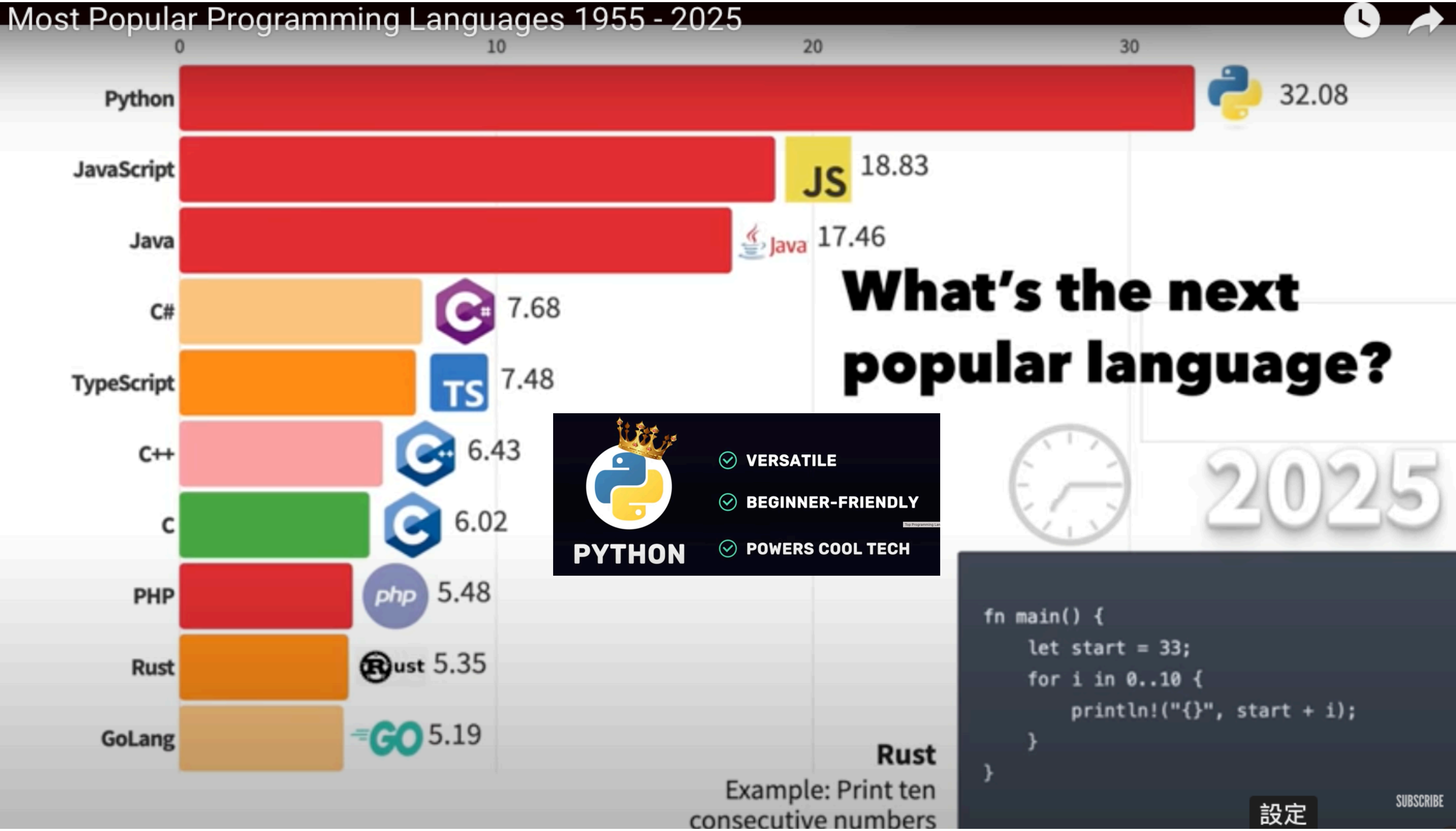
134.208.26.59

Software implementation and computation
Python Programming

An introduction to Python and Python modules



Most Popular Programming Languages 1955 - 2025

What's the next popular language?

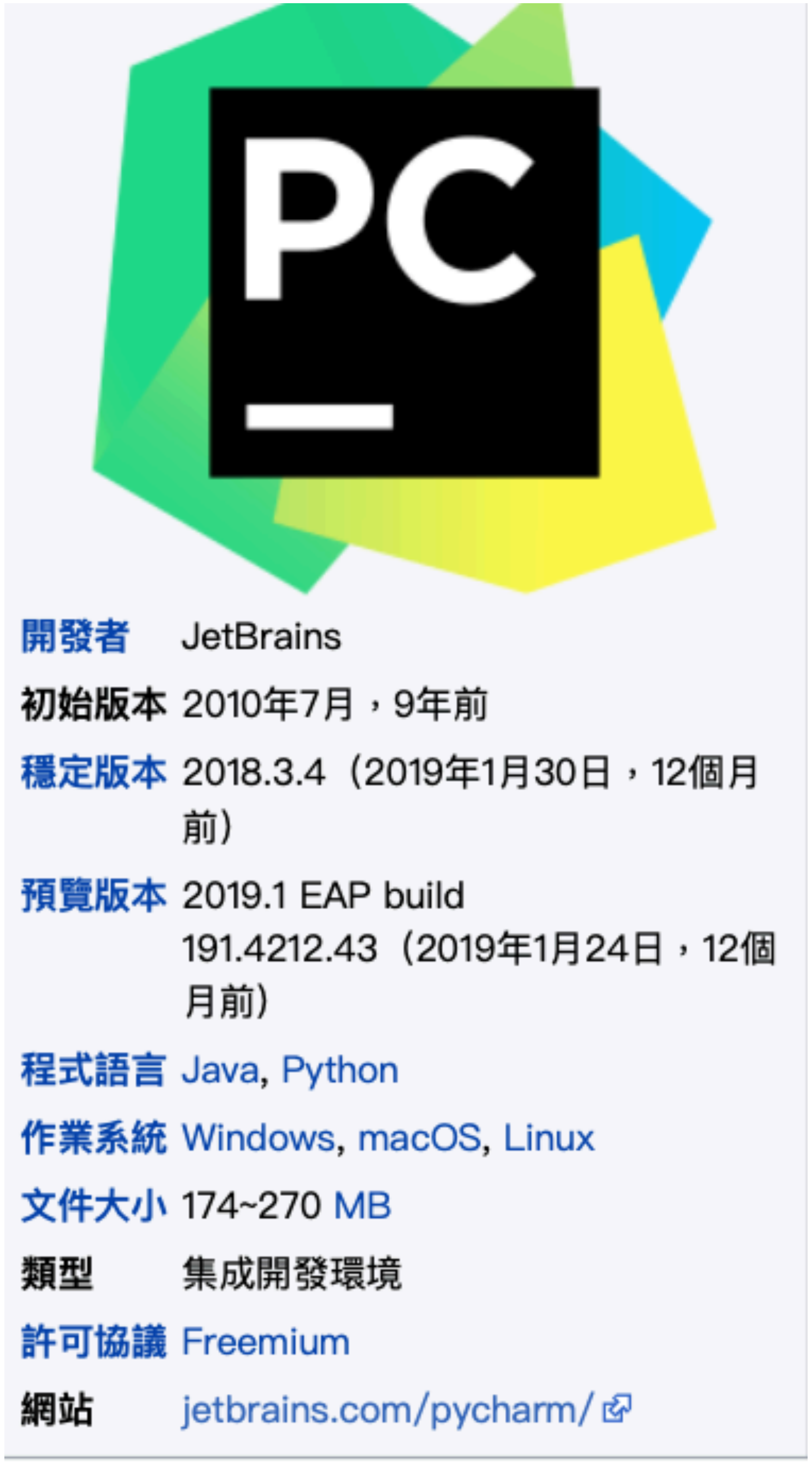
PYTHON

- ✓ VERSATILE
- ✓ BEGINNER-FRIENDLY
- ✓ POWERS COOL TECH

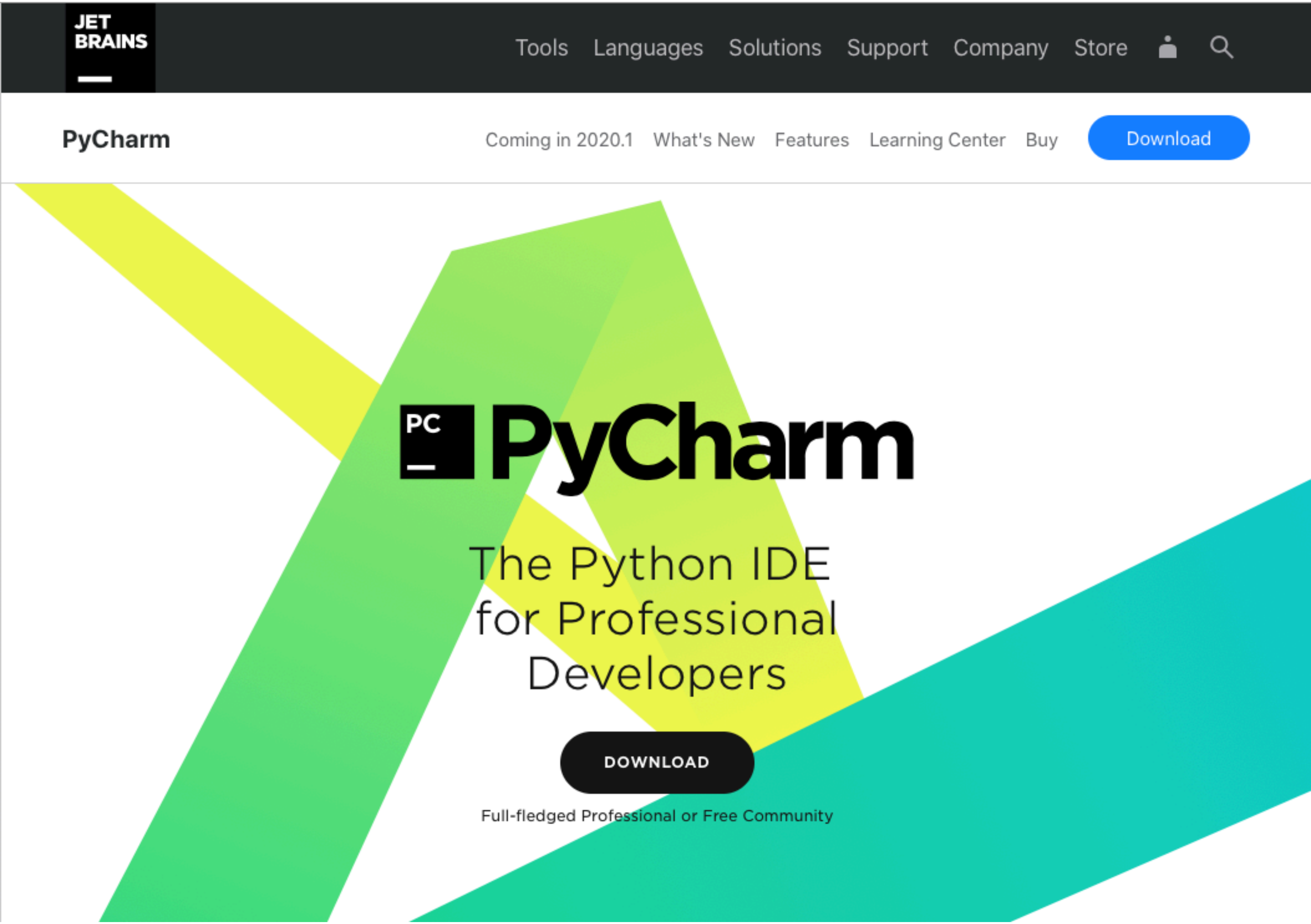
Rust

Example: Print ten consecutive numbers

```
fn main() {
    let start = 33;
    for i in 0..10 {
        println!("{}", start + i);
    }
}
```

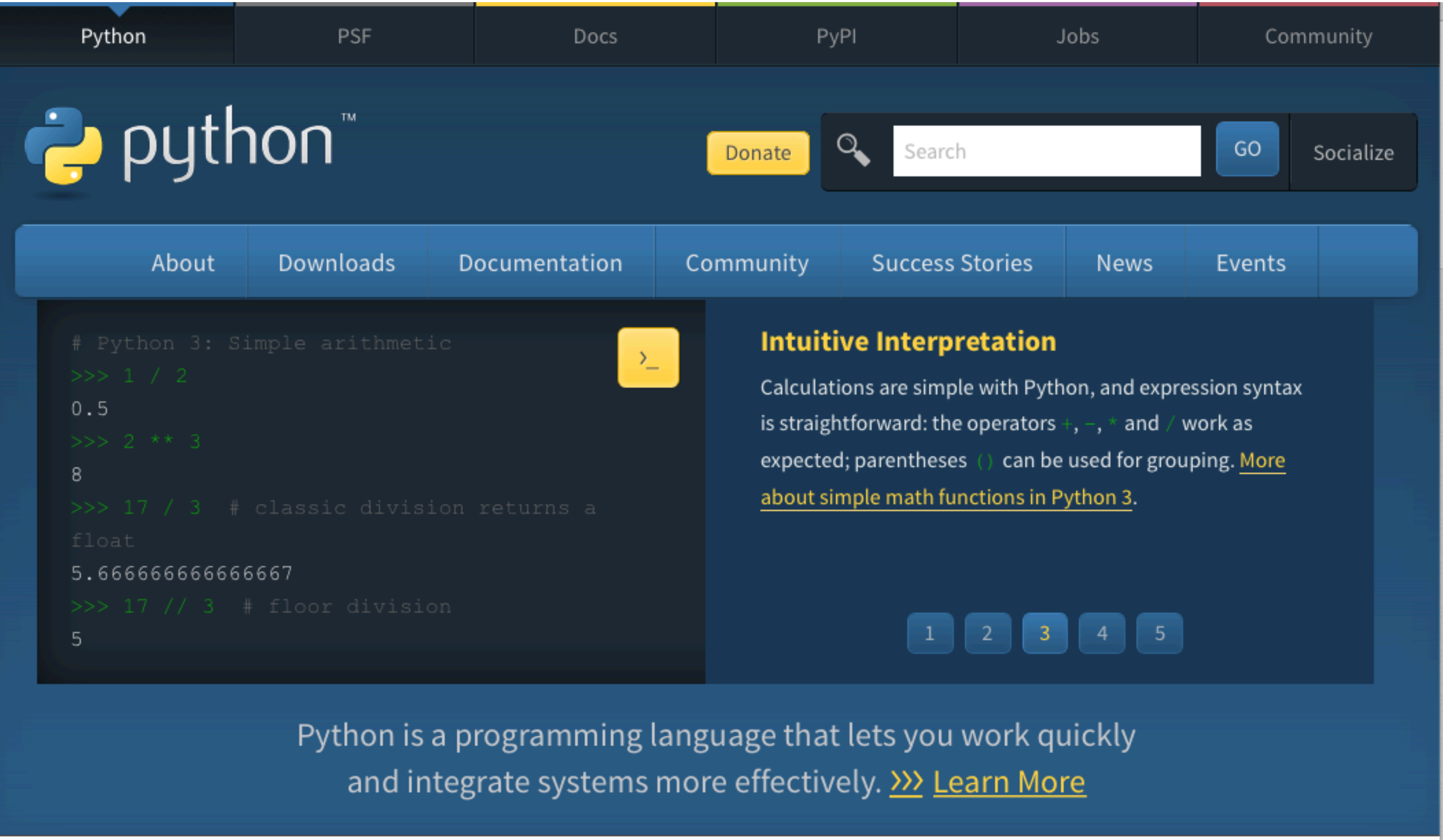


開發者	JetBrains
初始版本	2010年7月，9年前
穩定版本	2018.3.4（2019年1月30日，12個月前）
預覽版本	2019.1 EAP build 191.4212.43（2019年1月24日，12個月前）
程式語言	Java, Python
作業系統	Windows, macOS, Linux
文件大小	174~270 MB
類型	集成開發環境
許可協議	Freemium
網站	jetbrains.com/pycharm/



The Python IDE for Professional Developers

Full-fledged Professional or Free Community

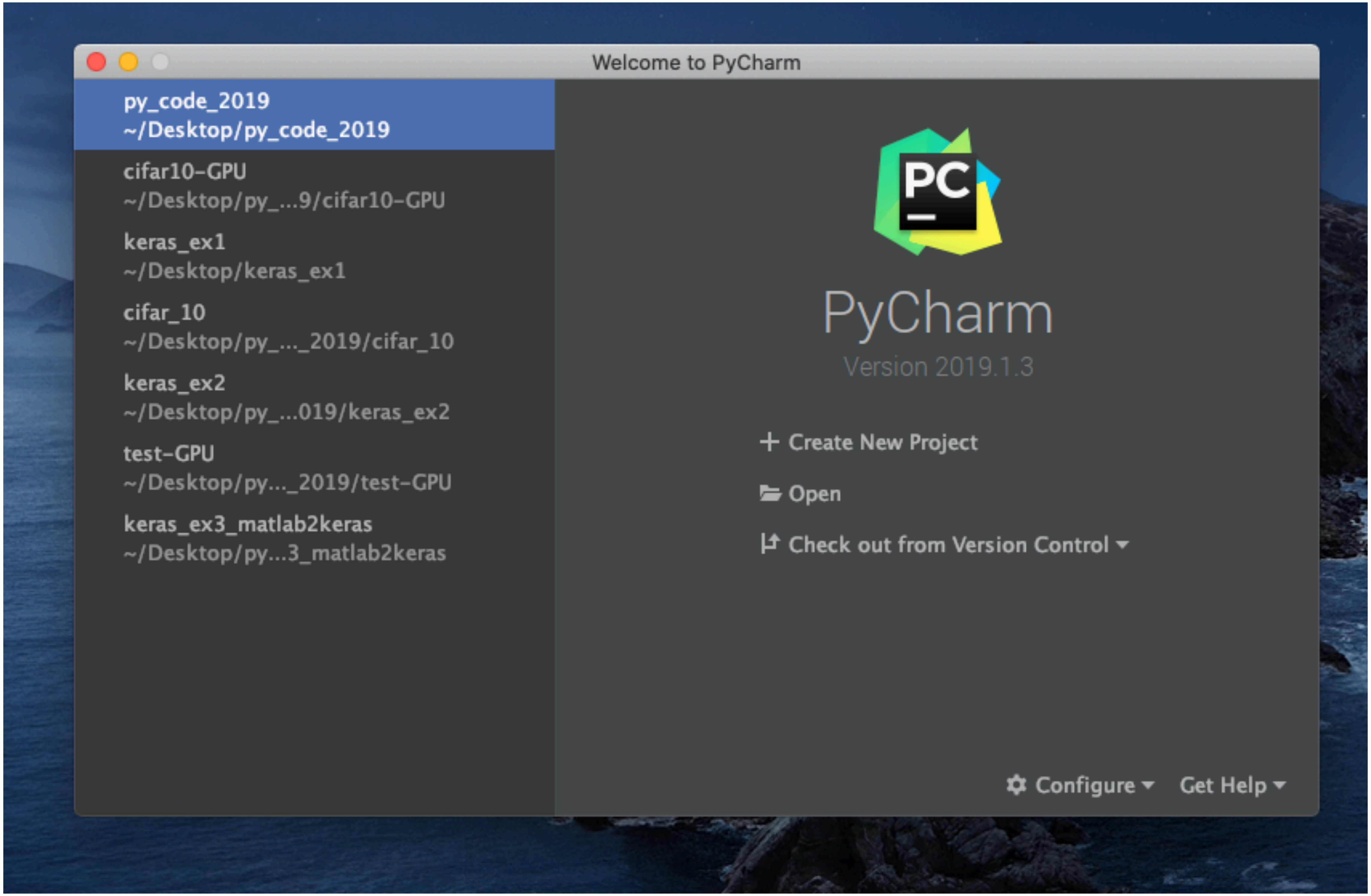


```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3  # classic division returns a
float
5.666666666666667
>>> 17 // 3  # floor division
5
```

Intuitive Interpretation

Calculations are simple with Python, and expression syntax is straightforward: the operators +, -, * and / work as expected; parentheses () can be used for grouping. More about simple math functions in Python 3.

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



Welcome to PyCharm

py_code_2019
~/Desktop/py_code_2019

cifar10-GPU
~/Desktop/py_...9/cifar10-GPU

keras_ex1
~/Desktop/keras_ex1

cifar_10
~/Desktop/py_..._2019/cifar_10

keras_ex2
~/Desktop/py_...019/keras_ex2

test-GPU
~/Desktop/py..._2019/test-GPU

keras_ex3_matlab2keras
~/Desktop/py...3_matlab2keras

PyCharm
Version 2019.1.3

探索全新科技的未來，透過我們的GPF技術，提升學習與交流的無障礙體驗，讓每個人都能輕鬆掌握知識，無論年齡與背景。

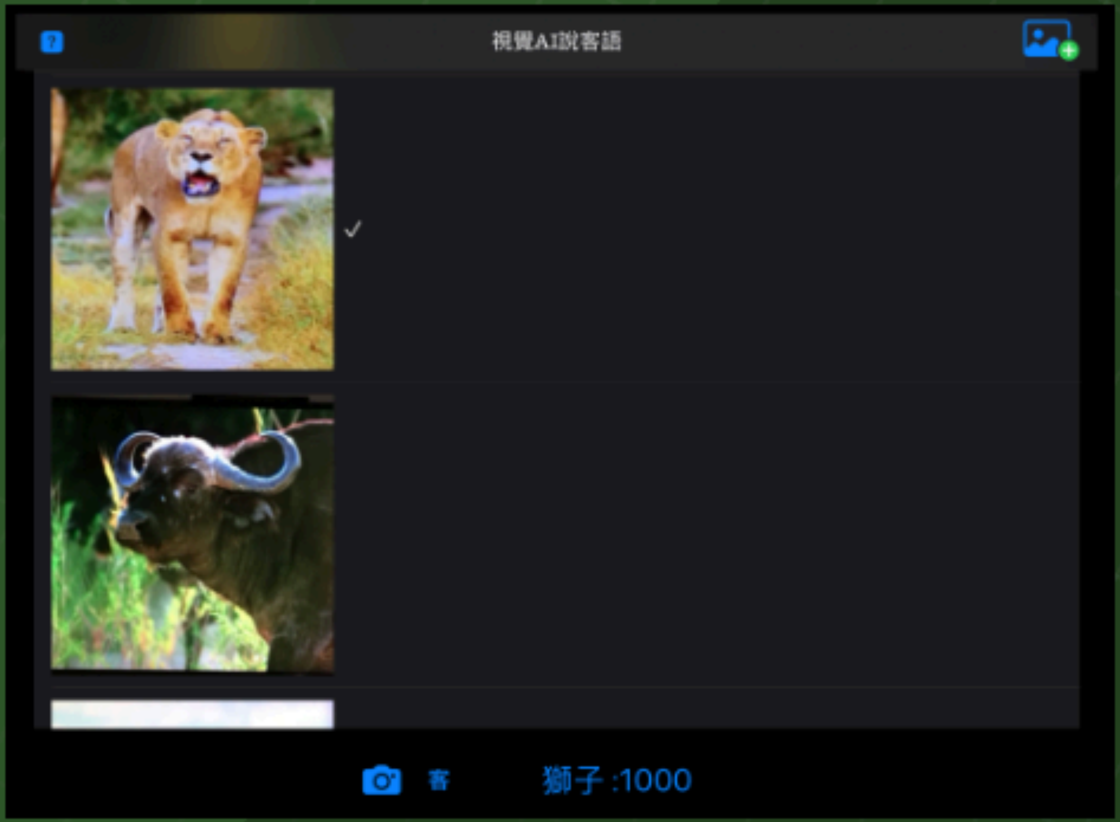
GPF
視覺AI說台語

Home About Us Contact Us

EXPLORE FEATURES OF GPF-VISION-AI

探索新一代的生成預訓練濾波器技術

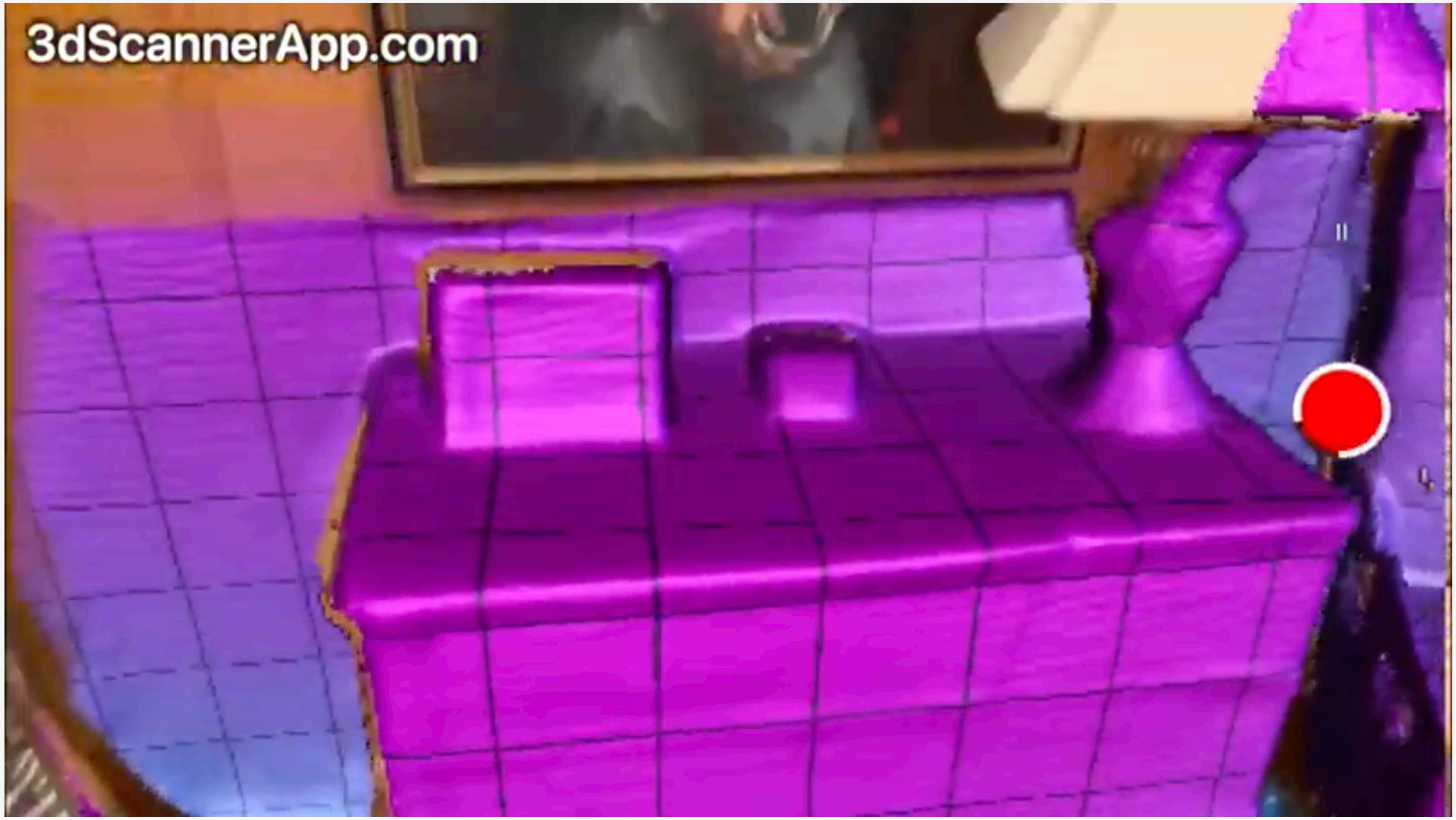
這款應用程式利用最新的GPF卷積神經網絡技術，幫助用戶即時識別物件影像，並提供精準的台語和客語語音翻譯，讓學習台語客語變得更輕鬆。



視覺AI說台語客語測試報名

體驗最先進的技術，提升您的學習效率，讓語言溝通無障礙。

image3d.png



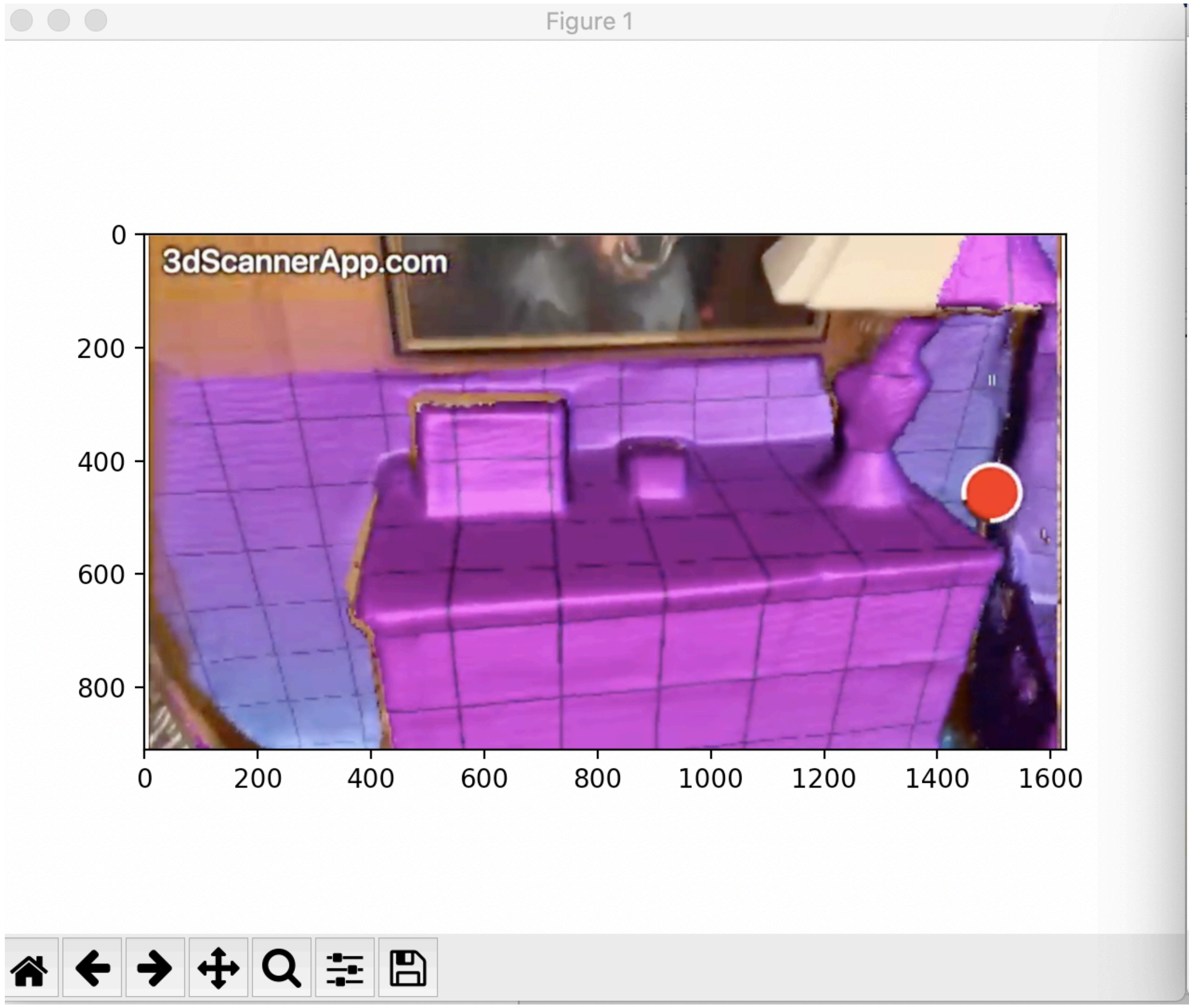
EX1: Read and Show an image

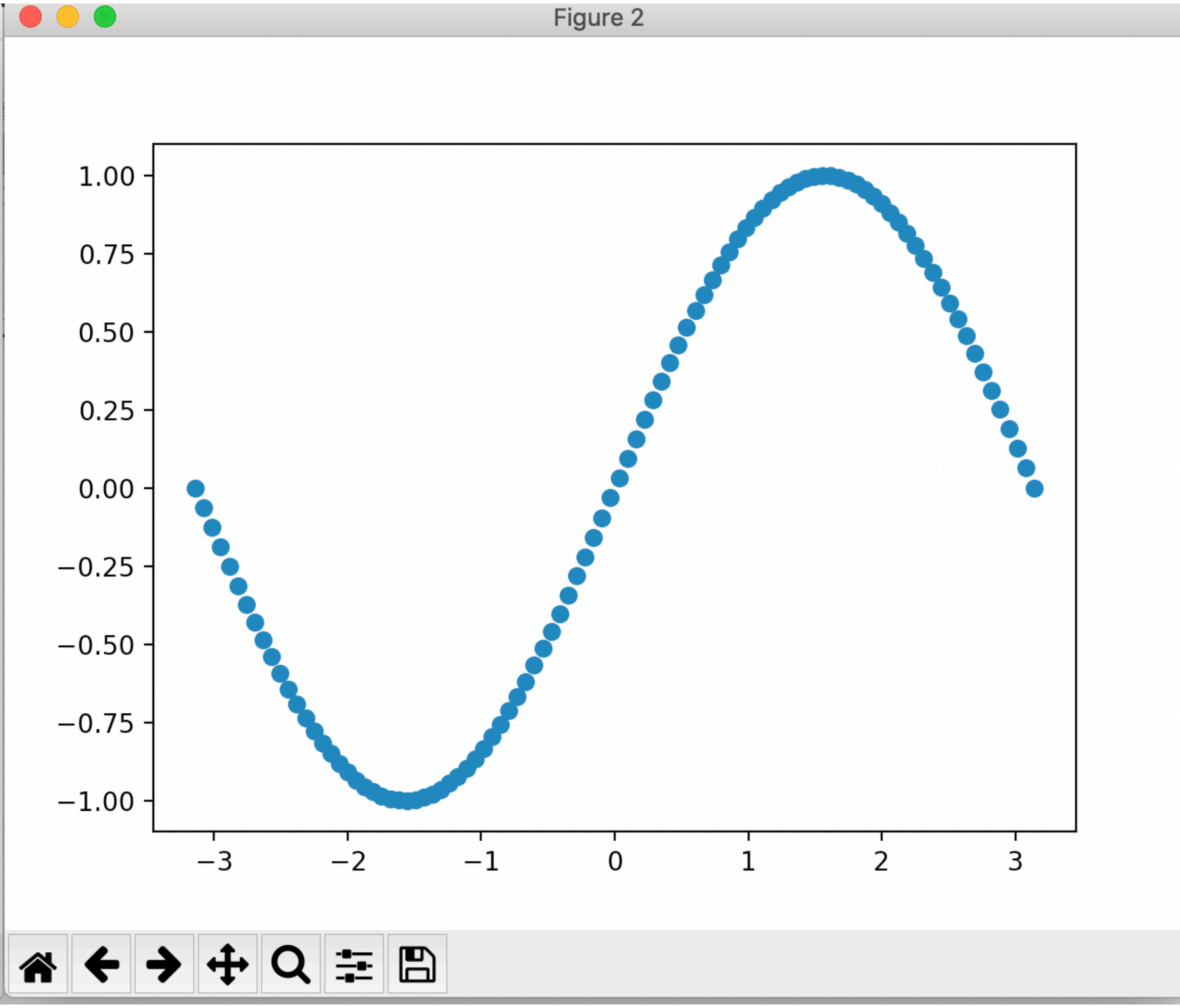
- Use PyCharm to new a project
- Place image3d.png to the project directory
- Install package matplotlib
- New a python file and write code to read and display an image

plt represents a module

```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
I = mpimg.imread('image3d.png')
plt.imshow(I)
plt.show()
```

Ex2: show an image and a sin plot





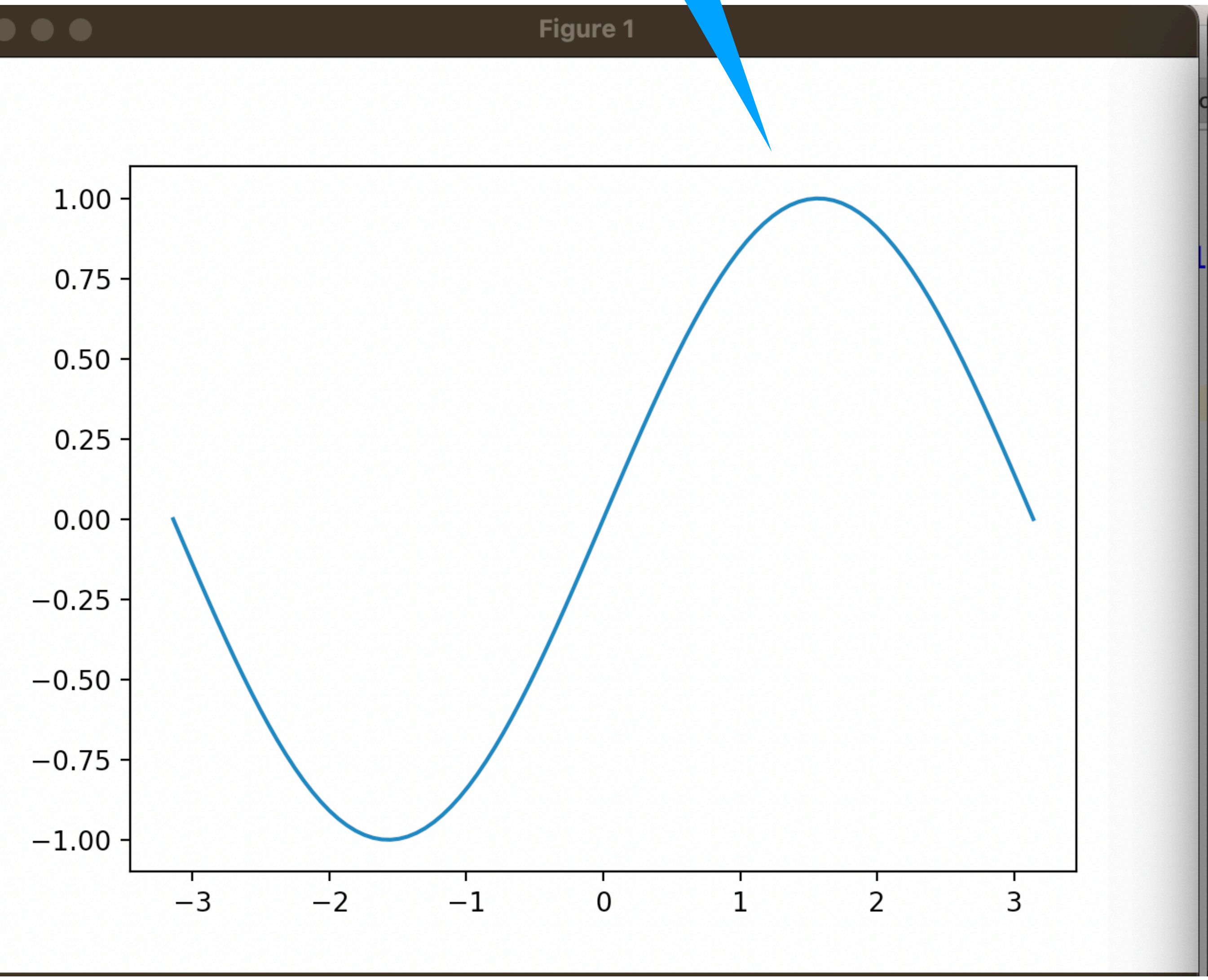
Ex2: show an image and a sin plot

- Steps in Ex1
- Install numpy
- Partition [-pi, pi] to 100 points and store them to x
- Set y to sin of x
- Plot points defined by x and y

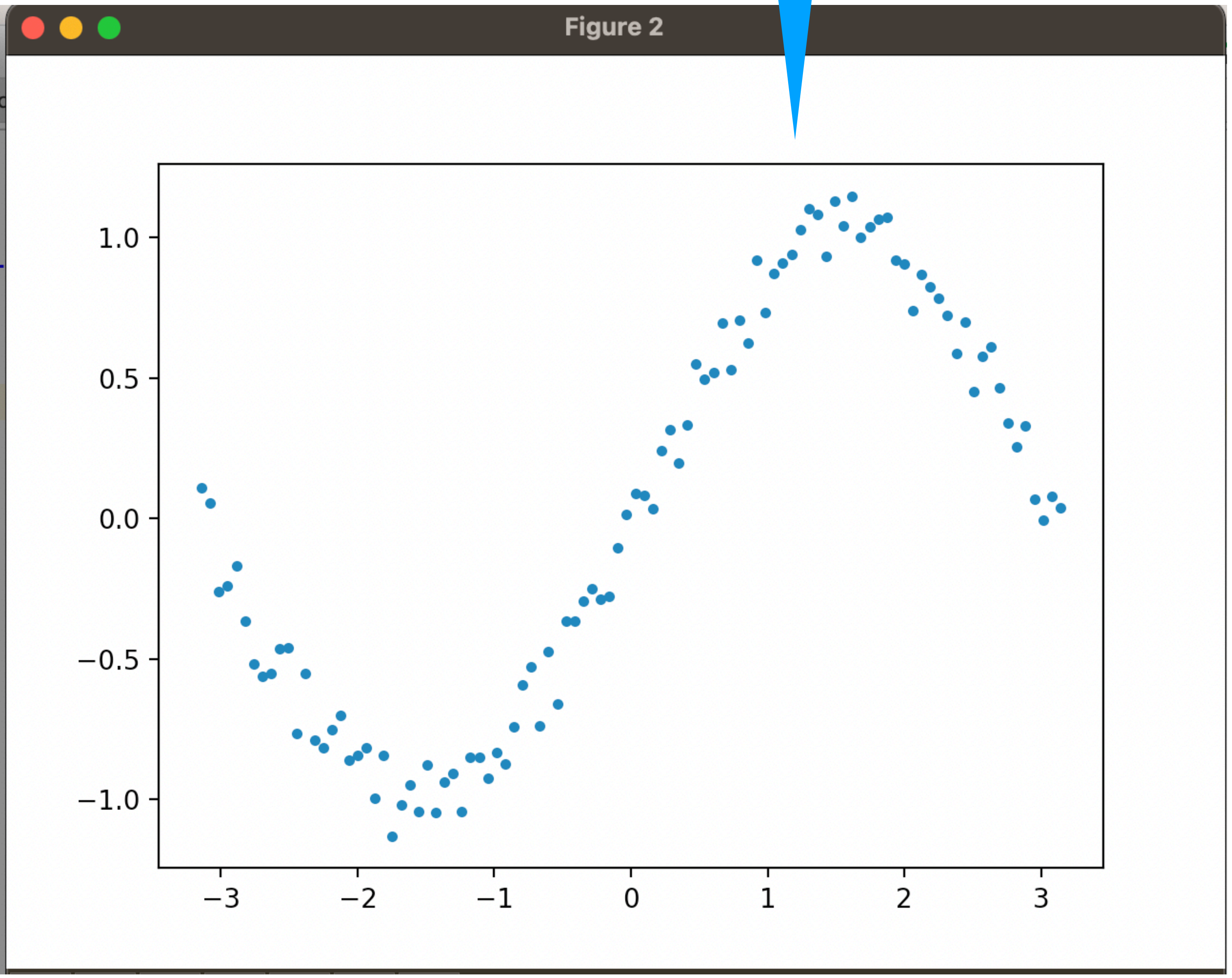
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
plt.figure(1)
plt.imshow(mpimg.imread('image3d.png'))


import numpy as np
x = np.linspace(-np.pi, np.pi, 100)
y = np.sin(x)
plt.figure(2)
plt.scatter(x, y, marker = 'o')
plt.show()
```

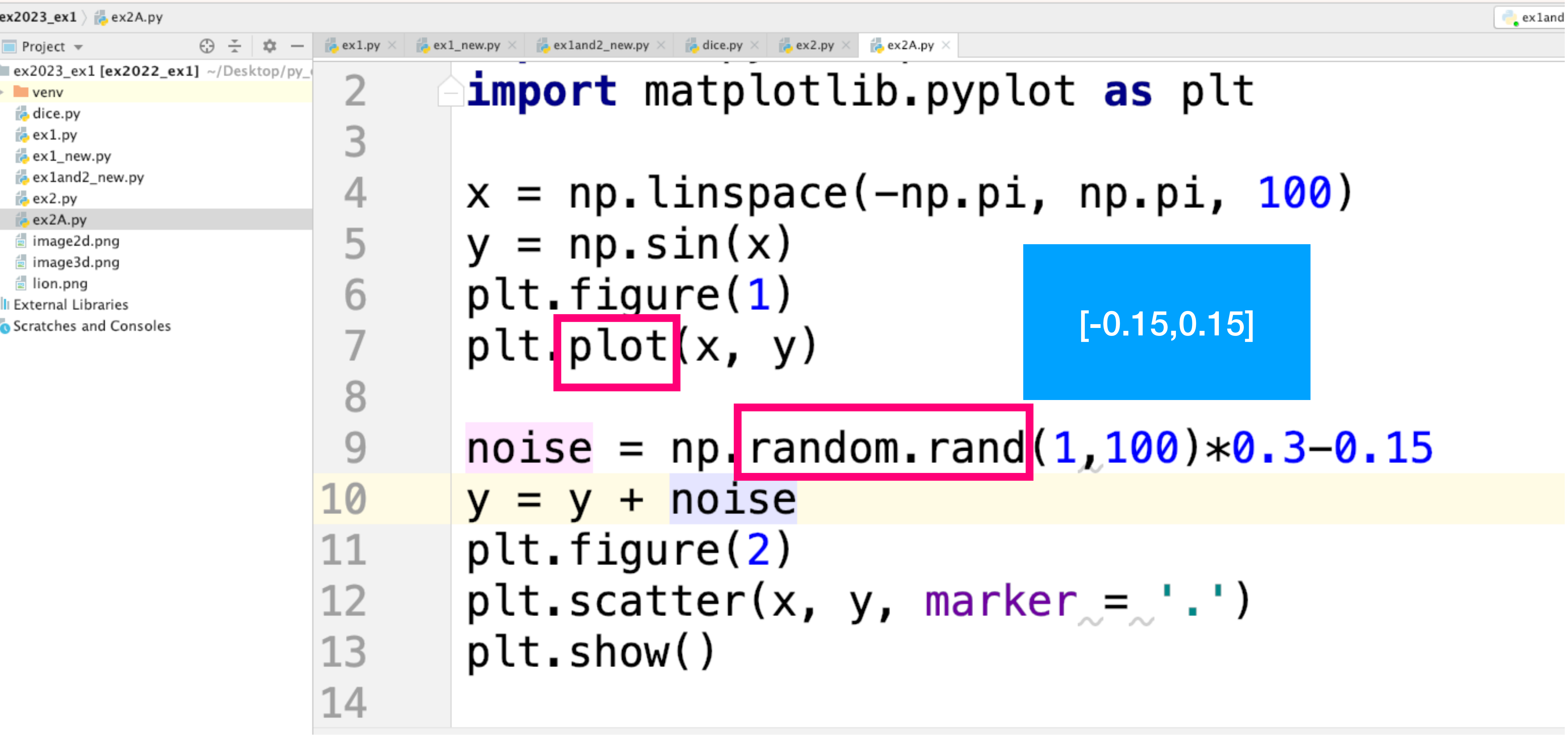
Ex2A

Use plt.plot to draw sin function



Try to generate $y = sin(x) + n,$ where $n \in [-0.15,0.15]$

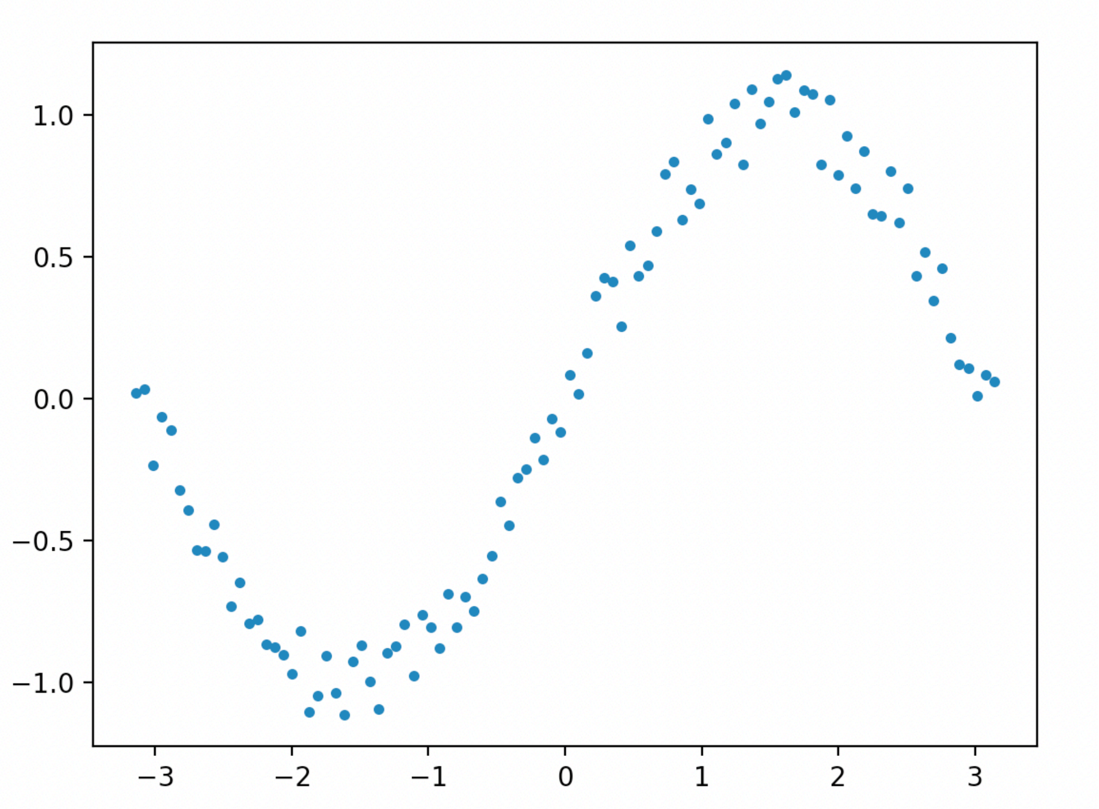


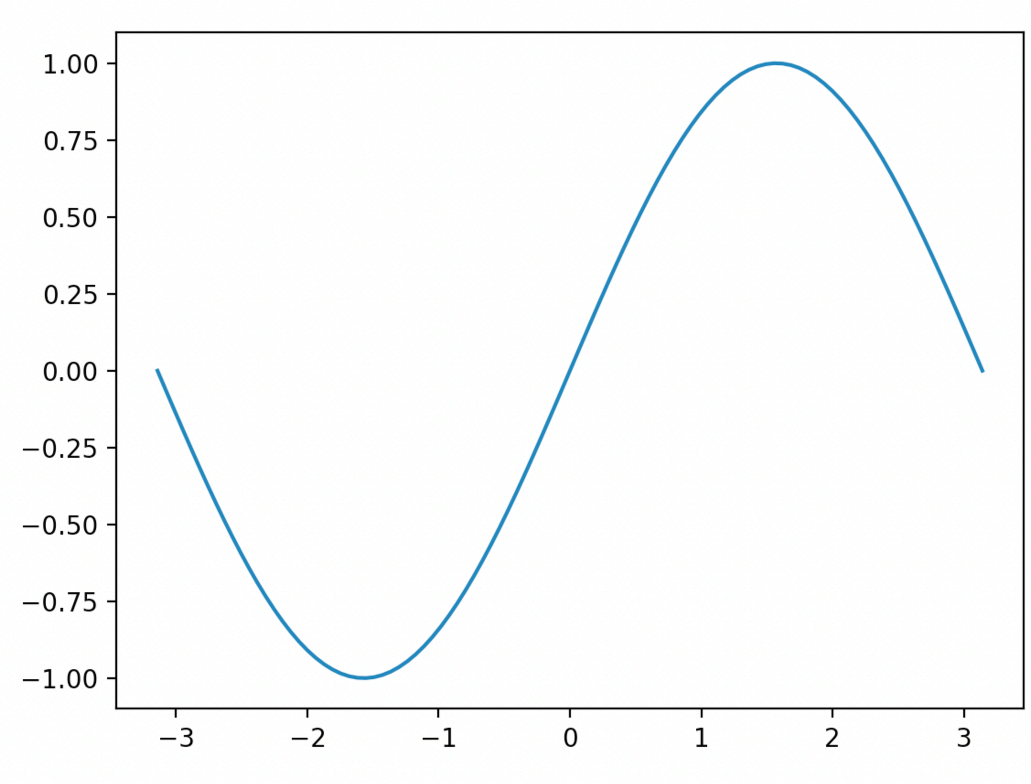


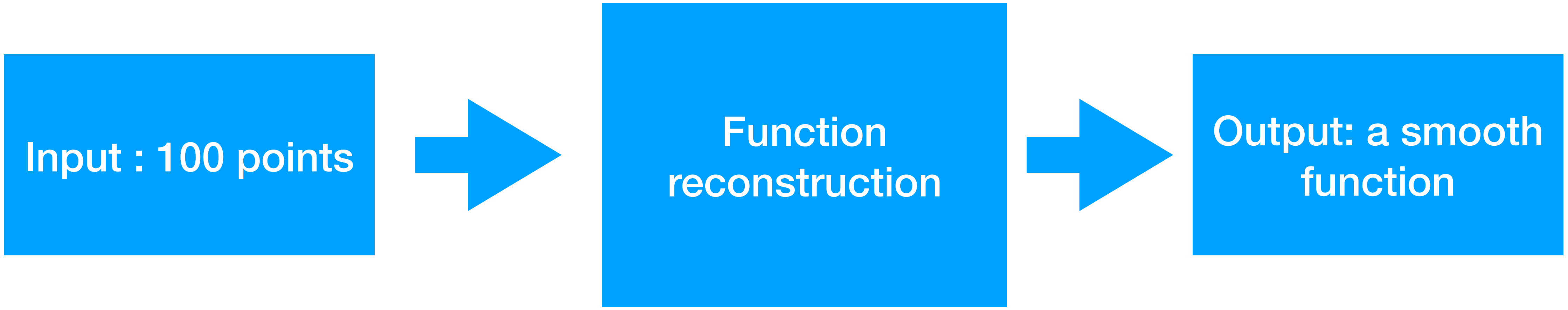
```python
import matplotlib.pyplot as plt

x = np.linspace(-np.pi, np.pi, 100)
y = np.sin(x)
plt.figure(1)
plt.plot(x, y)

noise = np.random.rand(1,100)*0.3-0.15
y = y + noise
plt.figure(2)
plt.scatter(x, y, marker = '.')
plt.show()

```

[-0.15,0.15]

Challenge







Input : 100 points

Function reconstruction

Output: a smooth function

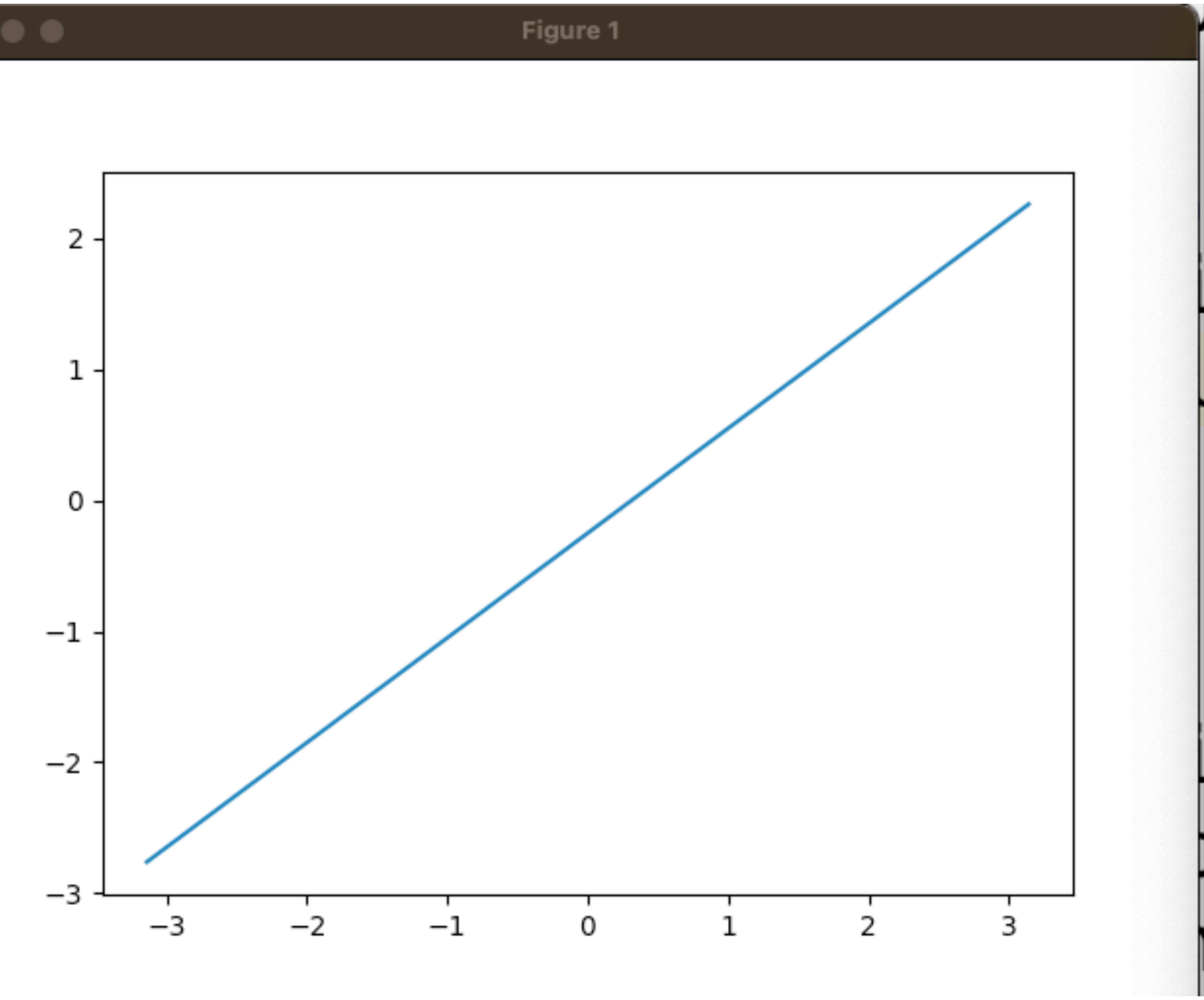


Figure 1
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

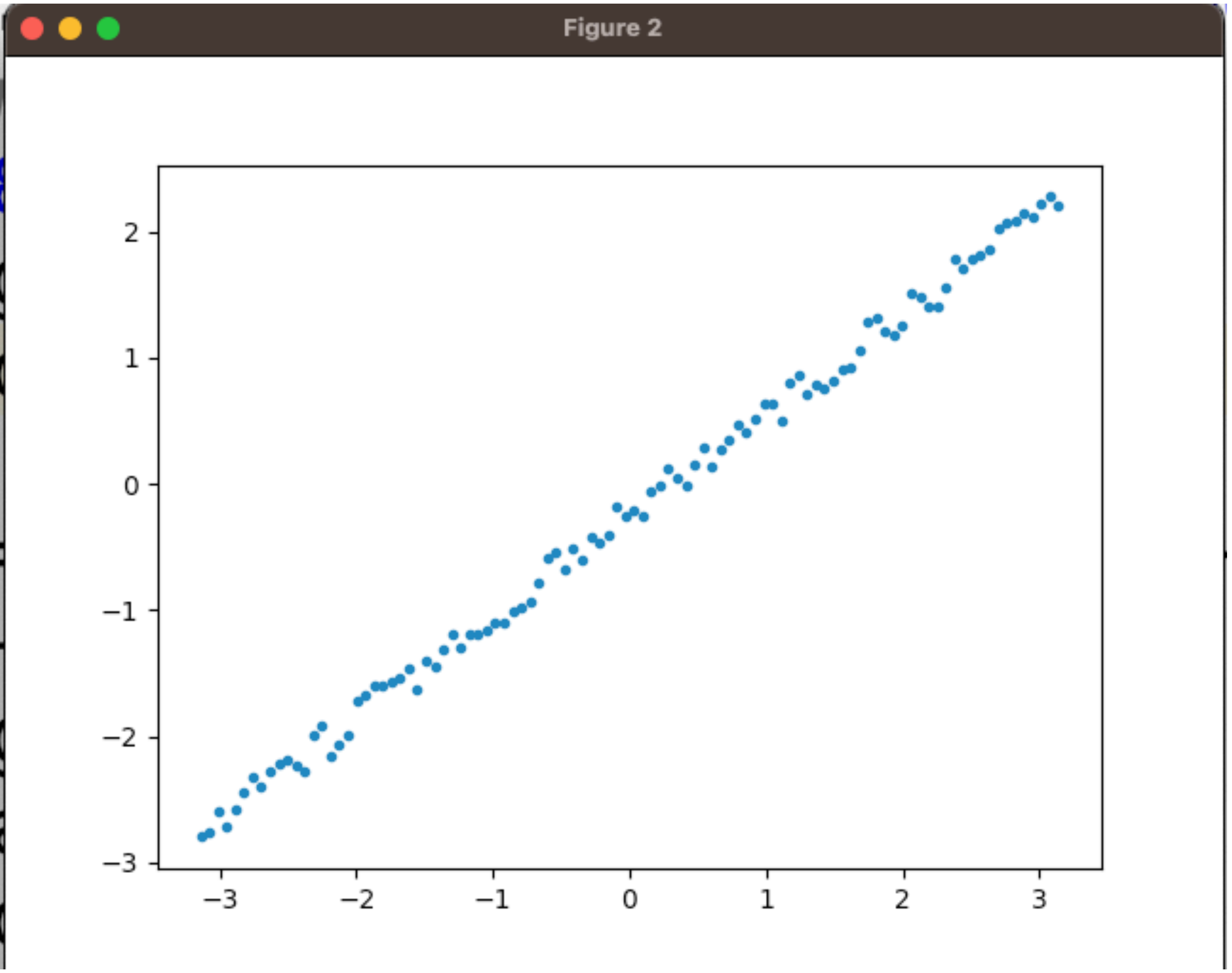


Figure 2
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

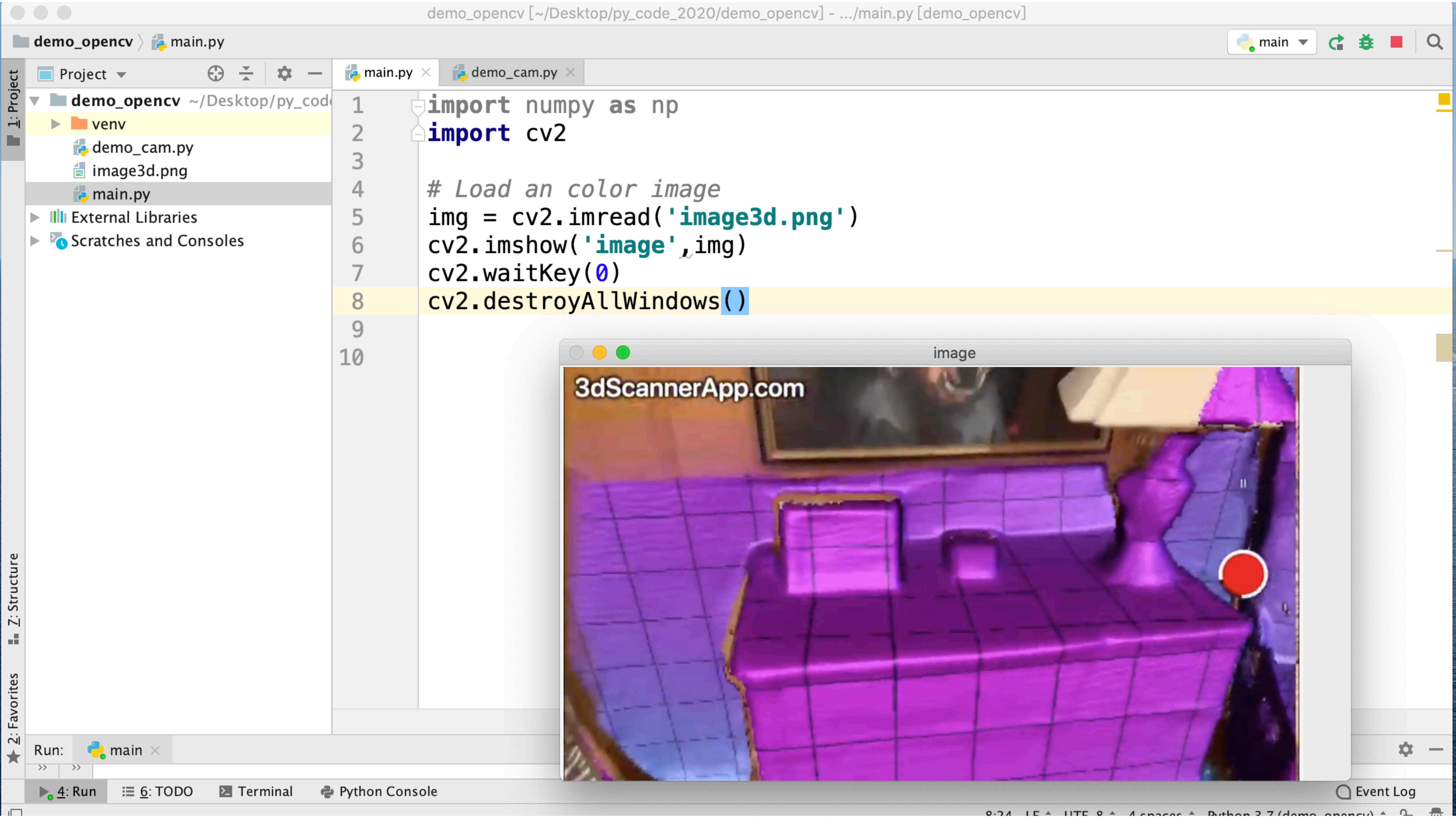
🏠 OpenCV-Python Tutorials

latest

cv2.imread(), cv2.imshow()

- Install opencv-python
- Import cv2
- Use cv2.imread to read an image
- Use cv2.imshow to show an image

```python
import numpy as np
import cv2

# Load an color image
img = cv2.imread('image3d.png')
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

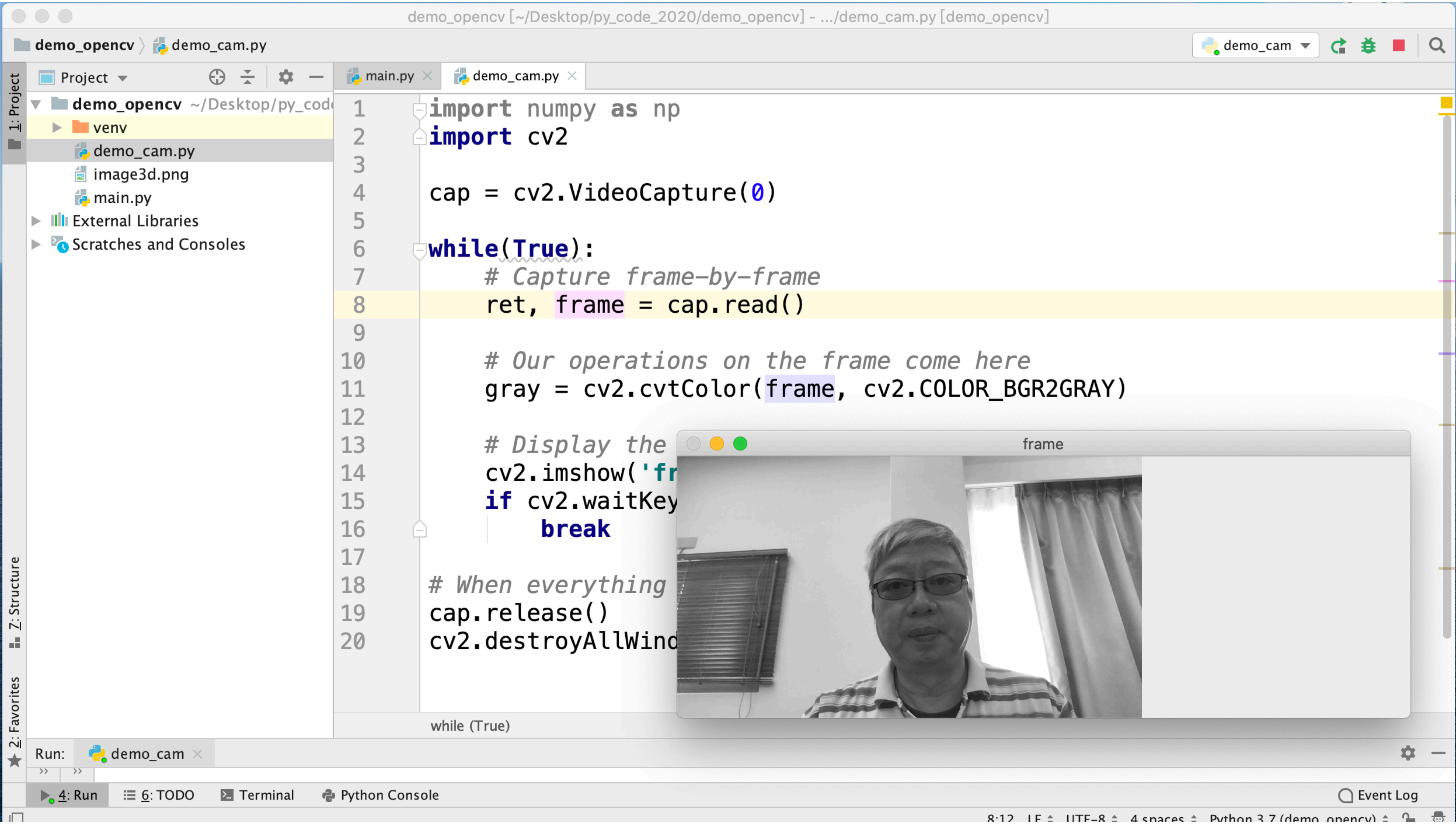


```
import numpy as np
import cv2

# Load an color image
img = cv2.imread('image3d.png')
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

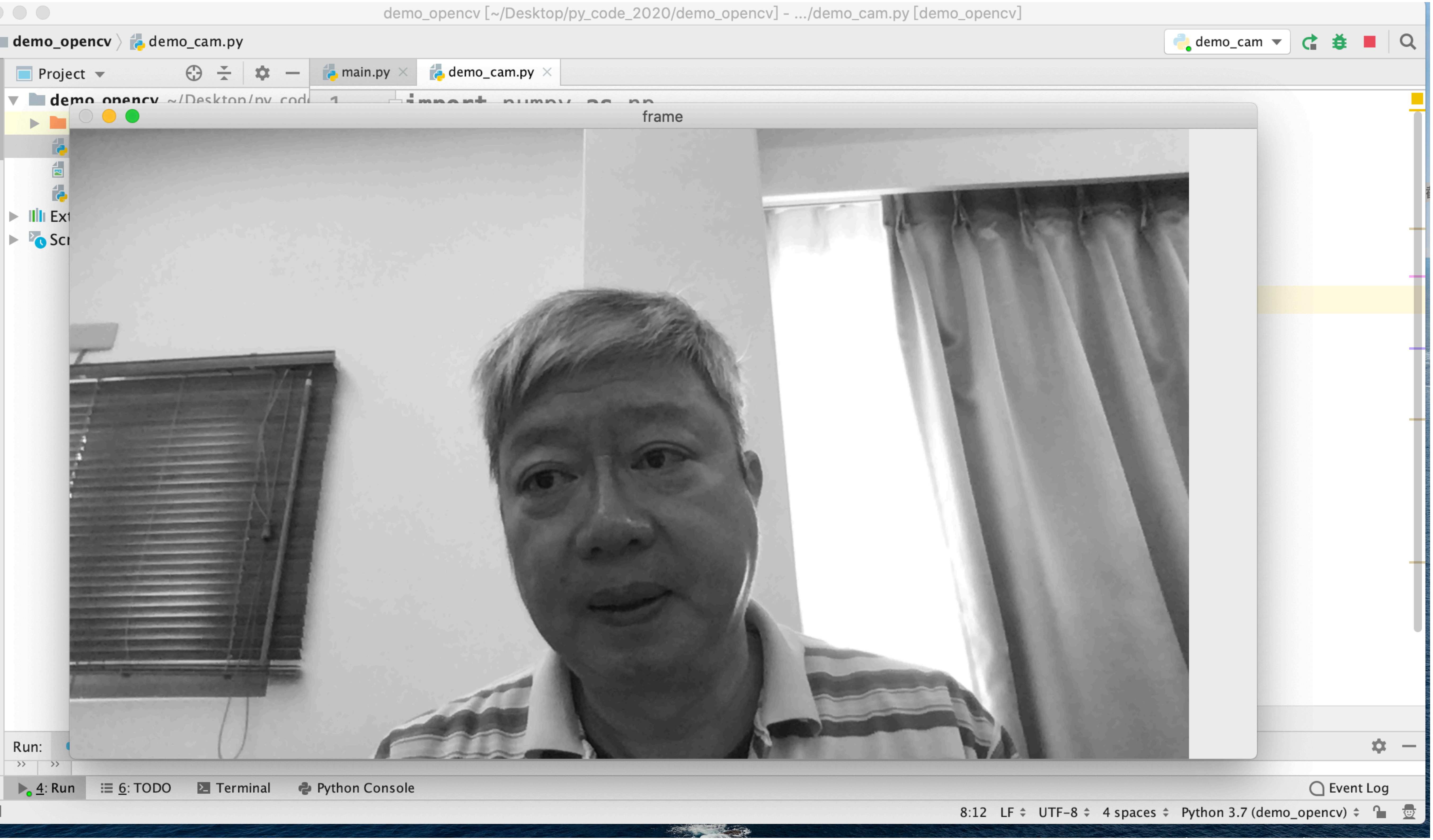
```
cap = cv2.VideoCapture(0)
cap.read()
```

- Set cap to cv2.VideoCapture(0)
- while(True):
  - Capture frame-by-frame
  - Display the resulting frame

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Display the resulting frame
    cv2.imshow('frame',gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```



```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Display the
    cv2.imshow('fr
    if cv2.waitKey
        break

# When everything
cap.release()
cv2.destroyAllWind
```


demo_opencv [~/Desktop/py_code_2020/demo_opencv] - .../demo_cam.py [demo_opencv]
demo_opencv
demo_cam.py
demo_cam
Project
main.py
demo_cam.py
demo_opencv ~/Desktop/py_code
import numpy as np
frame
Ext
Scr
Run:
4: Run
6: TODO
Terminal
Python Console
Event Log
8:12
LF
UTF-8
4 spaces
Python 3.7 (demo_opencv)

Porgramming Language I

- 1951 - Regional Assembly Language
- 1952 - Autocode
- 1954 - FORTRAN **
- 1954 - IPL (LISP的先驅)
- 1955 - FLOW-MATIC (COBOL的先驅)
- 1957 - COMTRAN (COBOL的先驅)
- 1958 - LISP **
- 1958 - ALGOL 58
- 1959 - FACT (COBOL的先驅)
- 1959 - COBOL **
- 1962 - APL
- 1962 - Simula
- 1962 - SNOBOL
- 1963 - CPL (C的先驅)
- 1964 - BASIC
- 1964 - PL/I
- 1967 - BCPL (C的先驅)
-

**有三個現代程式語言於1950年代被設計出來

這三者所衍生的語言直到今日仍舊廣泛地被採用

Porgramming Language II

確立了基礎範式

- 1968 - Logo
- 1970 - Pascal
- 1970 - Forth
- 1972 - C語言
- 1972 - Smalltalk
- 1972 - Prolog
- 1973 - ML
- 1975 - Scheme
- 1978 - SQL (起先只是一種查詢語言，擴充之後也具備了程式結構)
-

Porgramming Language III

1980年代：增強、模組、效能

- 1980 - Ada
- 1983 - C++ (就像有類別的C)
- 1984 - Common Lisp
- 1985 - Eiffel
- 1986 - Erlang
- 1987 - Perl
- 1988 - Tcl
- 1989 - FL (Backus)
-

C++合併了物件導向以及系統程式設計

Programming Languages for Internet

提升程式設計師的生產力

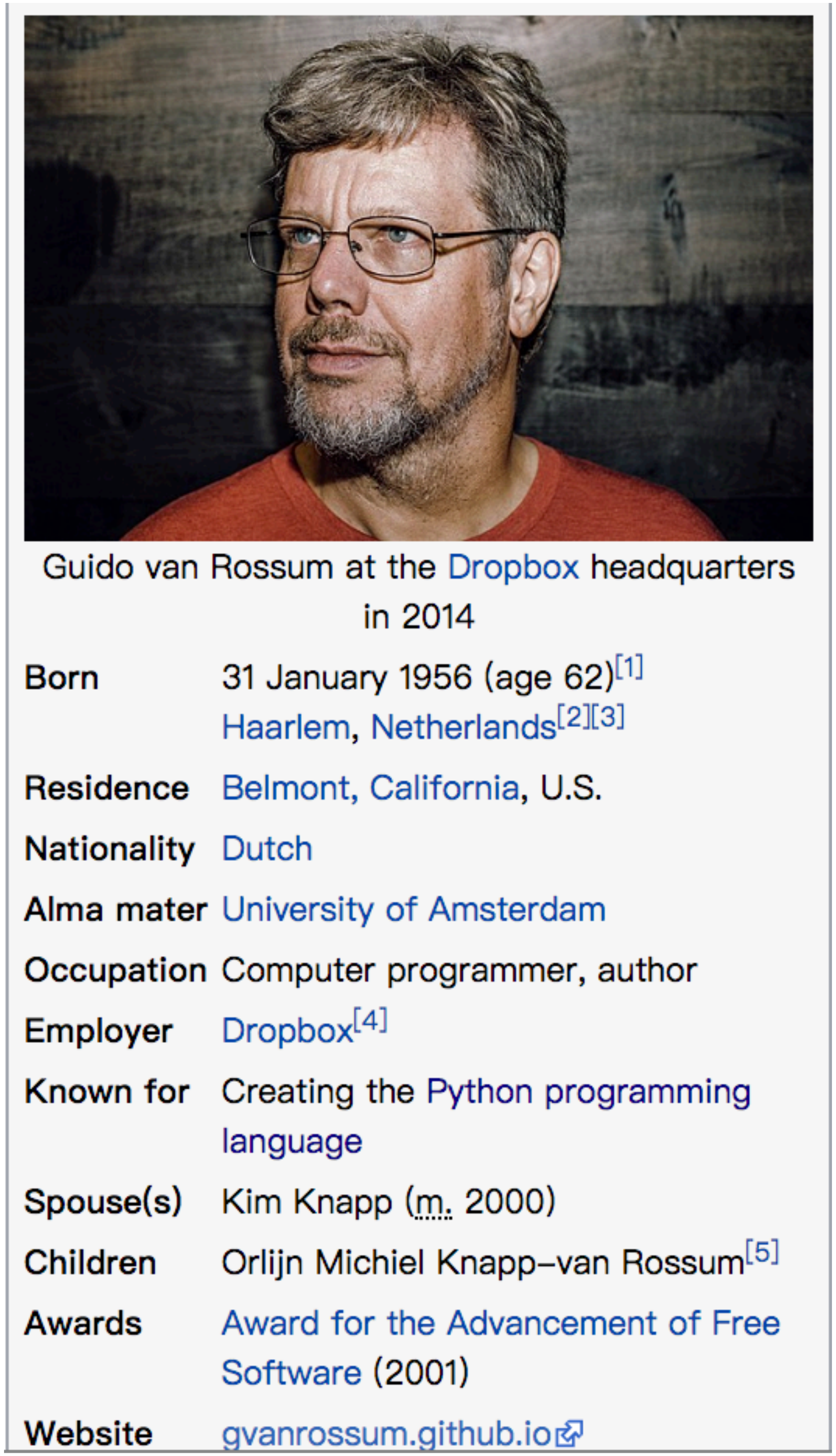
- 1990 - Haskell
- 1991 - Python
- 1991 - Visual Basic
- 1993 - Ruby
- 1993 - Lua
- 1994 - CLOS (part of ANSI Common Lisp)
- 1995 - Java
- 1995 - Delphi (Object Pascal)
- 1995 - JavaScript
- 1995 - PHP
- 1997 - REBOL
- 1999 - D
-

現今的趨勢

- 元件導向(component-oriented)軟體開發
- 更重視分散式及移動式的應用

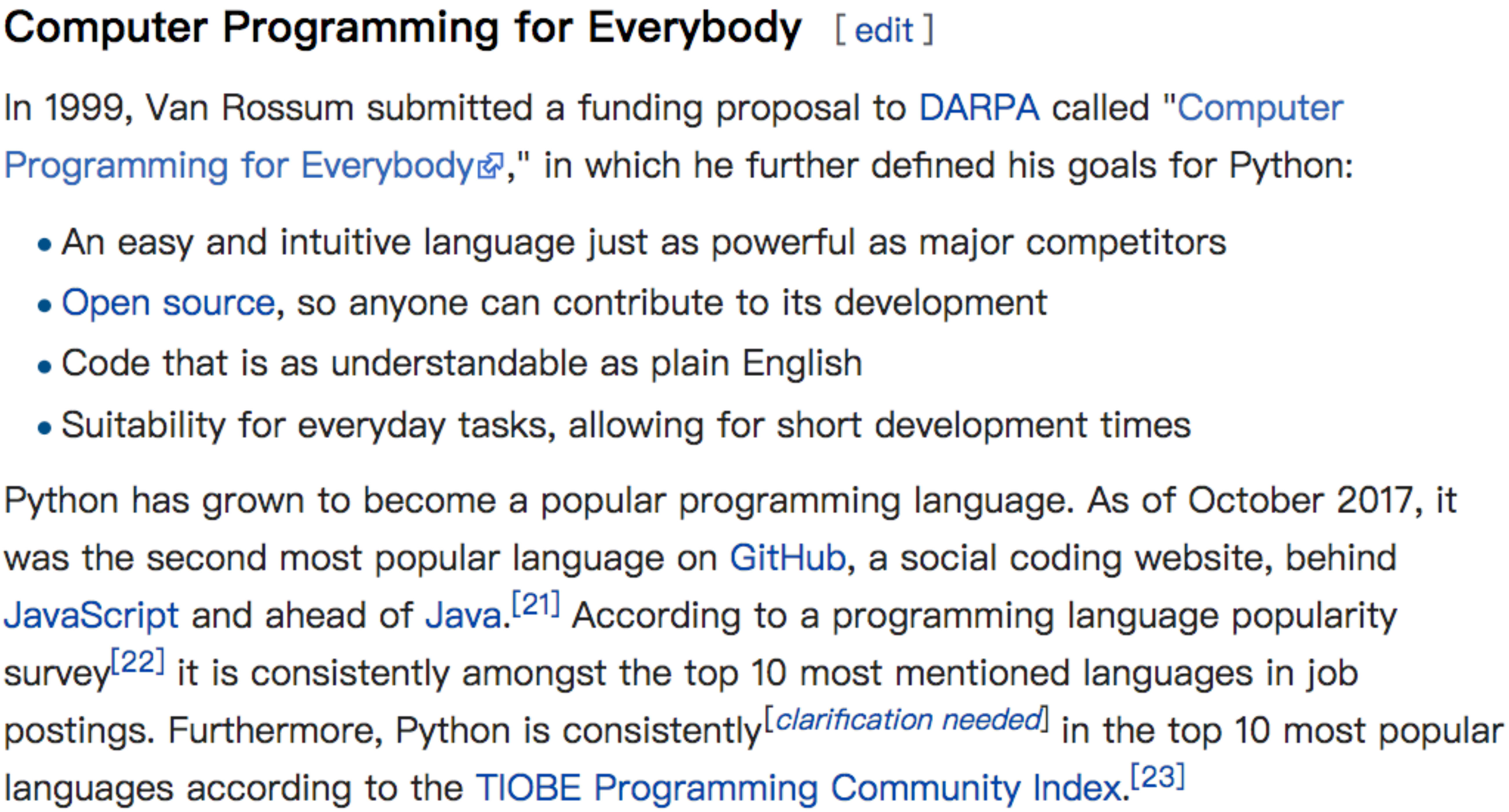
- 2001 - C#
- 2001 - Visual Basic .NET
- 2002 - F#
- 2003 - Scala
- 2003 - Factor
- 2006 - Windows PowerShell
- 2007 - Clojure
- 2009 - Go
- 2014 - Swift (程式語言)
-

Python



Guido van Rossum at the Dropbox headquarters in 2014

Born	31 January 1956 (age 62)[1] Haarlem, Netherlands[2][3]
Residence	Belmont, California, U.S.
Nationality	Dutch
Alma mater	University of Amsterdam
Occupation	Computer programmer, author
Employer	Dropbox[4]
Known for	Creating the Python programming language
Spouse(s)	Kim Knapp (m. 2000)
Children	Orlijn Michiel Knapp–van Rossum[5]
Awards	Award for the Advancement of Free Software (2001)
Website	gvanrossum.github.io

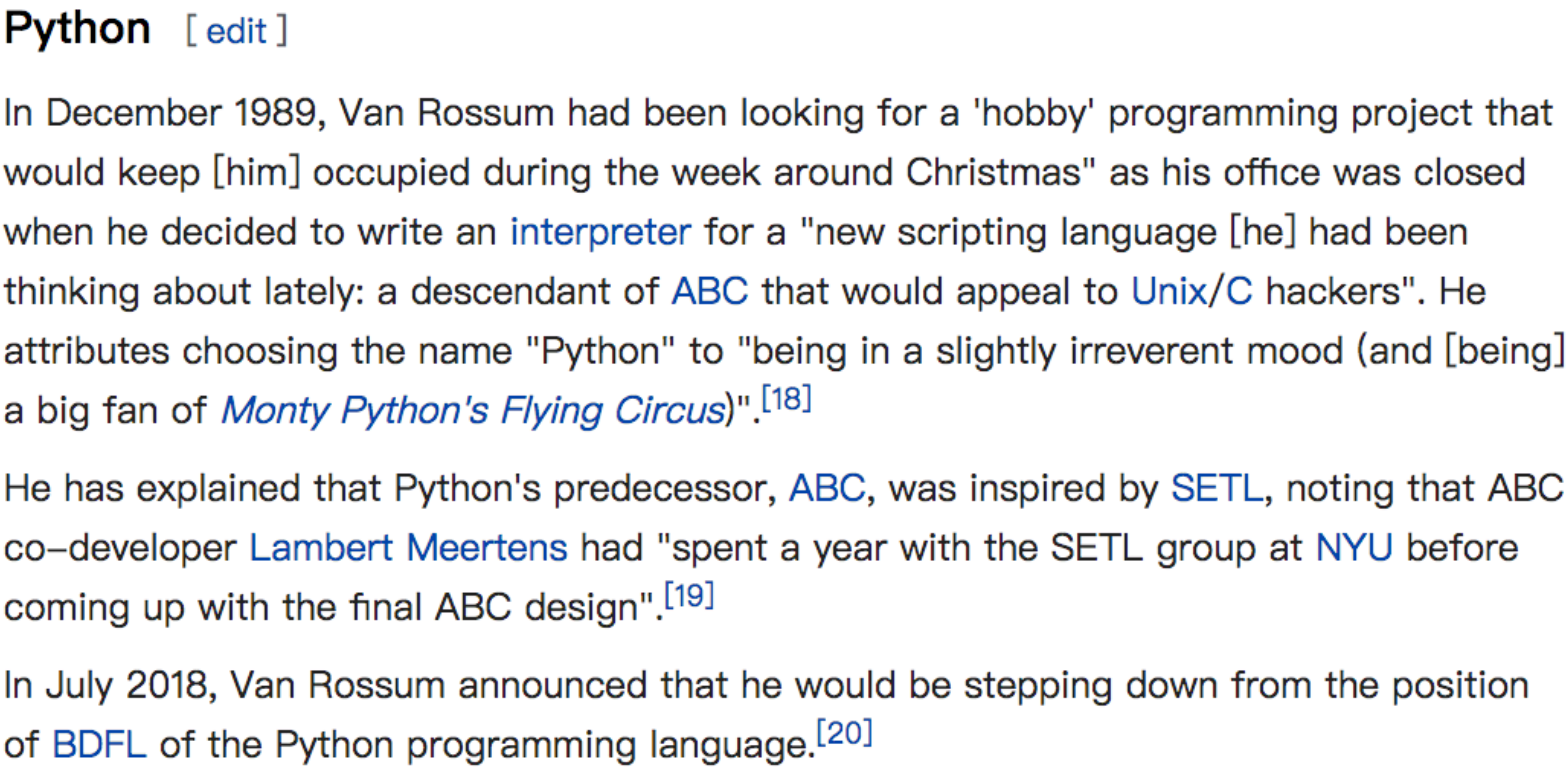


Computer Programming for Everybody [edit]

In 1999, Van Rossum submitted a funding proposal to DARPA called "Computer Programming for Everybody," in which he further defined his goals for Python:

- An easy and intuitive language just as powerful as major competitors
- Open source, so anyone can contribute to its development
- Code that is as understandable as plain English
- Suitability for everyday tasks, allowing for short development times

Python has grown to become a popular programming language. As of October 2017, it was the second most popular language on GitHub, a social coding website, behind JavaScript and ahead of Java.[21] According to a programming language popularity survey[22] it is consistently amongst the top 10 most mentioned languages in job postings. Furthermore, Python is consistently[*clarification needed*] in the top 10 most popular languages according to the TIOBE Programming Community Index.[23]



Python [edit]

In December 1989, Van Rossum had been looking for a 'hobby' programming project that would keep [him] occupied during the week around Christmas" as his office was closed when he decided to write an interpreter for a "new scripting language [he] had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers". He attributes choosing the name "Python" to "being in a slightly irreverent mood (and [being] a big fan of *Monty Python's Flying Circus*)".[18]

He has explained that Python's predecessor, ABC, was inspired by SETL, noting that ABC co-developer Lambert Meertens had "spent a year with the SETL group at NYU before coming up with the final ABC design".[19]

In July 2018, Van Rossum announced that he would be stepping down from the position of BDFL of the Python programming language.[20]

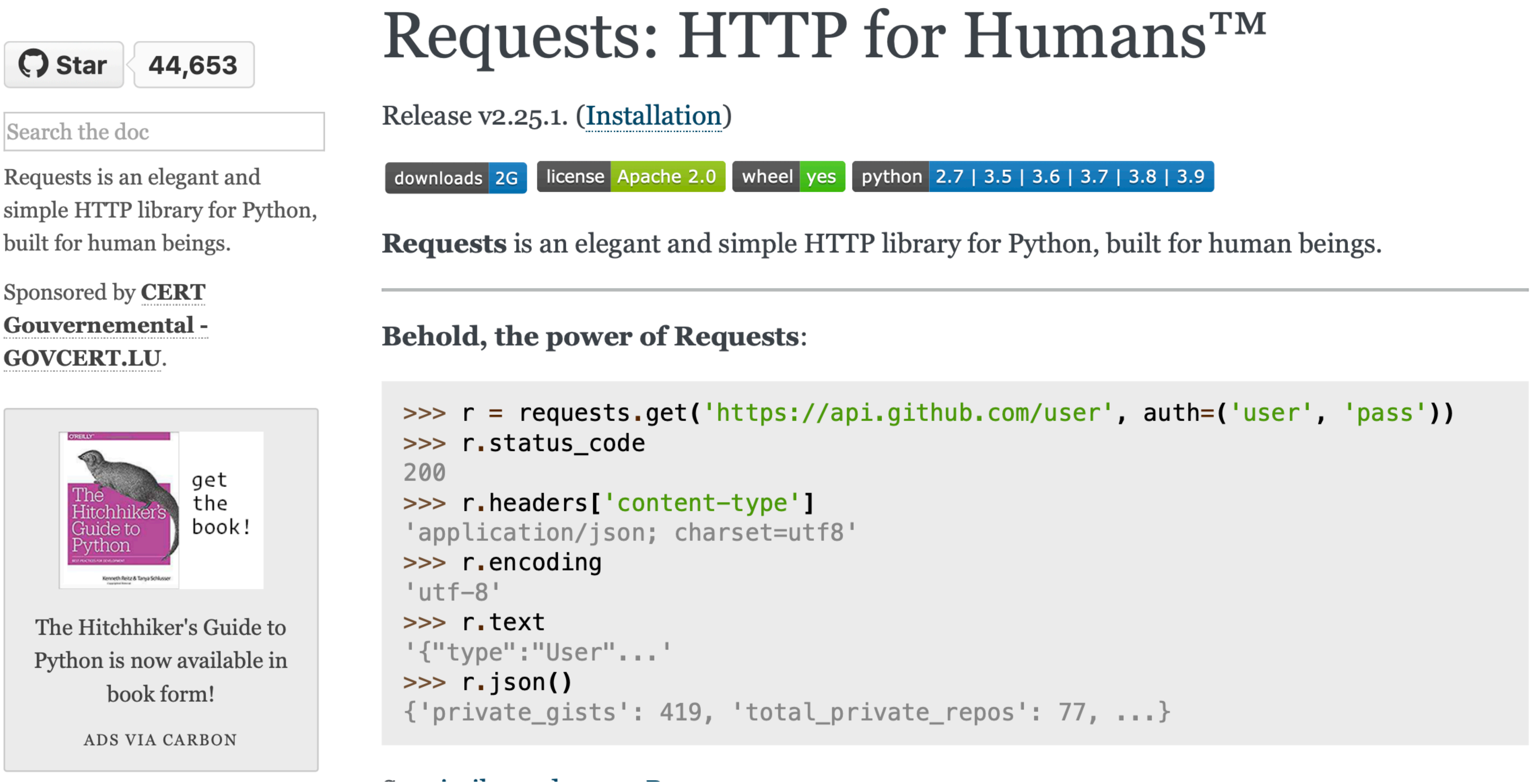
Python Tutorial

Release 3.7.0

Guido van Rossum
and the Python development team

Python modules for Web development

Web development



Requests is an elegant and simple HTTP library for Python, built for human beings.

Sponsored by **CERT Gouvernemental - GOVCERT.LU**.

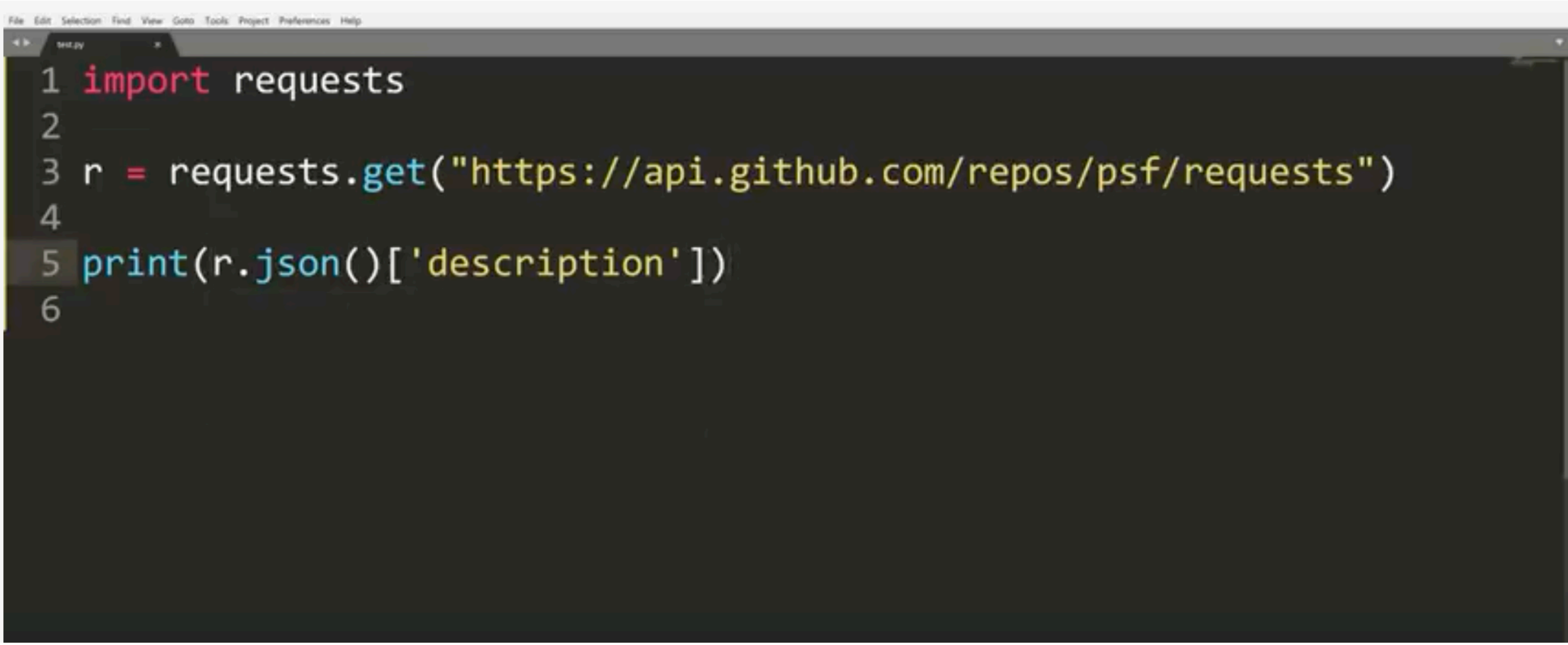
Requests: HTTP for Humans™

Release v2.25.1. (Installation)

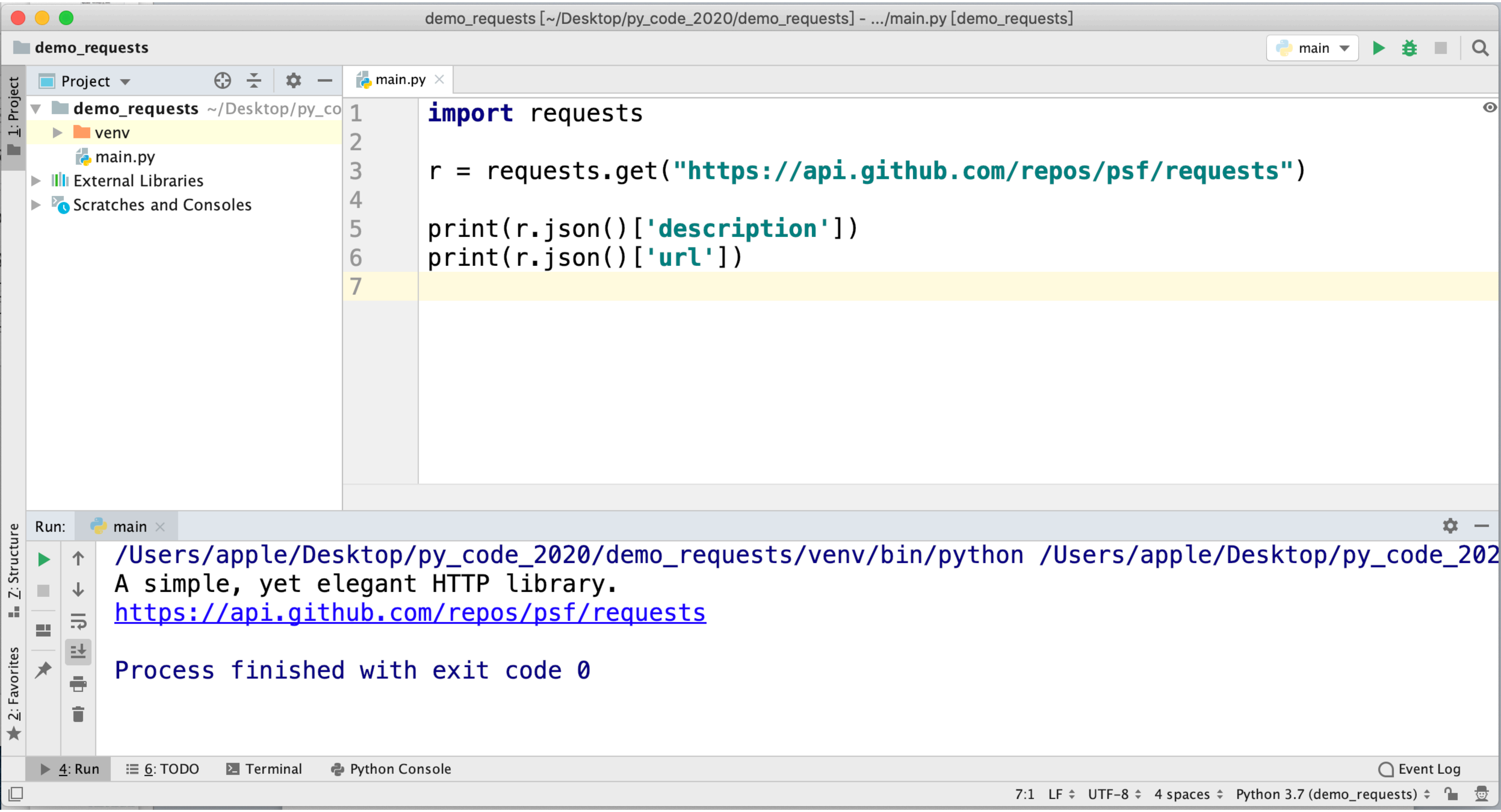
Requests is an elegant and simple HTTP library for Python, built for human beings.

Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type":"User"...'
>>> r.json()
{'private_gists': 419, 'total_private_repos': 77, ...}
```

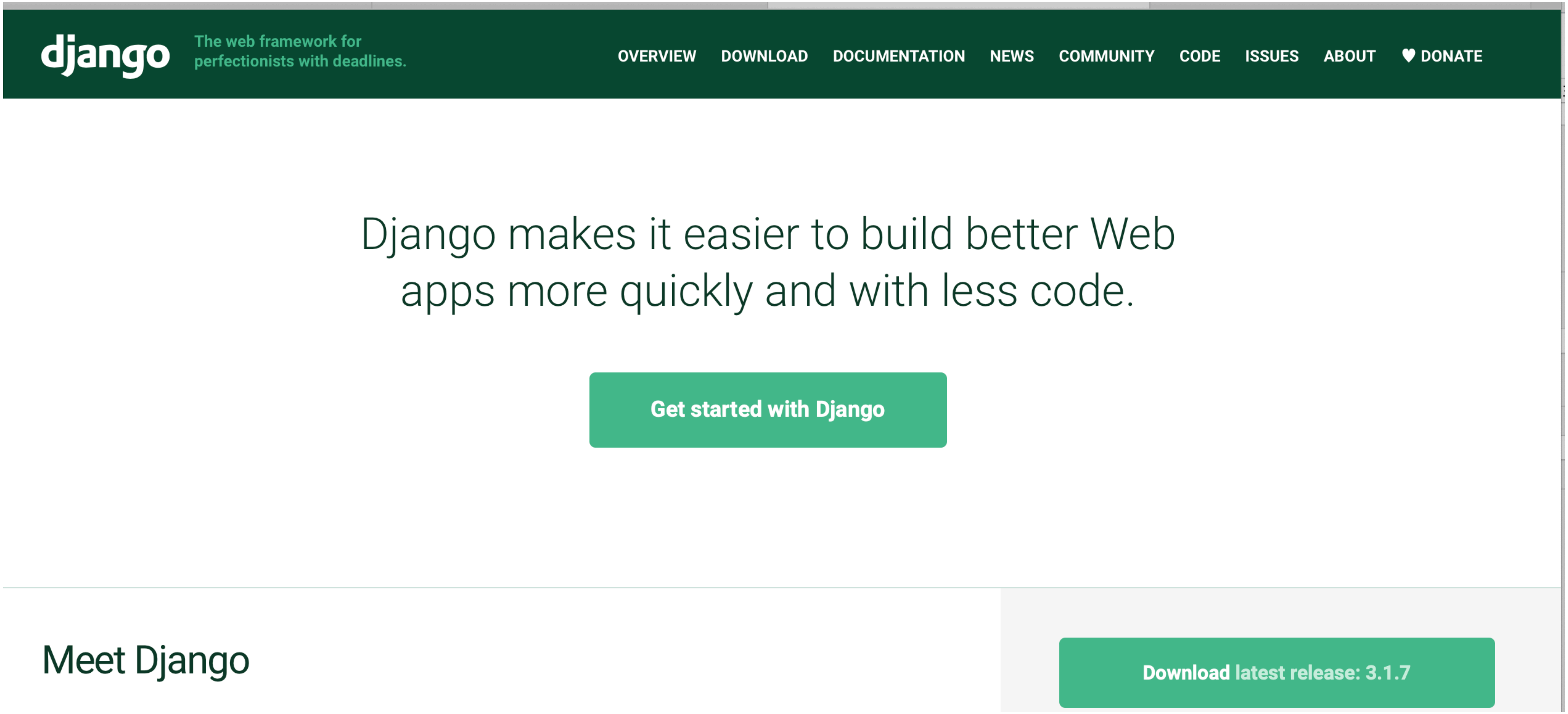


```python
import requests

r = requests.get("https://api.github.com/repos/psf/requests")

print(r.json()['description'])

```



```
import requests

r = requests.get("https://api.github.com/repos/psf/requests")

print(r.json()['description'])
print(r.json()['url'])
```

```
/Users/apple/Desktop/py_code_2020/demo_requests/venv/bin/python /Users/apple/Desktop/py_code_202
A simple, yet elegant HTTP library.
https://api.github.com/repos/psf/requests

Process finished with exit code 0
```

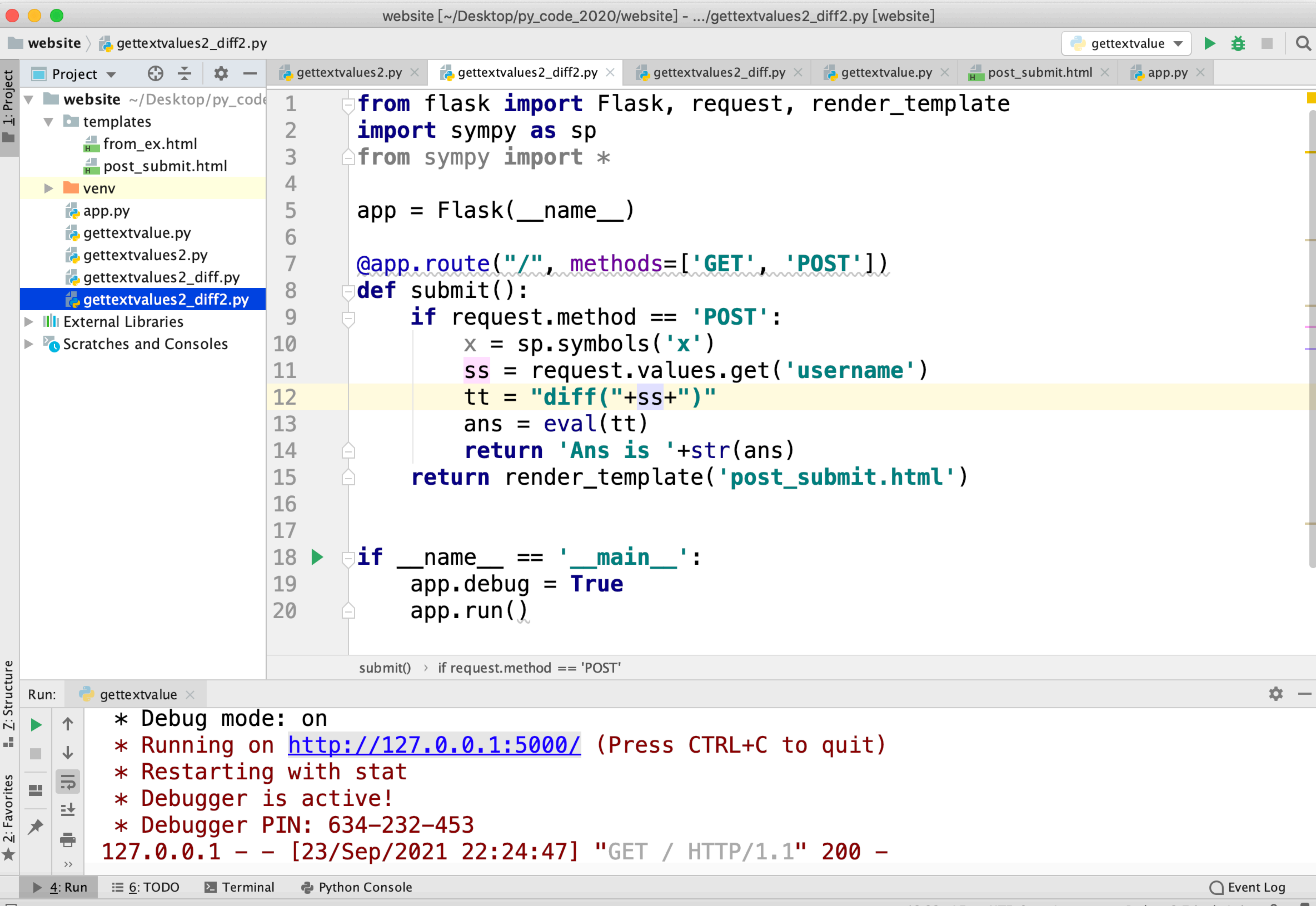


The web framework for perfectionists with deadlines.

Django makes it easier to build better Web apps more quickly and with less code.

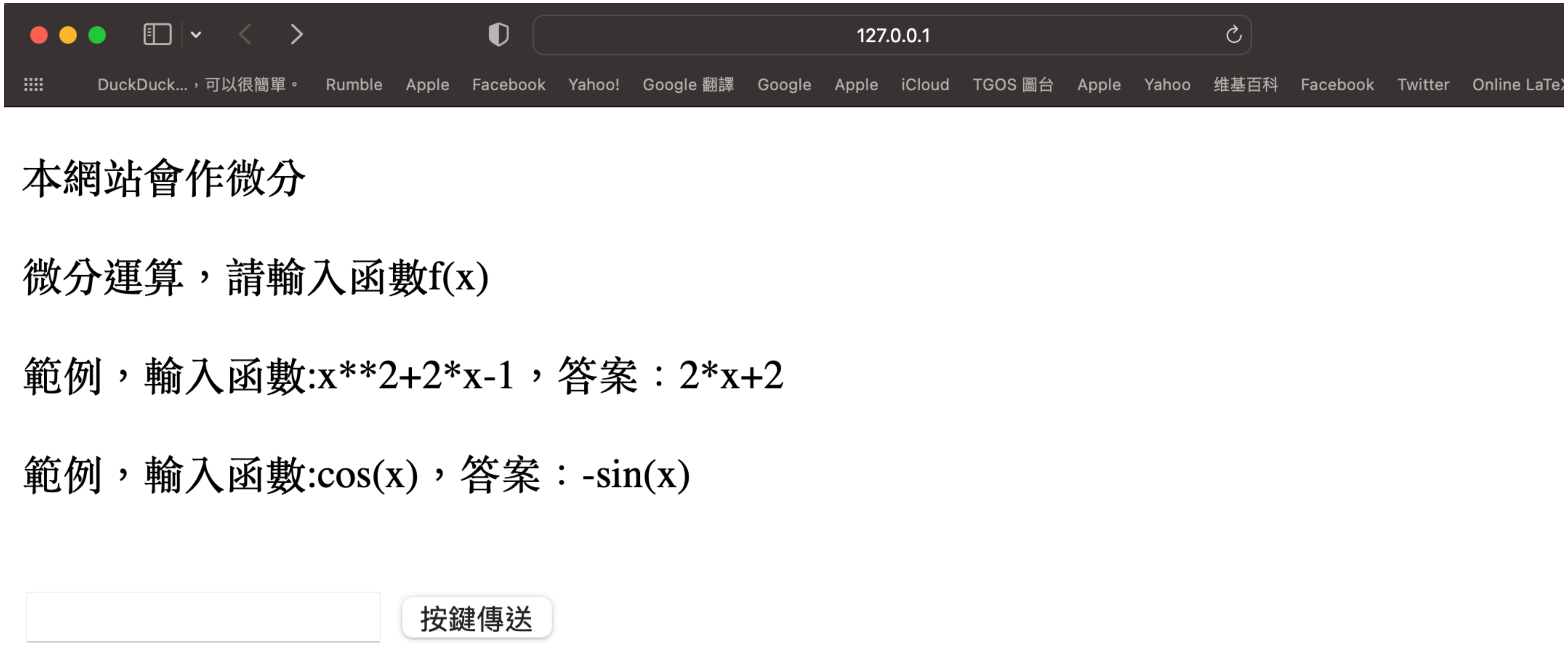
Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.



```
from flask import Flask, request, render_template
import sympy as sp
from sympy import *

app = Flask(__name__)

@app.route("/", methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        x = sp.symbols('x')
        ss = request.values.get('username')
        tt = "diff("+ss+")"
        ans = eval(tt)
        return 'Ans is '+str(ans)
    return render_template('post_submit.html')


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

```
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 634-232-453
127.0.0.1 - - [23/Sep/2021 22:24:47] "GET / HTTP/1.1" 200 -
```

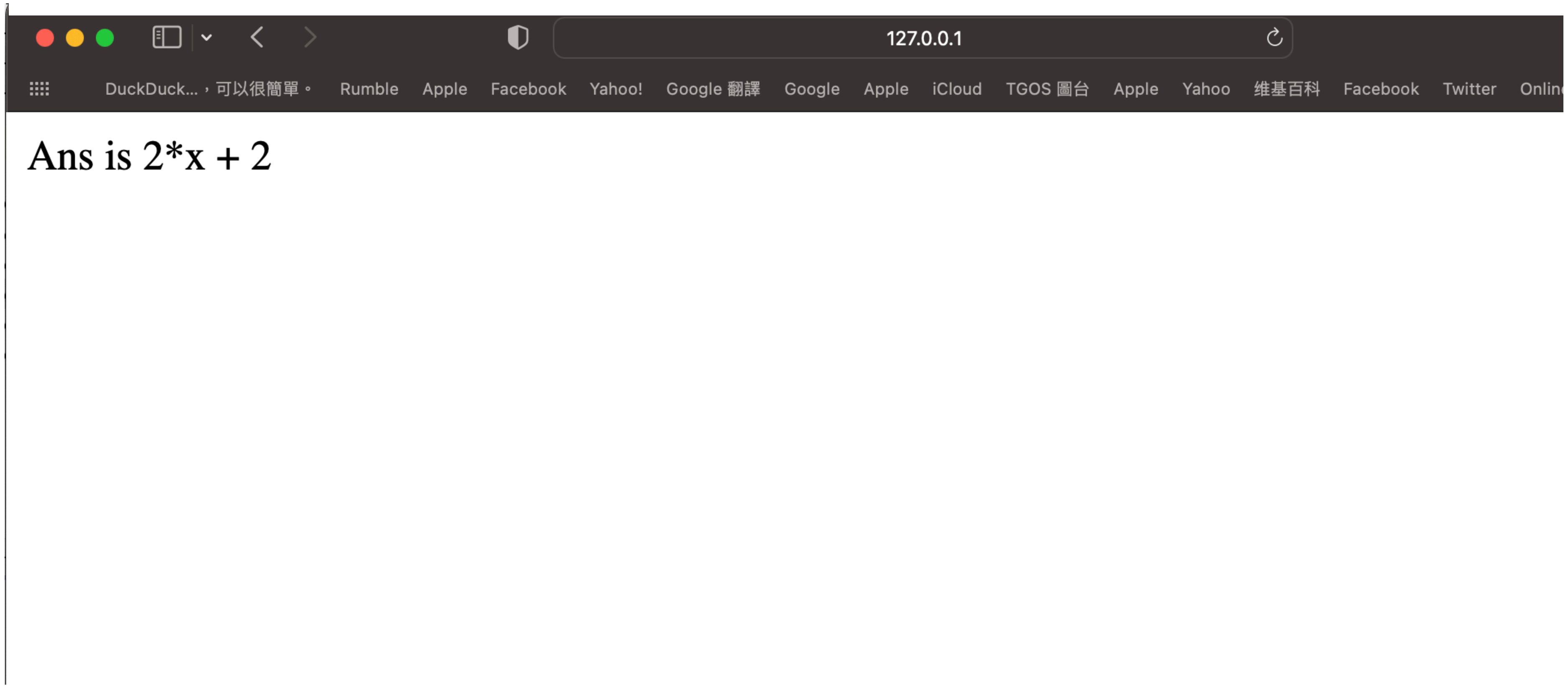


本網站會作微分

微分運算，請輸入函數f(x)

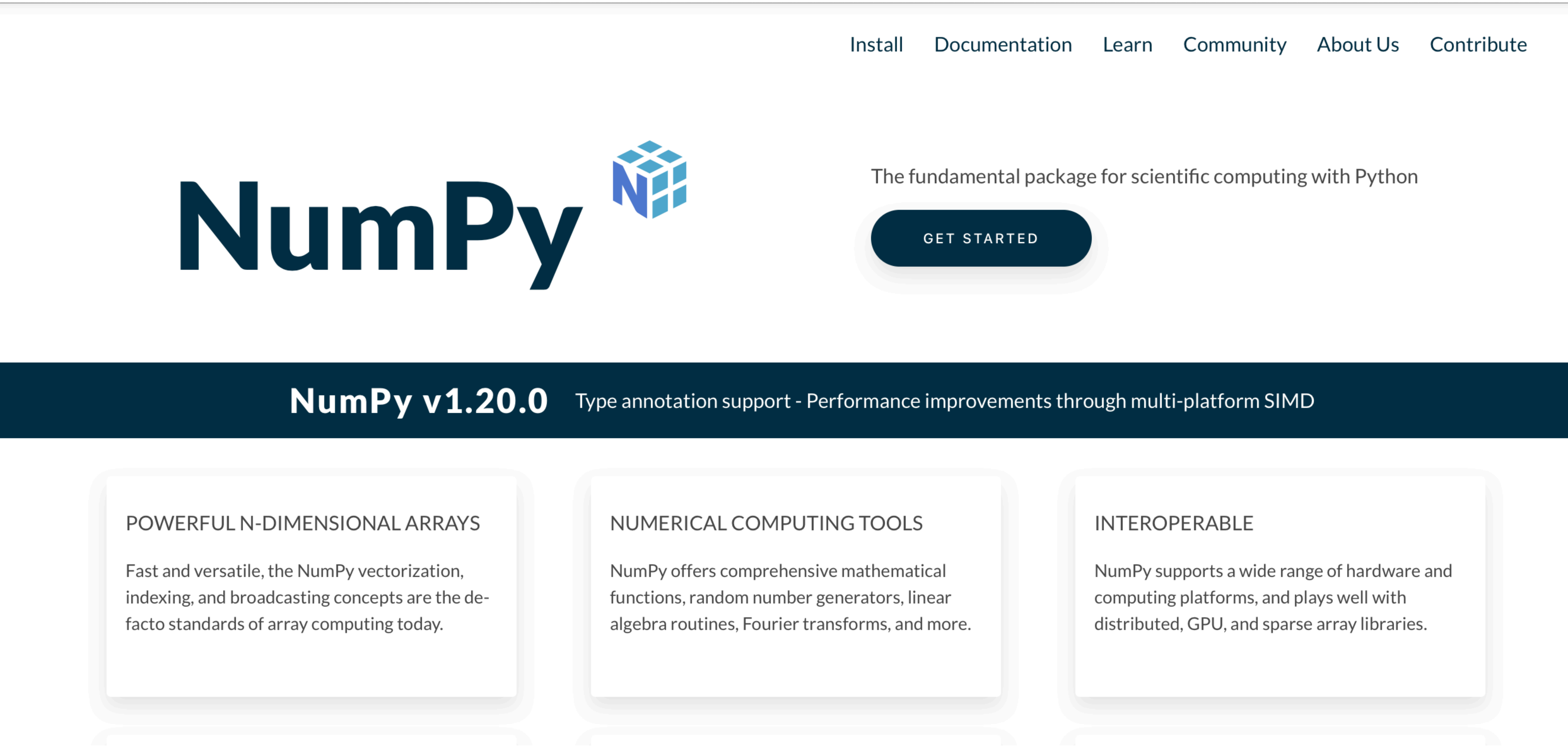
範例，輸入函數:x**2+2*x-1，答案：2*x+2

範例，輸入函數:cos(x)，答案：-sin(x)



Ans is 2*x + 2

Data Science



The fundamental package for scientific computing with Python

NumPy v1.20.0 Type annotation support - Performance improvements through multi-platform SIMD

POWERFUL N-DIMENSIONAL ARRAYS

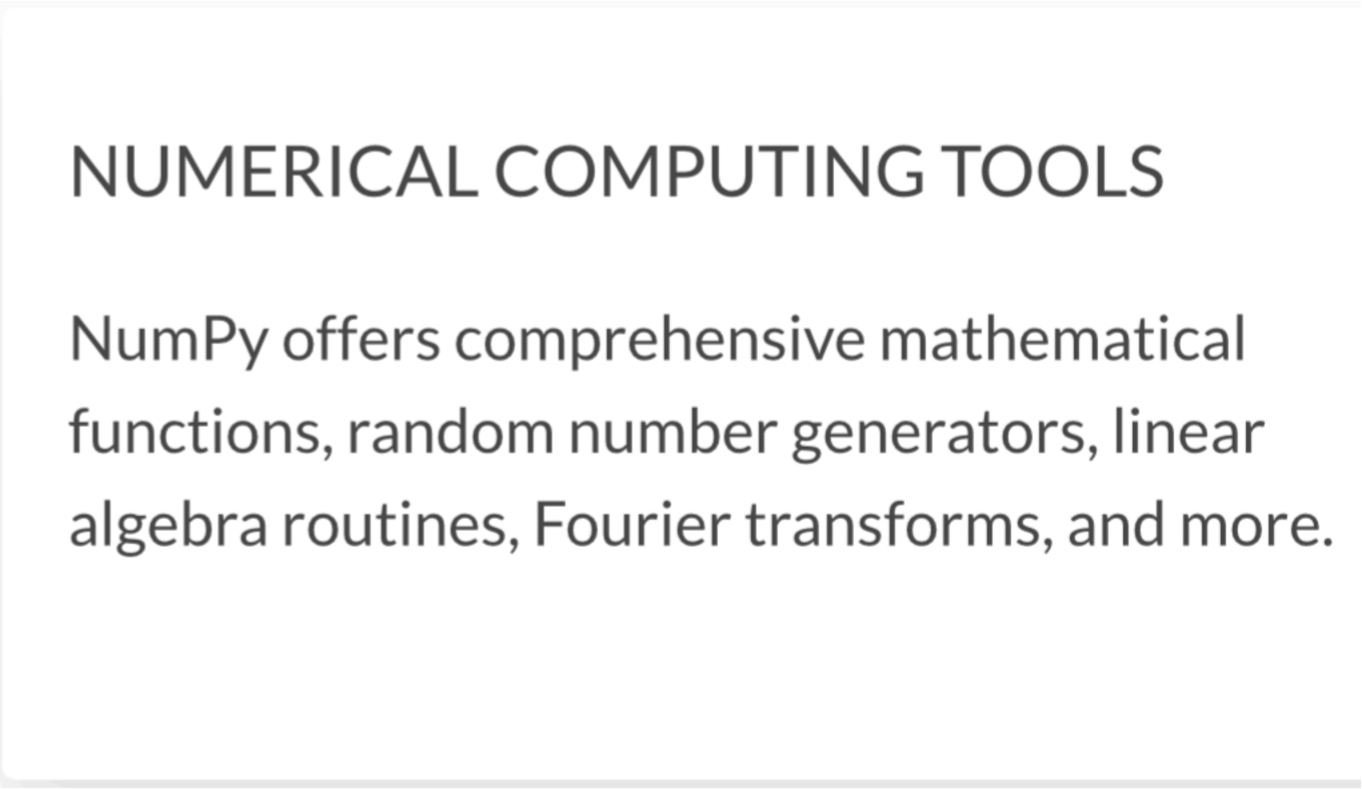
Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

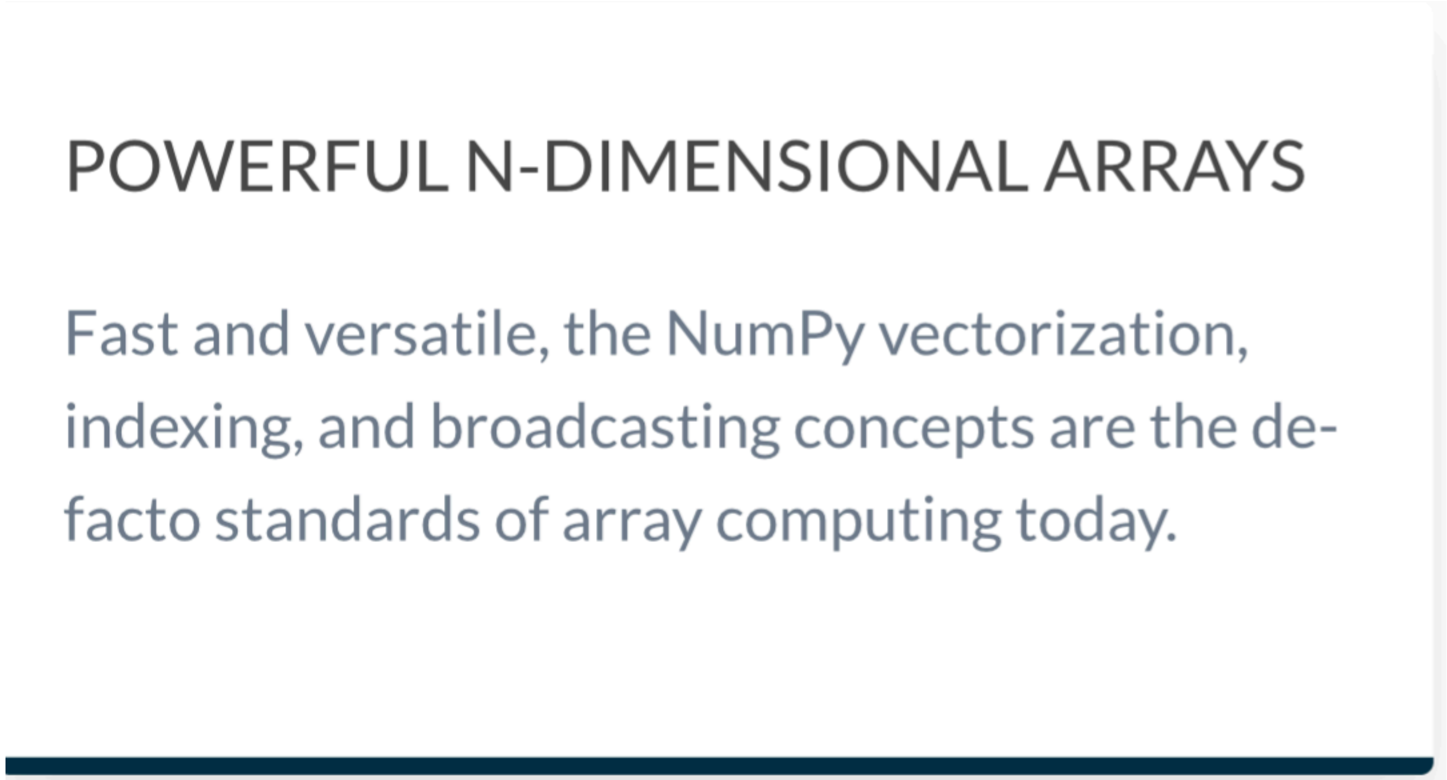
INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.



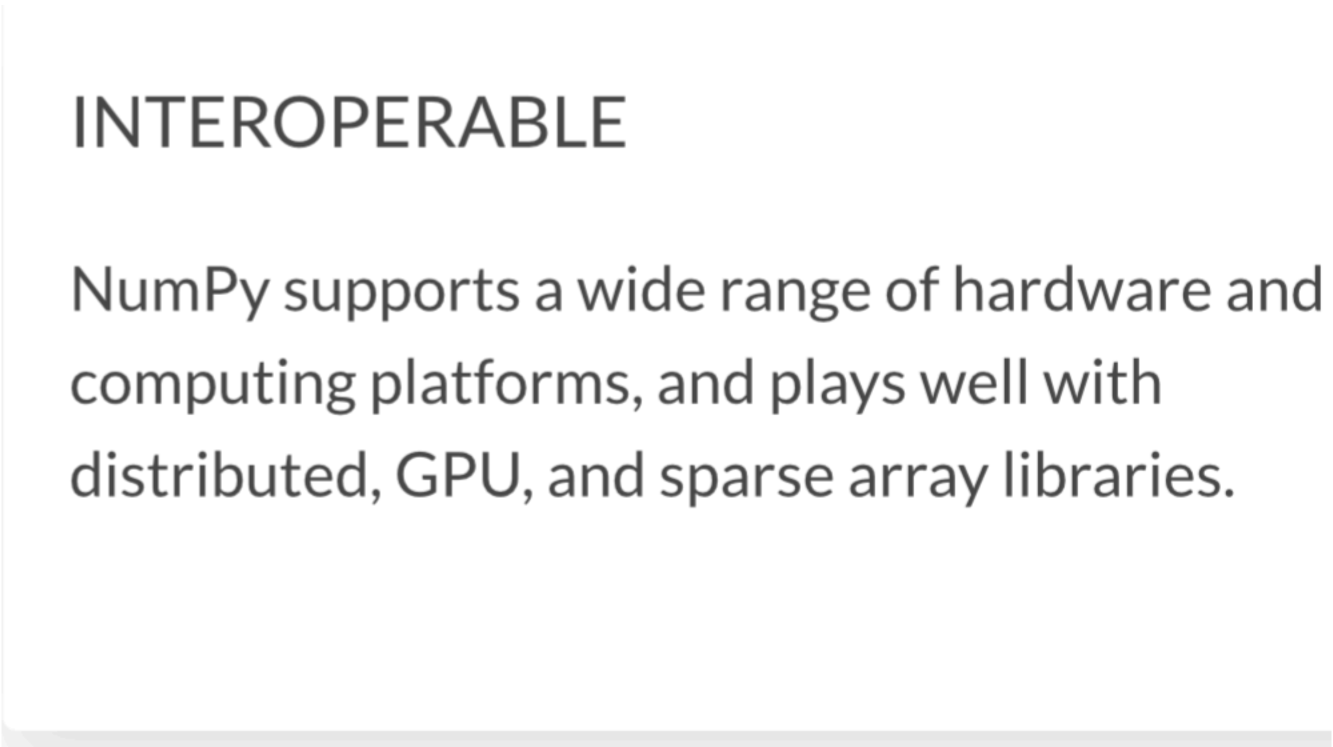
NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.



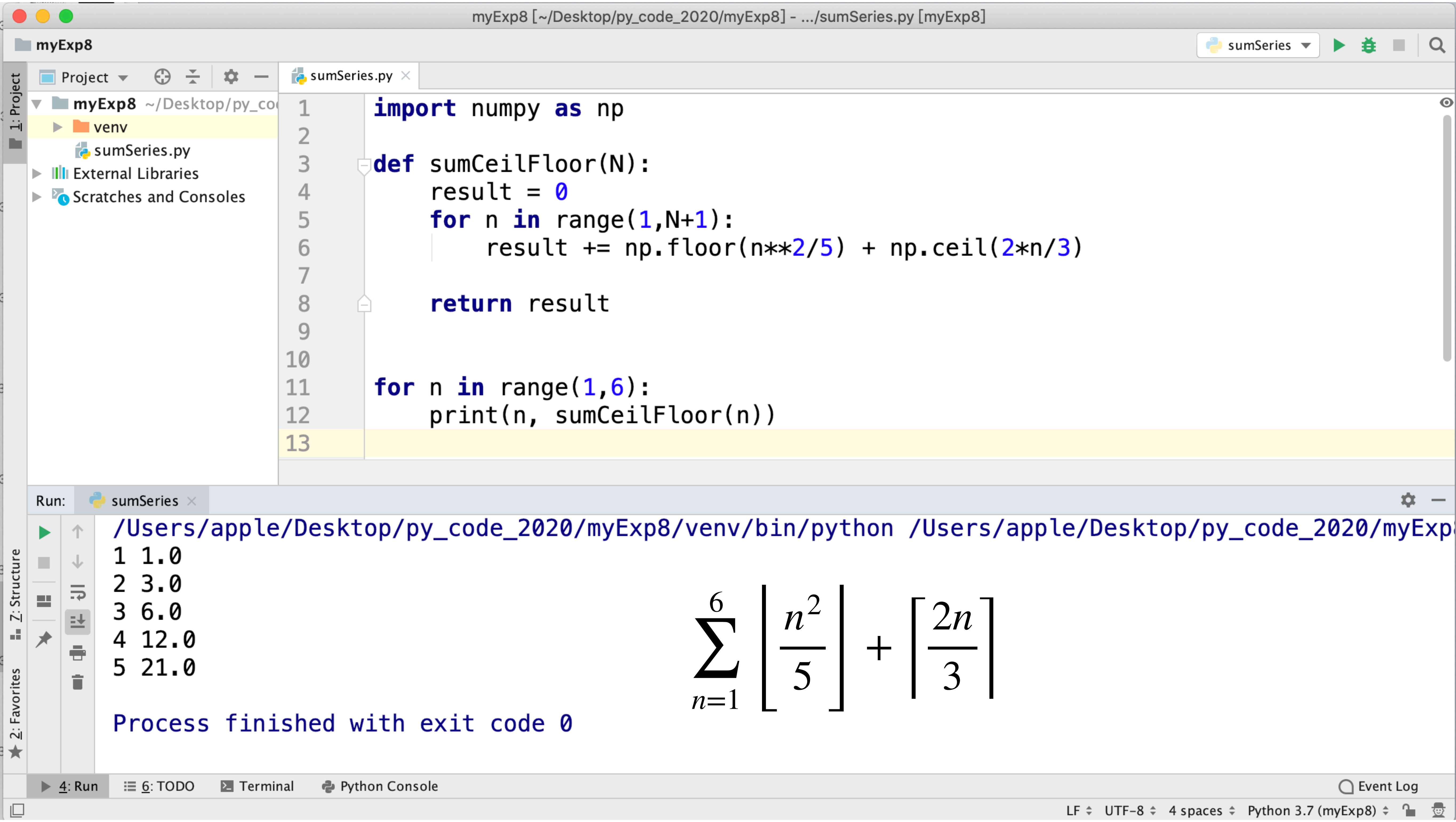
POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.



INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

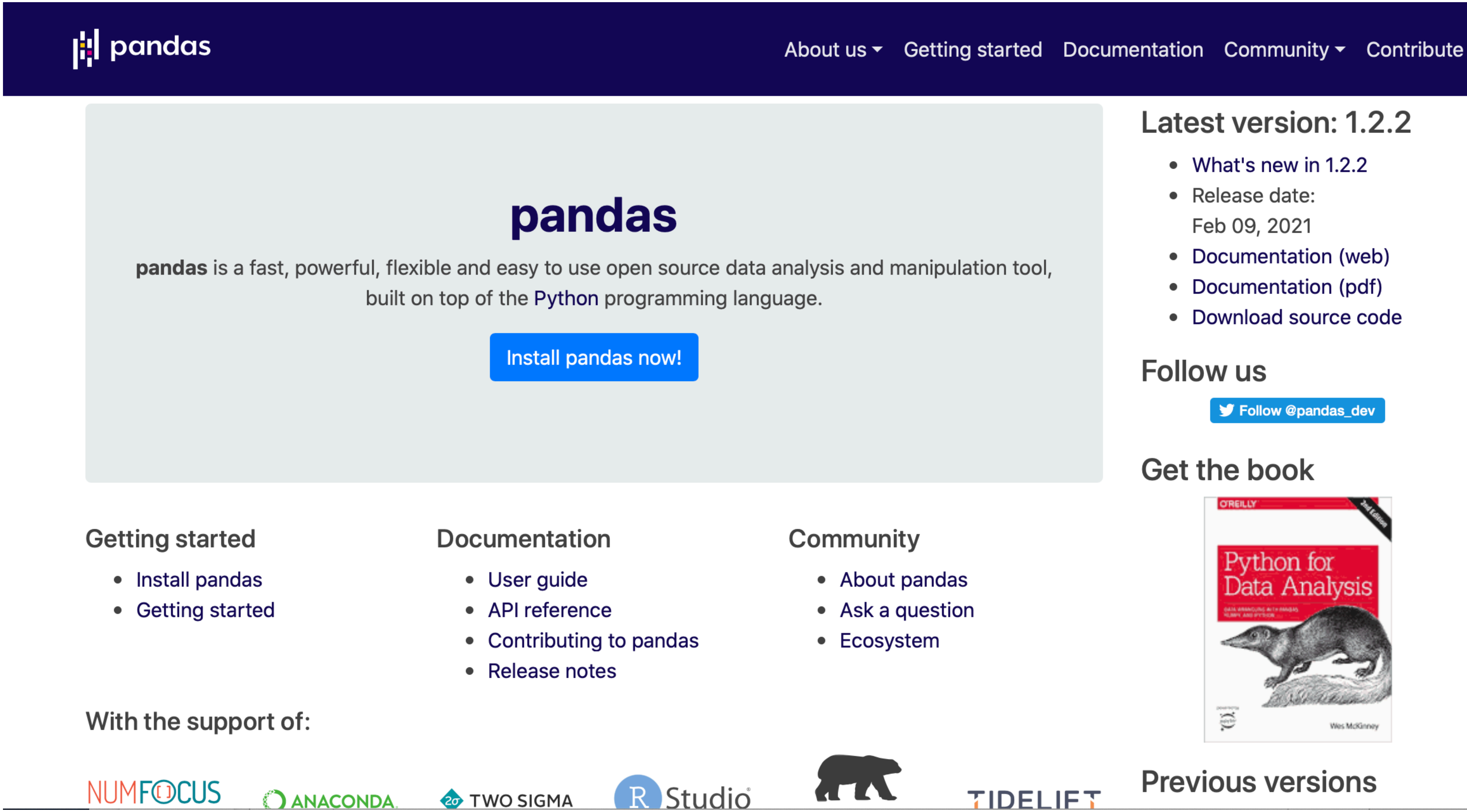


sumSeries.py

```python
import numpy as np

def sumCeilFloor(N):
    result = 0
    for n in range(1,N+1):
        result += np.floor(n**2/5) + np.ceil(2*n/3)

    return result


for n in range(1,6):
    print(n, sumCeilFloor(n))

```

```
/Users/apple/Desktop/py_code_2020/myExp8/venv/bin/python /Users/apple/Desktop/py_code_2020/myExp
1 1.0
2 3.0
3 6.0
4 12.0
5 21.0

Process finished with exit code 0
```

$$\sum_{n=1}^{6} \left\lfloor \frac{n^2}{5} \right\rfloor + \left\lceil \frac{2n}{3} \right\rceil$$



pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Latest version: 1.2.2

- What's new in 1.2.2
- Release date:
  Feb 09, 2021
- Documentation (web)
- Documentation (pdf)
- Download source code

Follow us

Get the book

Getting started

- Install pandas
- Getting started

Documentation

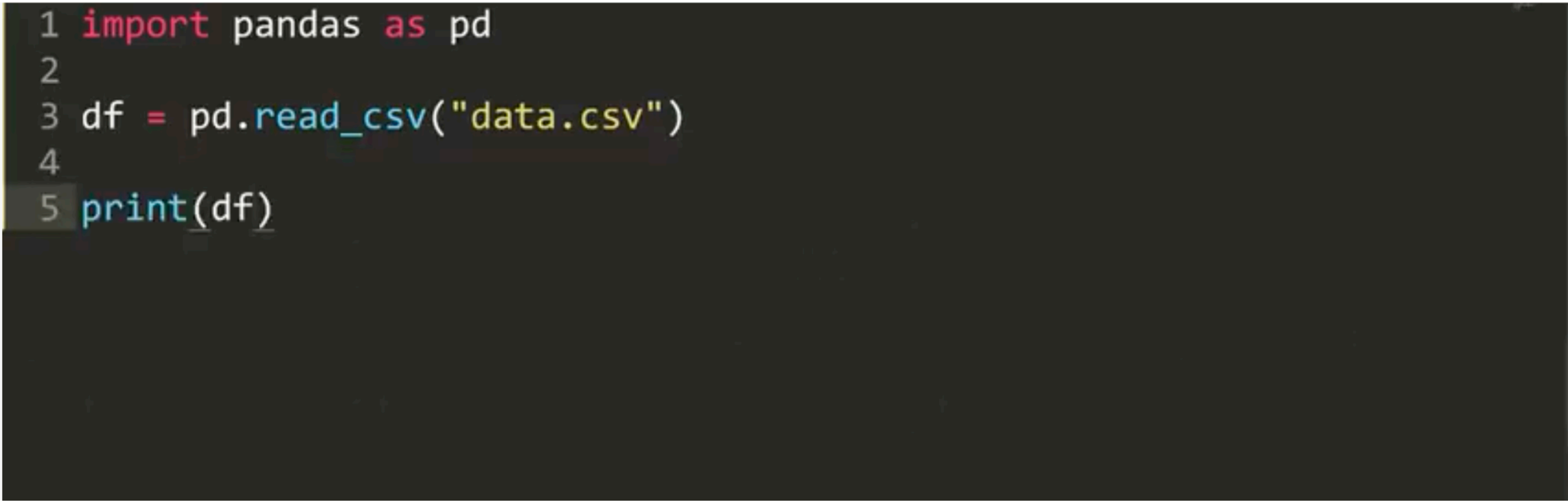
- User guide
- API reference
- Contributing to pandas
- Release notes

Community

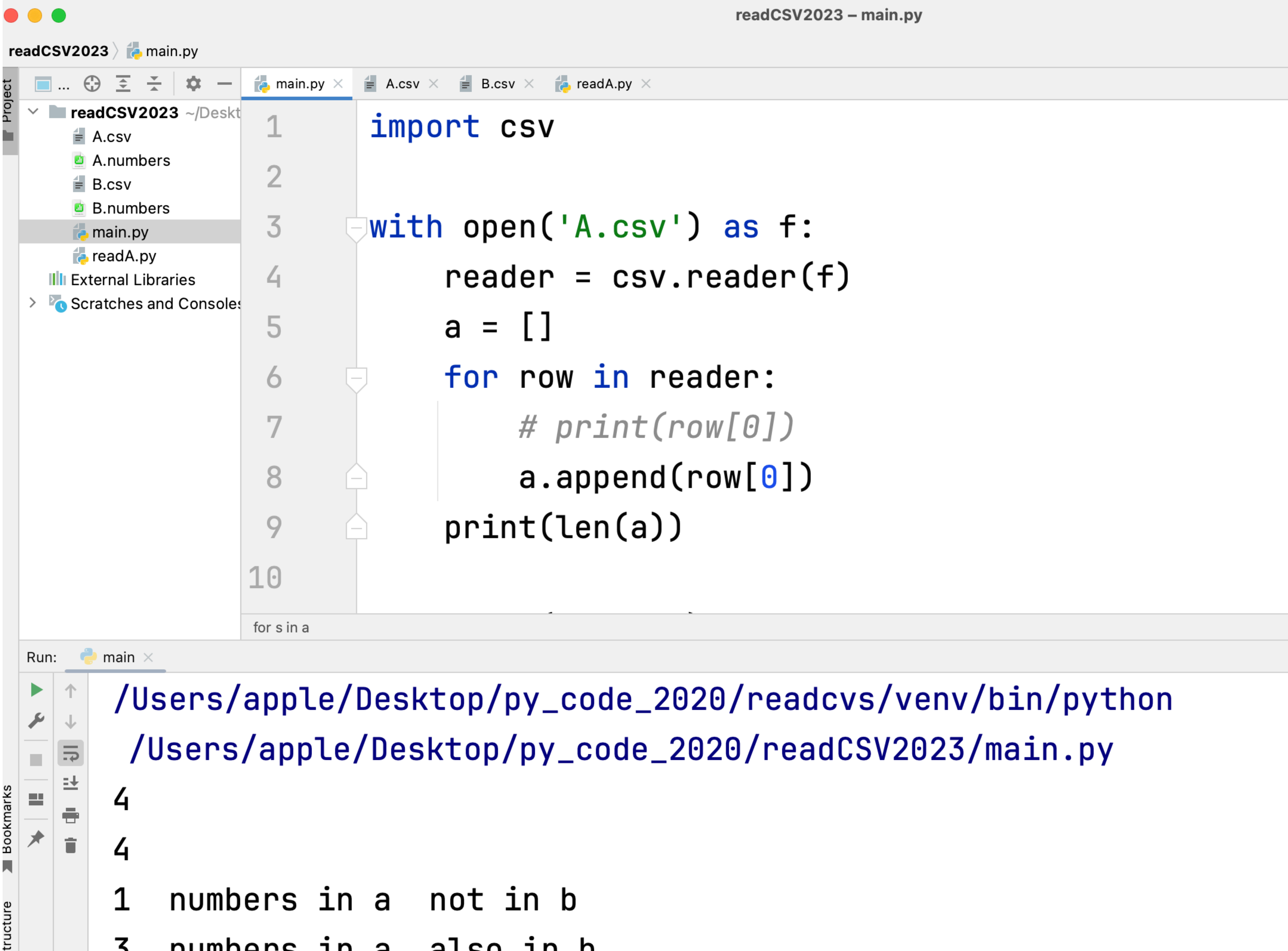
- About pandas
- Ask a question
- Ecosystem

With the support of:

Previous versions



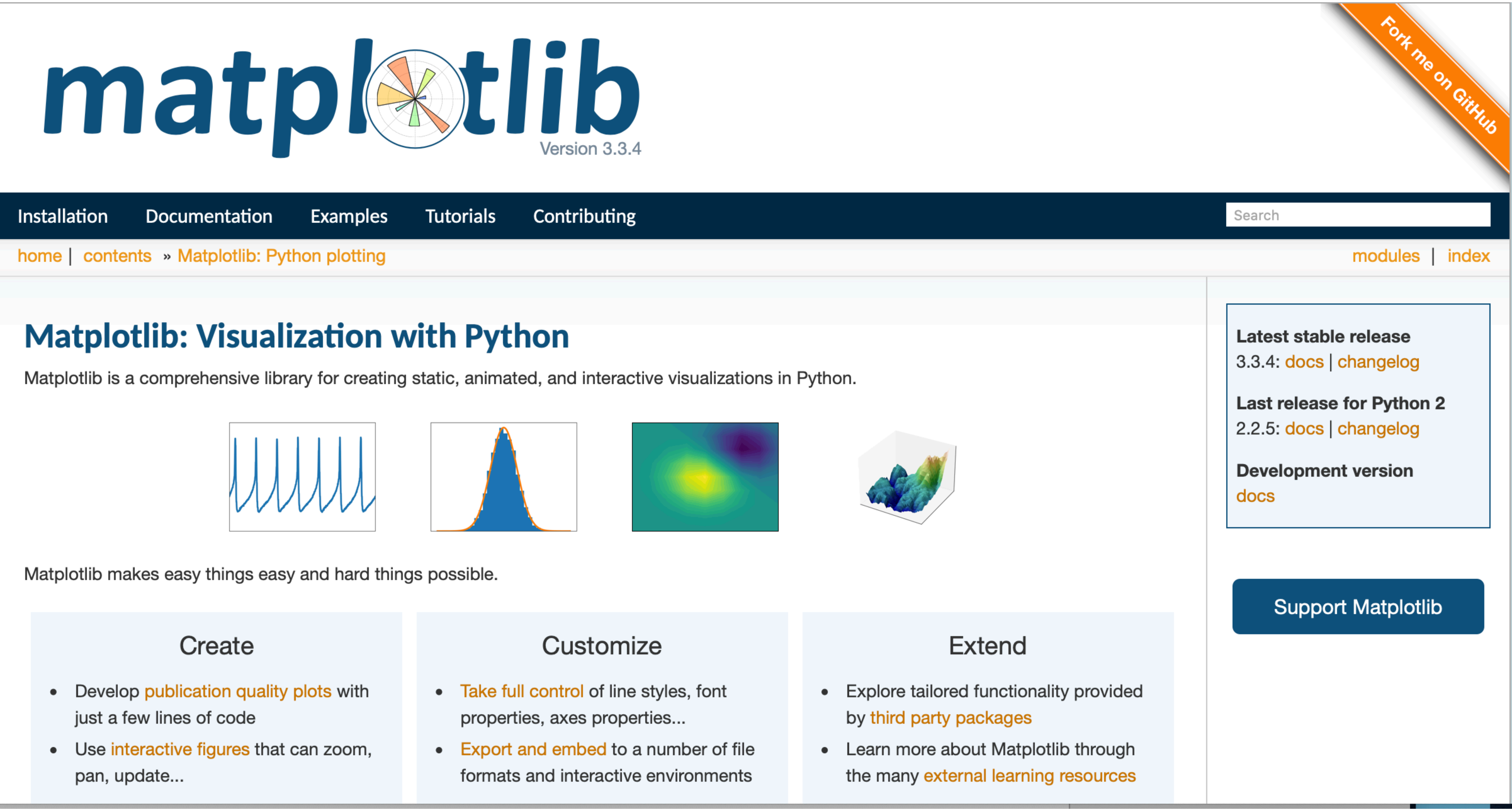
```python
import pandas as pd

df = pd.read_csv("data.csv")

print(df)
```



```python
import csv

with open('A.csv') as f:
    reader = csv.reader(f)
    a = []
    for row in reader:
        # print(row[0])
        a.append(row[0])
    print(len(a))

```

```
/Users/apple/Desktop/py_code_2020/readcvs/venv/bin/python
 /Users/apple/Desktop/py_code_2020/readCSV2023/main.py
4
4
1  numbers in a  not in b
3  numbers in a  also in b
```

Data Visualization



Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Matplotlib makes easy things easy and hard things possible.

Create

- Develop publication quality plots with just a few lines of code
- Use interactive figures that can zoom, pan, update...

Customize

- Take full control of line styles, font properties, axes properties...
- Export and embed to a number of file formats and interactive environments

Extend

- Explore tailored functionality provided by third party packages
- Learn more about Matplotlib through the many external learning resources

Latest stable release

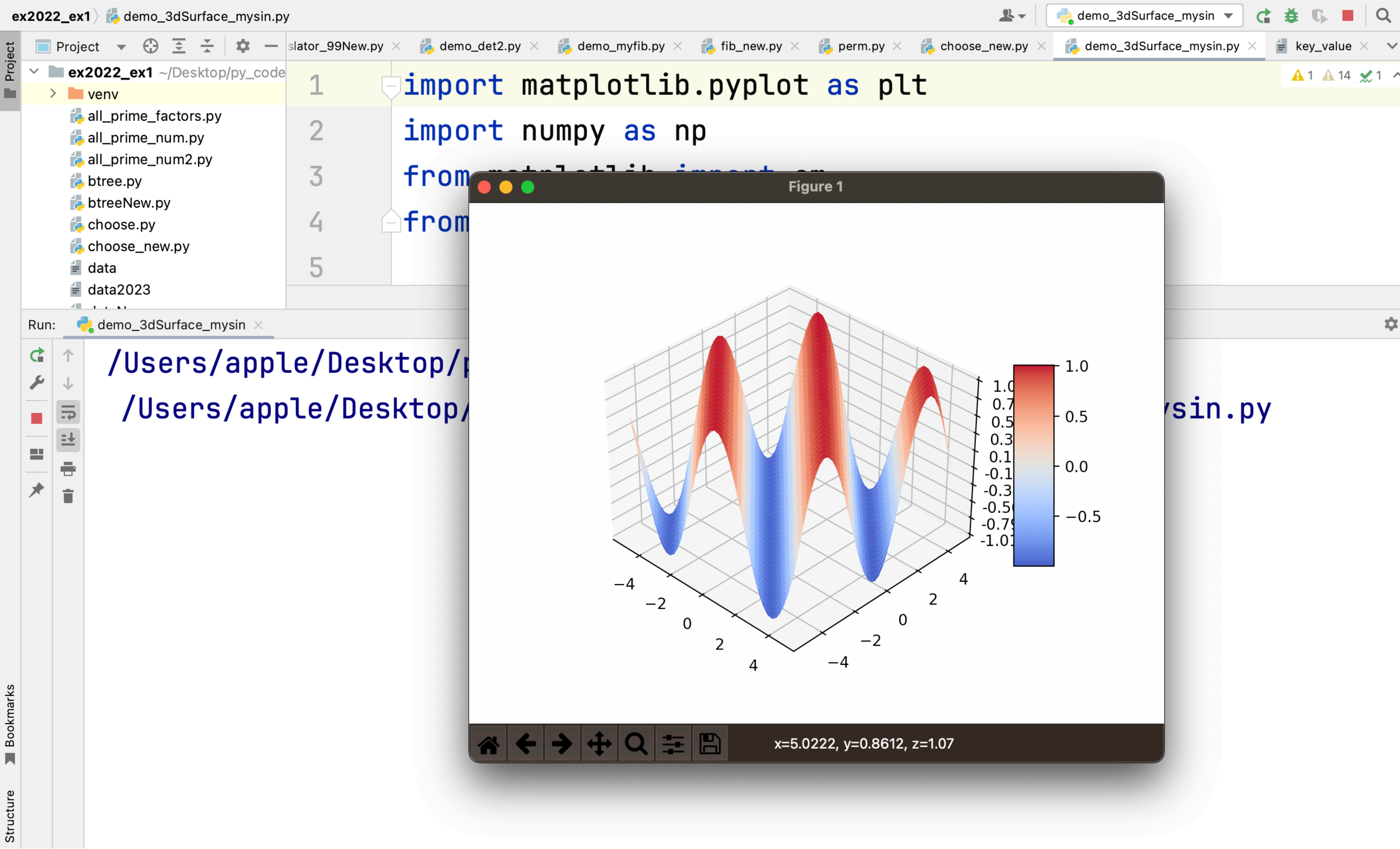
3.3.4: docs | changelog

Last release for Python 2

2.2.5: docs | changelog

Development version

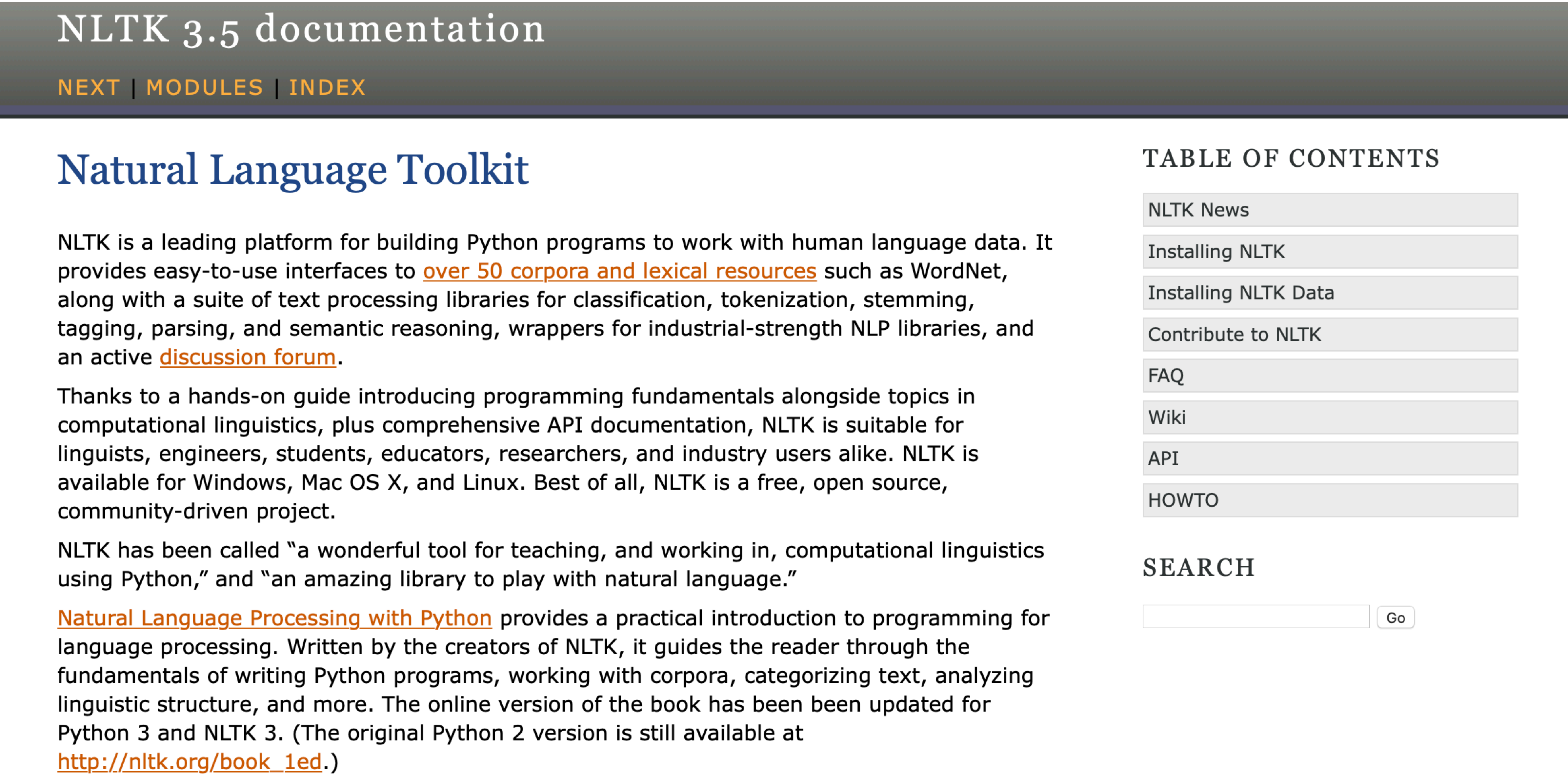
docs



```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import cm
from
```

```
/Users/apple/Desktop/p
 /Users/apple/Desktop/                                         sin.py
```

x=5.0222, y=0.8612, z=1.07



NLTK 3.5 documentation

NEXT | MODULES | INDEX

Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

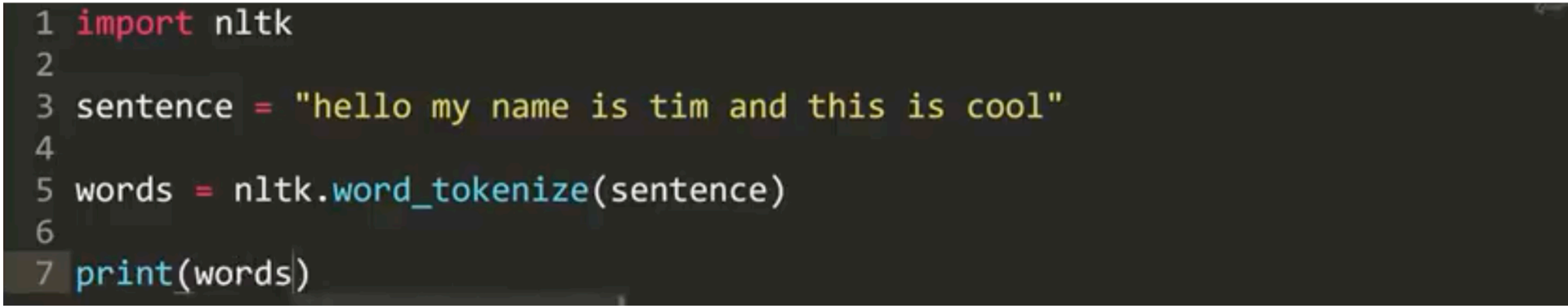
NLTK has been called "a wonderful tool for teaching, and working in, computational linguistics using Python," and "an amazing library to play with natural language."

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The online version of the book has been been updated for Python 3 and NLTK 3. (The original Python 2 version is still available at http://nltk.org/book_1ed.)

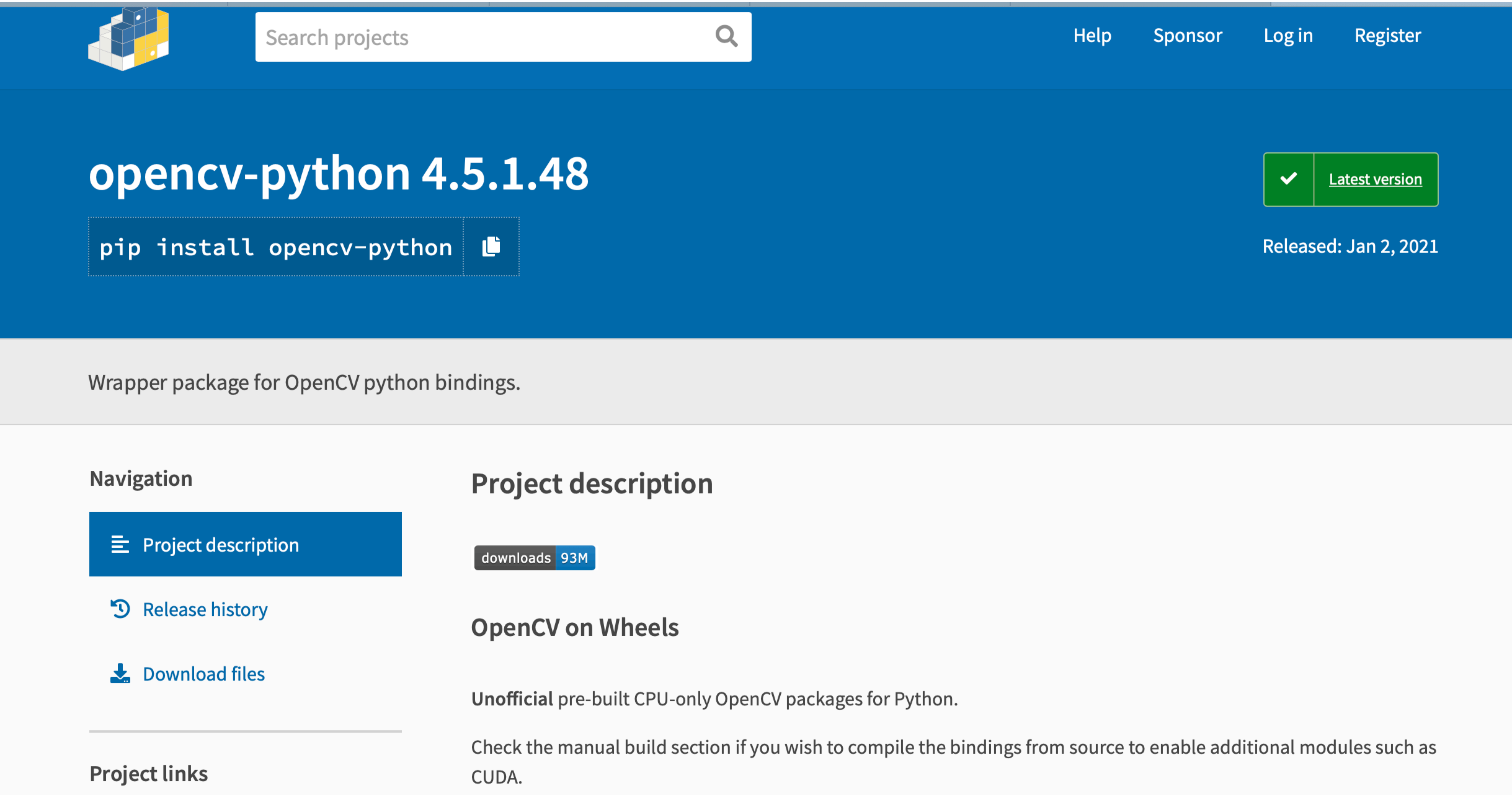
TABLE OF CONTENTS

NLTK News
Installing NLTK
Installing NLTK Data
Contribute to NLTK
FAQ
Wiki
API
HOWTO

SEARCH



```python
import nltk

sentence = "hello my name is tim and this is cool"

words = nltk.word_tokenize(sentence)

print(words)
```



opencv-python 4.5.1.48

```
pip install opencv-python
```

Released: Jan 2, 2021

Wrapper package for OpenCV python bindings.

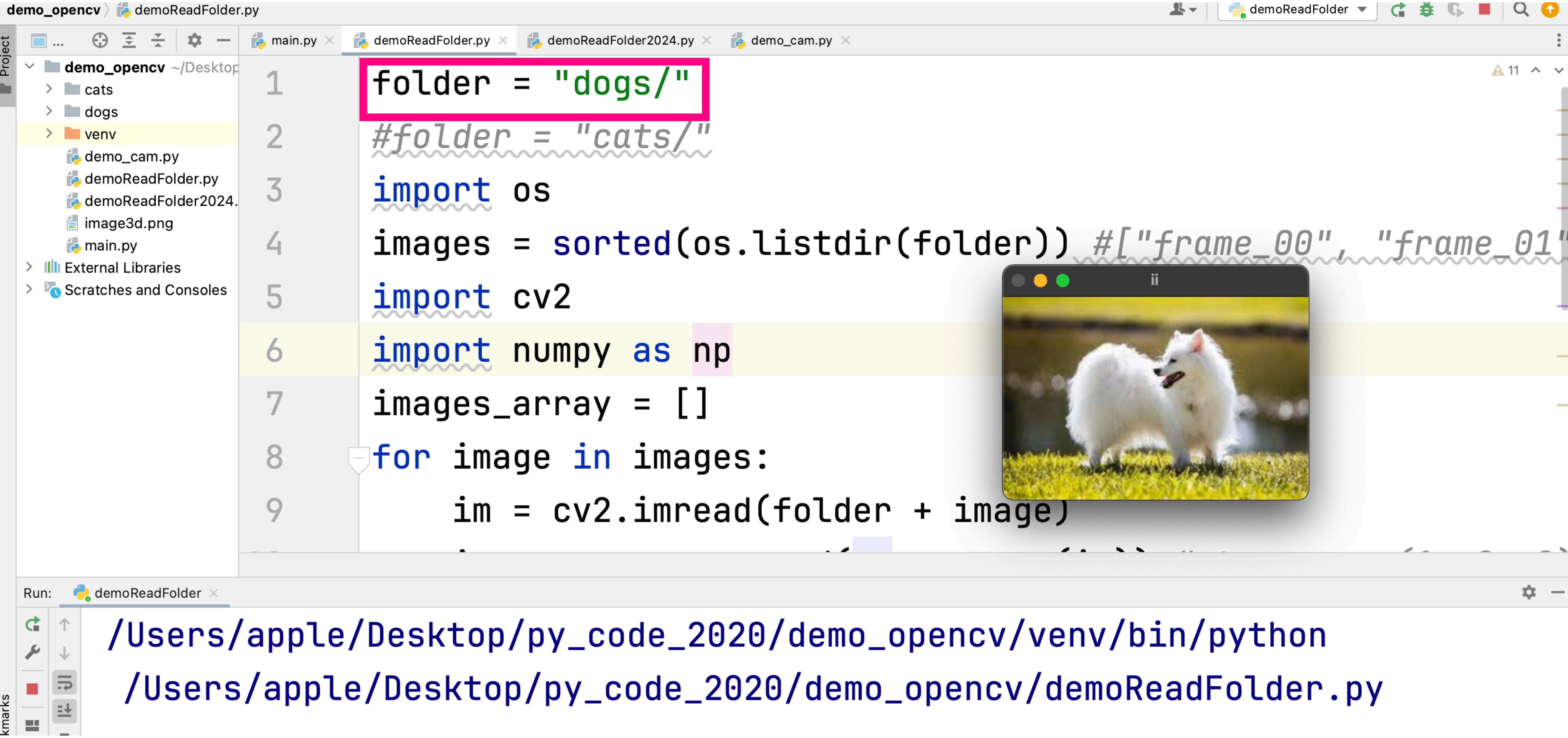
Project description

OpenCV on Wheels

Unofficial pre-built CPU-only OpenCV packages for Python.

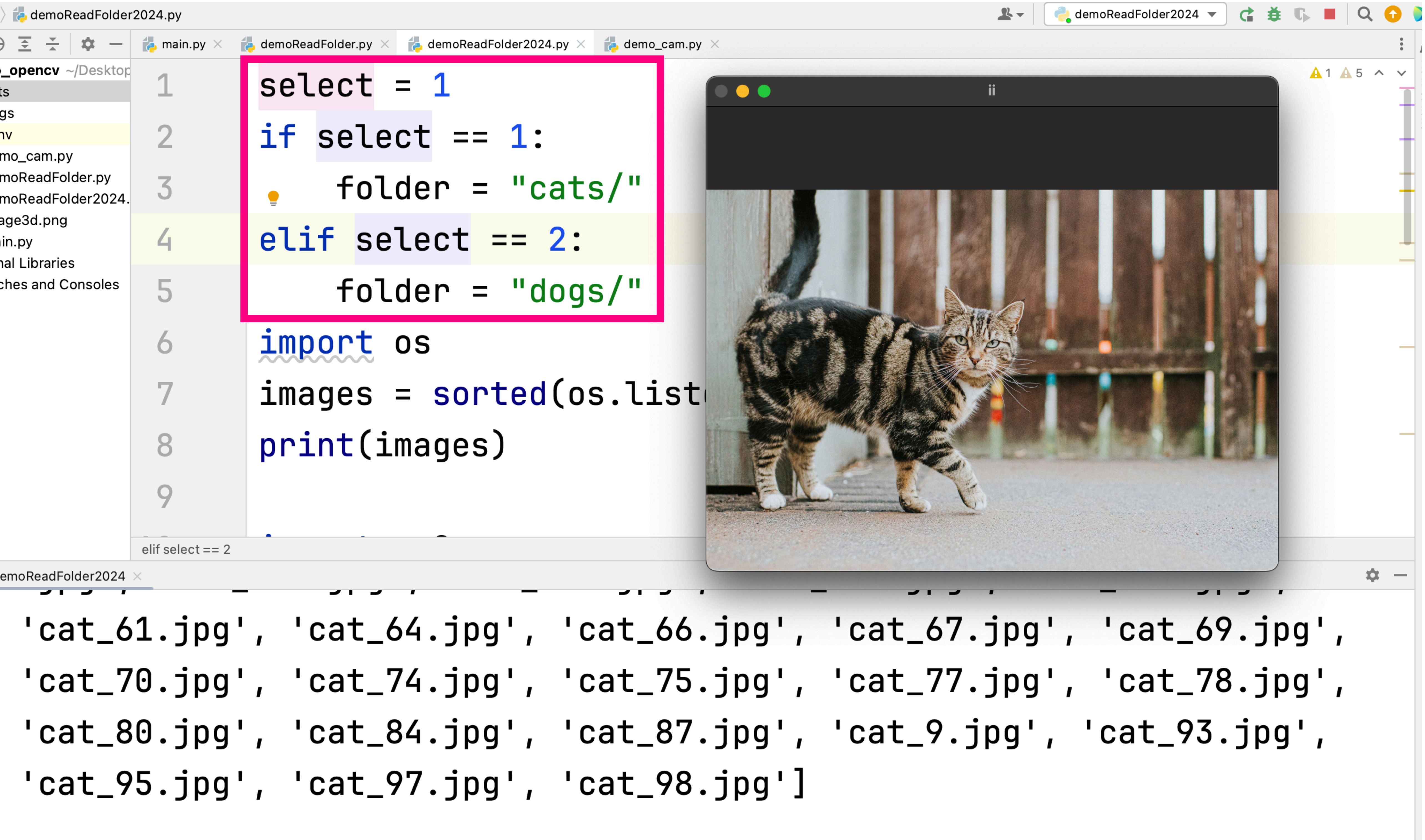
Check the manual build section if you wish to compile the bindings from source to enable additional modules such as CUDA.

Read images from a folder



```
folder = "dogs/"
#folder = "cats/"
import os
images = sorted(os.listdir(folder)) #["frame_00", "frame_01"
import cv2
import numpy as np
images_array = []
for image in images:
    im = cv2.imread(folder + image)
```

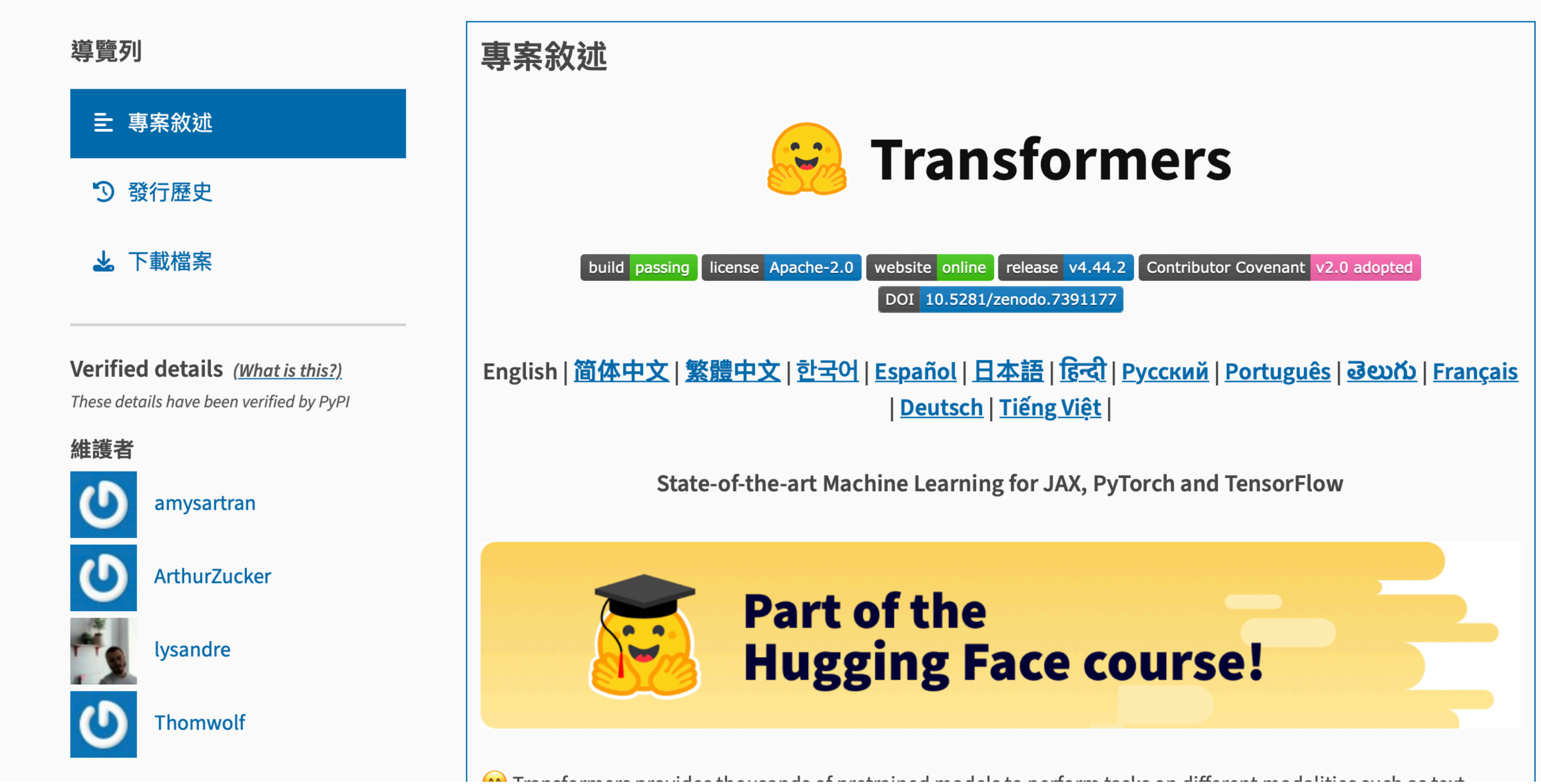
```
/Users/apple/Desktop/py_code_2020/demo_opencv/venv/bin/python
 /Users/apple/Desktop/py_code_2020/demo_opencv/demoReadFolder.py
```



```
select = 1
if select == 1:
    folder = "cats/"
elif select == 2:
    folder = "dogs/"
import os
images = sorted(os.listd
print(images)
```

```
'cat_61.jpg', 'cat_64.jpg', 'cat_66.jpg', 'cat_67.jpg', 'cat_69.jpg',
'cat_70.jpg', 'cat_74.jpg', 'cat_75.jpg', 'cat_77.jpg', 'cat_78.jpg',
'cat_80.jpg', 'cat_84.jpg', 'cat_87.jpg', 'cat_9.jpg', 'cat_93.jpg',
'cat_95.jpg', 'cat_97.jpg', 'cat_98.jpg']
```

Machine Learning and AI



導覽列

- 專案敘述
- 發行歷史
- 下載檔案

Verified details *(What is this?)*

These details have been verified by PyPI

維護者

- amysartran
- ArthurZucker
- lysandre
- Thomwolf

專案敘述

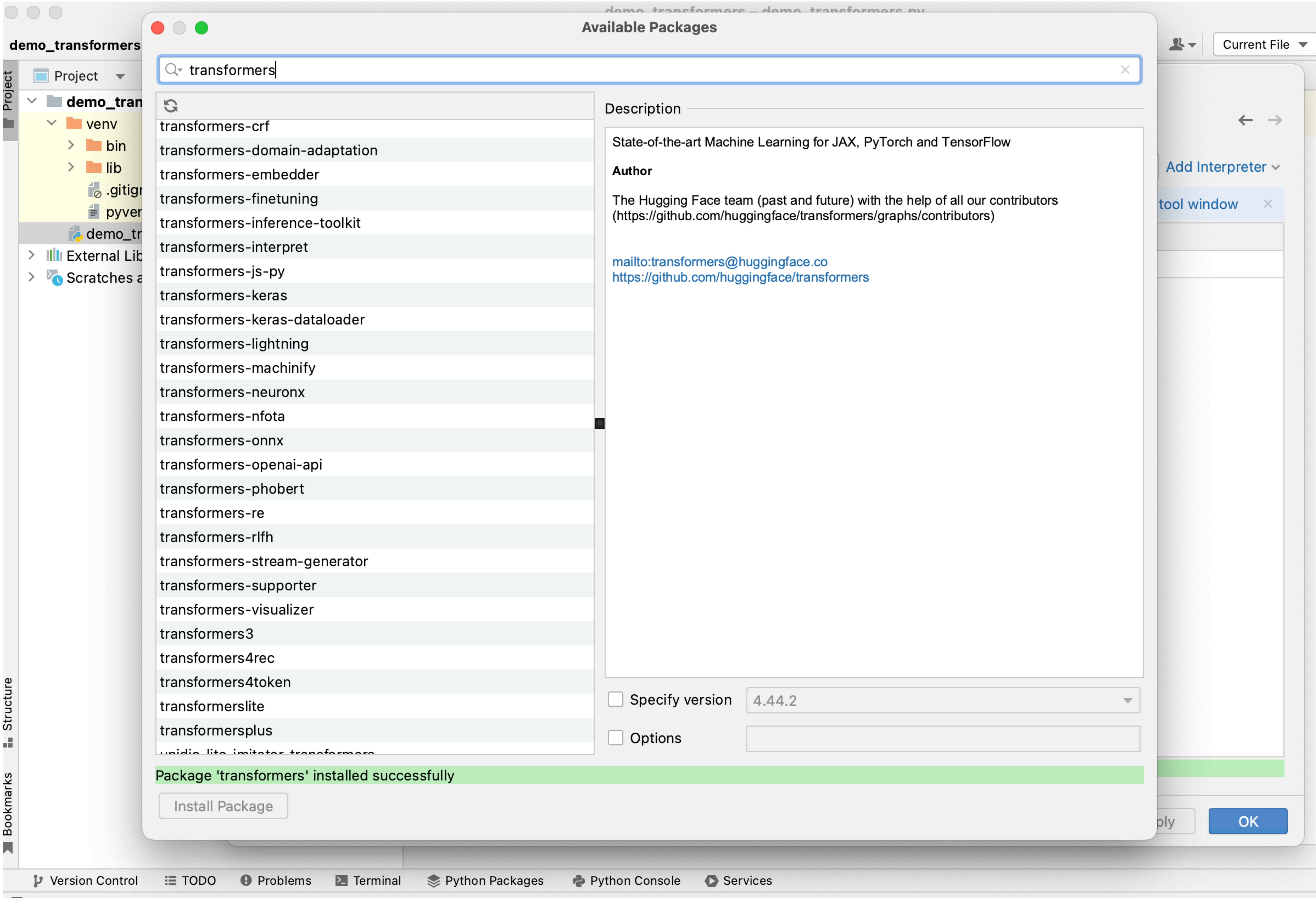
🤗 Transformers

build passing | license Apache-2.0 | website online | release v4.44.2 | Contributor Covenant v2.0 adopted | DOI 10.5281/zenodo.7391177

English | 简体中文 | 繁體中文 | 한국어 | Español | 日本語 | हिन्दी | Русский | Português | తెలుగు | Français | Deutsch | Tiếng Việt |

State-of-the-art Machine Learning for JAX, PyTorch and TensorFlow

🤗 Transformers provides thousands of pretrained models to perform tasks on different modalities such as text



Available Packages

transformers

transformers-crf
transformers-domain-adaptation
transformers-embedder
transformers-finetuning
transformers-inference-toolkit
transformers-interpret
transformers-js-py
transformers-keras
transformers-keras-dataloader
transformers-lightning
transformers-machinify
transformers-neuronx
transformers-nfota
transformers-onnx
transformers-openai-api
transformers-phobert
transformers-re
transformers-rlfh
transformers-stream-generator
transformers-supporter
transformers-visualizer
transformers3
transformers4rec
transformers4token
transformerslite
transformersplus
unidic-lite-imitator-transformers

Description

State-of-the-art Machine Learning for JAX, PyTorch and TensorFlow

Author

The Hugging Face team (past and future) with the help of all our contributors (https://github.com/huggingface/transformers/graphs/contributors)

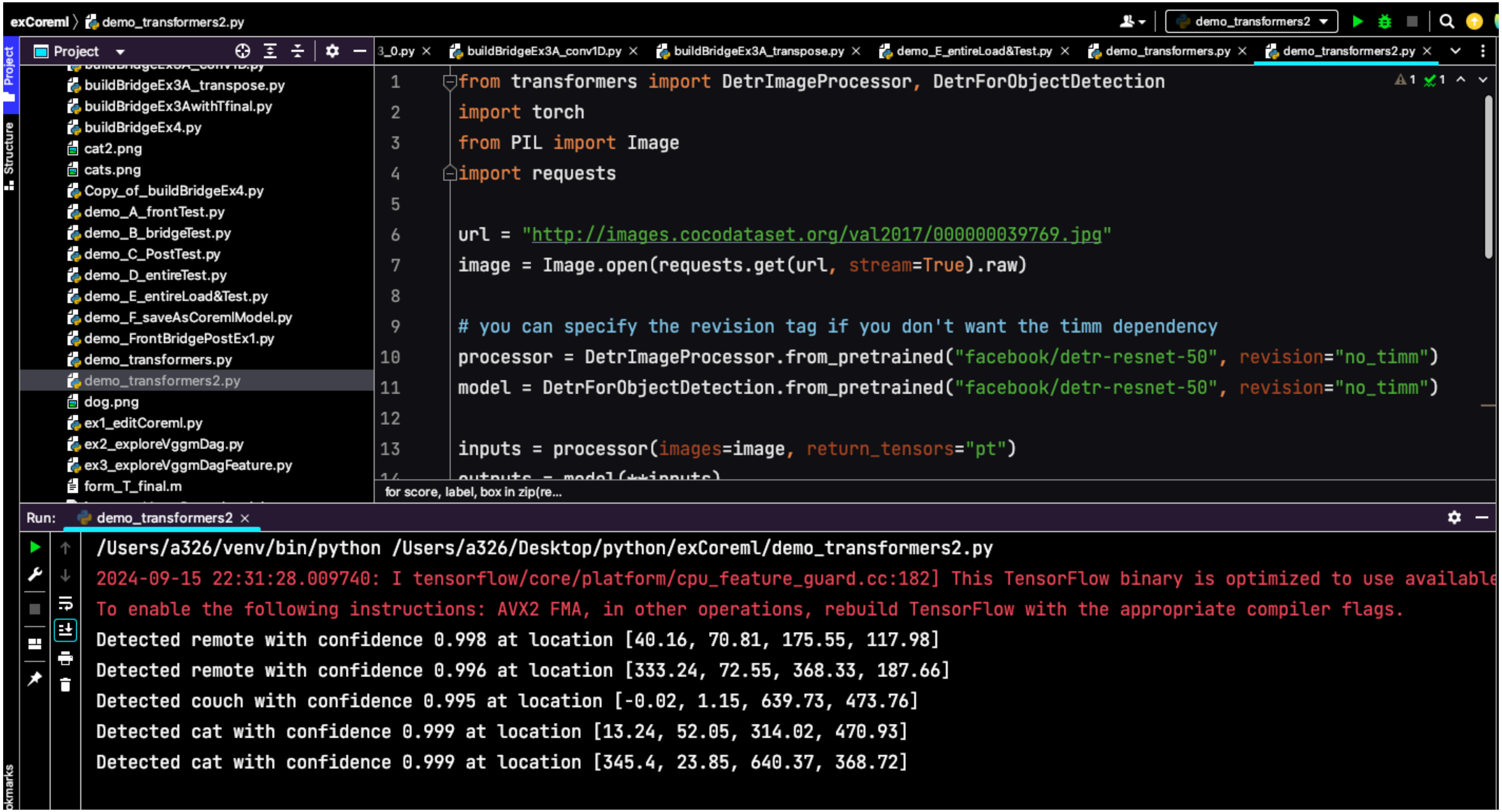
mailto:transformers@huggingface.co
https://github.com/huggingface/transformers

□ Specify version 4.44.2

□ Options

Package 'transformers' installed successfully

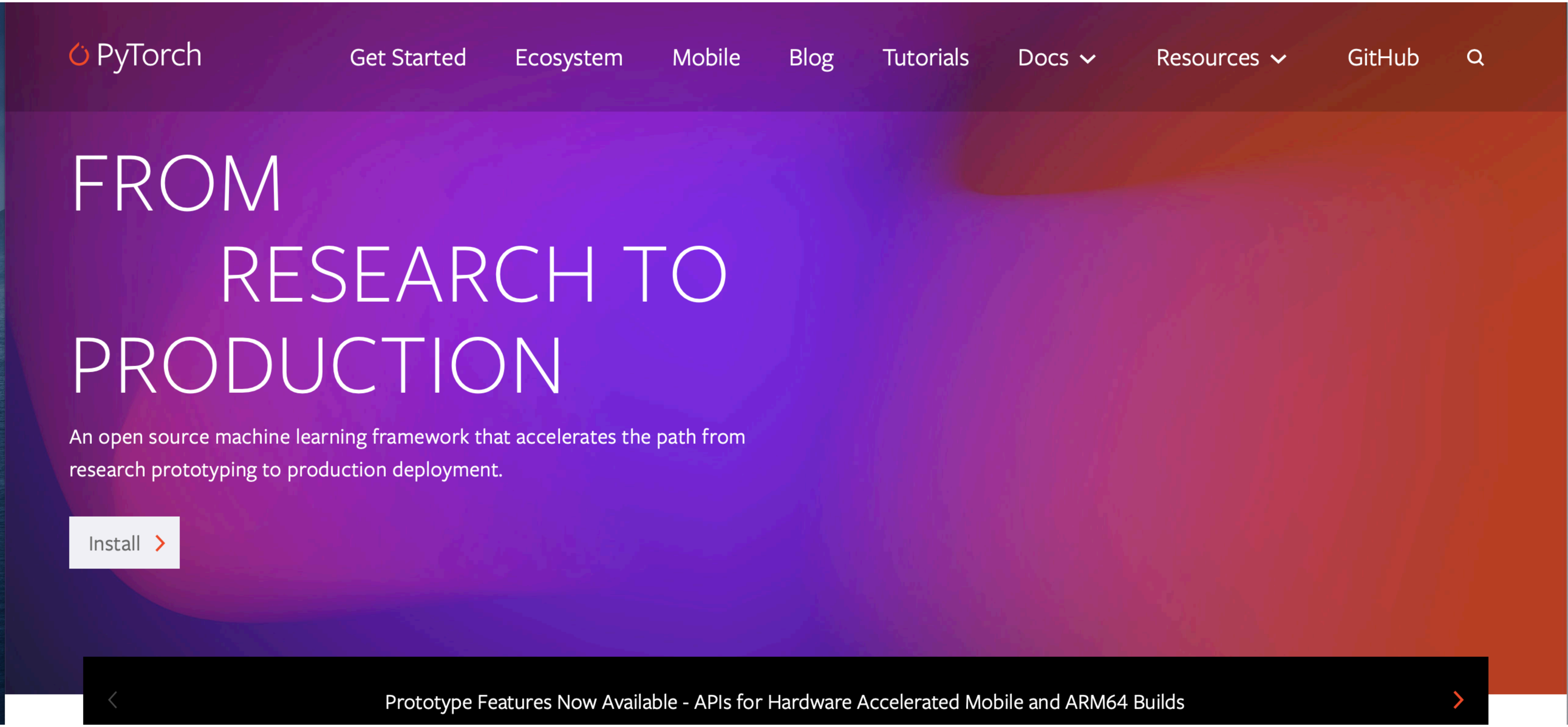
Install Package



```python
from transformers import DetrImageProcessor, DetrForObjectDetection
import torch
from PIL import Image
import requests

url = "http://images.cocodataset.org/val2017/000000039769.jpg"
image = Image.open(requests.get(url, stream=True).raw)

# you can specify the revision tag if you don't want the timm dependency
processor = DetrImageProcessor.from_pretrained("facebook/detr-resnet-50", revision="no_timm")
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50", revision="no_timm")

inputs = processor(images=image, return_tensors="pt")
outputs = model(**inputs)
```

for score, label, box in zip(re...

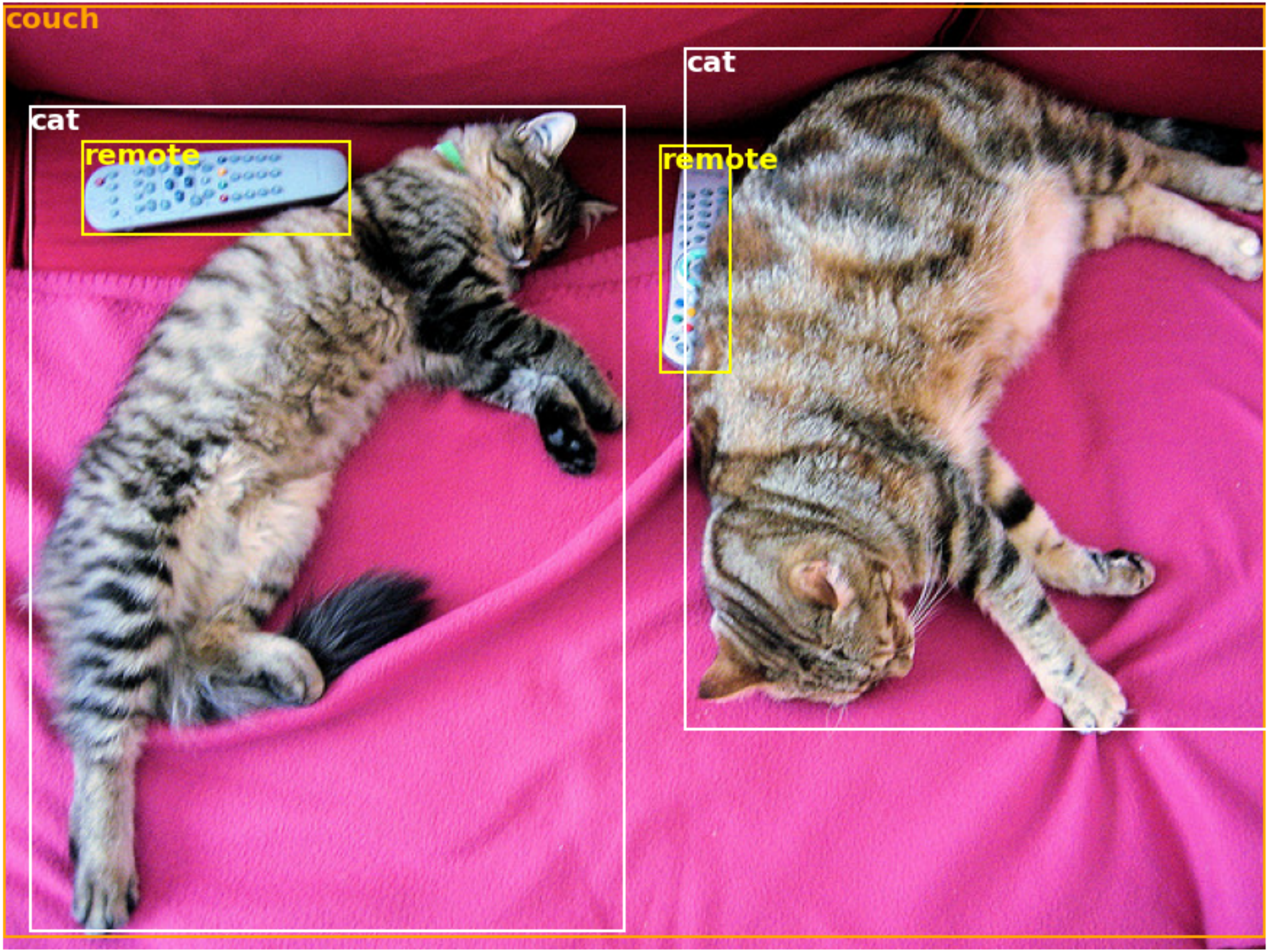
```
/Users/a326/venv/bin/python /Users/a326/Desktop/python/exCoreml/demo_transformers2.py
2024-09-15 22:31:28.009740: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Detected remote with confidence 0.998 at location [40.16, 70.81, 175.55, 117.98]
Detected remote with confidence 0.996 at location [333.24, 72.55, 368.33, 187.66]
Detected couch with confidence 0.995 at location [-0.02, 1.15, 639.73, 473.76]
Detected cat with confidence 0.999 at location [13.24, 52.05, 314.02, 470.93]
Detected cat with confidence 0.999 at location [345.4, 23.85, 640.37, 368.72]
```



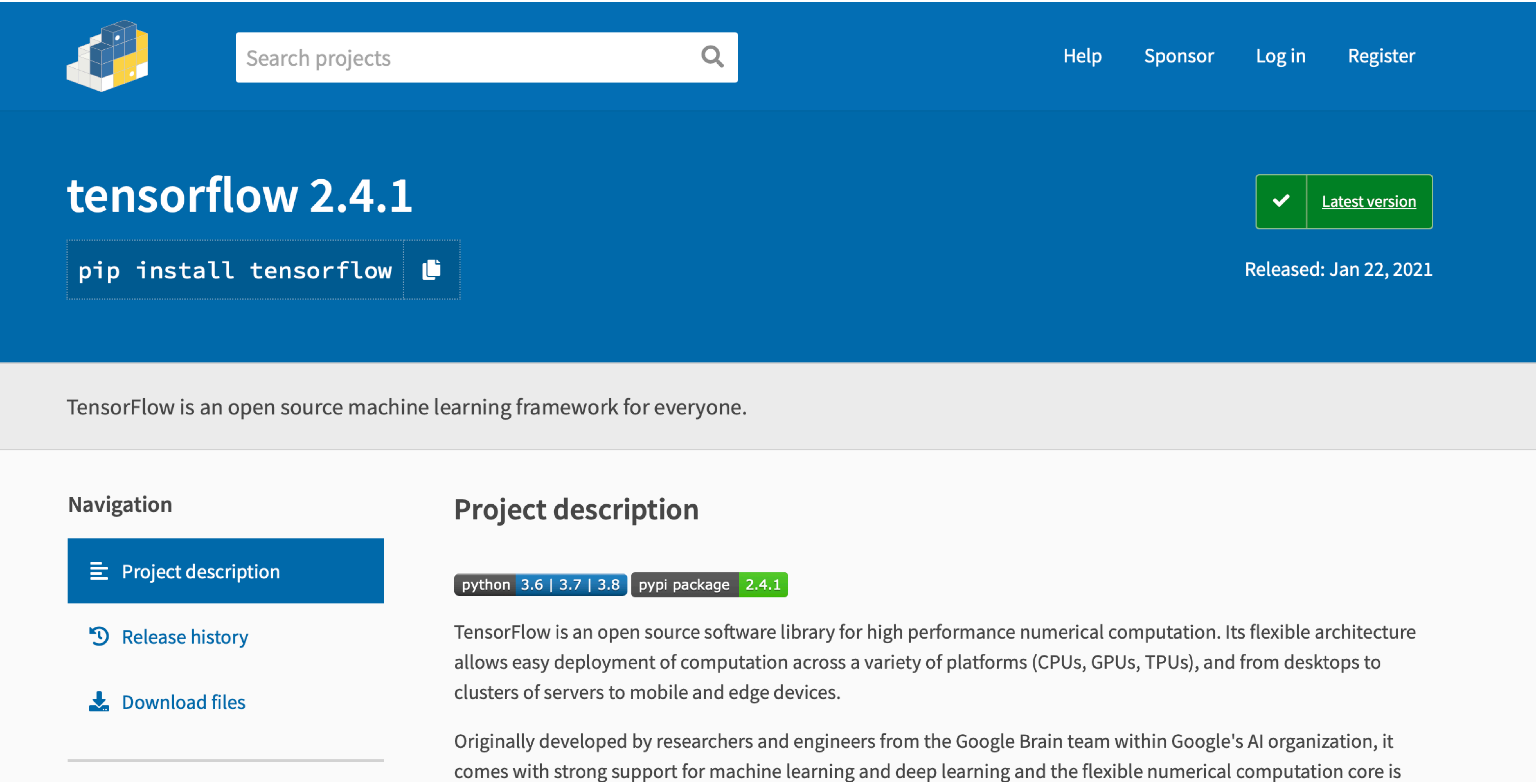
FROM
RESEARCH TO
PRODUCTION

An open source machine learning framework that accelerates the path from research prototyping to production deployment.

Prototype Features Now Available - APIs for Hardware Accelerated Mobile and ARM64 Builds


couch
cat
remote
cat
remote

Tensorflow



tensorflow 2.4.1

```
pip install tensorflow
```

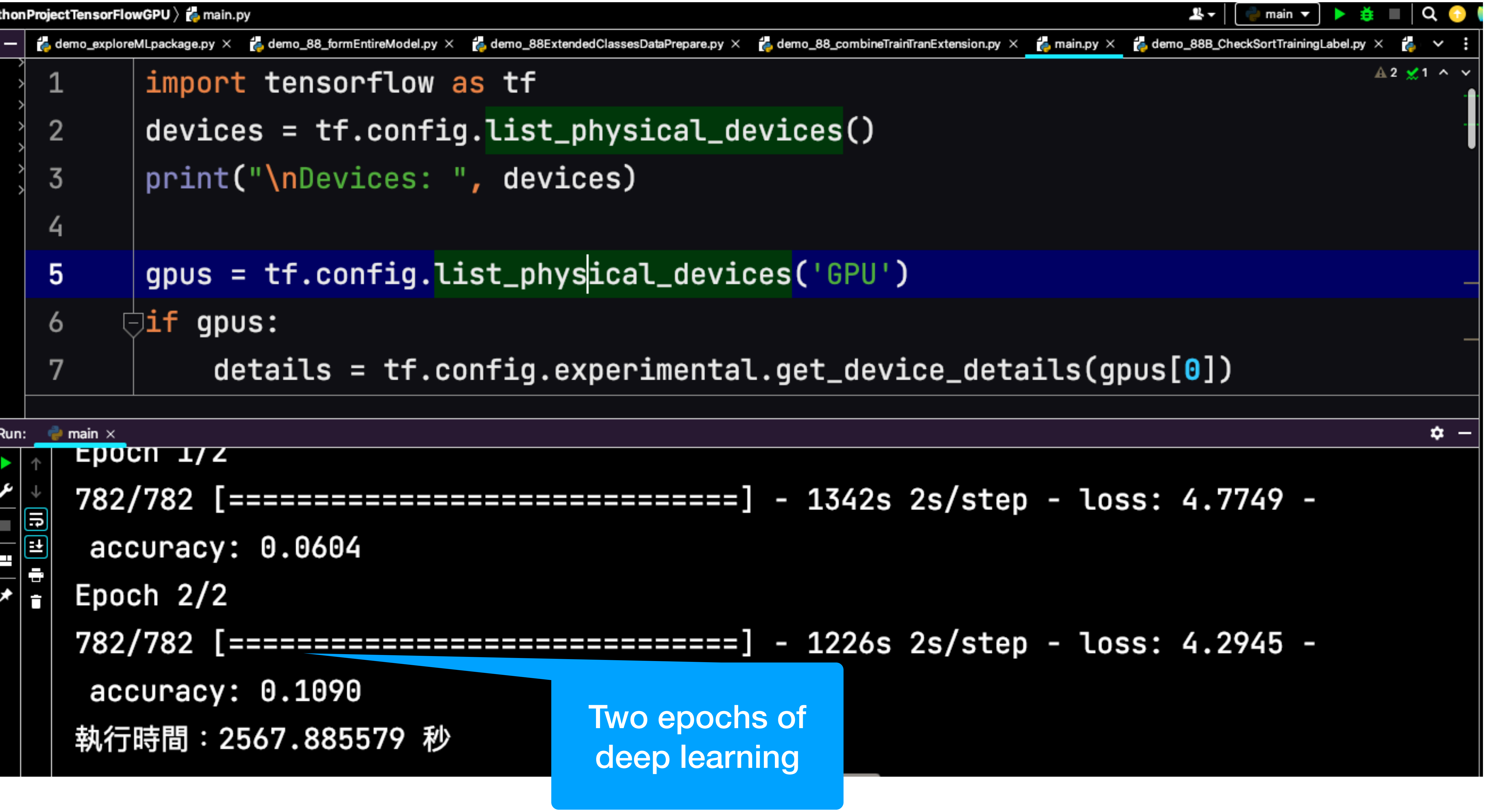
Released: Jan 22, 2021

TensorFlow is an open source machine learning framework for everyone.

Project description

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

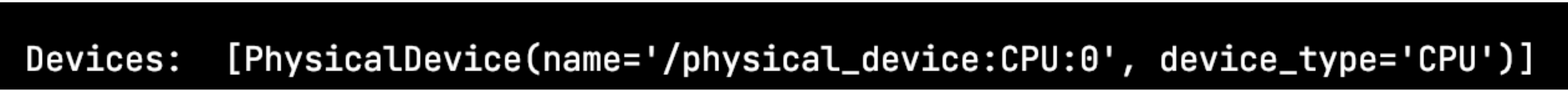
Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is


honProjectTensorFlowGPU 〉 main.py
main
demo_exploreMLpackage.py
demo_88_formEntireModel.py
demo_88ExtendedClassesDataPrepare.py
demo_88_combineTrainTranExtension.py
main.py
demo_88B_CheckSortTrainingLabel.py
⚠2 ✓1
Run:
main

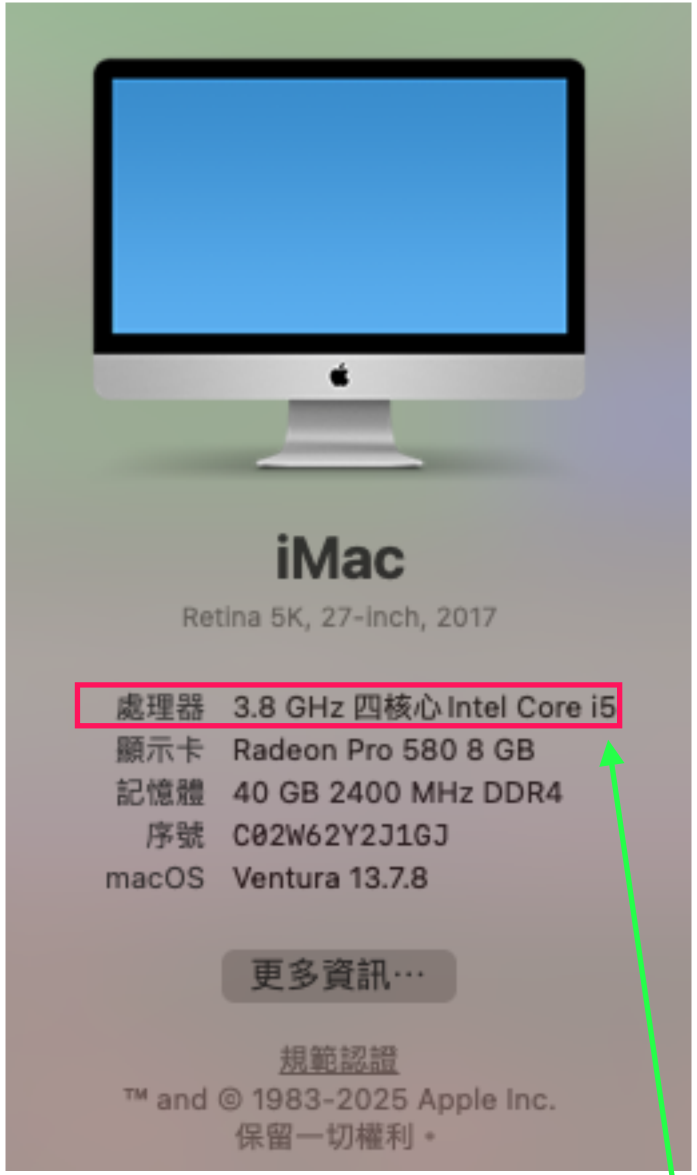
```
import tensorflow as tf
devices = tf.config.list_physical_devices()
print("\nDevices: ", devices)

gpus = tf.config.list_physical_devices('GPU')
if gpus:
    details = tf.config.experimental.get_device_details(gpus[0])
```

```
Epoch 1/2
782/782 [==============================] - 1342s 2s/step - loss: 4.7749 -
 accuracy: 0.0604
Epoch 2/2
782/782 [==============================] - 1226s 2s/step - loss: 4.2945 -
 accuracy: 0.1090
執行時間：2567.885579 秒
```

Two epochs of deep learning



```
Devices:  [PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')]
```



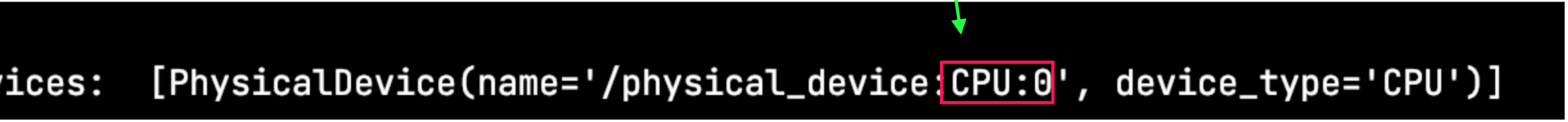
iMac

Retina 5K, 27-inch, 2017

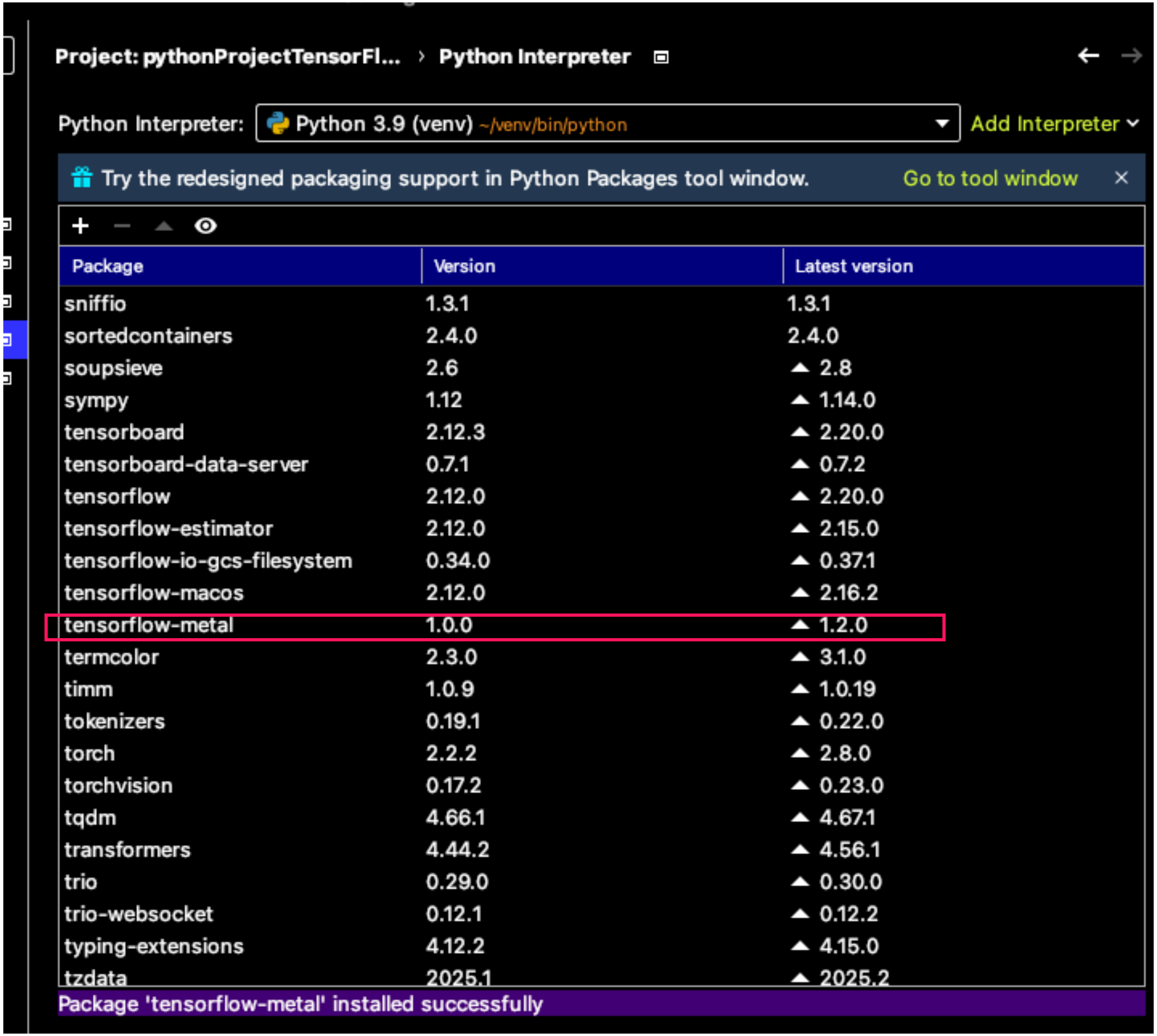
處理器	3.8 GHz 四核心Intel Core i5
顯示卡	Radeon Pro 580 8 GB
記憶體	40 GB 2400 MHz DDR4
序號	C02W62Y2J1GJ
macOS	Ventura 13.7.8

規範認證

™ and © 1983-2025 Apple Inc.
保留一切權利。



```
ices:  [PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')]
```



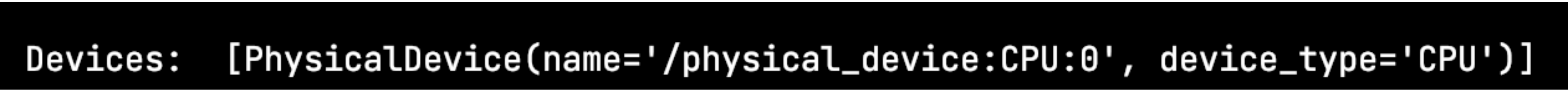
Project: pythonProjectTensorFl... › Python Interpreter

Python Interpreter: Python 3.9 (venv) ~/venv/bin/python

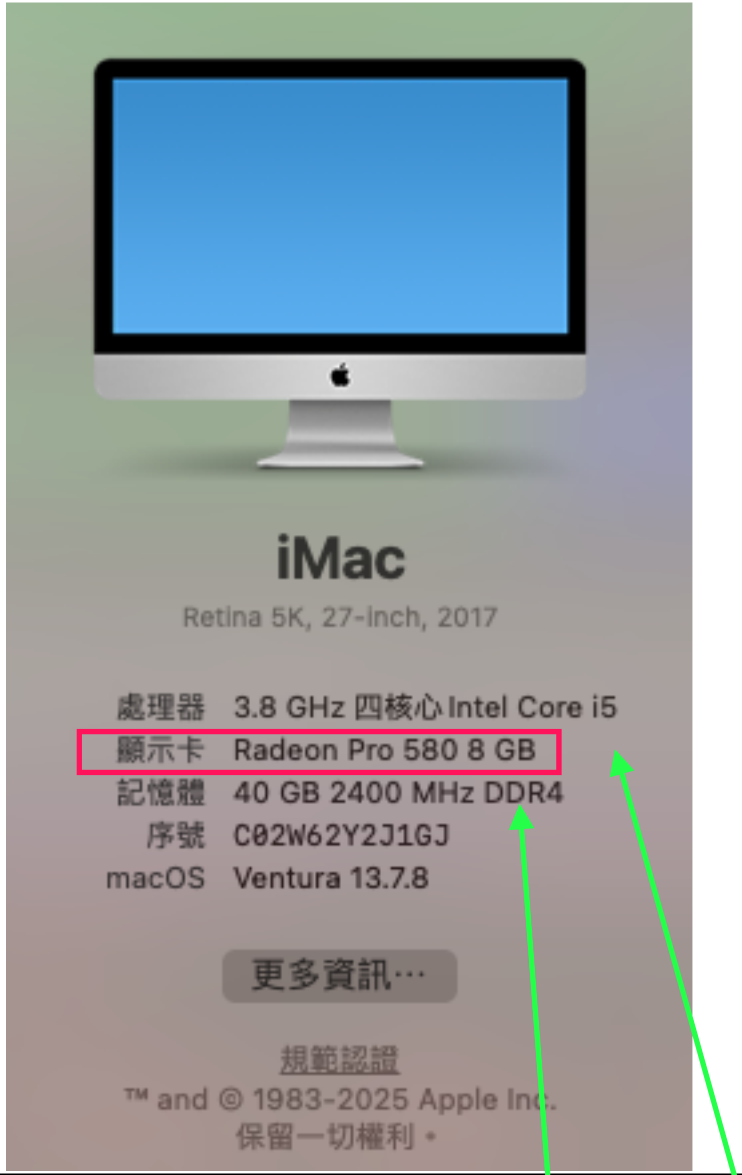
Try the redesigned packaging support in Python Packages tool window.

Package	Version	Latest version
sniffio	1.3.1	1.3.1
sortedcontainers	2.4.0	2.4.0
soupsieve	2.6	▲ 2.8
sympy	1.12	▲ 1.14.0
tensorboard	2.12.3	▲ 2.20.0
tensorboard-data-server	0.7.1	▲ 0.7.2
tensorflow	2.12.0	▲ 2.20.0
tensorflow-estimator	2.12.0	▲ 2.15.0
tensorflow-io-gcs-filesystem	0.34.0	▲ 0.37.1
tensorflow-macos	2.12.0	▲ 2.16.2
tensorflow-metal	1.0.0	▲ 1.2.0
termcolor	2.3.0	▲ 3.1.0
timm	1.0.9	▲ 1.0.19
tokenizers	0.19.1	▲ 0.22.0
torch	2.2.2	▲ 2.8.0
torchvision	0.17.2	▲ 0.23.0
tqdm	4.66.1	▲ 4.67.1
transformers	4.44.2	▲ 4.56.1
trio	0.29.0	▲ 0.30.0
trio-websocket	0.12.1	▲ 0.12.2
typing-extensions	4.12.2	▲ 4.15.0
tzdata	2025.1	▲ 2025.2

Package 'tensorflow-metal' installed successfully



```
Devices:  [PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')]
```

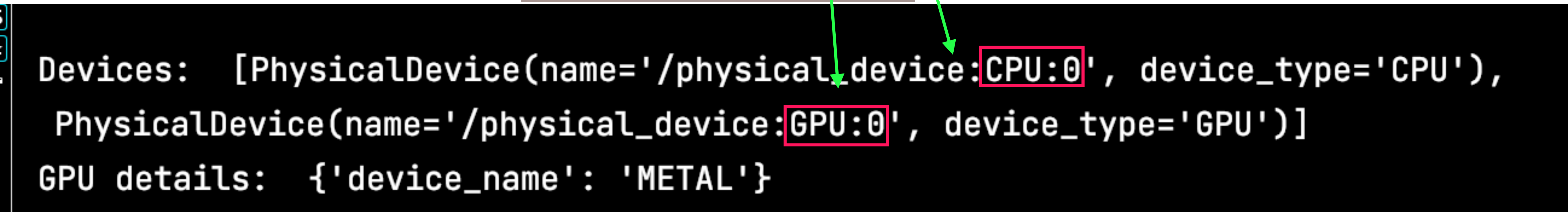


iMac

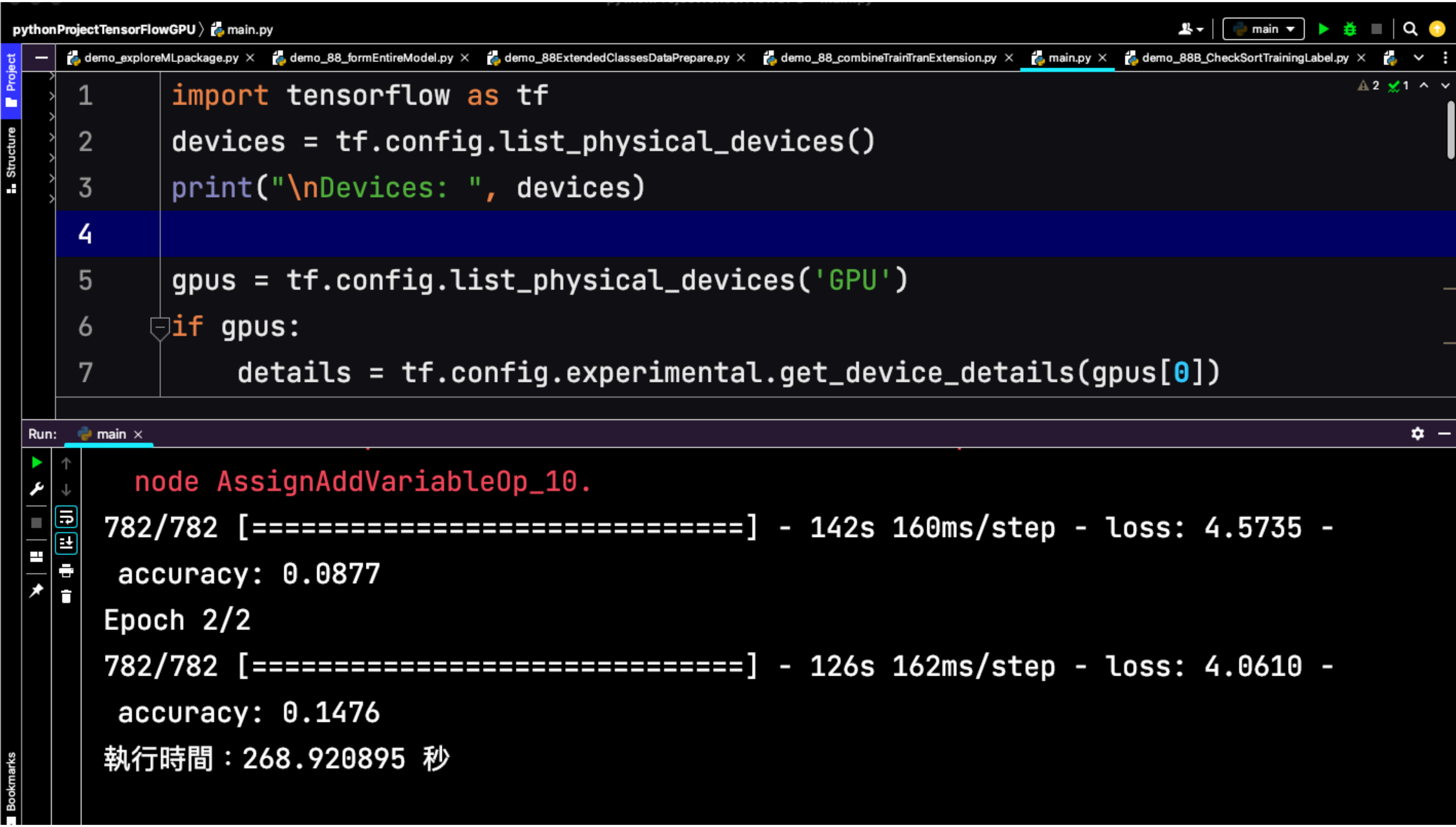
Retina 5K, 27-inch, 2017

處理器	3.8 GHz 四核心 Intel Core i5
顯示卡	Radeon Pro 580 8 GB
記憶體	40 GB 2400 MHz DDR4
序號	C02W62Y2J1GJ
macOS	Ventura 13.7.8

™ and © 1983-2025 Apple Inc.
保留一切權利。

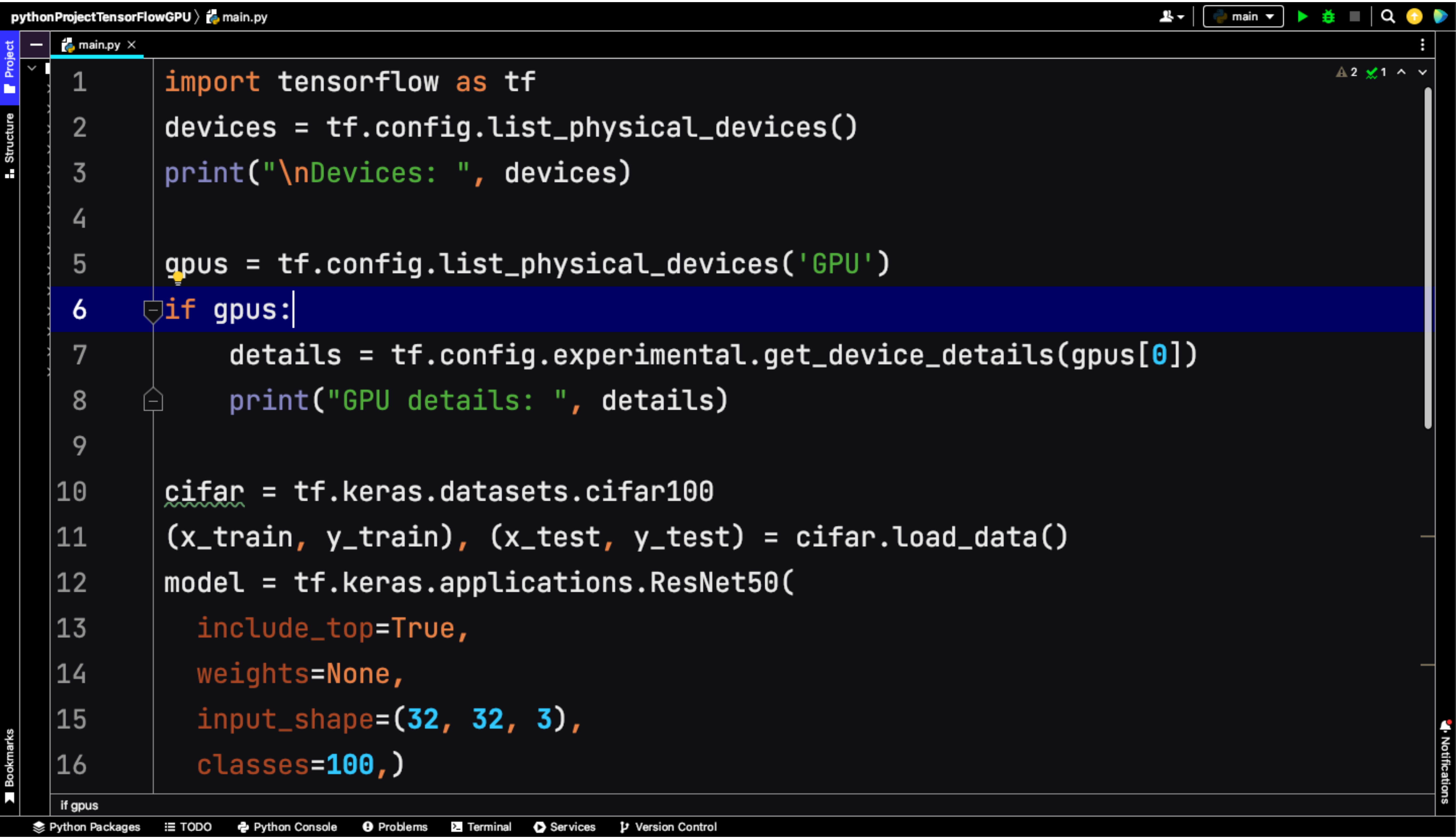


```
Devices:  [PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU'),
 PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
GPU details:  {'device_name': 'METAL'}
```

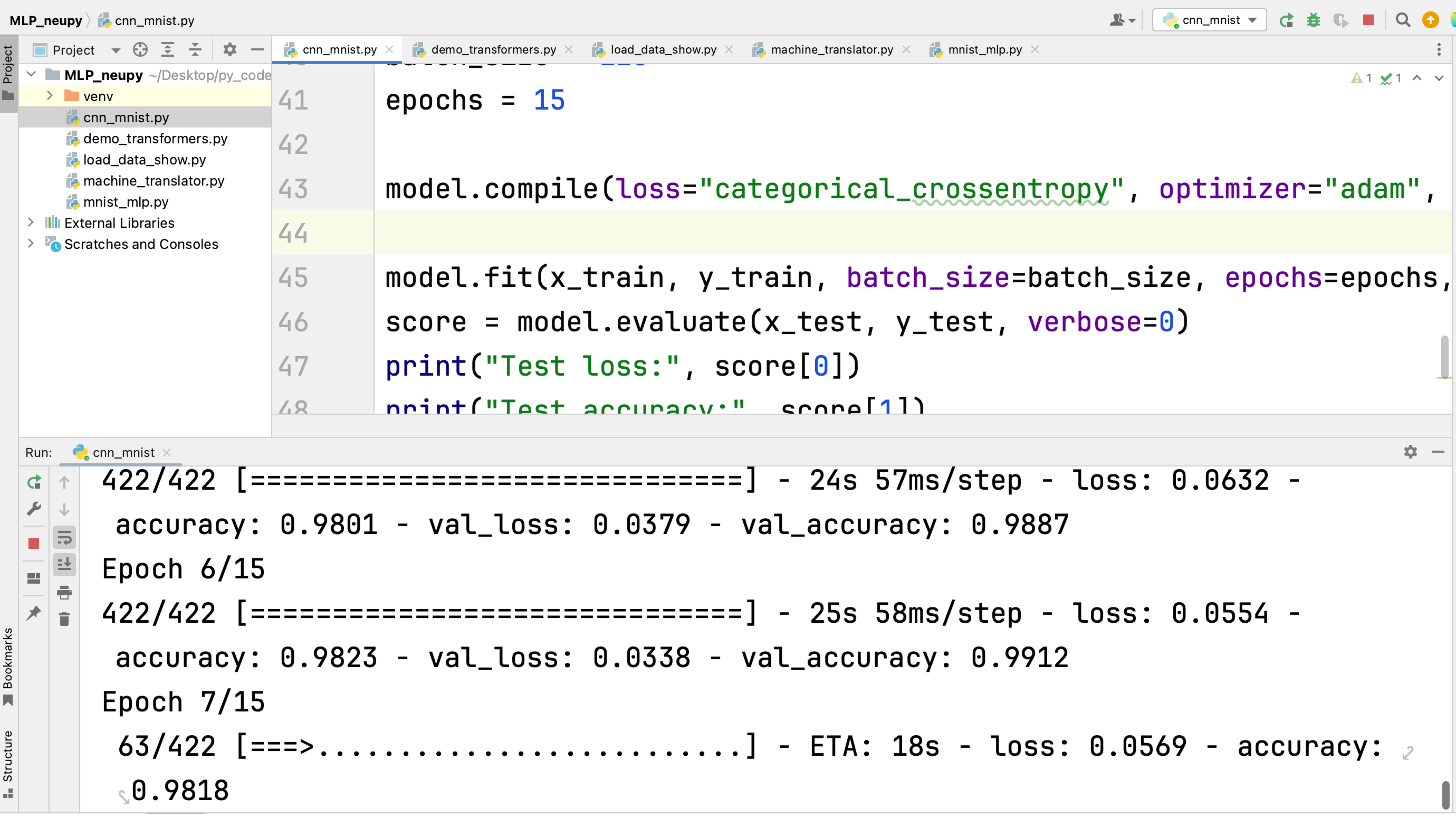

pythonProjectTensorFlowGPU 〉 main.py
main
demo_exploreMLpackage.py
demo_88_formEntireModel.py
demo_88ExtendedClassesDataPrepare.py
demo_88_combineTrainTranExtension.py
main.py
demo_88B_CheckSortTrainingLabel.py
Project
Structure
Bookmarks
Run:
main

```python
import tensorflow as tf
devices = tf.config.list_physical_devices()
print("\nDevices: ", devices)

gpus = tf.config.list_physical_devices('GPU')
if gpus:
    details = tf.config.experimental.get_device_details(gpus[0])
```

```
 node AssignAddVariableOp_10.
782/782 [==============================] - 142s 160ms/step - loss: 4.5735 -
 accuracy: 0.0877
Epoch 2/2
782/782 [==============================] - 126s 162ms/step - loss: 4.0610 -
 accuracy: 0.1476
執行時間：268.920895 秒
```

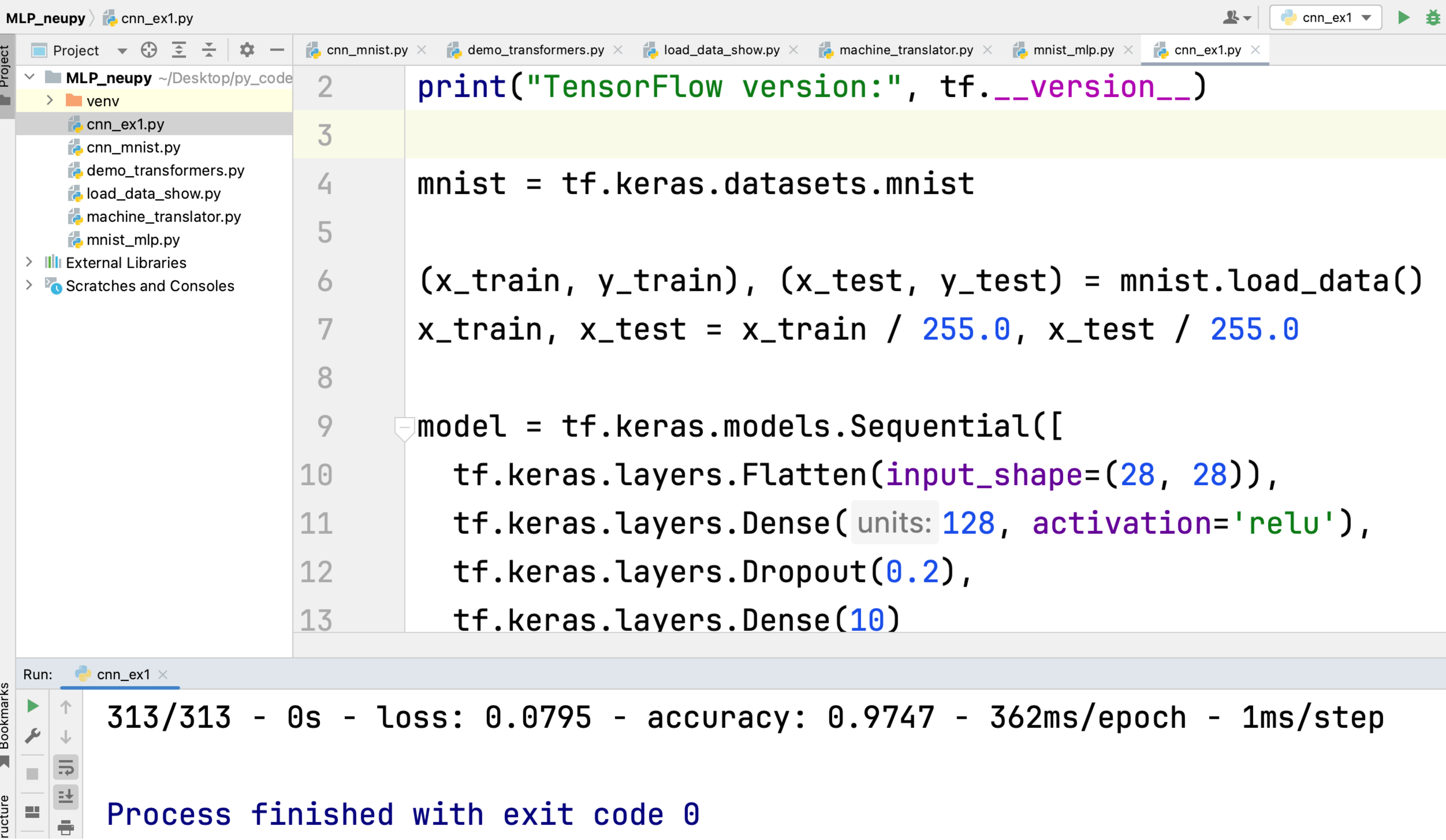


```python
import tensorflow as tf
devices = tf.config.list_physical_devices()
print("\nDevices: ", devices)

gpus = tf.config.list_physical_devices('GPU')
if gpus:
    details = tf.config.experimental.get_device_details(gpus[0])
    print("GPU details: ", details)

cifar = tf.keras.datasets.cifar100
(x_train, y_train), (x_test, y_test) = cifar.load_data()
model = tf.keras.applications.ResNet50(
  include_top=True,
  weights=None,
  input_shape=(32, 32, 3),
  classes=100,)
```



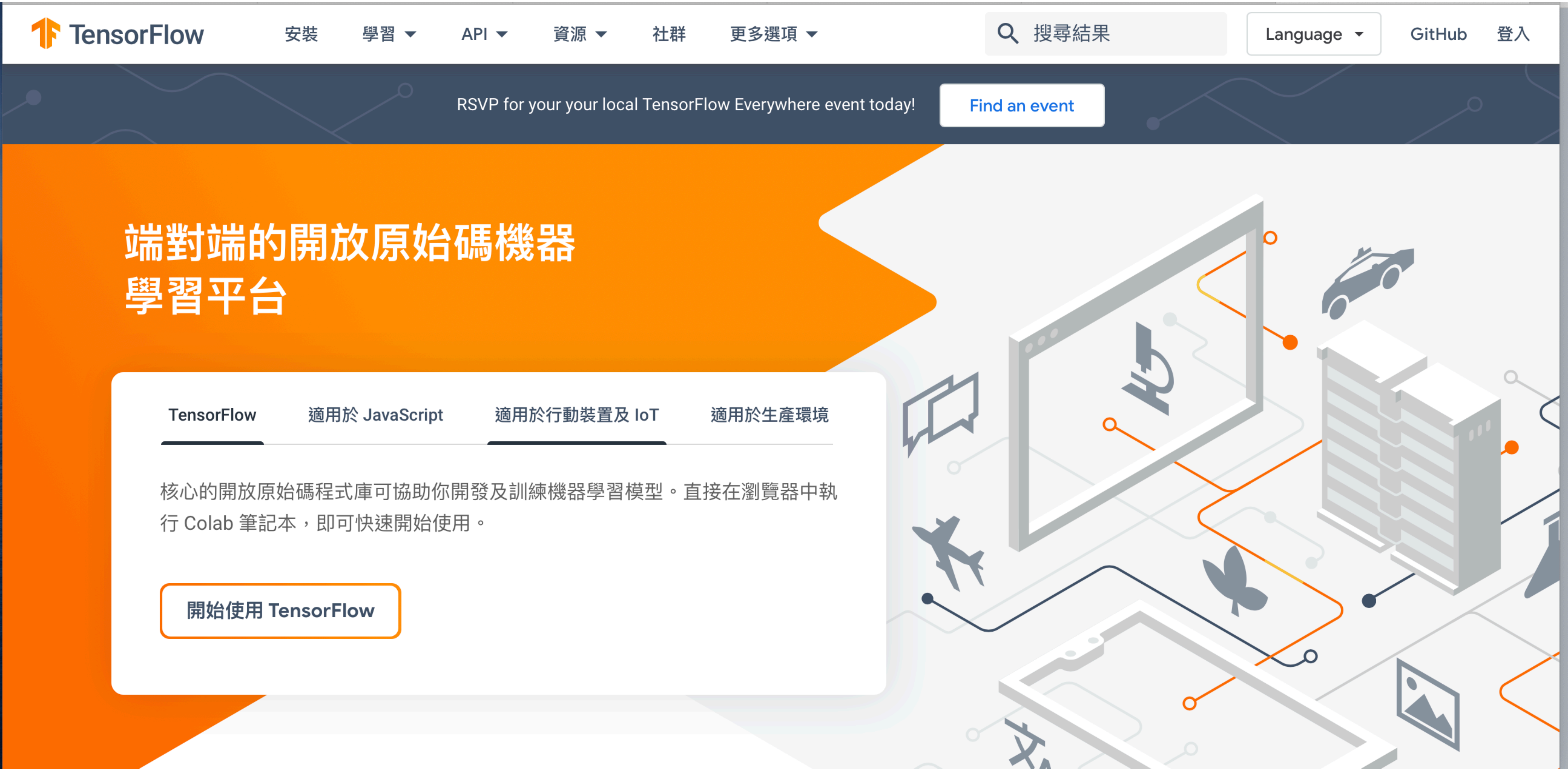
```
epochs = 15

model.compile(loss="categorical_crossentropy", optimizer="adam",

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
422/422 [==============================] - 24s 57ms/step - loss: 0.0632 -
 accuracy: 0.9801 - val_loss: 0.0379 - val_accuracy: 0.9887
Epoch 6/15
422/422 [==============================] - 25s 58ms/step - loss: 0.0554 -
 accuracy: 0.9823 - val_loss: 0.0338 - val_accuracy: 0.9912
Epoch 7/15
 63/422 [===>..........................] - ETA: 18s - loss: 0.0569 - accuracy:
 0.9818
```



```python
print("TensorFlow version:", tf.__version__)

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense( units: 128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
```

```
313/313 - 0s - loss: 0.0795 - accuracy: 0.9747 - 362ms/epoch - 1ms/step


Process finished with exit code 0
```

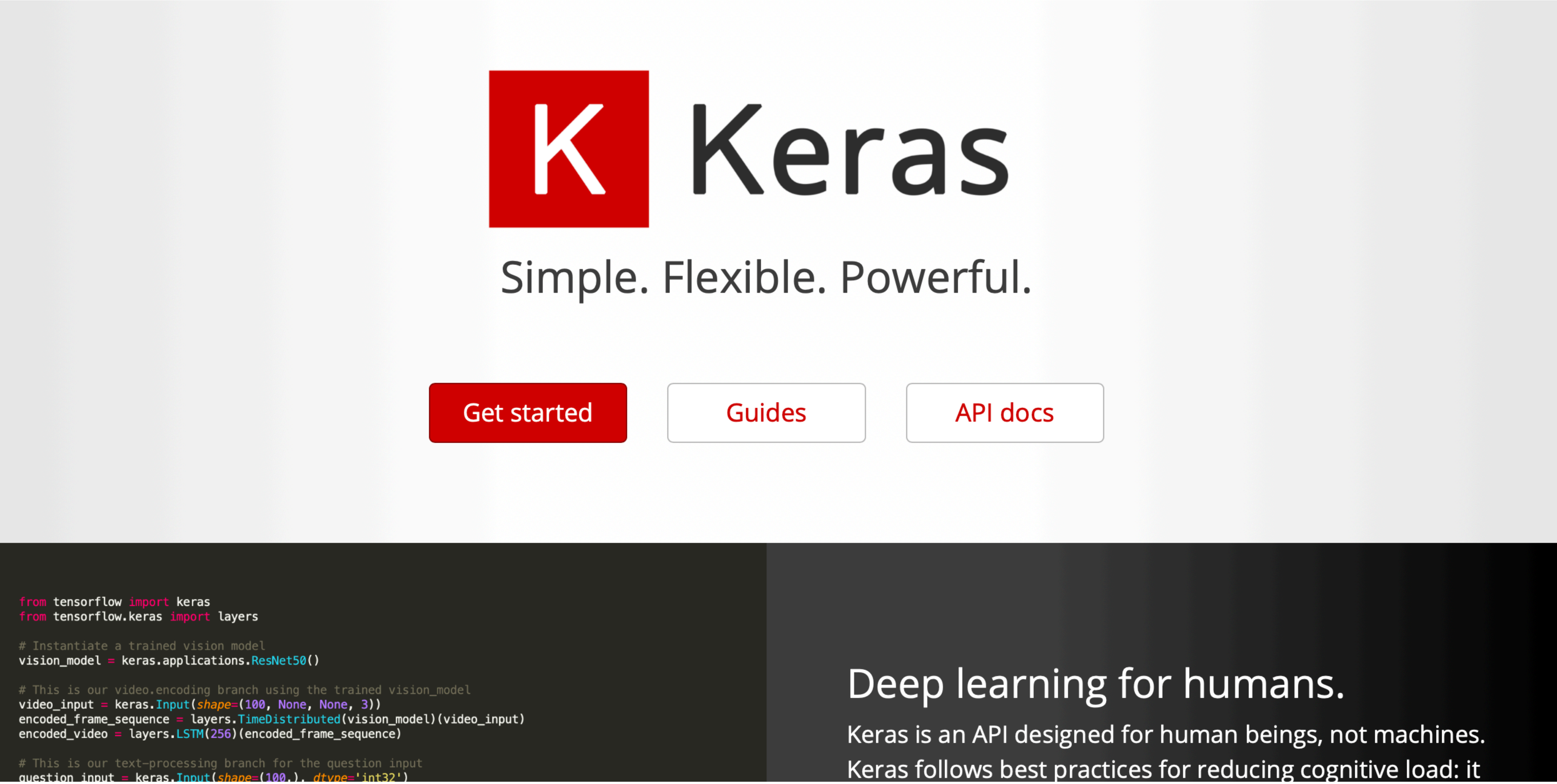


RSVP for your your local TensorFlow Everywhere event today!

端對端的開放原始碼機器學習平台

核心的開放原始碼程式庫可協助你開發及訓練機器學習模型。直接在瀏覽器中執行 Colab 筆記本，即可快速開始使用。

Keras

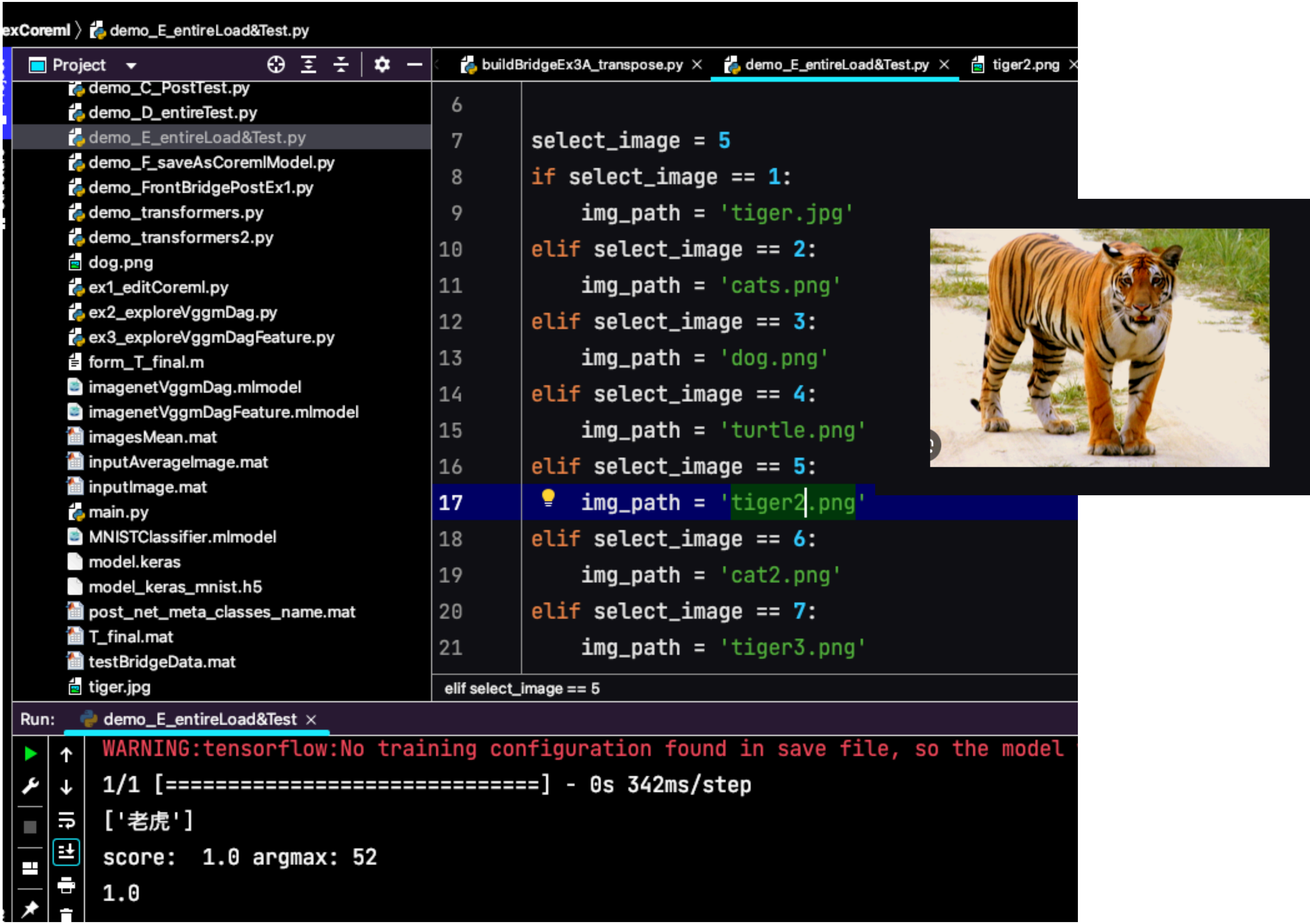


Simple. Flexible. Powerful.

```
from tensorflow import keras
from tensorflow.keras import layers

# Instantiate a trained vision model
vision_model = keras.applications.ResNet50()

# This is our video.encoding branch using the trained vision_model
video_input = keras.Input(shape=(100, None, None, 3))
encoded_frame_sequence = layers.TimeDistributed(vision_model)(video_input)
encoded_video = layers.LSTM(256)(encoded_frame_sequence)

# This is our text-processing branch for the question input
question_input = keras.Input(shape=(100,), dtype='int32')
```

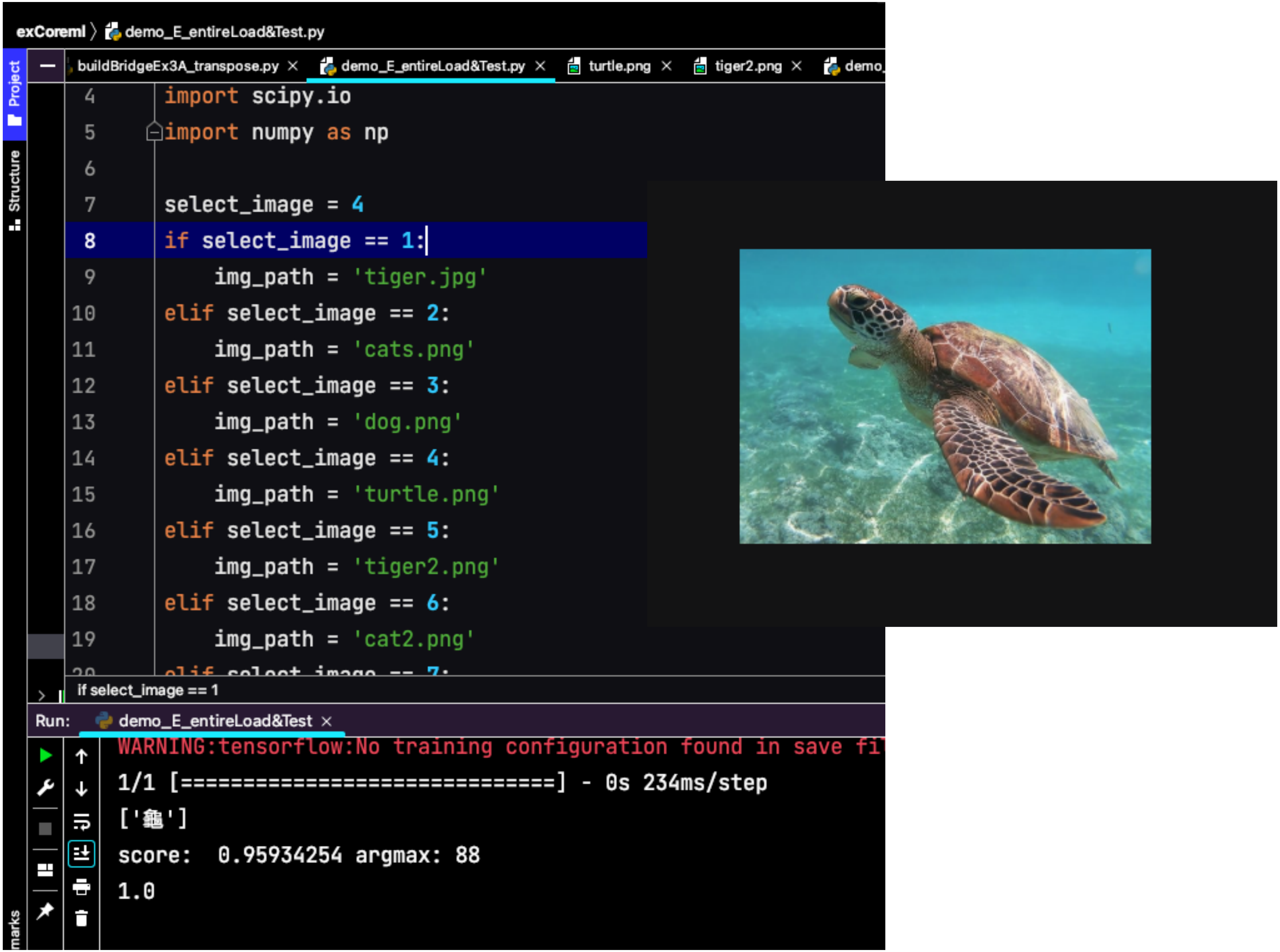
Deep learning for humans.

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it



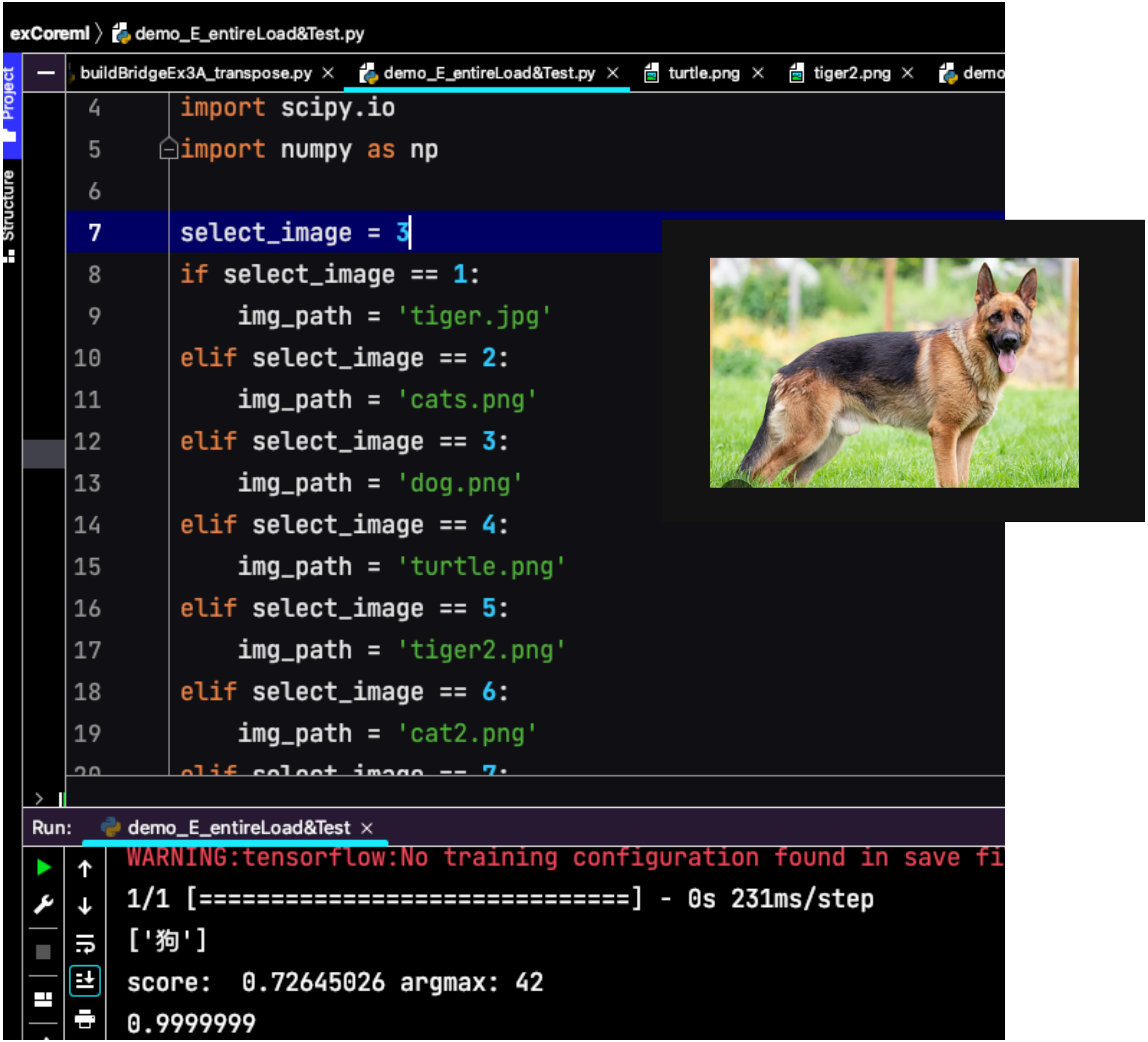
```
select_image = 5
if select_image == 1:
    img_path = 'tiger.jpg'
elif select_image == 2:
    img_path = 'cats.png'
elif select_image == 3:
    img_path = 'dog.png'
elif select_image == 4:
    img_path = 'turtle.png'
elif select_image == 5:
    img_path = 'tiger2.png'
elif select_image == 6:
    img_path = 'cat2.png'
elif select_image == 7:
    img_path = 'tiger3.png'
```

```
WARNING:tensorflow:No training configuration found in save file, so the model
1/1 [==============================] - 0s 342ms/step
['老虎']
score:  1.0 argmax: 52
1.0
```



```python
import scipy.io
import numpy as np

select_image = 4
if select_image == 1:
    img_path = 'tiger.jpg'
elif select_image == 2:
    img_path = 'cats.png'
elif select_image == 3:
    img_path = 'dog.png'
elif select_image == 4:
    img_path = 'turtle.png'
elif select_image == 5:
    img_path = 'tiger2.png'
elif select_image == 6:
    img_path = 'cat2.png'
elif select_image == 7:
```

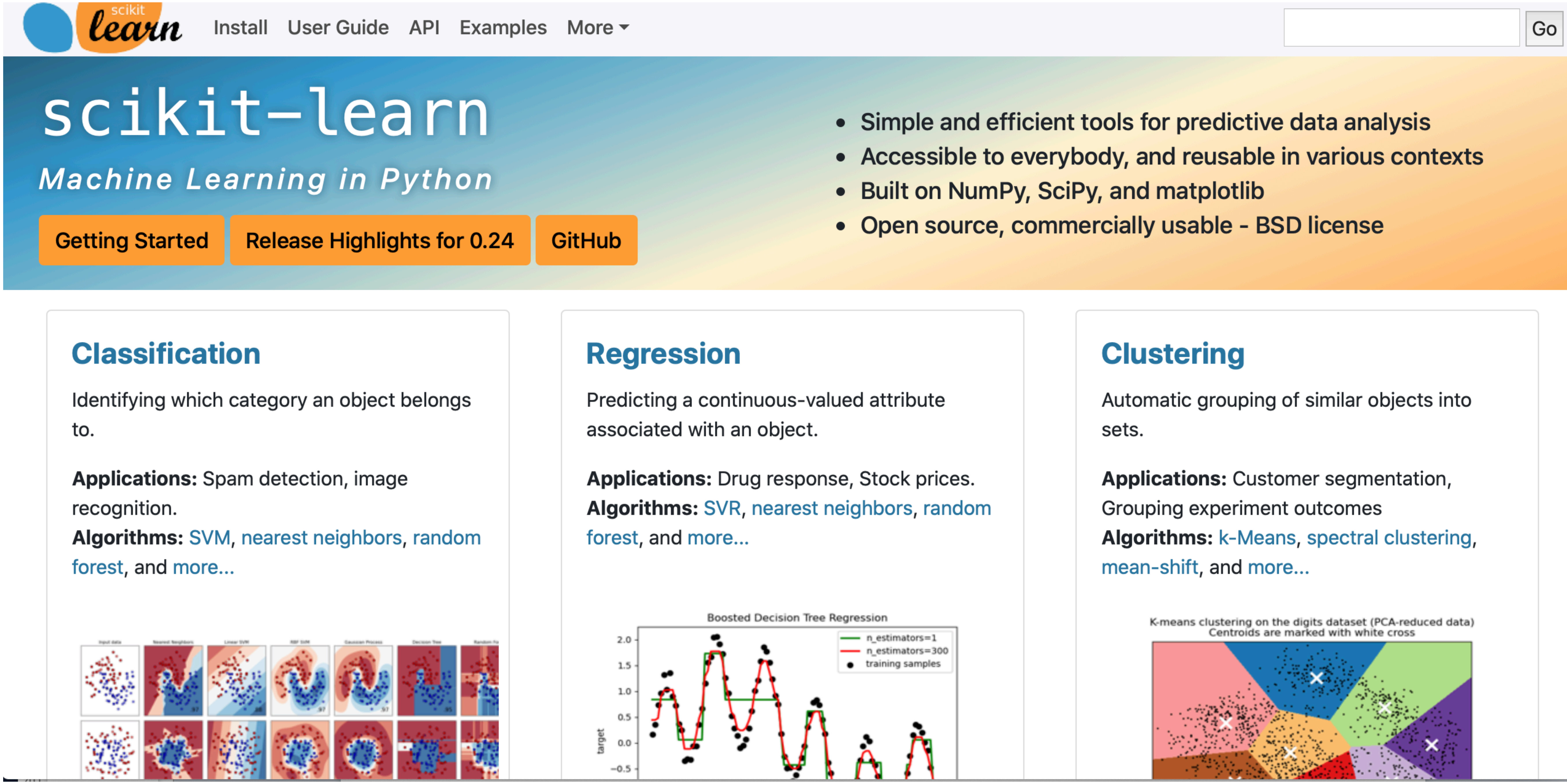
```
WARNING:tensorflow:No training configuration found in save fi
1/1 [==============================] - 0s 234ms/step
['龜']
score:  0.95934254 argmax: 88
1.0
```



exCoreml 〉 demo_E_entireLoad&Test.py
buildBridgeEx3A_transpose.py ×
demo_E_entireLoad&Test.py ×
turtle.png ×
tiger2.png ×
demo
Project
Structure
Run:
demo_E_entireLoad&Test ×

```python
import scipy.io
import numpy as np

select_image = 3
if select_image == 1:
    img_path = 'tiger.jpg'
elif select_image == 2:
    img_path = 'cats.png'
elif select_image == 3:
    img_path = 'dog.png'
elif select_image == 4:
    img_path = 'turtle.png'
elif select_image == 5:
    img_path = 'tiger2.png'
elif select_image == 6:
    img_path = 'cat2.png'
elif select_image == 7:
```

```
WARNING:tensorflow:No training configuration found in save fi
1/1 [==============================] - 0s 231ms/step
['狗']
score:  0.72645026 argmax: 42
0.9999999
```



scikit-learn

Machine Learning in Python

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

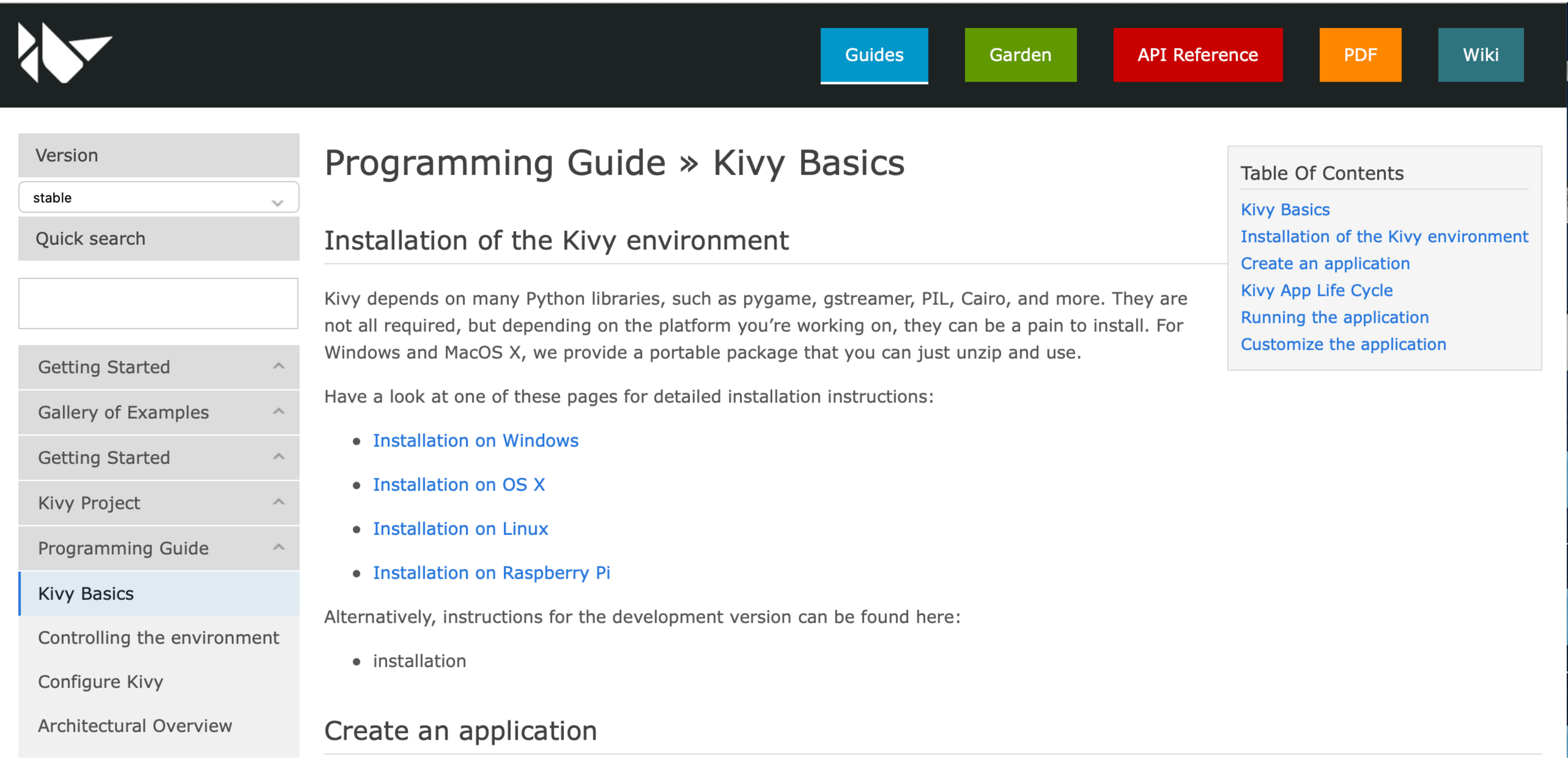
Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross

GUI



Programming Guide » Kivy Basics

Table Of Contents

- Kivy Basics
- Installation of the Kivy environment
- Create an application
- Kivy App Life Cycle
- Running the application
- Customize the application

Installation of the Kivy environment

Kivy depends on many Python libraries, such as pygame, gstreamer, PIL, Cairo, and more. They are not all required, but depending on the platform you're working on, they can be a pain to install. For Windows and MacOS X, we provide a portable package that you can just unzip and use.

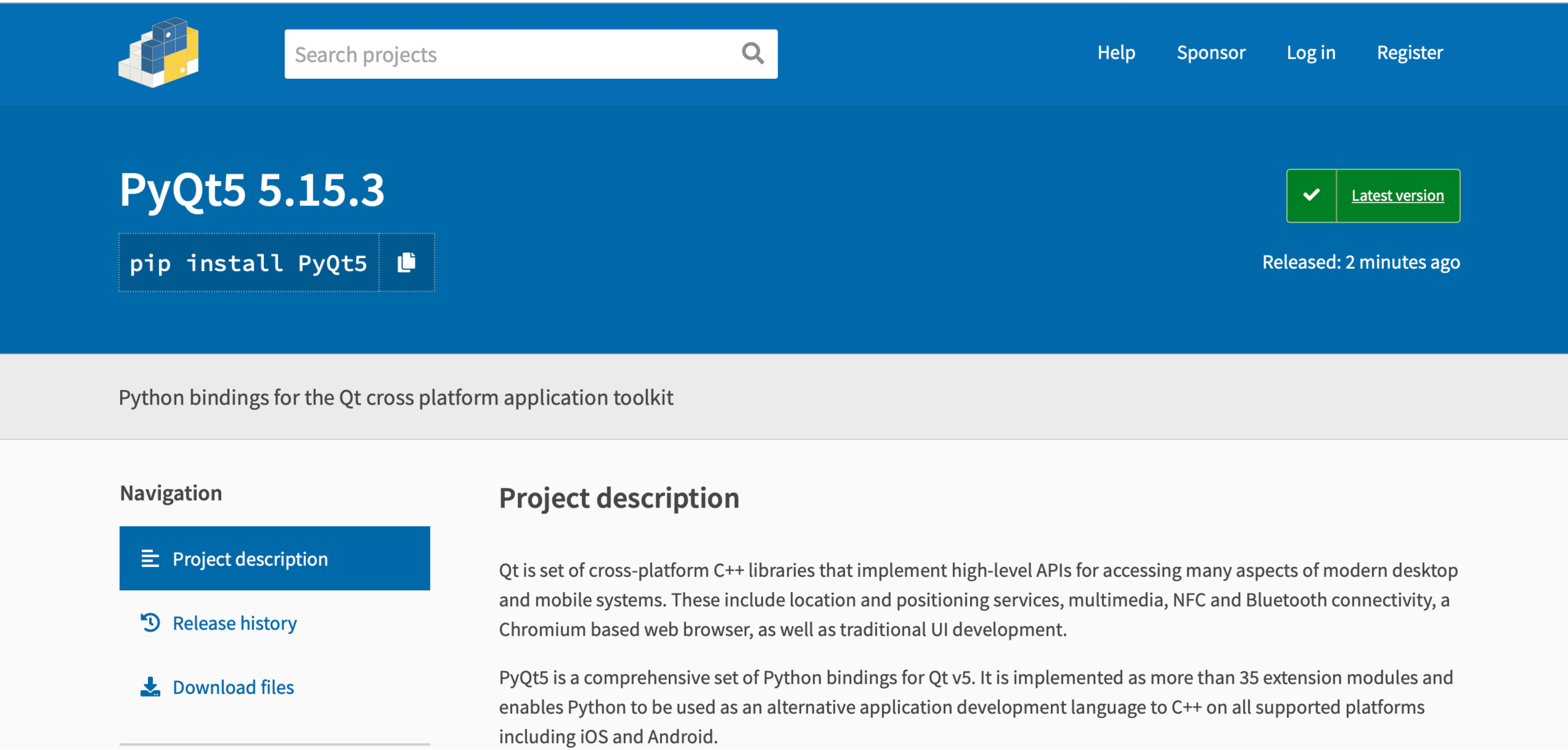
Have a look at one of these pages for detailed installation instructions:

- Installation on Windows
- Installation on OS X
- Installation on Linux
- Installation on Raspberry Pi

Alternatively, instructions for the development version can be found here:

- installation

Create an application



PyQt5 5.15.3

```
pip install PyQt5
```

Released: 2 minutes ago

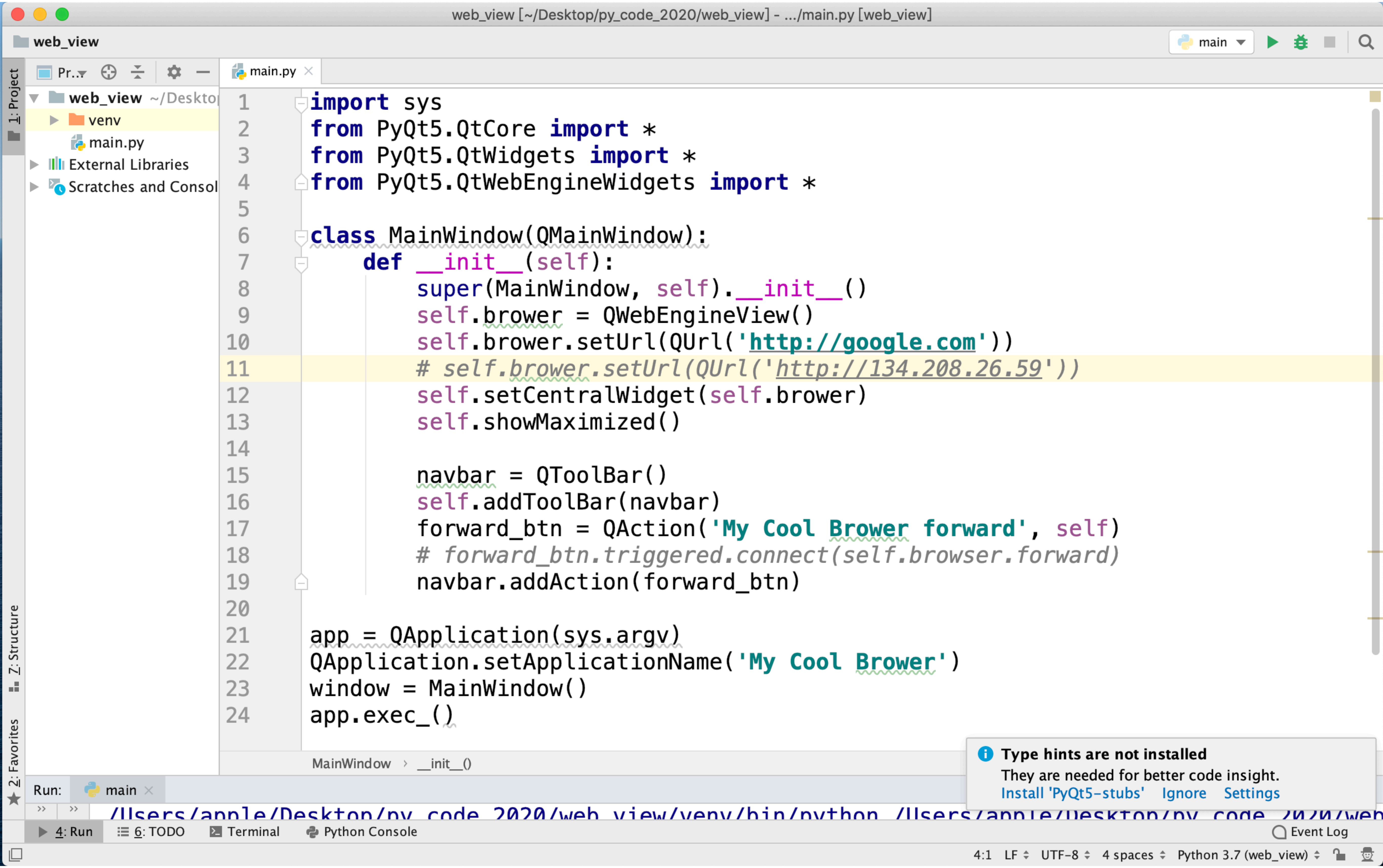
Python bindings for the Qt cross platform application toolkit

Project description

Qt is set of cross-platform C++ libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems. These include location and positioning services, multimedia, NFC and Bluetooth connectivity, a Chromium based web browser, as well as traditional UI development.

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android.

```
import sys
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from PyQt5.QtWebEngineWidgets import *

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.brower = QWebEngineView()
        self.brower.setUrl(QUrl('http://google.com'))
        # self.brower.setUrl(QUrl('http://134.208.26.59'))
        self.setCentralWidget(self.brower)
        self.showMaximized()

        navbar = QToolBar()
        self.addToolBar(navbar)
        forward_btn = QAction('My Cool Brower forward', self)
        # forward_btn.triggered.connect(self.browser.forward)
        navbar.addAction(forward_btn)

app = QApplication(sys.argv)
QApplication.setApplicationName('My Cool Brower')
window = MainWindow()
app.exec_()
```



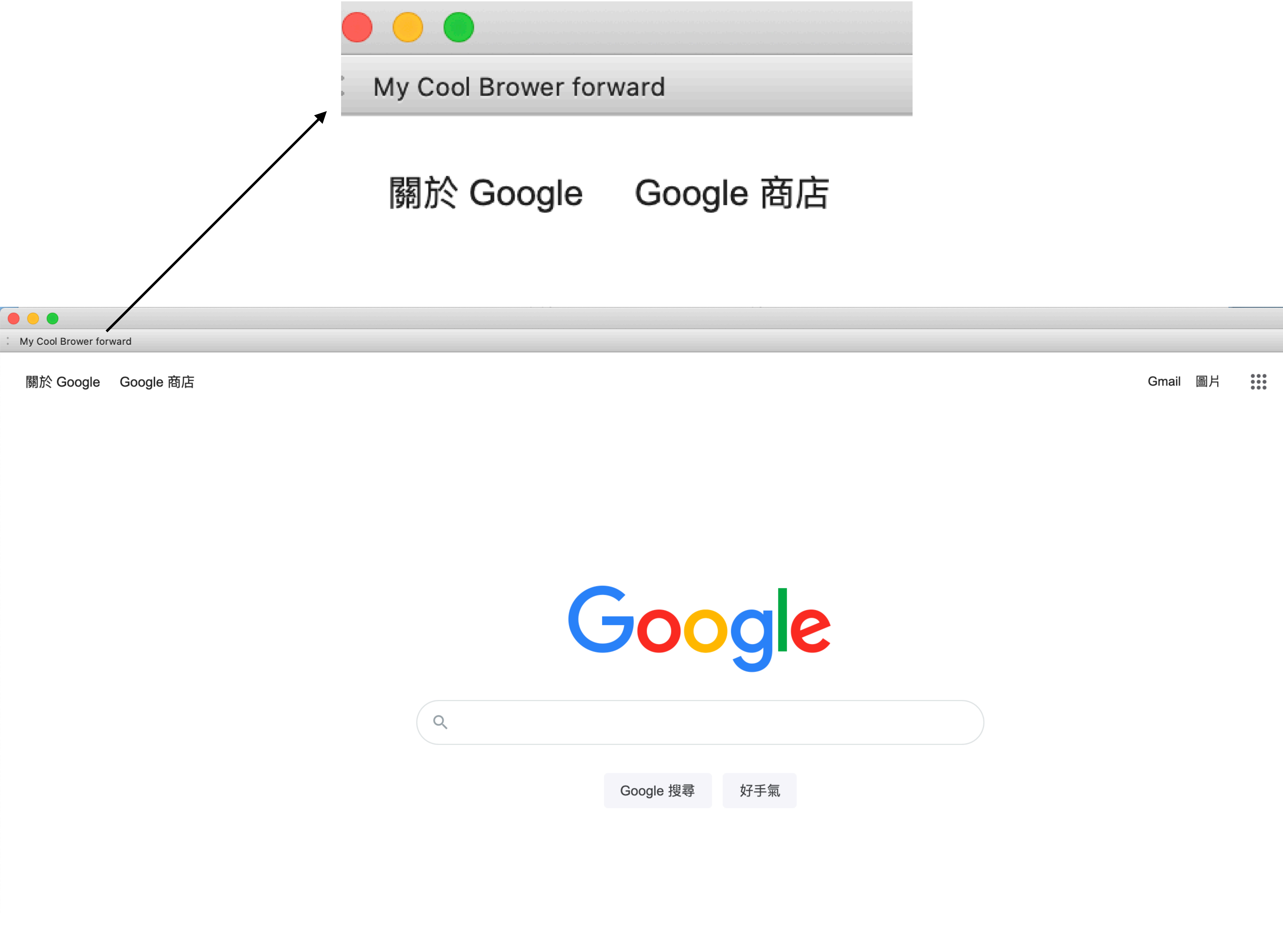
```python
import sys
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from PyQt5.QtWebEngineWidgets import *

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.brower = QWebEngineView()
        self.brower.setUrl(QUrl('http://google.com'))
        # self.brower.setUrl(QUrl('http://134.208.26.59'))
        self.setCentralWidget(self.brower)
        self.showMaximized()

        navbar = QToolBar()
        self.addToolBar(navbar)
        forward_btn = QAction('My Cool Brower forward', self)
        # forward_btn.triggered.connect(self.browser.forward)
        navbar.addAction(forward_btn)

app = QApplication(sys.argv)
QApplication.setApplicationName('My Cool Brower')
window = MainWindow()
app.exec_()
```

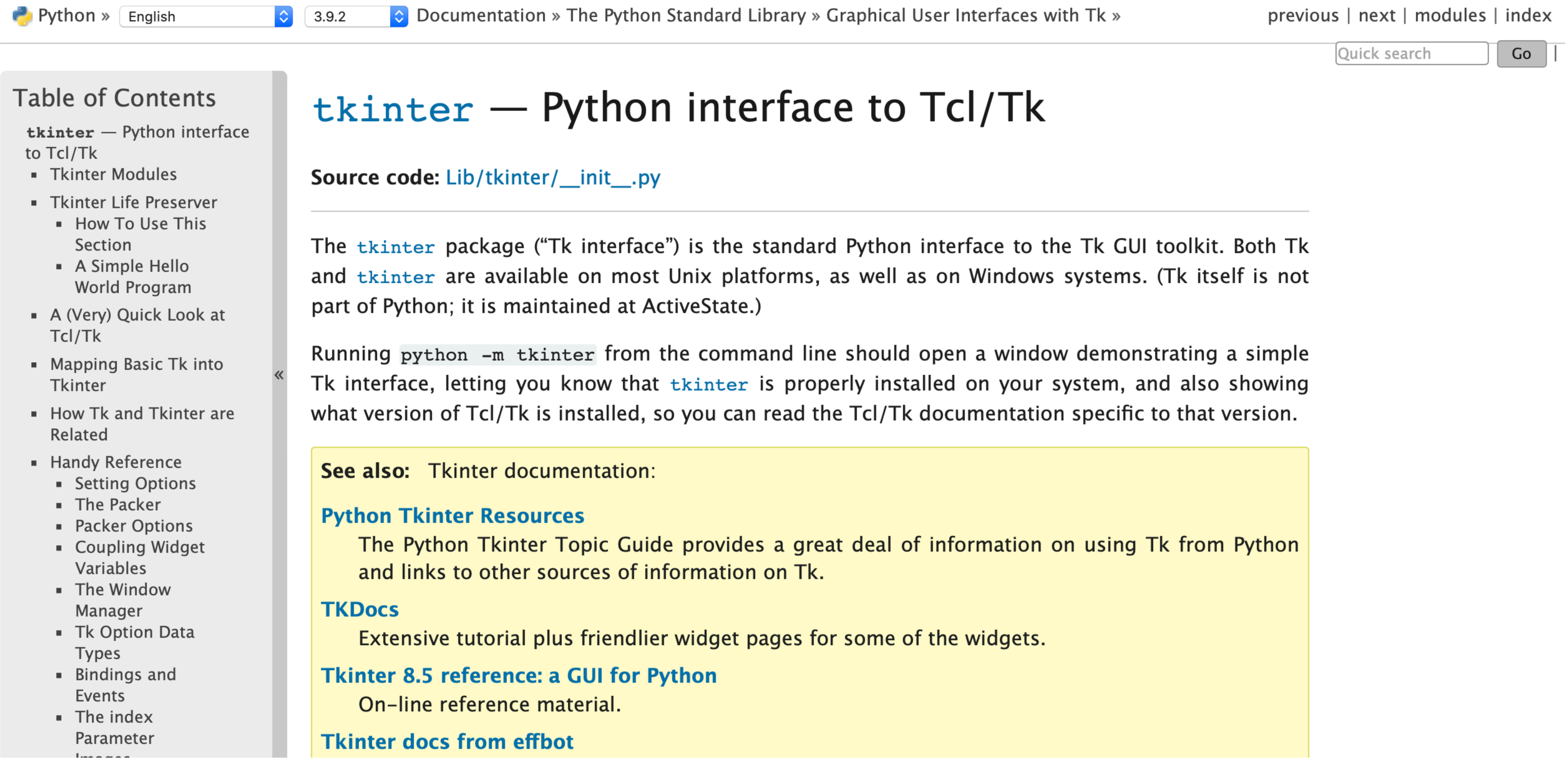
```
/Users/apple/Desktop/py_code_2020/web_view/venv/bin/python /Users/apple/Desktop/py_code_2020/web
```

Type hints are not installed

They are needed for better code insight.

Install 'PyQt5-stubs'   Ignore   Settings


My Cool Brower forward
關於 Google
Google 商店
My Cool Brower forward
關於 Google
Google 商店
Gmail
圖片
Google
Google 搜尋
好手氣



Python » English 3.9.2 Documentation » The Python Standard Library » Graphical User Interfaces with Tk »

previous | next | modules | index

Quick search Go |

Table of Contents

tkinter — Python interface to Tcl/Tk

- Tkinter Modules
- Tkinter Life Preserver
  - How To Use This Section
  - A Simple Hello World Program
- A (Very) Quick Look at Tcl/Tk
- Mapping Basic Tk into Tkinter
- How Tk and Tkinter are Related
- Handy Reference
  - Setting Options
  - The Packer
  - Packer Options
  - Coupling Widget Variables
  - The Window Manager
  - Tk Option Data Types
  - Bindings and Events
  - The index Parameter

`tkinter` — Python interface to Tcl/Tk

Source code: Lib/tkinter/__init__.py

The `tkinter` package ("Tk interface") is the standard Python interface to the Tk GUI toolkit. Both Tk and `tkinter` are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at ActiveState.)

Running `python -m tkinter` from the command line should open a window demonstrating a simple Tk interface, letting you know that `tkinter` is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

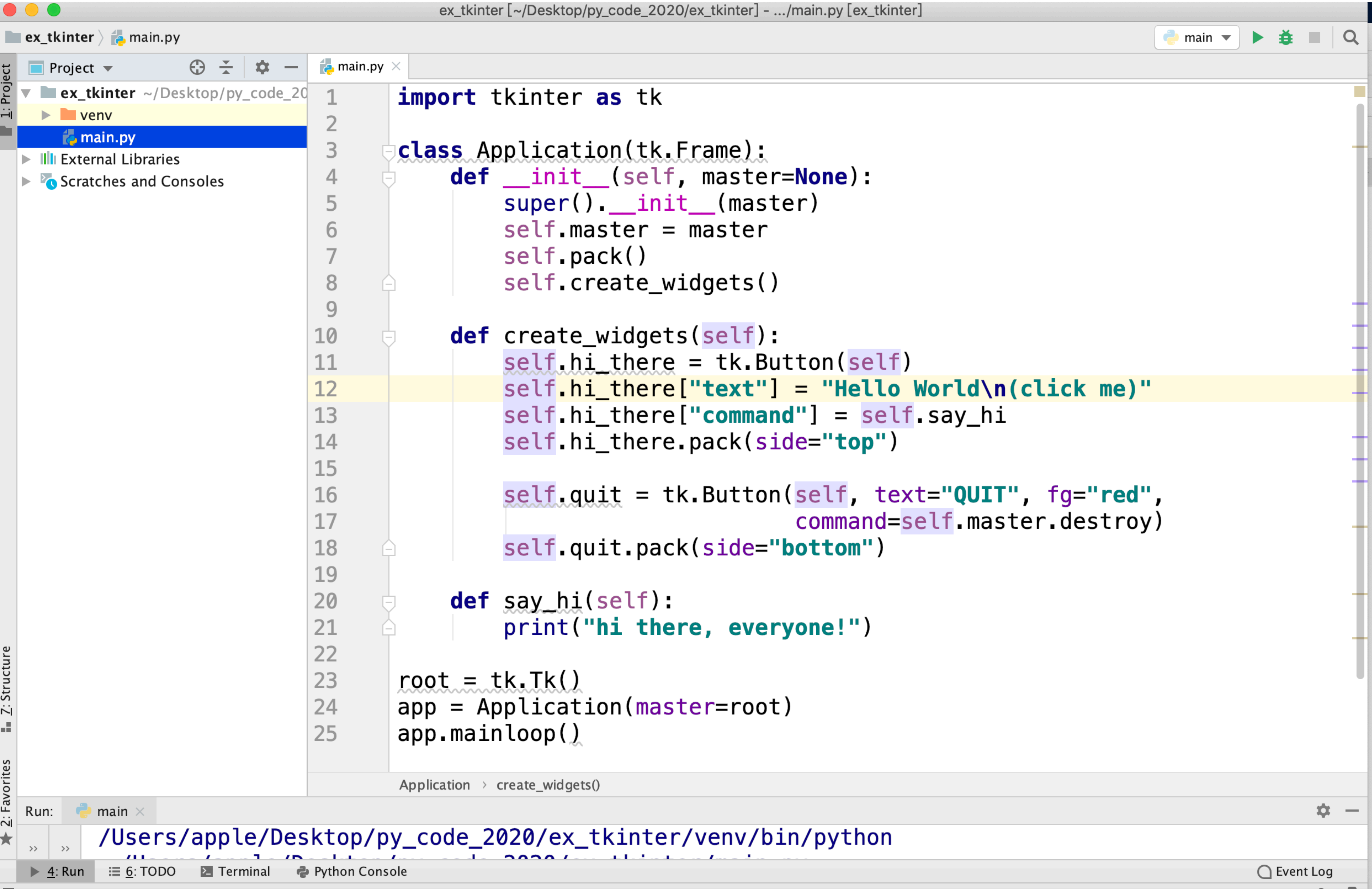
See also: Tkinter documentation:

Python Tkinter Resources
The Python Tkinter Topic Guide provides a great deal of information on using Tk from Python and links to other sources of information on Tk.

TKDocs
Extensive tutorial plus friendlier widget pages for some of the widgets.

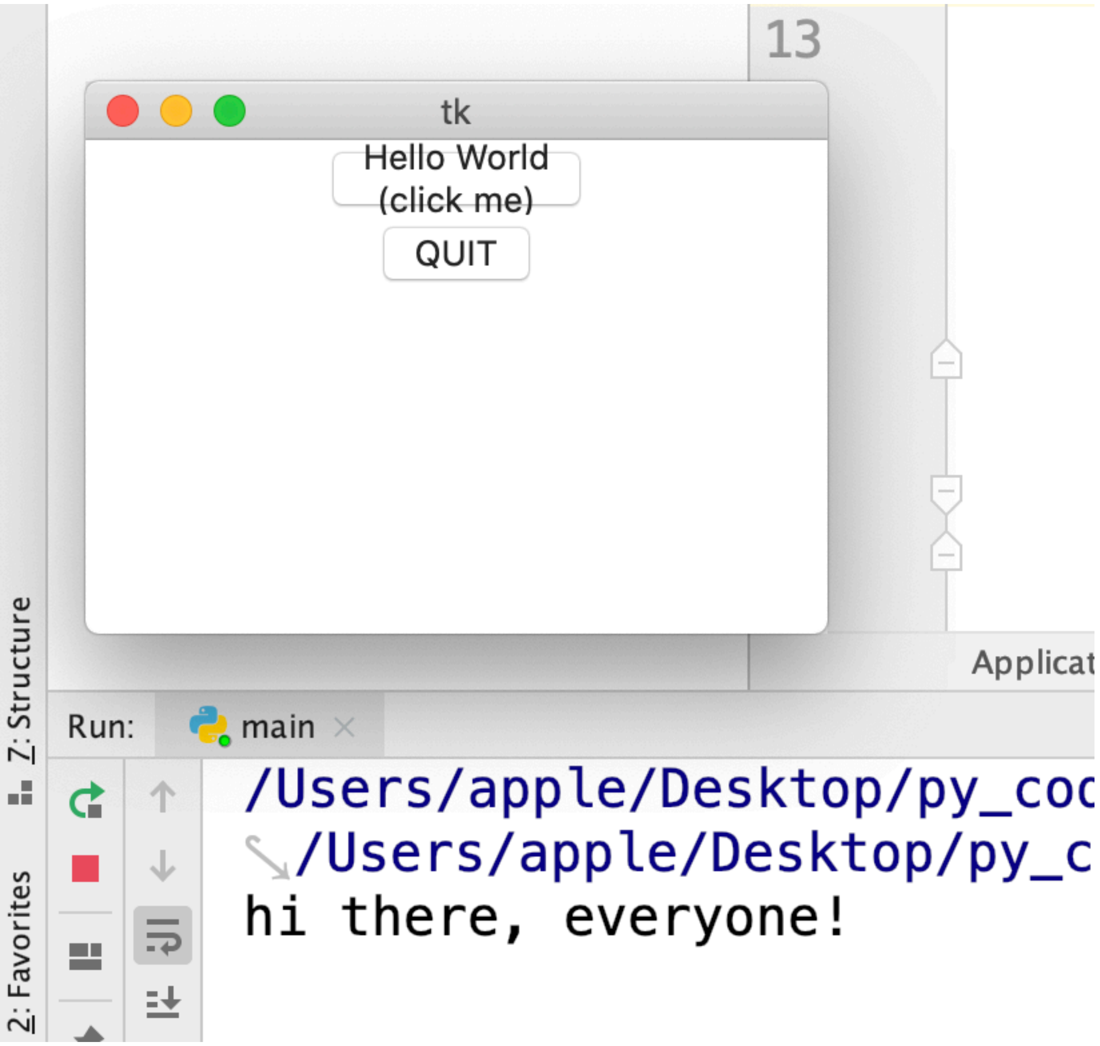
Tkinter 8.5 reference: a GUI for Python
On-line reference material.

Tkinter docs from effbot

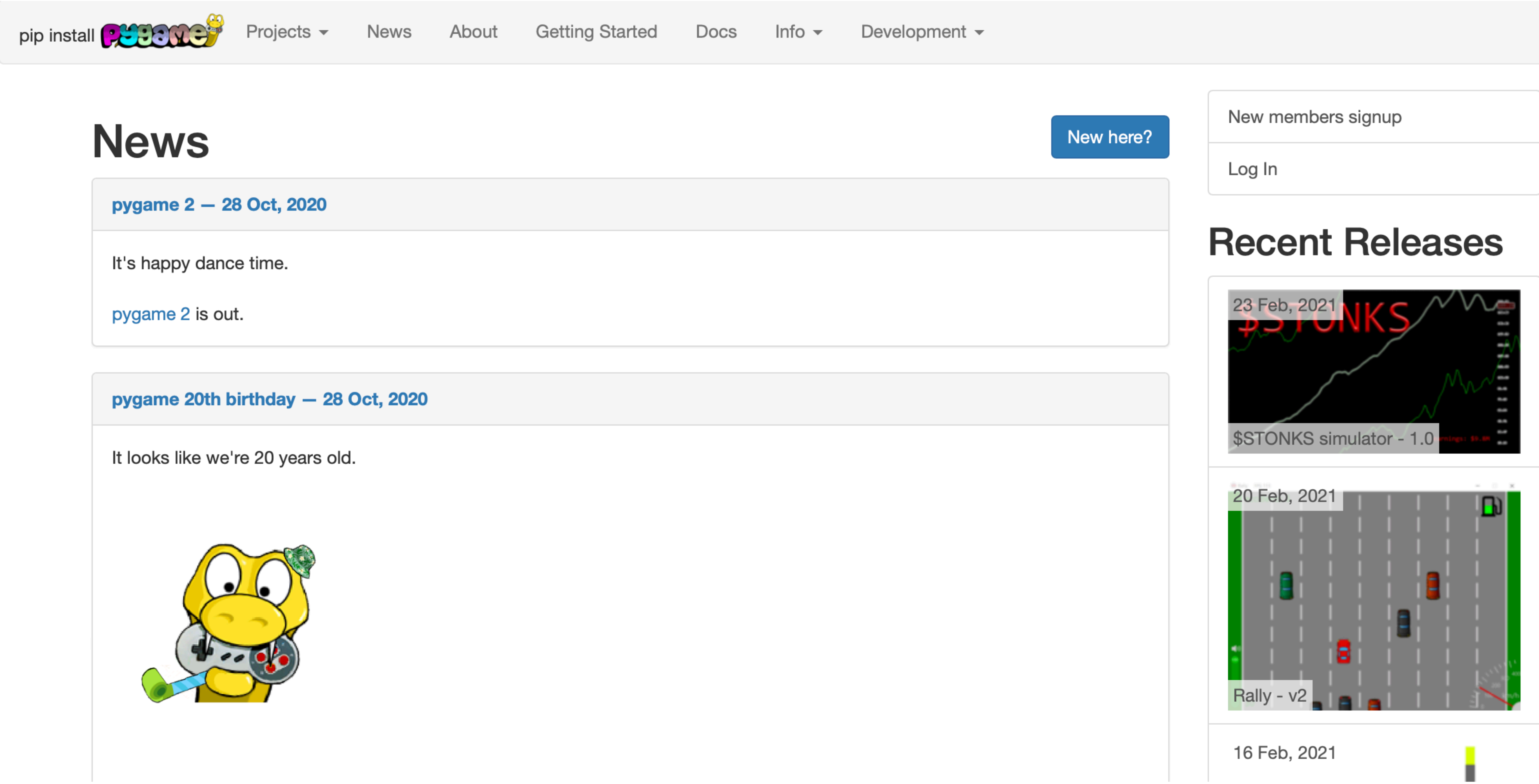


```python
import tkinter as tk

class Application(tk.Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.master = master
        self.pack()
        self.create_widgets()

    def create_widgets(self):
        self.hi_there = tk.Button(self)
        self.hi_there["text"] = "Hello World\n(click me)"
        self.hi_there["command"] = self.say_hi
        self.hi_there.pack(side="top")

        self.quit = tk.Button(self, text="QUIT", fg="red",
                              command=self.master.destroy)
        self.quit.pack(side="bottom")

    def say_hi(self):
        print("hi there, everyone!")

root = tk.Tk()
app = Application(master=root)
app.mainloop()
```

```
/Users/apple/Desktop/py_code_2020/ex_tkinter/venv/bin/python
/Users/apple/Desktop/py_code_2020/ex_tkinter/main.py
```



```
/Users/apple/Desktop/py_coc
/Users/apple/Desktop/py_c
hi there, everyone!
```



News

pygame 2 — 28 Oct, 2020

It's happy dance time.

pygame 2 is out.

pygame 20th birthday — 28 Oct, 2020

It looks like we're 20 years old.

Recent Releases

23 Feb, 2021

$STONKS simulator - 1.0

20 Feb, 2021

Rally - v2

16 Feb, 2021