Python Workshop

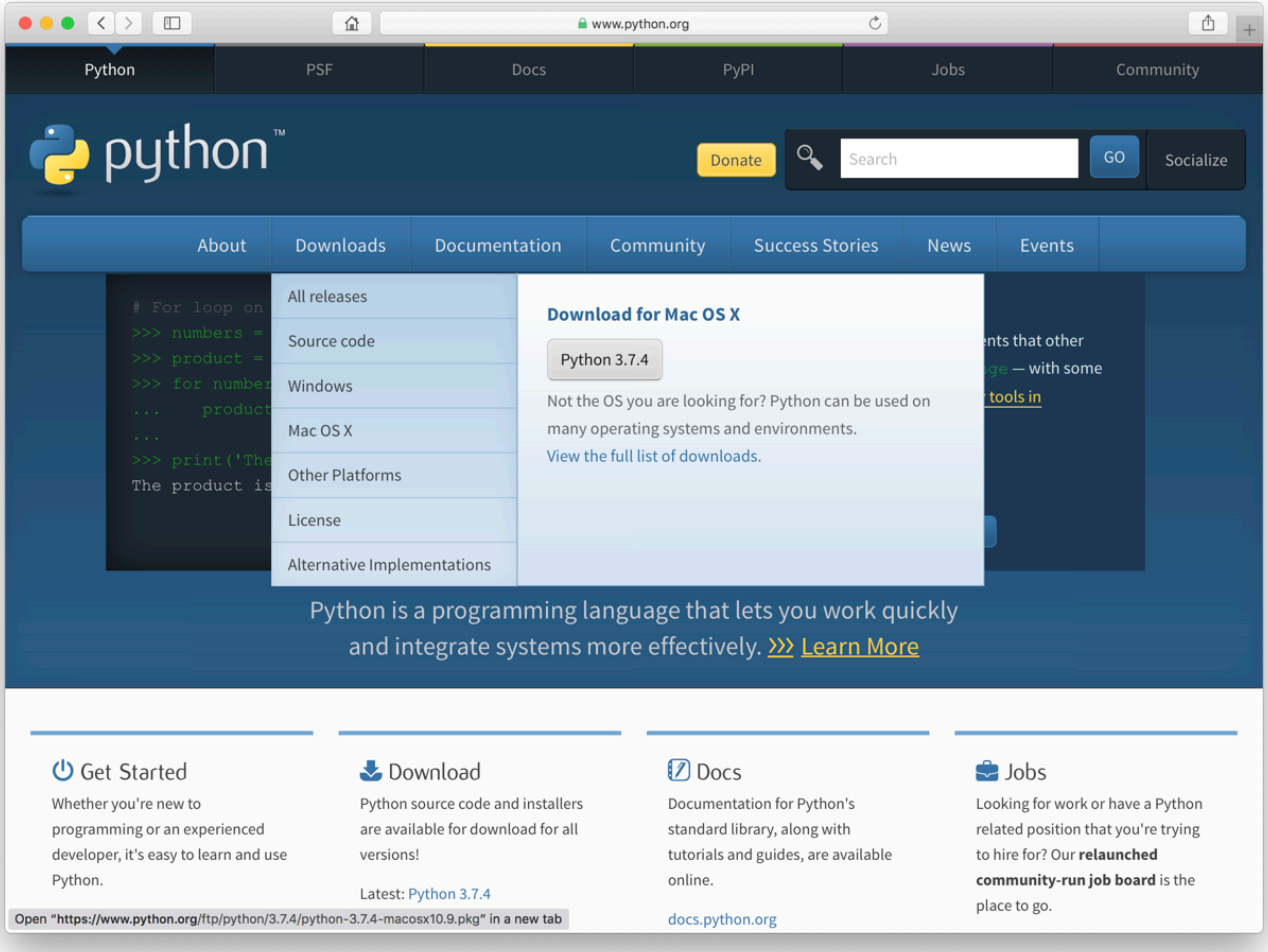
KnoxPy

September 5, 2019

Slides available at
github.com/knoxpy/presentations

Getting started

Download and install Python 3



```
# For loop on
>>> numbers =
>>> product =
>>> for number
...     product
...
>>> print('The
The product is
```

Download for Mac OS X

Not the OS you are looking for? Python can be used on many operating systems and environments.
View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Get Started

Whether you're new to programming or an experienced developer, it's easy to learn and use Python.

Download

Python source code and installers are available for download for all versions!

Latest: Python 3.7.4

Docs

Documentation for Python's standard library, along with tutorials and guides, are available online.

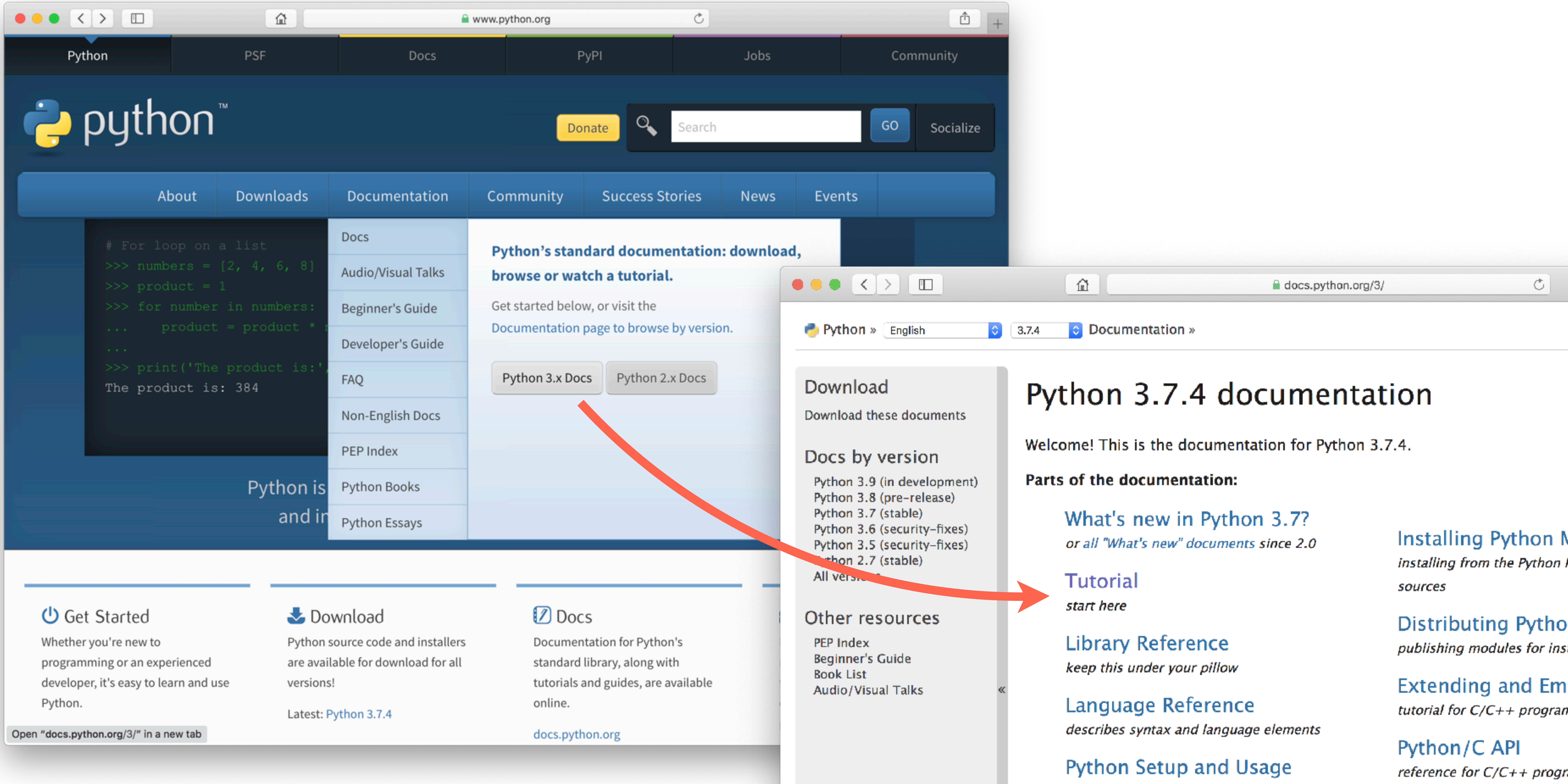
docs.python.org

Jobs

Looking for work or have a Python related position that you're trying to hire for? Our **relaunched community-run job board** is the place to go.

Open “https://www.python.org/ftp/python/3.7.4/python-3.7.4-macosx10.9.pkg” in a new tab

Python tutorial



```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * n
...
>>> print('The product is:',
The product is: 384
```

Python is
and in

Python's standard documentation: download, browse or watch a tutorial.

Get started below, or visit the Documentation page to browse by version.

Whether you're new to programming or an experienced developer, it's easy to learn and use Python.

Python source code and installers are available for download for all versions!

Latest: Python 3.7.4

Documentation for Python's standard library, along with tutorials and guides, are available online.

docs.python.org

Open "docs.python.org/3/" in a new tab

Download

Download these documents

Docs by version

Python 3.9 (in development)
Python 3.8 (pre-release)
Python 3.7 (stable)
Python 3.6 (security-fixes)
Python 3.5 (security-fixes)
Python 2.7 (stable)
All versions

Other resources

PEP Index
Beginner's Guide
Book List
Audio/Visual Talks

Python 3.7.4 documentation

Welcome! This is the documentation for Python 3.7.4.

Parts of the documentation:

What's new in Python 3.7?
or all "What's new" documents since 2.0

Tutorial
start here

Library Reference
keep this under your pillow

Language Reference
describes syntax and language elements

Python Setup and Usage

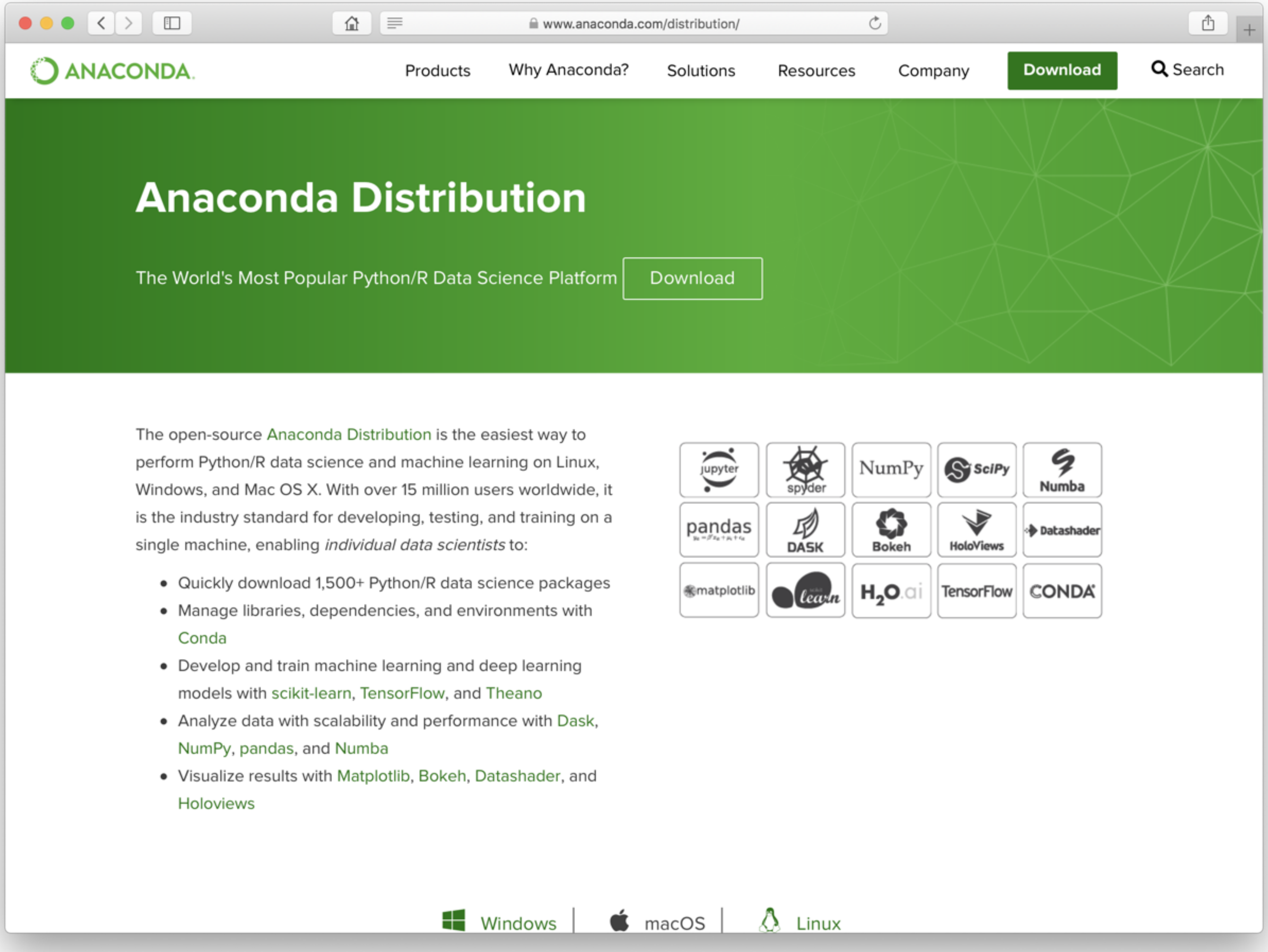
Installing Python M
installing from the Python I sources

Distributing Pytho
publishing modules for ins

Extending and Em
tutorial for C/C++ program

Python/C API
reference for C/C++ progr

Python for science



Anaconda Distribution

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

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

Python programming language

Prompts, comments, and variables

Command prompt input is represented by **>>>**

Command prompt output is represented by absence of **>>>**

```
>>> 2 + 2
4
>>> 50 - 5*6
20
```

Comments in Python begin with the **#** character

```
# this is the first comment
spam = 1  # and this is the second comment
          # ... and now a third!
text = "# This is not a comment because it's inside quotes."
```

The equal sign **=** is used to assign a value to a variable

```
>>> width = 20
>>> height = 5 * 9
>>> width * height
900
```

Numbers

```
>>> 2 + 2
4

>>> 50 - 5*6
20

>>> (50 - 5*6) / 4
5.0

>>> 8 / 5       # division always returns a floating point number
1.6

>>> 10 / 3      # classic division returns a float
3.33333335

>>> 17 // 3     # floor division discards the fractional part
5

>>> 17 % 3      # the % operator returns the remainder of the division
2
```

```
>>> 5 * 3 + 2   # result * divisor + remainder
17

>>> 5 ** 2      # 5 squared
25

>>> 2 ** 7      # 2 to the power of 7
128

>>> 4 * 3.75 - 1
14.0
```

Strings

```
>>> 'spam eggs'     # single quotes
'spam eggs'

>>> 'doesn\'t'      # use \' to escape the single quote...
"doesn't"

>>> "doesn't"       # ...or use double quotes instead
"doesn't"

>>> '"Yes," they said.'
'"Yes," they said.'

>>> "\"Yes,\" they said."
'"Yes," they said.'

>>> '"Isn\'t," they said.'
'"Isn\'t," they said.'

>>> 3 * 'un' + 'ium'    # 3 times 'un', followed by 'ium'
'unununium'

>>> 'Py' 'thon'         # two or more string literals automatically combine
'Python'

>>> prefix + 'thon'     # concatenate variables or a variable and a literal
'Python'
```

Lists

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]

>>> squares[0]  # indexing returns the item
1

>>> squares[-1]
25

>>> squares[-3:]  # slicing returns a new list
[9, 16, 25]

>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

>>> cubes = [1, 8, 27, 65, 125]
>>> cubes[3] = 64
>>> cubes
[1, 8, 27, 64, 125]

>>> letters = ['a', 'b', 'c', 'd']
>>> len(letters)
4
```

Tuples

```
>>> t = 12345, 54321, 'hello!'

>>> t[0]
12345

>>> t
(12345, 54321, 'hello!')

>>> u = t, (1, 2, 3, 4, 5)  # tuples may be nested
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))

>>> t[0] = 88888            # tuples are immutable
TypeError: 'tuple' object does not support item assignment

>>> empty = ()              # construct an empty tuple
>>> len(empty)
0

>>> singleton = 'hello',    # construct a one item tuple
>>> len(singleton)
1
>>> singleton
('hello',)

>>> t = 12345, 54321, 'hello!'  # tuple packing
>>> x, y, z = t                 # tuple unpacking
```

Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'jack': 4098, 'sape': 4139, 'guido': 4127}

>>> tel['jack']
4098

>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'jack': 4098, 'guido': 4127, 'irv': 4127}

>>> list(tel)
['jack', 'guido', 'irv']

>>> sorted(tel)
['guido', 'irv', 'jack']

>>> 'guido' in tel
True

>>> 'jack' not in tel
False
```

Sets

```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
>>> basket
{'orange', 'banana', 'pear', 'apple'}   # duplicates have been removed

>>> 'orange' in basket                  # fast membership testing
True
>>> 'crabgrass' in basket
False

>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                                  # unique letters in a
{'a', 'r', 'b', 'c', 'd'}

>>> a - b                              # letters in a but not in b
{'r', 'd', 'b'}

>>> a | b                              # letters in a or b or both
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}

>>> a & b                              # letters in both a and b
{'a', 'c'}

>>> a ^ b                              # letters in a or b but not both
{'r', 'd', 'b', 'm', 'z', 'l'}
```

Control flow with “if” and “for” statements

```
>>> x = 42

>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')

More
```

```
>>> words = ['cat', 'window', 'defenestrate']

>>> for w in words:
...     print(w, len(w))

cat 3
window 6
defenestrate 12
```

```
>>> for w in words[:]:  # Loop over a slice copy of the entire list.
...     if len(w) > 6:
...         words.insert(0, w)

>>> words
['defenestrate', 'cat', 'window', 'defenestrate']
```

```
>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print(i, a[i])

0 Mary
1 had
2 a
3 little
4 lamb
```

Functions

```
def fib2(n):
    """
    Return a list containing the Fibonacci
    series up to n.
    """
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result

>>> f100 = fib2(100)    # call it

>>> f100                # write the result
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)

>>> ask_ok('Do you really want to quit?')

>>> ask_ok('OK to overwrite the file?', 2)

>>> ask_ok('OK to overwrite the file?', 2, 'Come on, only yes or no!')
```

Classes

```
class Dog:

    def __init__(self, name):
        self.name = name
        self.tricks = []    # creates a new empty list for each dog

    def add_trick(self, trick):
        self.tricks.append(trick)

>>> d = Dog('Fido')

>>> e = Dog('Buddy')

>>> d.add_trick('roll over')

>>> e.add_trick('play dead')

>>> d.tricks
['roll over']

>>> e.tricks
['play dead']
```

Modules

fibo.py

```python
# Fibonacci numbers module
# This code is saved into a file named fibo.py

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

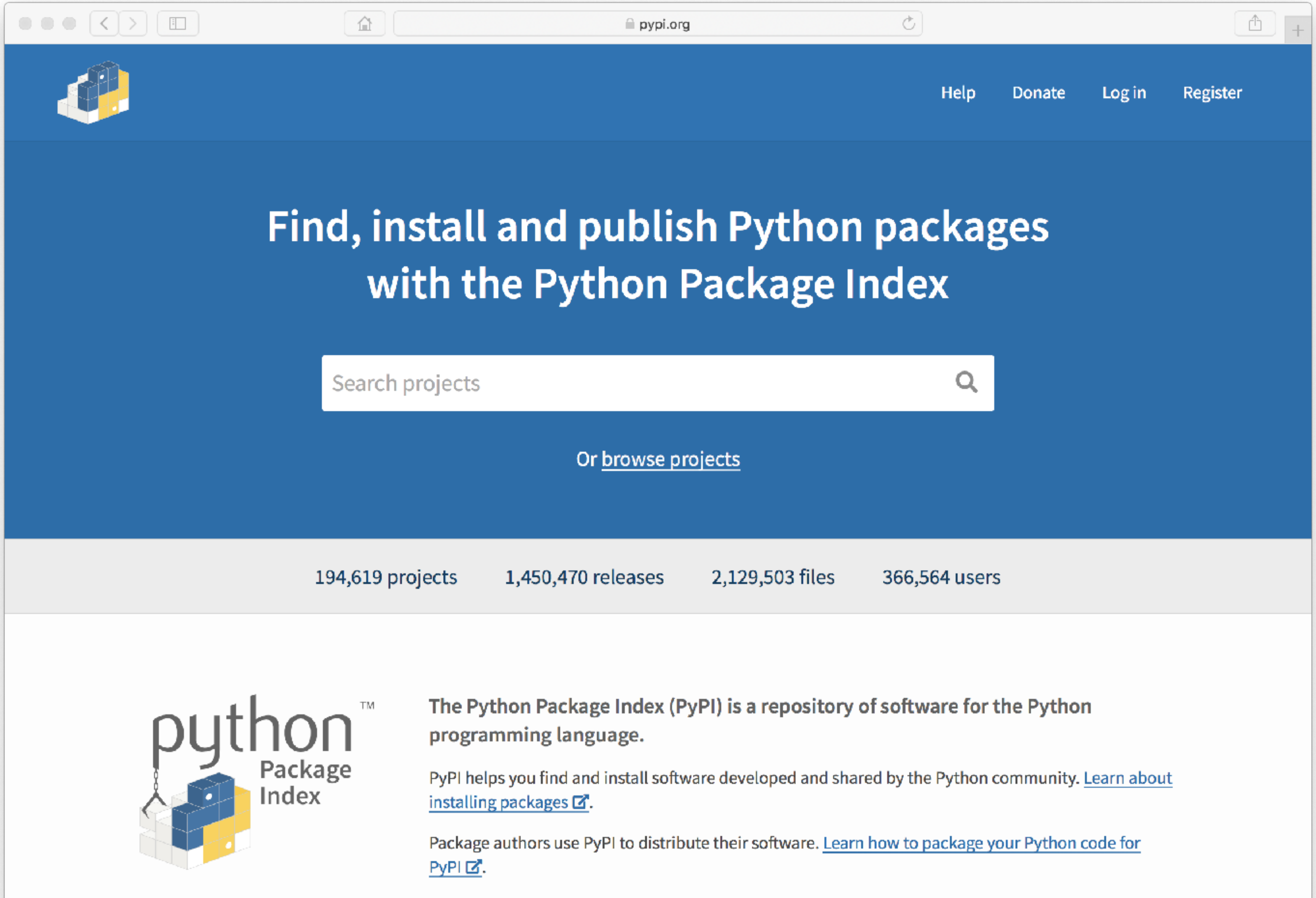
Python interpreter

```
>>> import fibo

>>> fibo.fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

>>> fibo.fib2(100)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Python Package Index (PyPI)



Find, install and publish Python packages with the Python Package Index

Or browse projects

194,619 projects | 1,450,470 releases | 2,129,503 files | 366,564 users

The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages.

Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

Resources

IDEs and Text Editors

PyCharm
https://www.jetbrains.com/pycharm/

Sublime Text
https://www.sublimetext.com

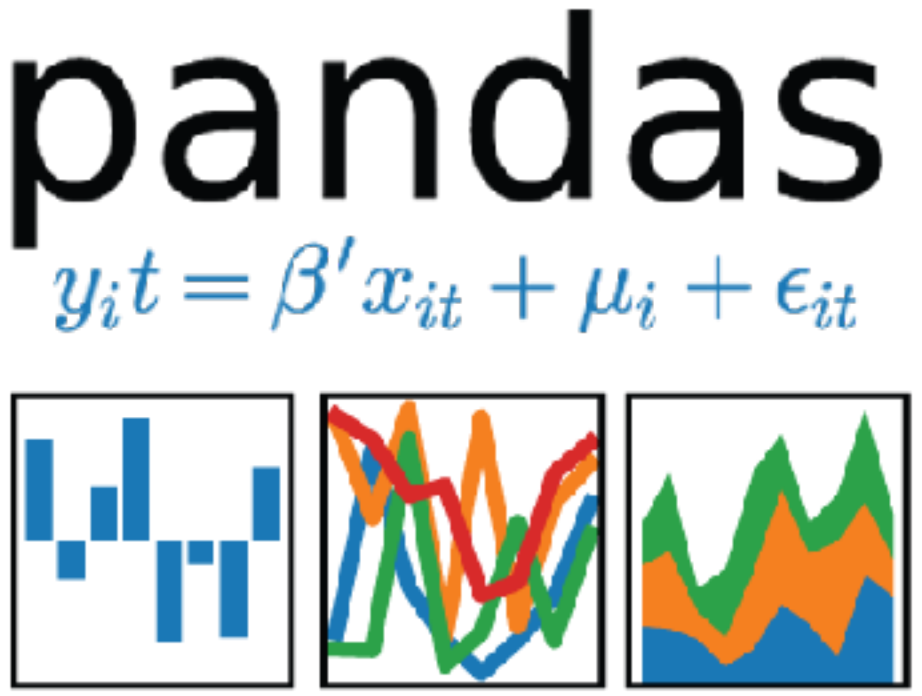
Spyder IDE
https://github.com/spyder-ide/spyder

iPython
https://ipython.org

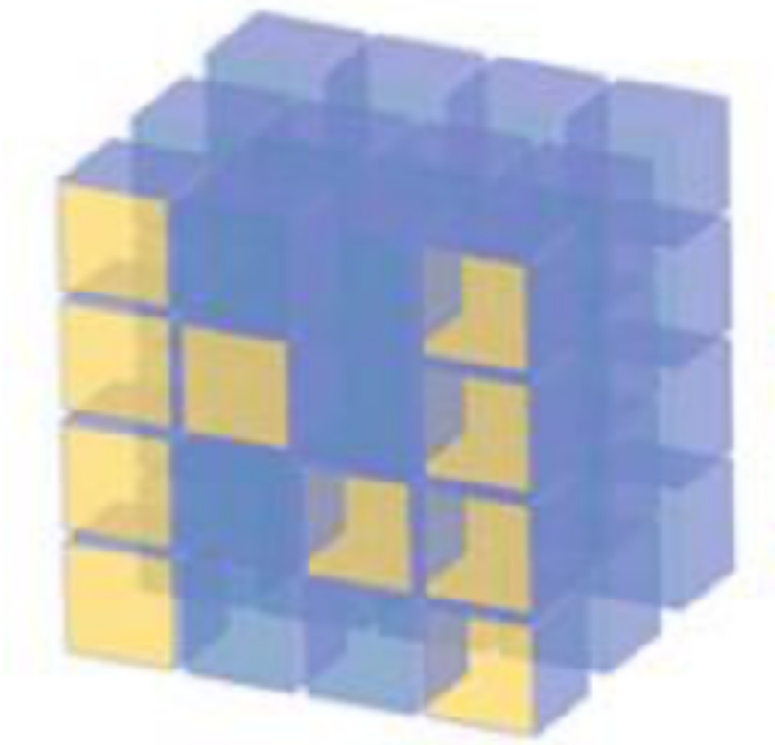
Jupyter Notebook
http://jupyter.org

Visual Studio Code
https://code.visualstudio.com

Modules (Packages)



Pandas
http://pandas.pydata.org



NumPy
http://www.numpy.org

SciPy
https://www.scipy.org

Flask
http://flask.pocoo.org

Matplotlib
http://matplotlib.org

Requests
http://docs.python-requests.org

Modules (Packages)

MicroPython
https://micropython.org

CircuitPython
https://github.com/adafruit/circuitpython

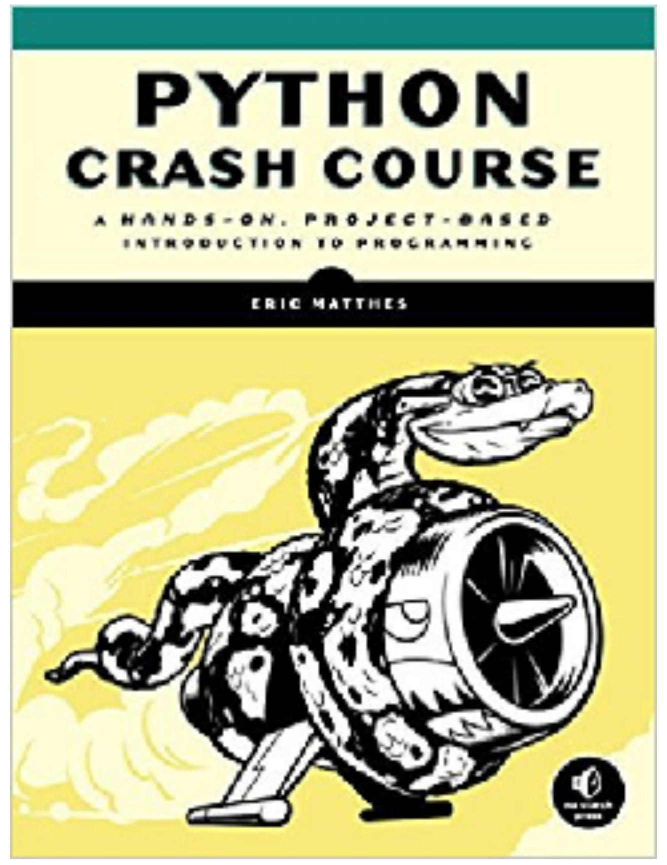
SQLAlchemy
https://www.sqlalchemy.org

Bokeh
https://bokeh.pydata.org

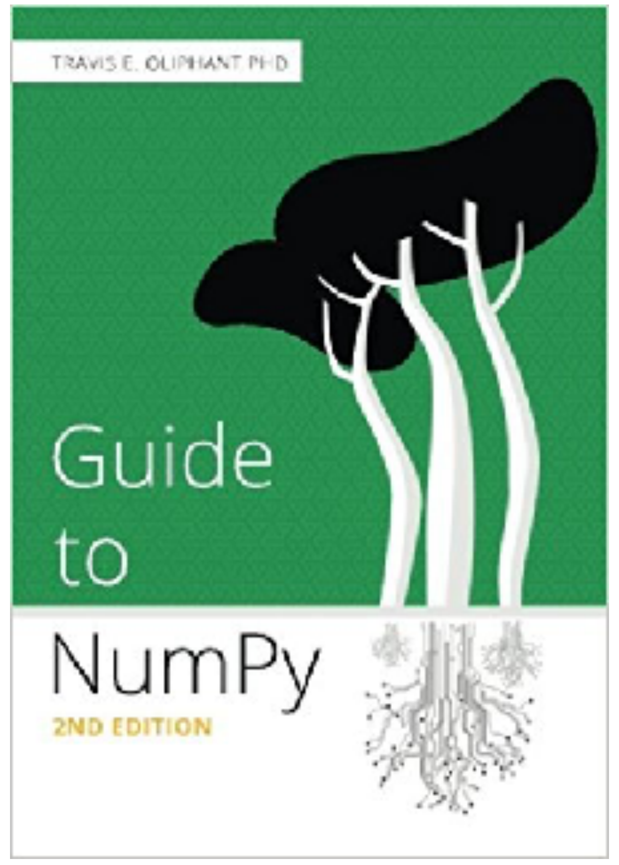
Django
https://www.djangoproject.com

SymPy
http://www.sympy.org

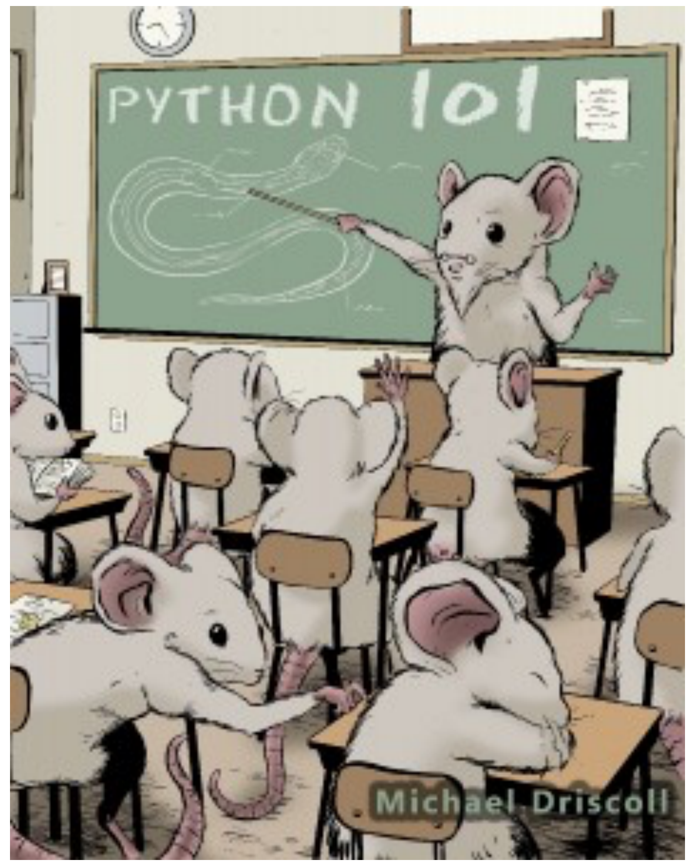
Books



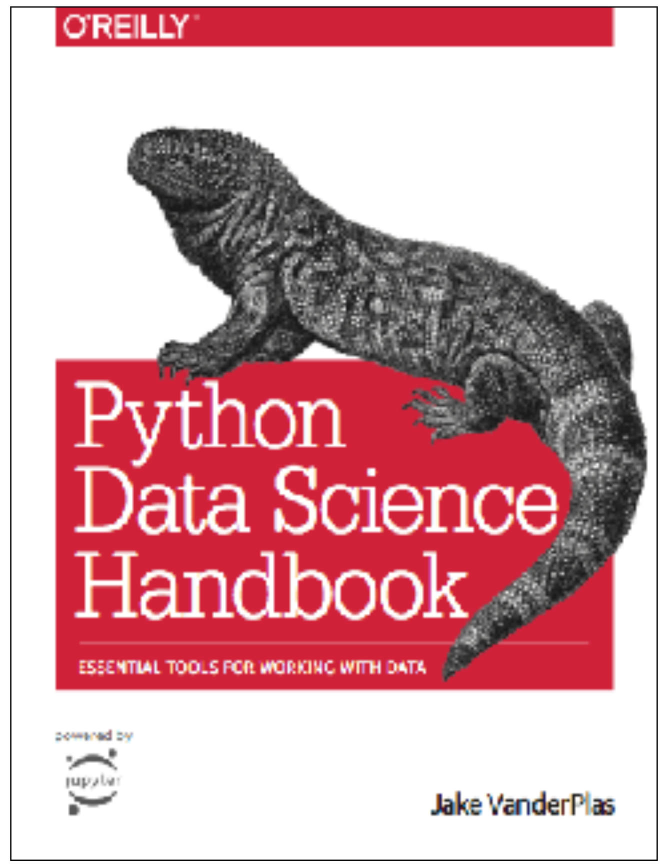
Python Crash Course
by Eric Matthes



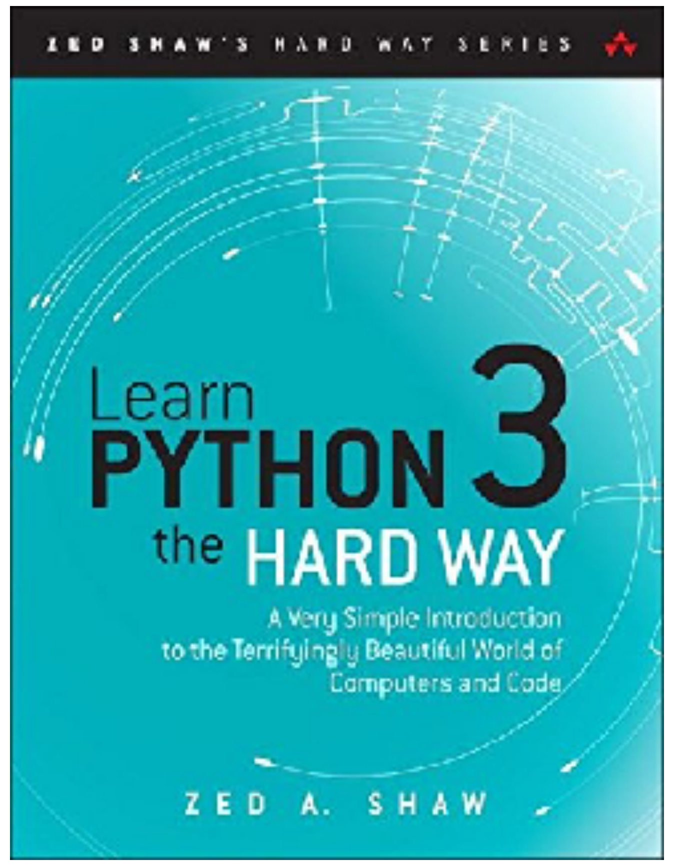
Guide to NumPy
by Travis Oliphant



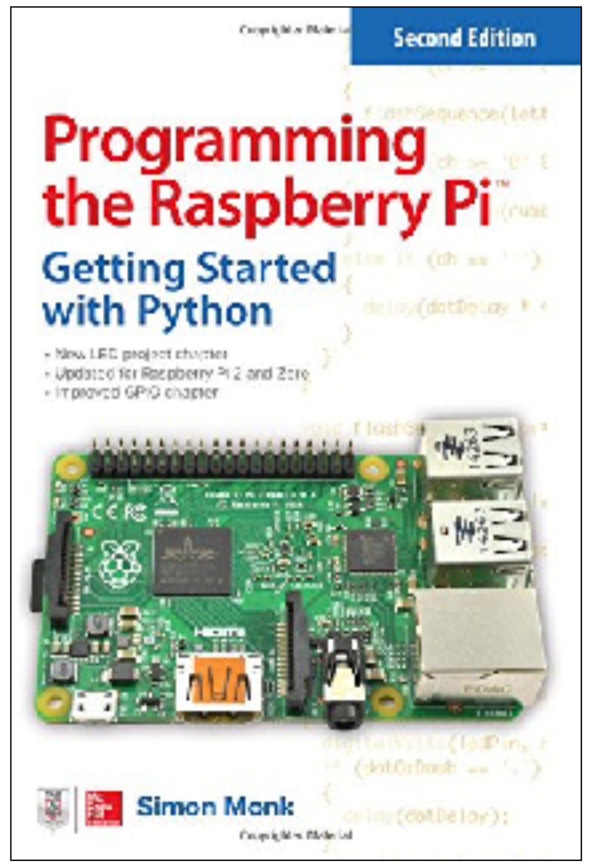
Python 101
by Michael Driscoll


O'REILLY
Python Data Science Handbook
ESSENTIAL TOOLS FOR WORKING WITH DATA
Jake VanderPlas

Python Data Science Handbook
by Jake VanderPlas
https://jakevdp.github.io/PythonDataScienceHandbook/


ZED SHAW'S HARD WAY SERIES
Learn PYTHON 3 the HARD WAY
A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code
ZED A. SHAW

Learn Python 3 the Hard Way
by Zed Shaw



Programming the Raspberry Pi
by Simon Monk

Online Courses and Tutorials

DataCamp
https://www.datacamp.com

Snakify
https://snakify.org

Test Driven Development
http://testdriven.io

LearnPython
https://www.learnpython.org

Coursera
https://www.coursera.org

Udacity
https://www.udacity.com

Conferences

SciPy
scientific computing conference
https://conference.scipy.org

PyCon
largest gathering for open-source python
https://us.pycon.org

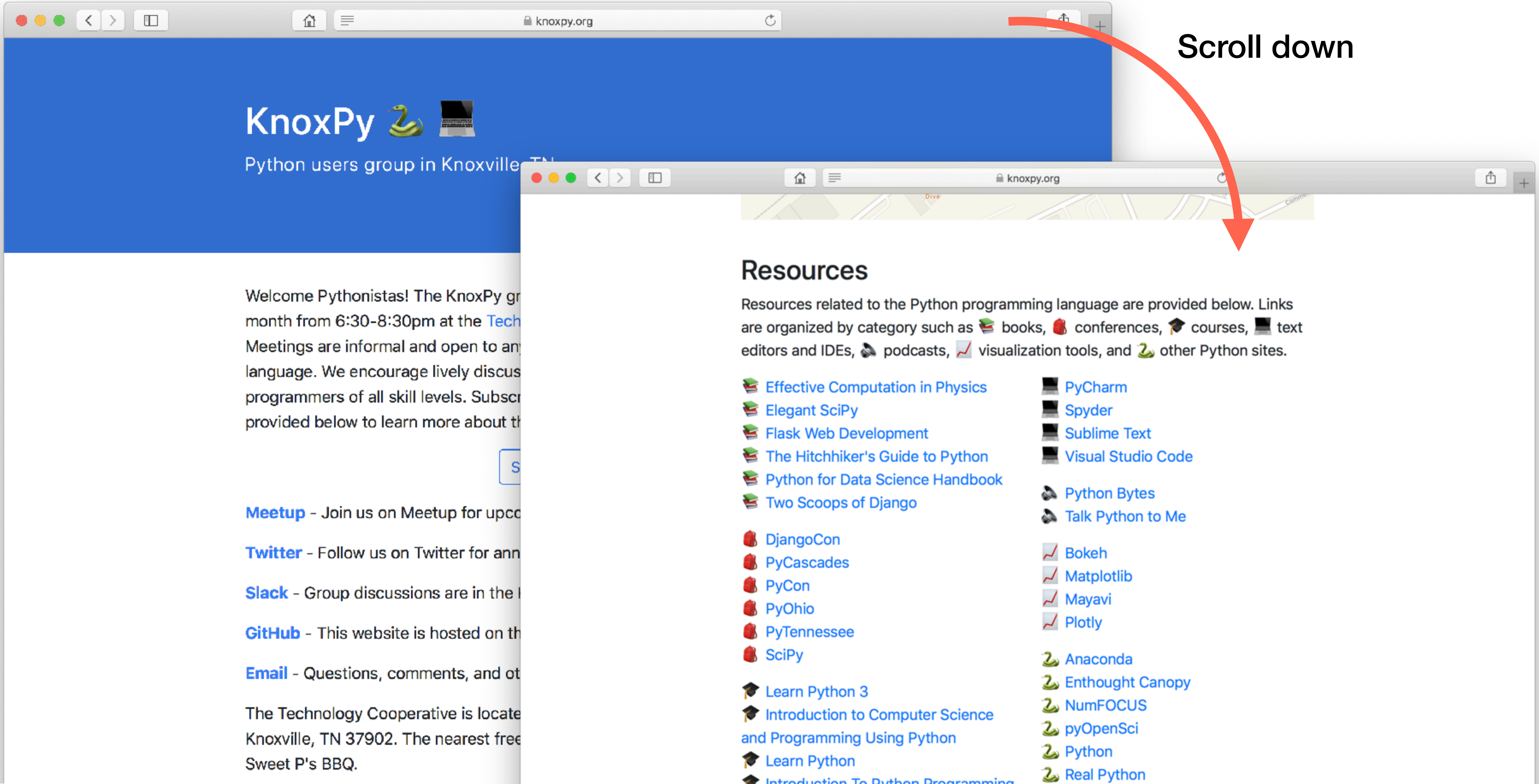
PyTennessee
regional conference in Nashville
https://www.pytennessee.org

AnacondaCon
open data science conference
https://anacondacon18.io

EuroPython
largest European python conference
https://ep2017.europython.eu/en/

PyOhio
free annual python conference
https://pyohio.org

More Resources



Scroll down

knoxpy.org

KnoxPy 🐍 💻

Python users group in Knoxville, TN

Welcome Pythonistas! The KnoxPy gr
month from 6:30-8:30pm at the Tech
Meetings are informal and open to an
language. We encourage lively discus
programmers of all skill levels. Subscr
provided below to learn more about th

Meetup - Join us on Meetup for upco

Twitter - Follow us on Twitter for ann

Slack - Group discussions are in the

GitHub - This website is hosted on th

Email - Questions, comments, and ot

The Technology Cooperative is locate
Knoxville, TN 37902. The nearest free
Sweet P's BBQ.

knoxpy.org

Resources

Resources related to the Python programming language are provided below. Links are organized by category such as 📚 books, 🎒 conferences, 🎓 courses, 💻 text editors and IDEs, 🔈 podcasts, 📈 visualization tools, and 🐍 other Python sites.

- 📚 Effective Computation in Physics
- 📚 Elegant SciPy
- 📚 Flask Web Development
- 📚 The Hitchhiker's Guide to Python
- 📚 Python for Data Science Handbook
- 📚 Two Scoops of Django

- 🎒 DjangoCon
- 🎒 PyCascades
- 🎒 PyCon
- 🎒 PyOhio
- 🎒 PyTennessee
- 🎒 SciPy

- 🎓 Learn Python 3
- 🎓 Introduction to Computer Science and Programming Using Python
- 🎓 Learn Python
- 🎓 Introduction To Python Programming

- 💻 PyCharm
- 💻 Spyder
- 💻 Sublime Text
- 💻 Visual Studio Code

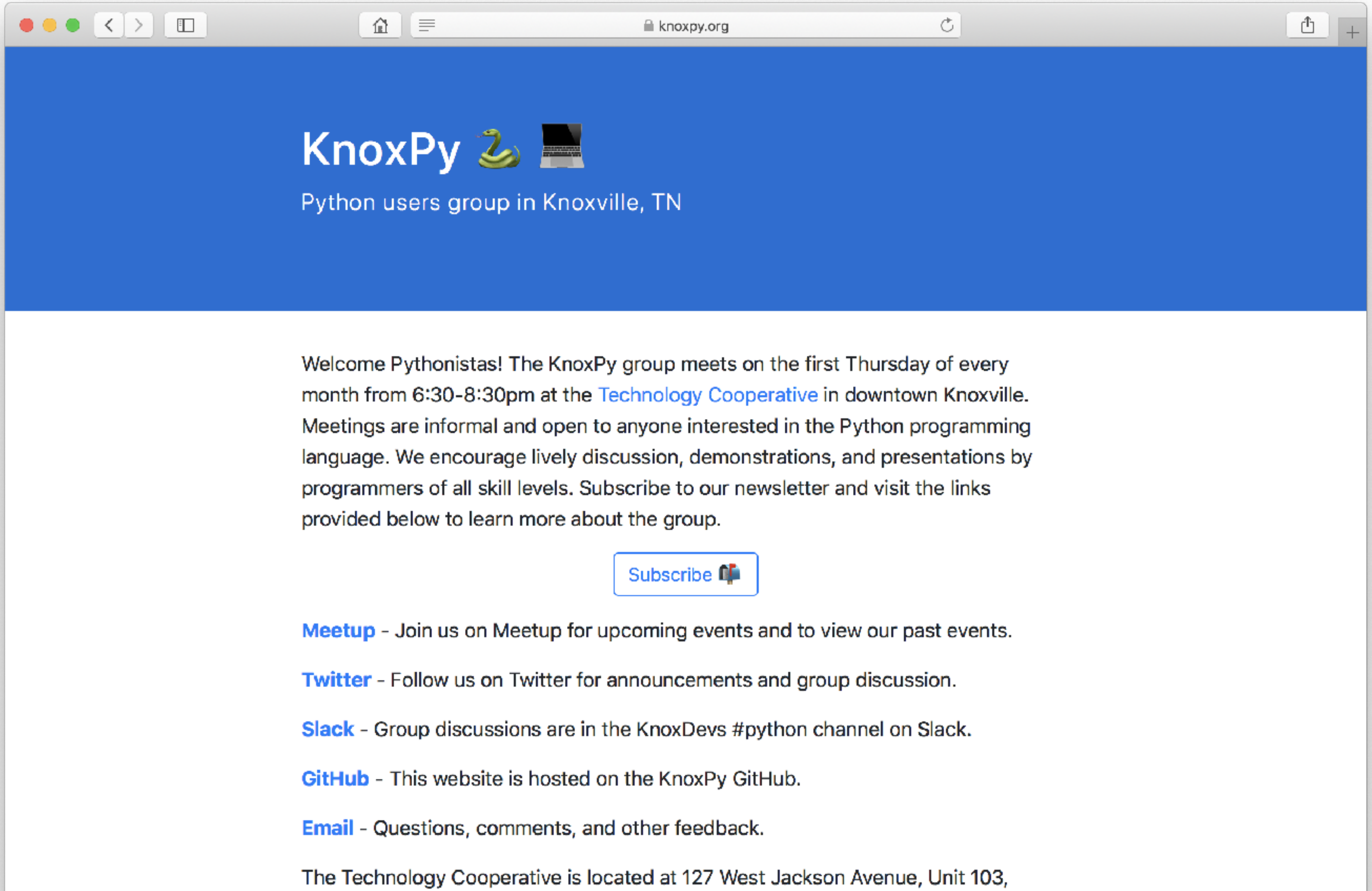
- 🔈 Python Bytes
- 🔈 Talk Python to Me

- 📈 Bokeh
- 📈 Matplotlib
- 📈 Mayavi
- 📈 Plotly

- 🐍 Anaconda
- 🐍 Enthought Canopy
- 🐍 NumFOCUS
- 🐍 pyOpenSci
- 🐍 Python
- 🐍 Real Python

Next steps…

Attend more KnoxPy meetings! https://knoxpy.org



KnoxPy 🐍 💻

Python users group in Knoxville, TN

Welcome Pythonistas! The KnoxPy group meets on the first Thursday of every month from 6:30-8:30pm at the Technology Cooperative in downtown Knoxville. Meetings are informal and open to anyone interested in the Python programming language. We encourage lively discussion, demonstrations, and presentations by programmers of all skill levels. Subscribe to our newsletter and visit the links provided below to learn more about the group.

Subscribe 📫

Meetup - Join us on Meetup for upcoming events and to view our past events.

Twitter - Follow us on Twitter for announcements and group discussion.

Slack - Group discussions are in the KnoxDevs #python channel on Slack.

GitHub - This website is hosted on the KnoxPy GitHub.

Email - Questions, comments, and other feedback.

The Technology Cooperative is located at 127 West Jackson Avenue, Unit 103,