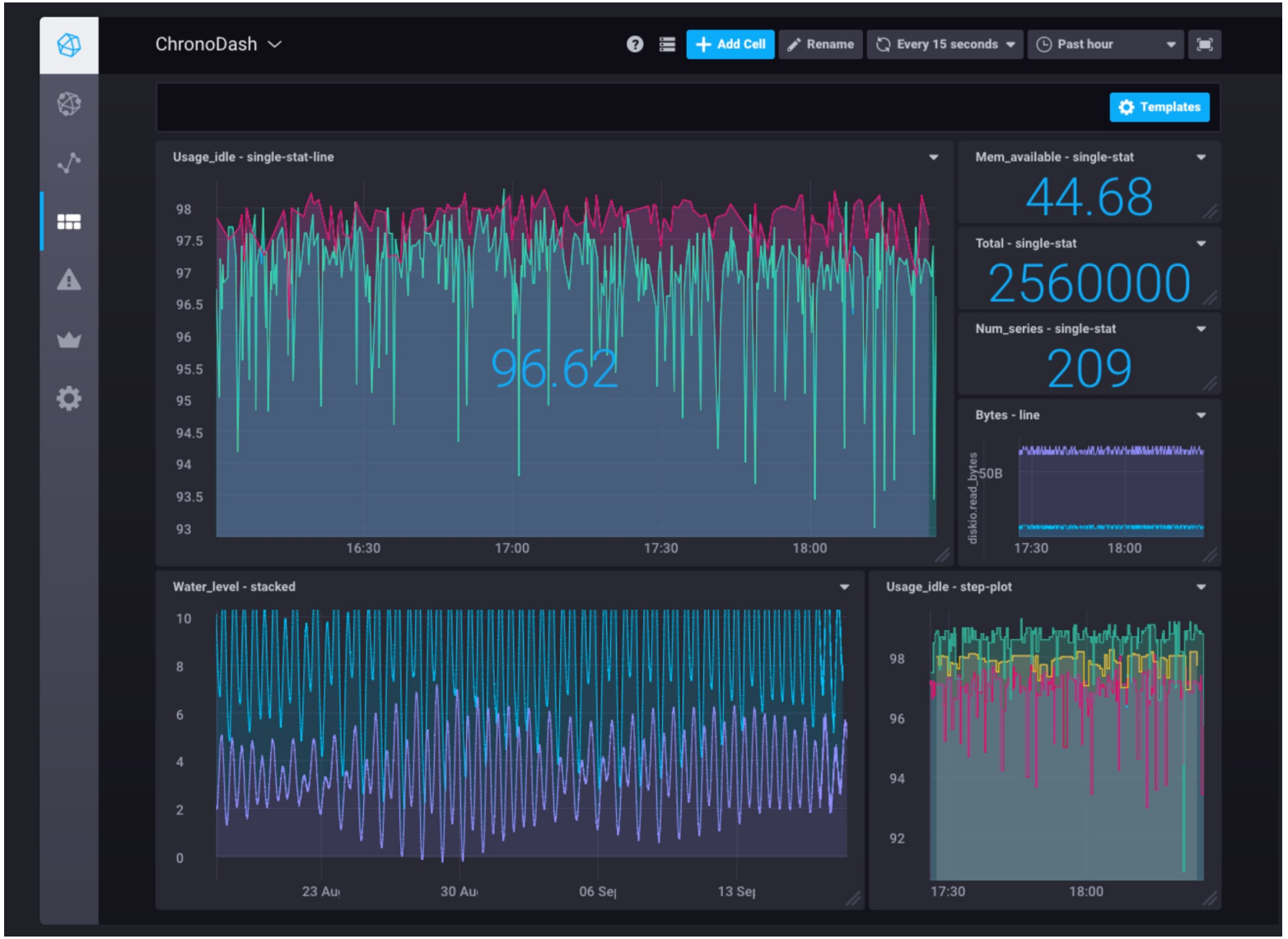
Open App

Monitoring Data Pipelines

- Application / service / network configuration errors
- Balance flexibility with restrictions aimed at encouraging users to do the right thing
- Sandbox testing/ Automated unit testing to identify edge cases
- Automated rollout / rollback of code changes (DevOps)
- Monitoring / Dashboards / Telemetry for error detection, performance tracking



Dashboards – cont.



$ conda install dash

```
The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    brotli-python-1.0.9        |   py39hfd1d529_7         736 KB  conda-forge
    dash-2.3.1                 |     pyhd8ed1ab_0         7.5 MB  conda-forge
    flask-compress-1.11        |     pyhd8ed1ab_0          14 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         8.3 MB

The following NEW packages will be INSTALLED:

  brotli-python      conda-forge/osx-64::brotli-python-1.0.9-py39hfd1d529_7
  dash               conda-forge/noarch::dash-2.3.1-pyhd8ed1ab_0
  flask-compress     conda-forge/noarch::flask-compress-1.11-pyhd8ed1ab_0
```

$ conda install jupyter-dash

If you prefer using Jupyter Notebooks or Jupyter Lab

```
The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ansi2html-1.7.0            |   py39h6e9494a_1          39 KB  conda-forge
    jupyter-dash-0.4.2         |     pyhd8ed1ab_1          20 KB  conda-forge
    retrying-1.3.3             |               py_2          11 KB  conda-forge
    ------------------------------------------------------------
                                           Total:          71 KB

The following NEW packages will be INSTALLED:

  ansi2html          conda-forge/osx-64::ansi2html-1.7.0-py39h6e9494a_1
  jupyter-dash       conda-forge/noarch::jupyter-dash-0.4.2-pyhd8ed1ab_1
  retrying           conda-forge/noarch::retrying-1.3.3-py_2
```

Dash installation verification: $ conda list dash

```
(base) [512 uranus:dash ] conda list dash
# packages in environment at /Users/rachlin/opt/anaconda3:
#
# Name                    Version                   Build  Channel
dash                      2.3.1              pyhd8ed1ab_0    conda-forge
jupyter-dash              0.4.2              pyhd8ed1ab_1    conda-forge
```

Also: **conda list plotly** (plotly and plotly express already installed)

```
(base) [513 uranus:dash ] condalist plotly
plotly                    5.5.0              pyhd8ed1ab_0    conda-forge
plotly_express            0.4.1                     py_0    conda-forge
```

Dash HTML Components

```python
from dash import html

html.Div([
    html.H1('Hello Dash'),
    html.Div([
        html.P('Dash converts Python classes into HTML'),
        html.P("This conversion happens behind the scenes by Dash's JavaScript front-end")
    ])
])
```

which gets converted (behind the scenes) into the following HTML in your web-app:

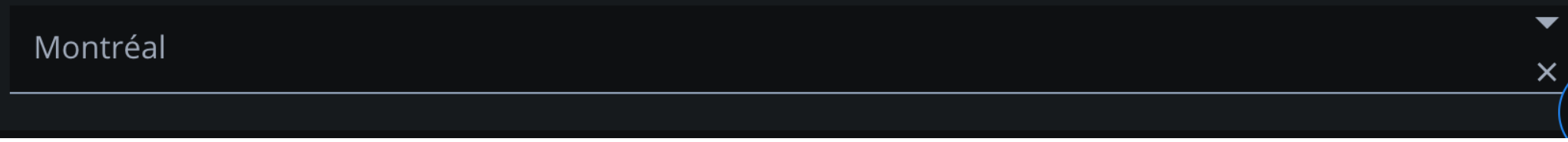
```html
<div>
    <h1>Hello Dash</h1>
    <div>
        <p>Dash converts Python classes into HTML</p>
        <p>This conversion happens behind the scenes by Dash's JavaScript front-end</p>
    </div>
</div>
```


Source: https://dash.plotly.com/dash-html-components

Dash Core Components

Dropdown

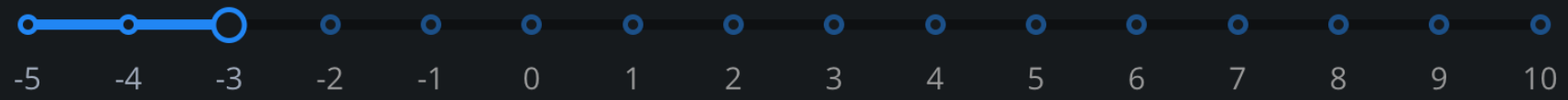
```
from dash import Dash, html, dcc

app = Dash(__name__)

app.layout = html.Div([
    dcc.Dropdown(['New York City', 'Montréal', 'San Francisco'], 'Montréal')
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



Dash Core Components → localhost:8050

Slider

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Slider(-5, 10, 1, value=-3)
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



Dash Core Components

Radio Items

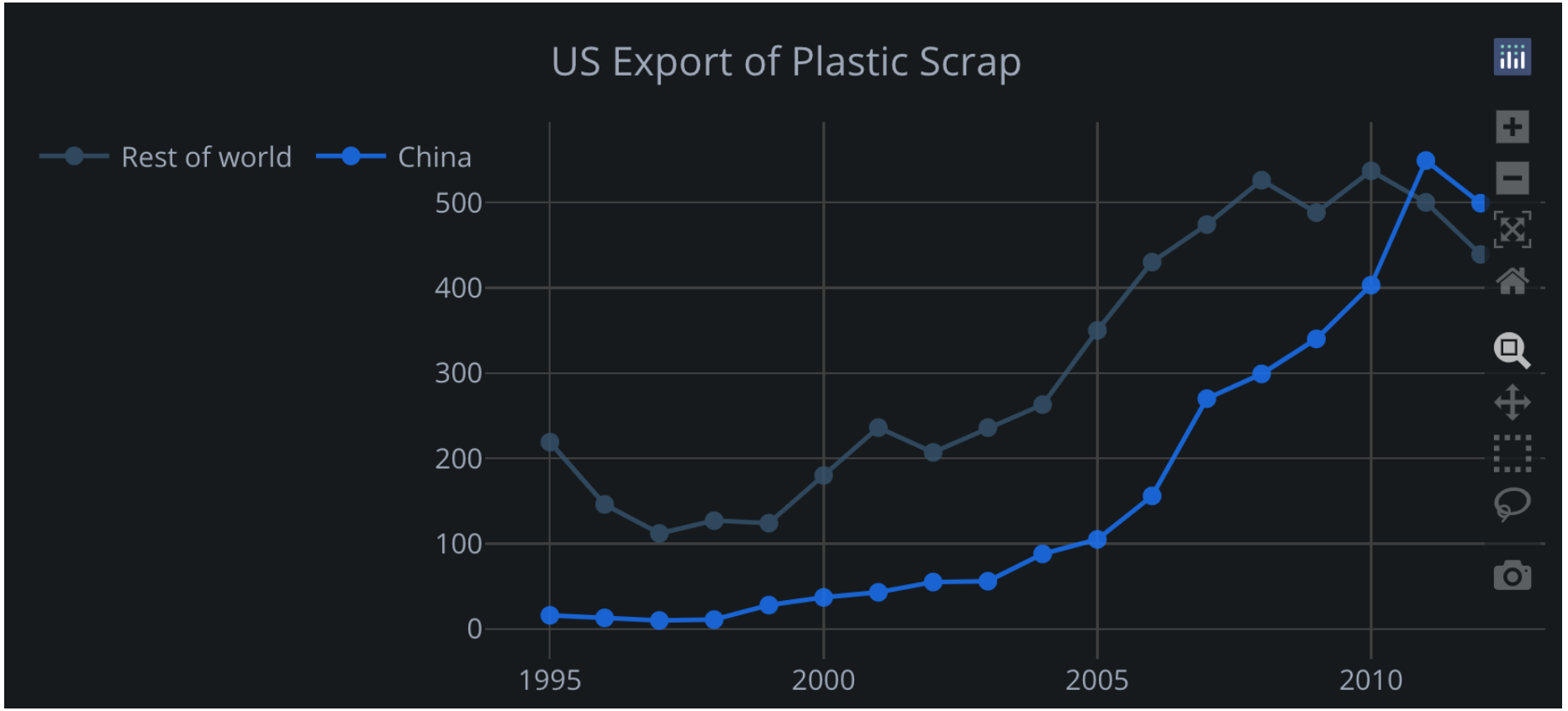
```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.RadioItems(['New York City', 'Montréal', 'San Francisco'], 'Montréal')
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

- ○ New York City
- ◉ Montréal
- ○ San Francisco

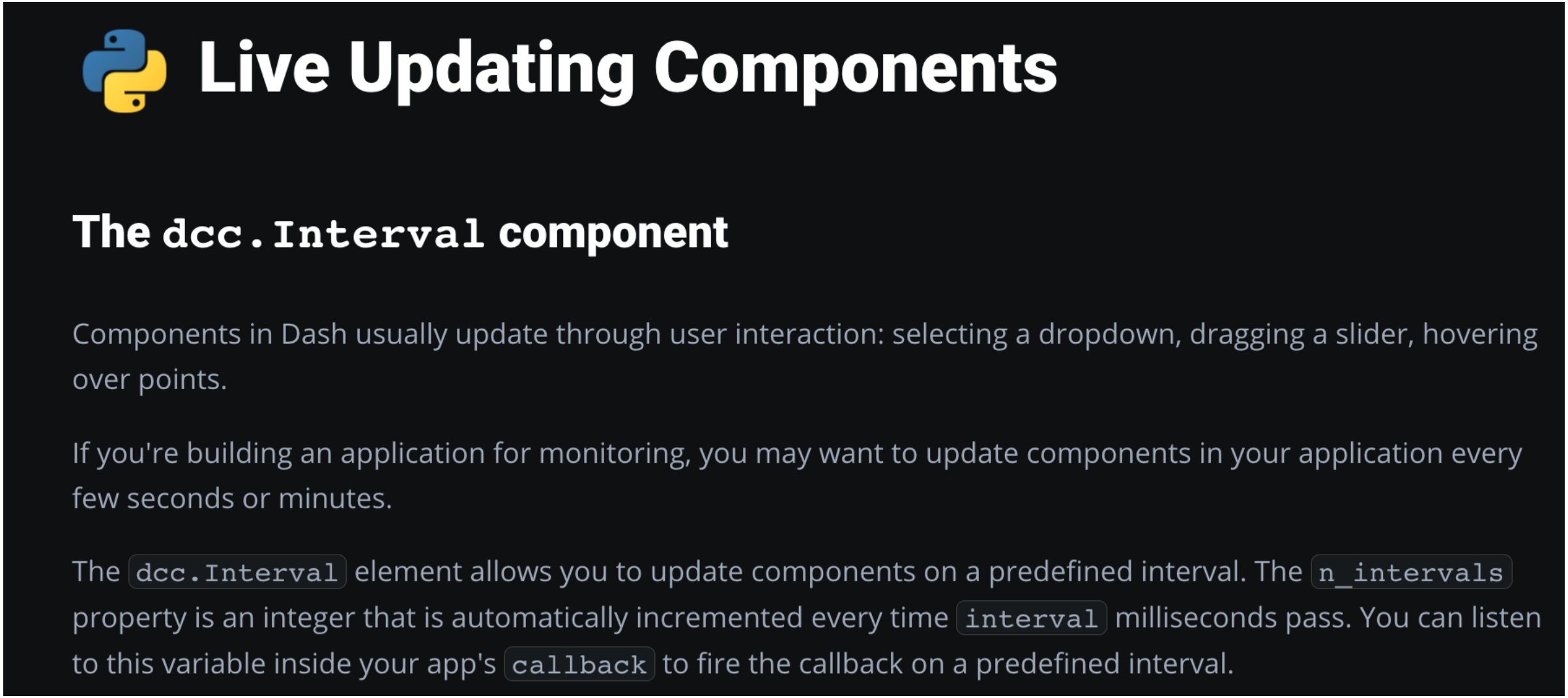
Graphs! (Including our friend, the SanKey Diagram)

Graphs

The `Graph` component shares the same syntax as the open-source `plotly.py` library. View the **plotly.py docs** to learn more.



Live Updates



Live Updating Components

The `dcc.Interval` component

Components in Dash usually update through user interaction: selecting a dropdown, dragging a slider, hovering over points.

If you're building an application for monitoring, you may want to update components in your application every few seconds or minutes.

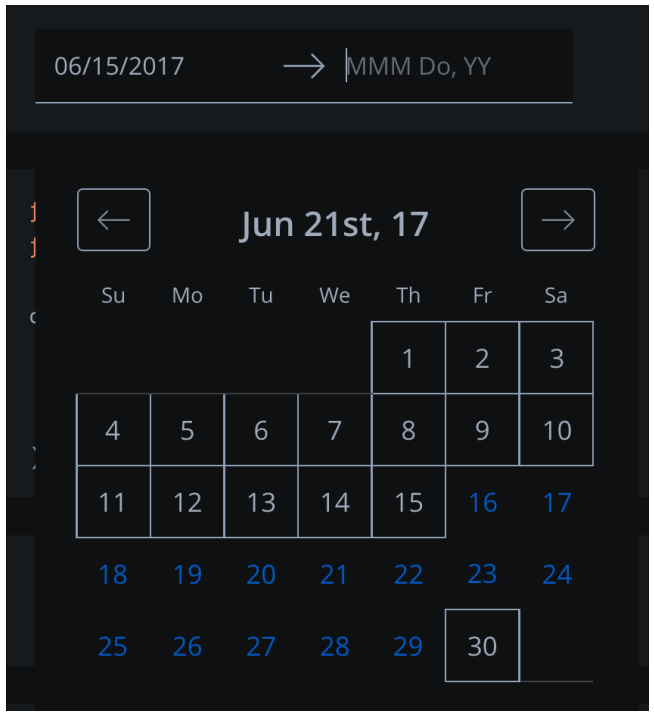
The `dcc.Interval` element allows you to update components on a predefined interval. The `n_intervals` property is an integer that is automatically incremented every time `interval` milliseconds pass. You can listen to this variable inside your app's `callback` to fire the callback on a predefined interval.

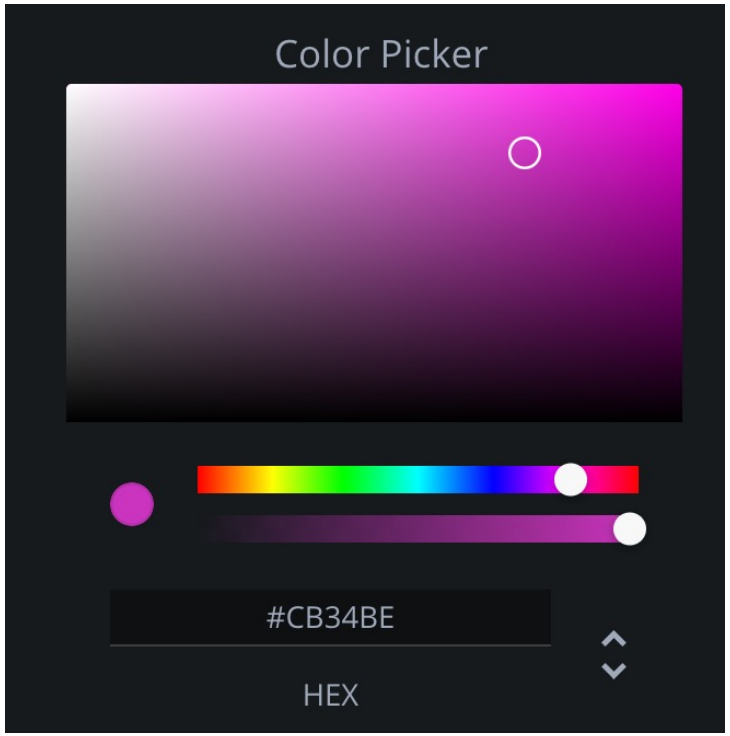
Dash Tables

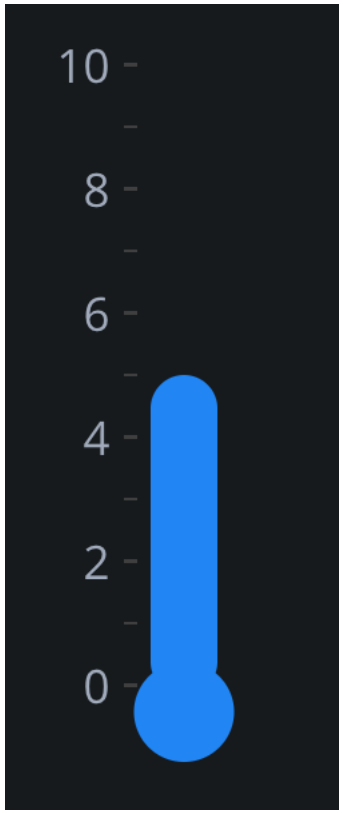
index	continent	country	gdpPercap	lifeExp	pop
1	Asia	Afghanistan	974.5803384	43.828	31889923
2	Europe	Albania	5937.029525999999	76.423	3600523
3	Africa	Algeria	6223.367465	72.301	33333216
4	Africa	Angola	4797.231267	42.731	12420476
5	Americas	Argentina	12779.37964	75.32	40301927

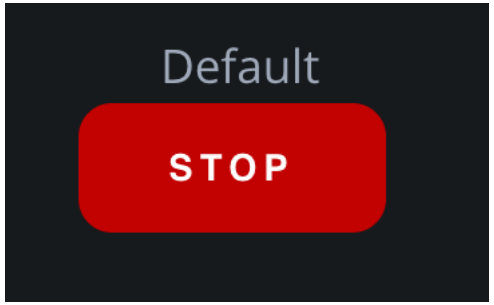
« ‹ 1 › »

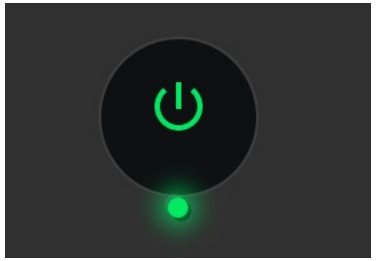
Dash DAQ: Web development framework









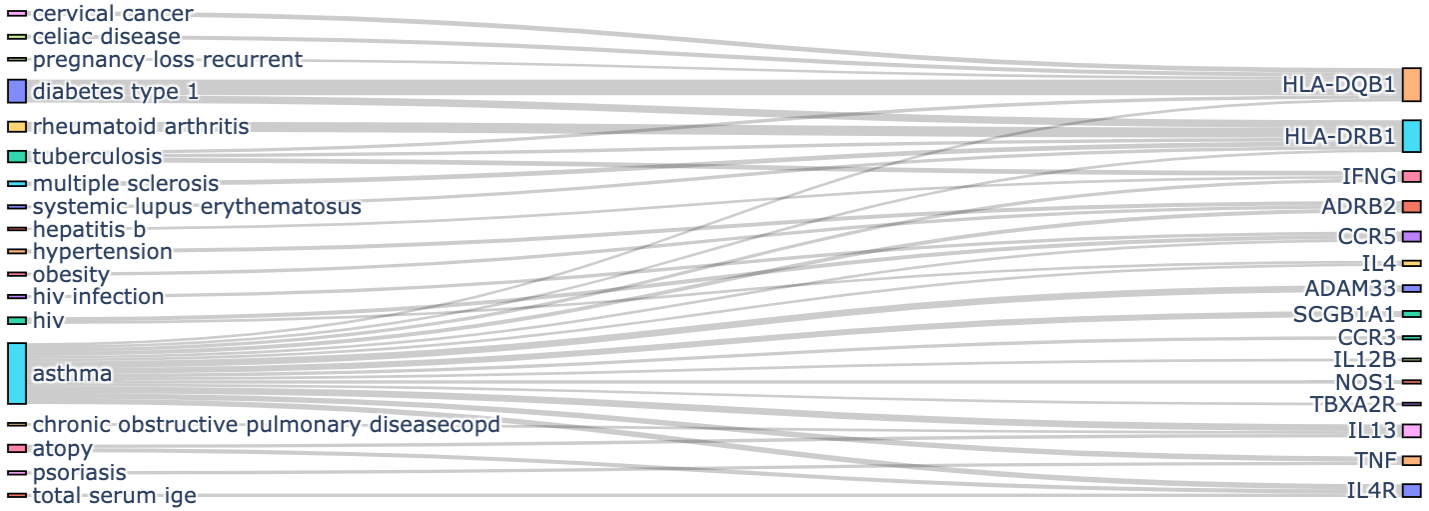


Callbacks

Callbacks make your dashboards interactive

```python
from dash import Dash, dcc, html, Input, Output

app = Dash(__name__)
app.layout = html.Div([
    dcc.Dropdown(['NYC', 'MTL', 'SF'], 'NYC', id='demo-dropdown'),
    html.Div(id='dd-output-container')
])


@app.callback(
    Output('dd-output-container', 'children'),
    Input('demo-dropdown', 'value')
)
def update_output(value):
    return f'You have selected {value}'


if __name__ == '__main__':
    app.run_server(debug=True)
```

Putting it all together…..



Select phenotype:

asthma

Minimum number of confirming publications

1 2 3 4 5 6 7 8 9 10

An interactive database-driven web application for visualizing gene-disease associations and identifying related phenotypes based on gene overlap.

Array Processing with Numpy

(DS2500 Review)

Types of Arrays

Type	Description	Advantage	Disadvantage	
Lists	Standard built-in python list collection	Simplicity Non-homogeneous	Slower performance Limited functionality	[[1, 2, 3], [4, 5, 6]]
ndarray (NumPy)	High-performance array module for scientific applications.	High performance Flexible operations	Homogeneous values Numeric indexing only	array([1,2,3])
Series (Pandas)	Enhanced one-dimensional array	Custom indexing Missing-value support Many capabilities	Complexity	Wally 87 Eva 100 Sam 94 dtype: int64
DataFrame (Pandas)	Enhanced two-dimensional array (e.g., “tables”)	Flexible support for manipulating tabular data: filtering, sorting	Complexity	A B C 0 10 True 5.4 1 15 False 8.8 2 13 False 6.6

Array operations

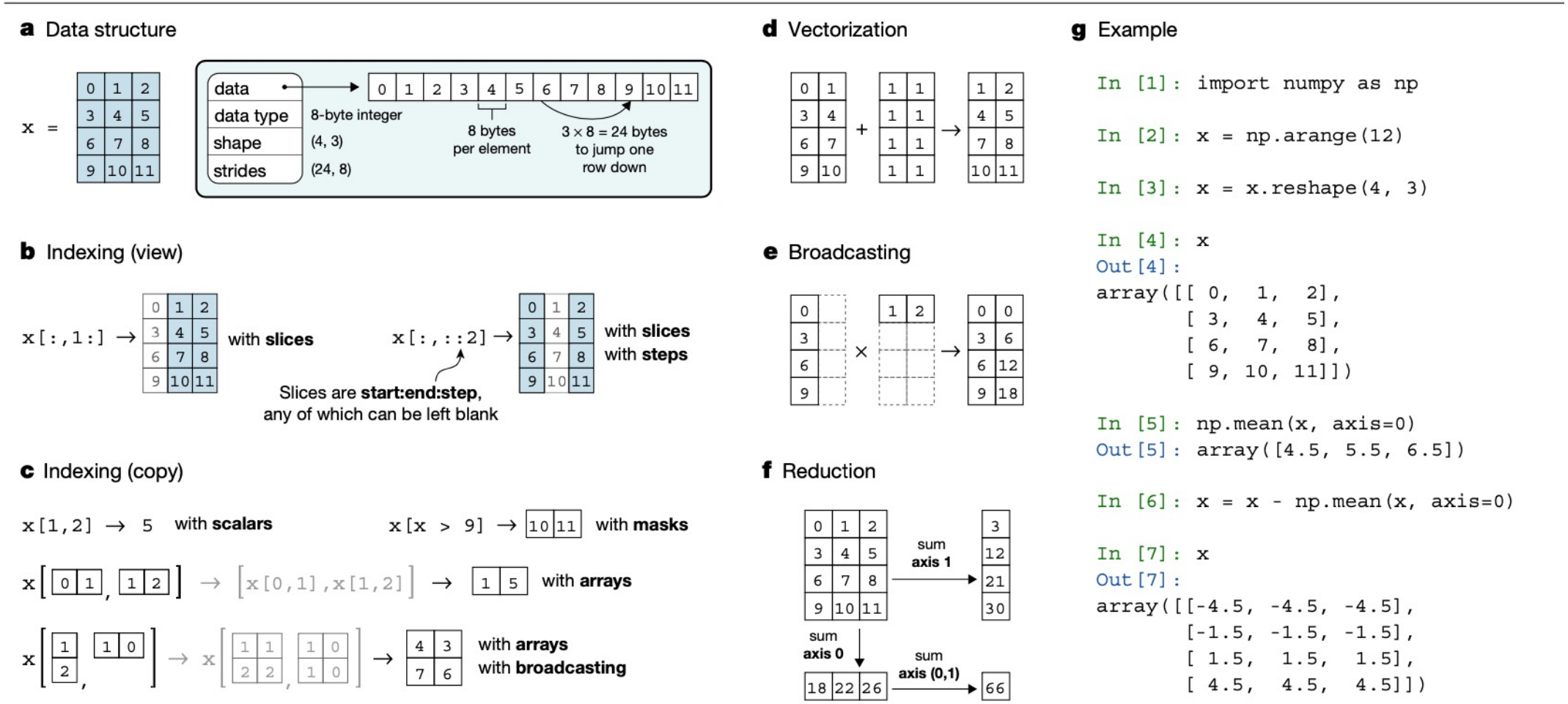


Fig. 1 | The NumPy array incorporates several fundamental array concepts. **a**, The NumPy array data structure and its associated metadata fields. **b**, Indexing an array with slices and steps. These operations return a 'view' of the original data. **c**, Indexing an array with masks, scalar coordinates or other arrays, so that it returns a 'copy' of the original data. In the bottom example, an array is indexed with other arrays; this broadcasts the indexing arguments before performing the lookup. **d**, Vectorization efficiently applies operations to groups of elements. **e**, Broadcasting in the multiplication of two-dimensional arrays. **f**, Reduction operations act along one or more axes. In this example, an array is summed along select axes to produce a vector, or along two axes consecutively to produce a scalar. **g**, Example NumPy code, illustrating some of these concepts.

NumPy is the base of the scientific Python ecosystem

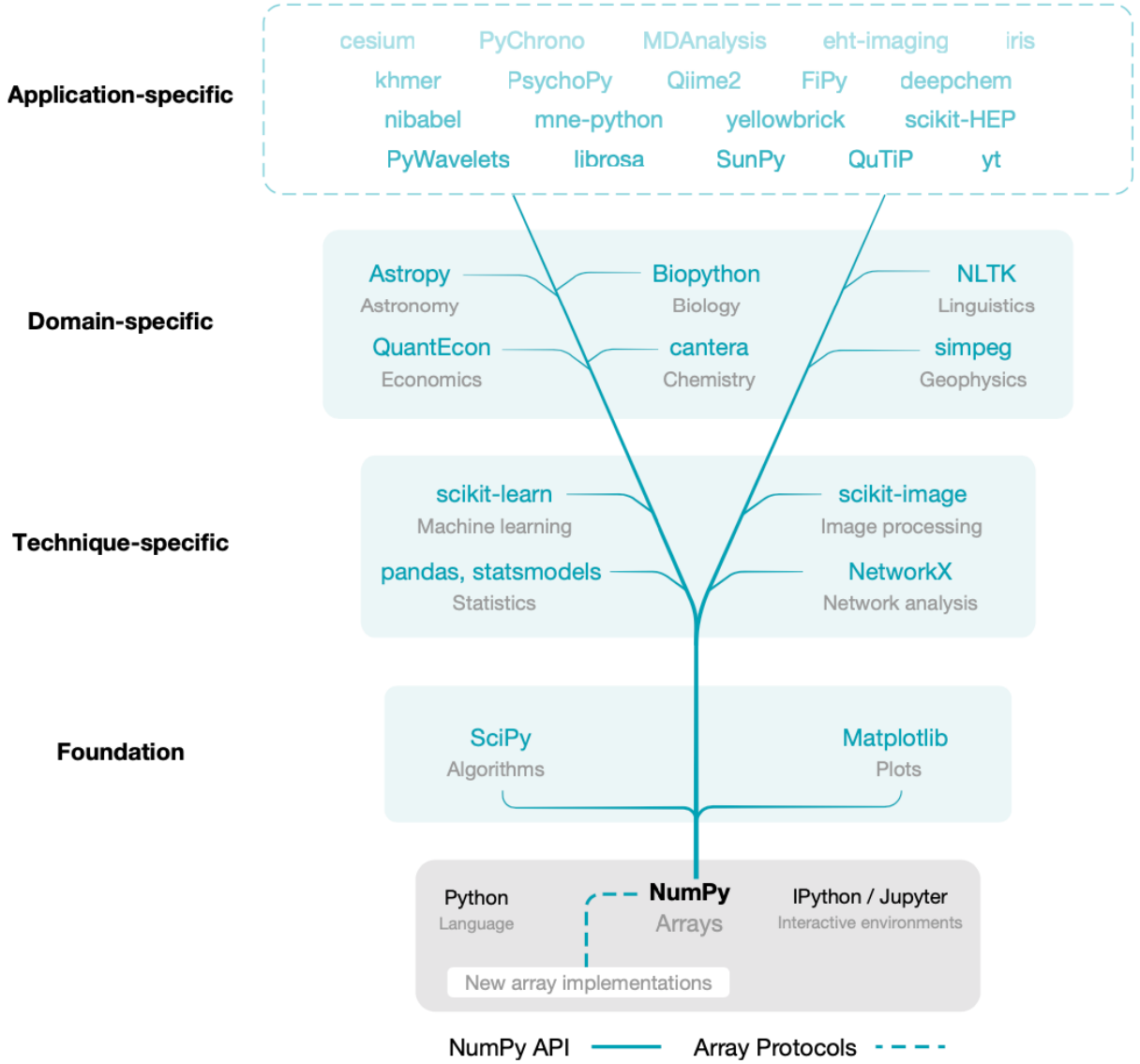


Fig. 2 | NumPy is the base of the scientific Python ecosystem. Essential libraries and projects that depend on NumPy's API gain access to new array implementations that support NumPy's array protocols (Fig. 3).

NumPy Image Analysis

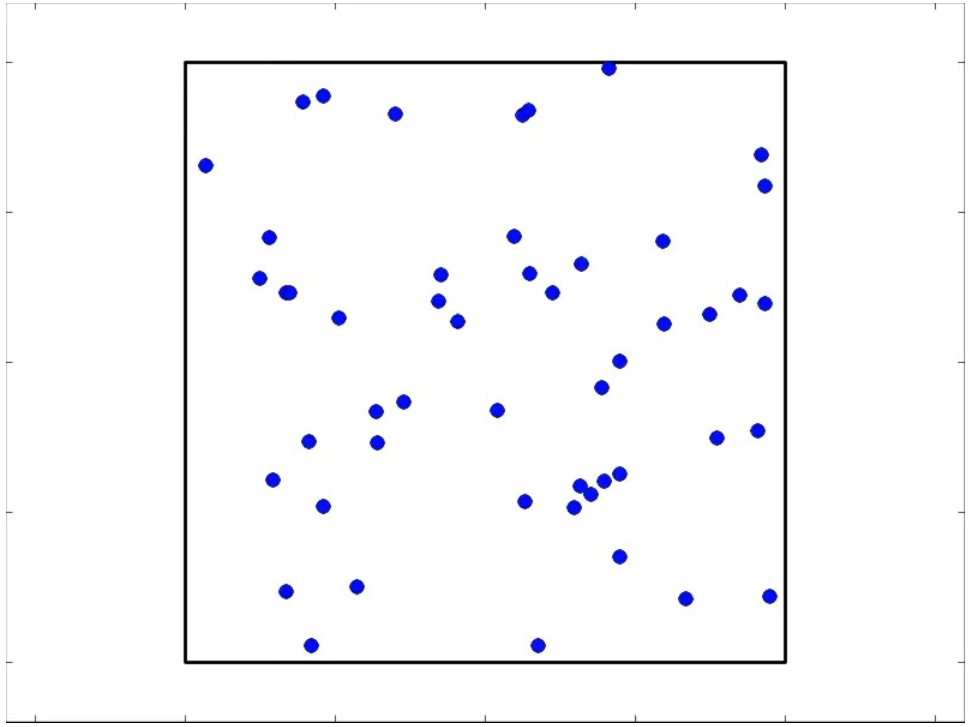
If the image is colored, then each pixel is represented by three numbers - a value for each of red, green, and blue. In that case we need a 3rd dimension (because each cell can only contain one number). So a colored image is represented by an ndarray of dimensions: (height x width x 3).

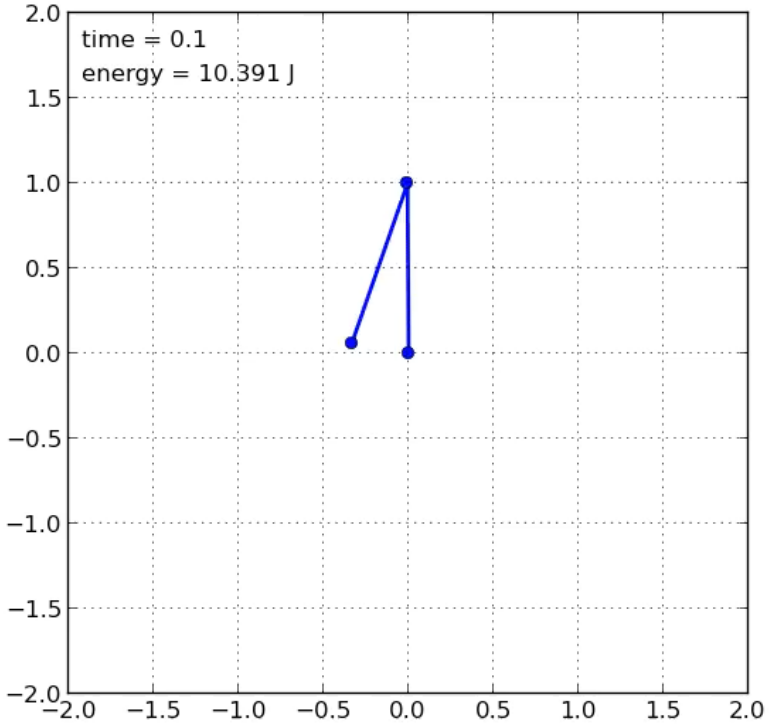
Source: https://jalammar.github.io/visual-numpy/

Animation

Created animated visuals using Matplotlib

Example Animations





Source: https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/

FuncAnimation

matplotlib.animation.FuncAnimation

class matplotlib.animation.**FuncAnimation**(*fig*, *func*, *frames=None*, *init_func=None*, *fargs=None*, *save_count=None*, *, *cache_frame_data=True*, **kwargs*) [source]

Makes an animation by repeatedly calling a function *func*.

Note

You must store the created Animation in a variable that lives as long as the animation should run. Otherwise, the Animation object will be garbage-collected and the animation stops.

fig : plt.figure()
func : The animation function – called each frame
frames : The number of frames
fargs : Additional arguments to func (if needed)
interval : Time (ms) between frames

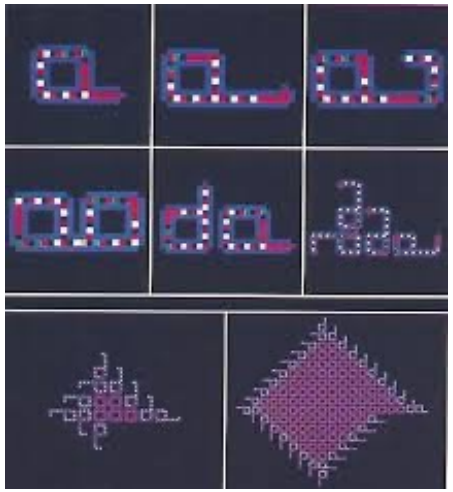
Artificial Life

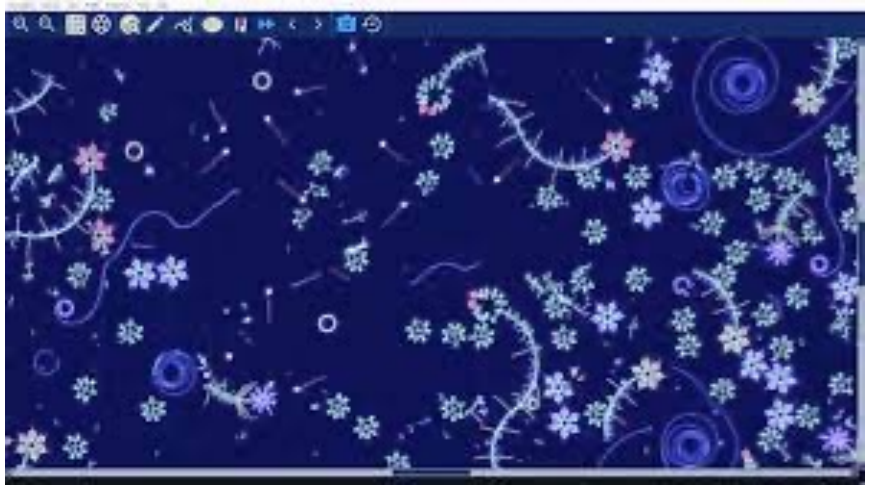
Life as it could be

Artificial Life Defined

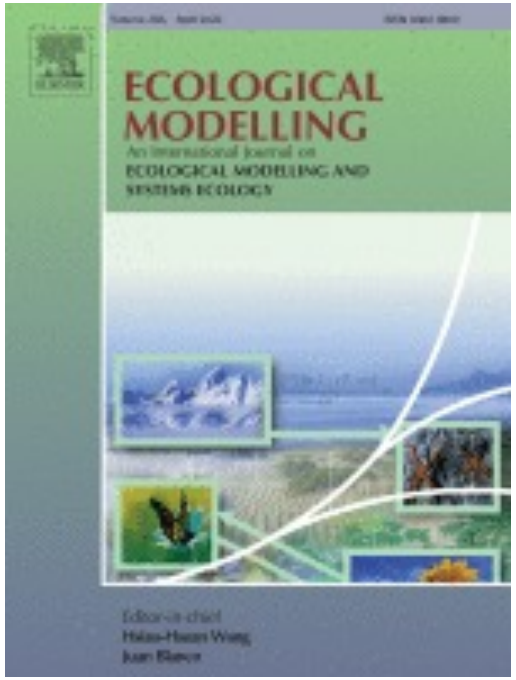
Artificial life (often abbreviated **ALife** or **A-Life**) is a field of study wherein researchers examine systems related to natural life, its processes, and its evolution, through the use of simulations with computer models, robotics, and biochemistry.

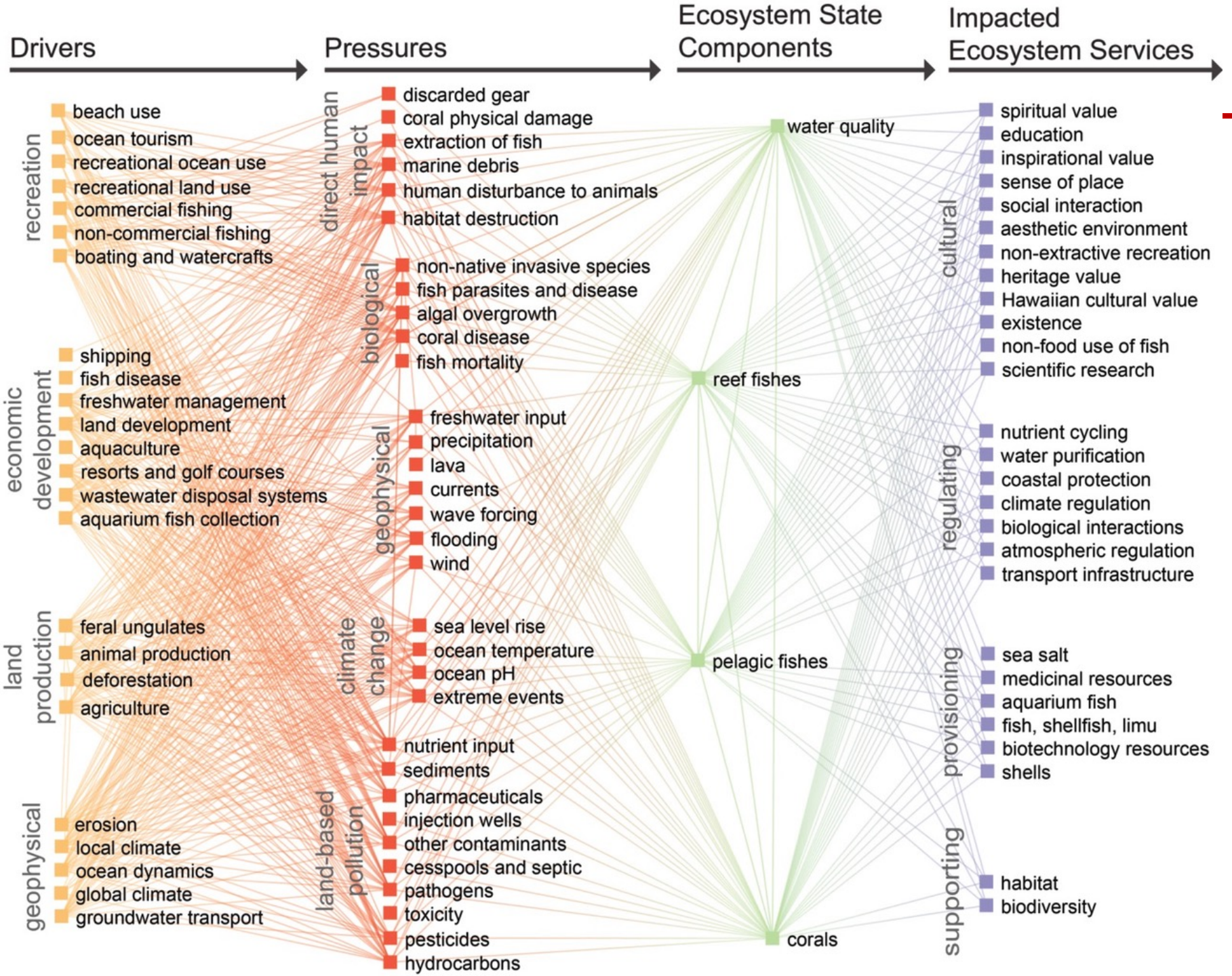
The discipline was named by Christopher Langton, an American theoretical biologist, in 1986.





Ecosystems


ECOLOGICAL
MODELLING

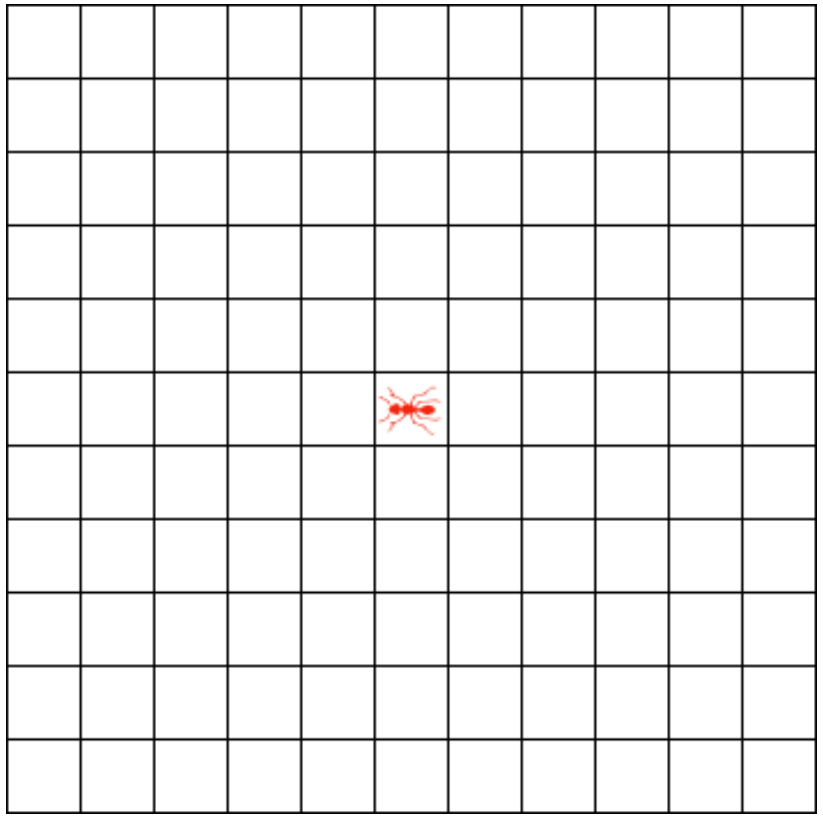

Drivers
Pressures
Ecosystem State Components
Impacted Ecosystem Services
recreation
beach use
ocean tourism
recreational ocean use
recreational land use
commercial fishing
non-commercial fishing
boating and watercrafts
economic development
shipping
fish disease
freshwater management
land development
aquaculture
resorts and golf courses
wastewater disposal systems
aquarium fish collection
land production
feral ungulates
animal production
deforestation
agriculture
geophysical
erosion
local climate
ocean dynamics
global climate
groundwater transport
direct human impact
discarded gear
coral physical damage
extraction of fish
marine debris
human disturbance to animals
habitat destruction
biological
non-native invasive species
fish parasites and disease
algal overgrowth
coral disease
fish mortality
geophysical
freshwater input
precipitation
lava
currents
wave forcing
flooding
wind
climate change
sea level rise
ocean temperature
ocean pH
extreme events
land-based pollution
nutrient input
sediments
pharmaceuticals
injection wells
other contaminants
cesspools and septic
pathogens
toxicity
pesticides
hydrocarbons
water quality
reef fishes
pelagic fishes
corals
cultural
spiritual value
education
inspirational value
sense of place
social interaction
aesthetic environment
non-extractive recreation
heritage value
Hawaiian cultural value
existence
non-food use of fish
scientific research
regulating
nutrient cycling
water purification
coastal protection
climate regulation
biological interactions
atmospheric regulation
transport infrastructure
provisioning
sea salt
medicinal resources
aquarium fish
fish, shellfish, limu
biotechnology resources
shells
supporting
habitat
biodiversity

Langton's Ants

Modeling ants responding to pheromone signals.

An example of how the repeated application of simple rules → complexity

White: Turn Right, **Red**, 1 Step. **Red**: Turn Left, **White**, 1 Step



Langton's ant after 11,000 steps. A red pixel shows the ant's location.