Python series

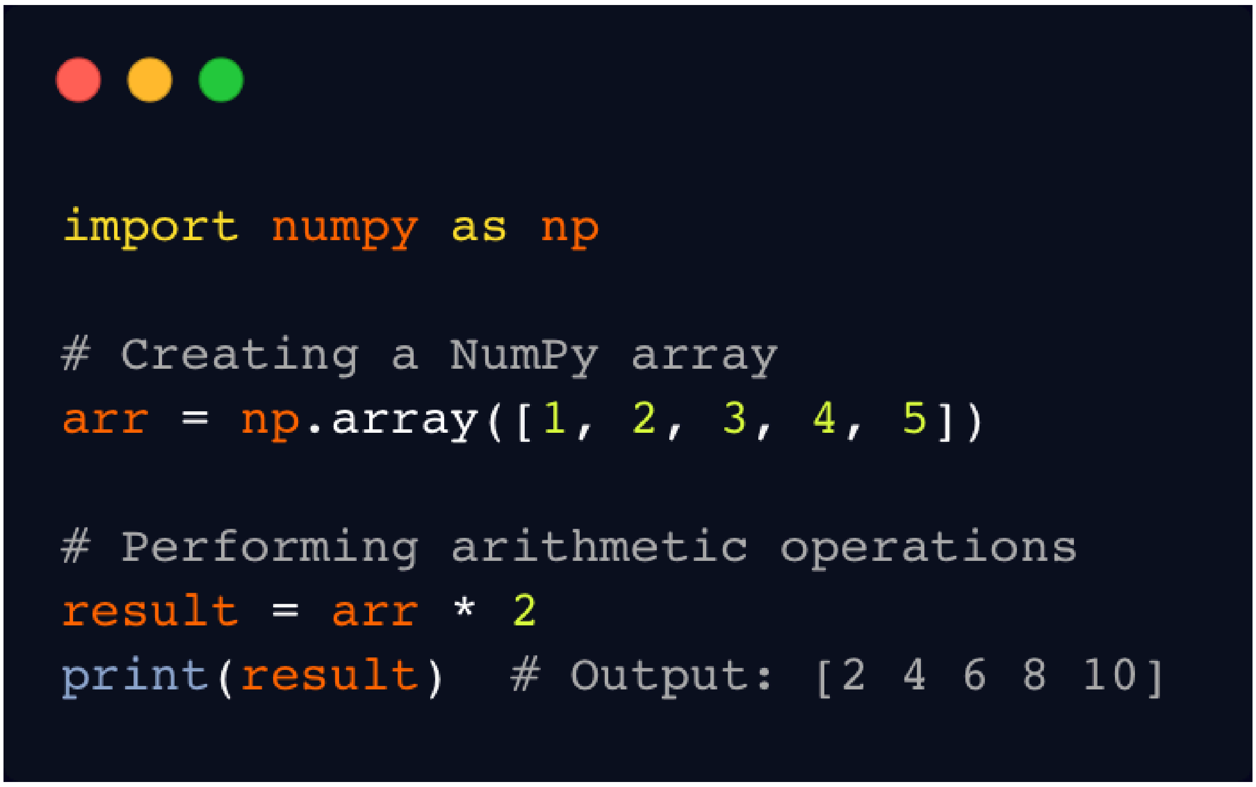
Introduction to Python libraries

With Code Examples

save for later

NumPy

NumPy is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays.



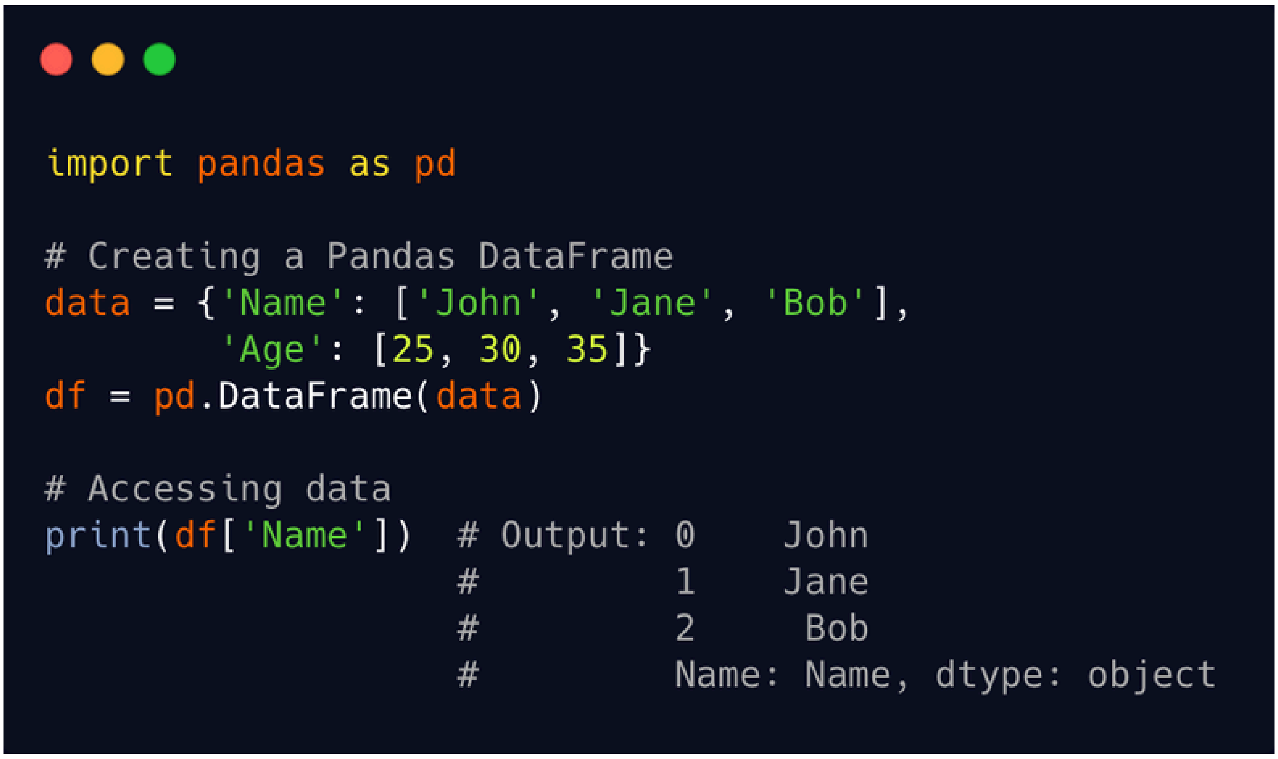
```
import numpy as np

# Creating a NumPy array
arr = np.array([1, 2, 3, 4, 5])

# Performing arithmetic operations
result = arr * 2
print(result)  # Output: [2 4 6 8 10]
```

Swipe next ⟶

follow for more

Pandas

Pandas is a powerful data manipulation and analysis library. It provides data structures and data analysis tools for working with structured (tabular, multidimensional, potentially heterogeneous) and time series data.



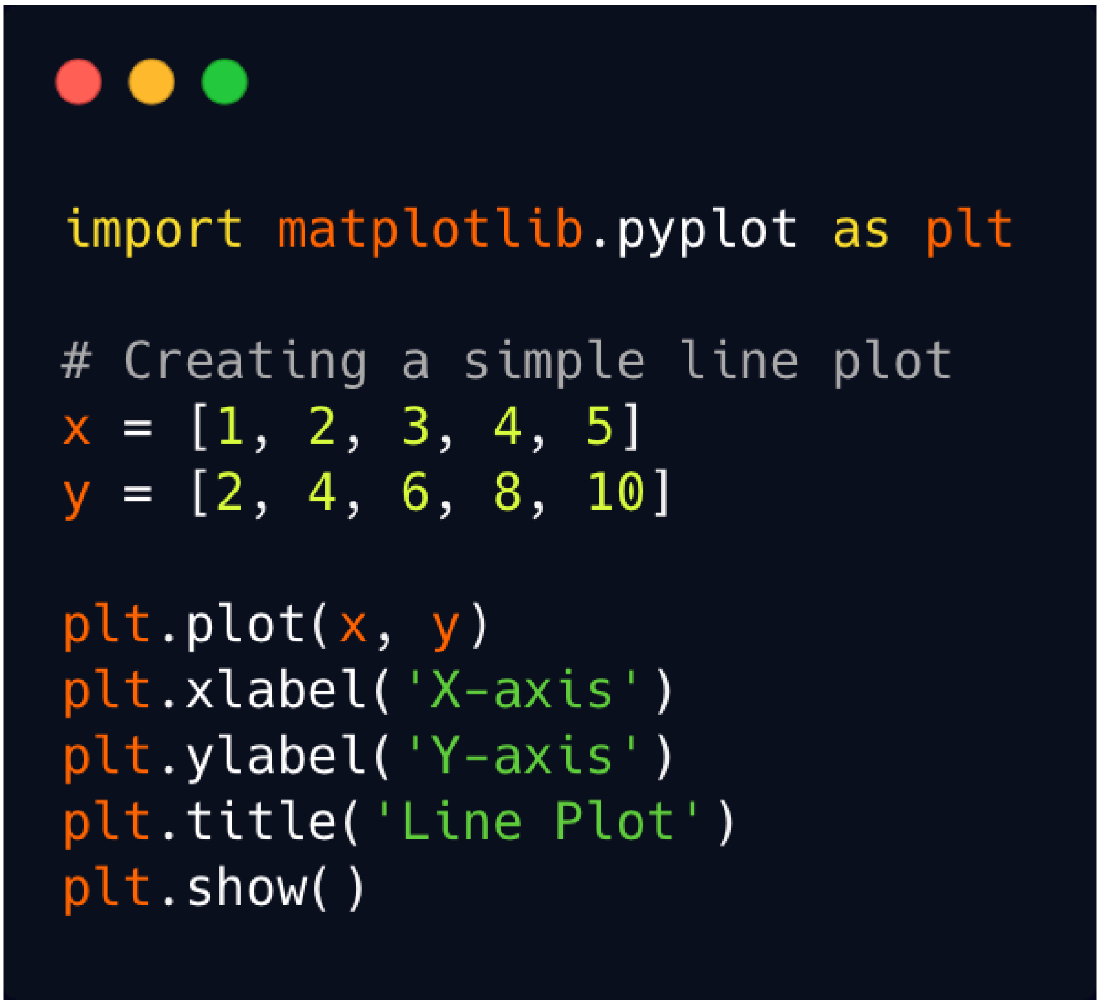
```python
import pandas as pd

# Creating a Pandas DataFrame
data = {'Name': ['John', 'Jane', 'Bob'],
        'Age': [25, 30, 35]}
df = pd.DataFrame(data)

# Accessing data
print(df['Name'])  # Output: 0    John
                   #         1    Jane
                   #         2     Bob
                   #         Name: Name, dtype: object
```

Swipe next ⟶

save for later

Matplotlib

Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. It can be used in Python scripts, the Python and IPython shells, web application servers, and various graphical user interface toolkits.



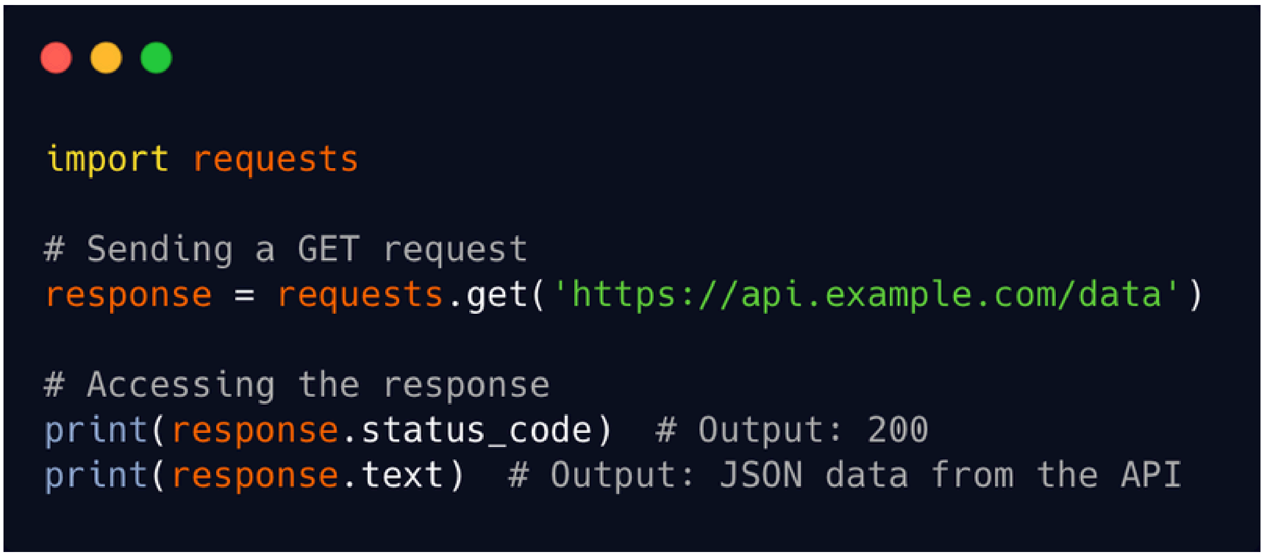
```python
import matplotlib.pyplot as plt

# Creating a simple line plot
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

plt.plot(x, y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot')
plt.show()
```

Swipe next ⟶

follow for more

Requests

The Requests library is designed to be the simplest way to make HTTP requests in Python. It provides a high-level, user-friendly interface for sending HTTP/1.1 requests, handling cookies, file uploads, and more.



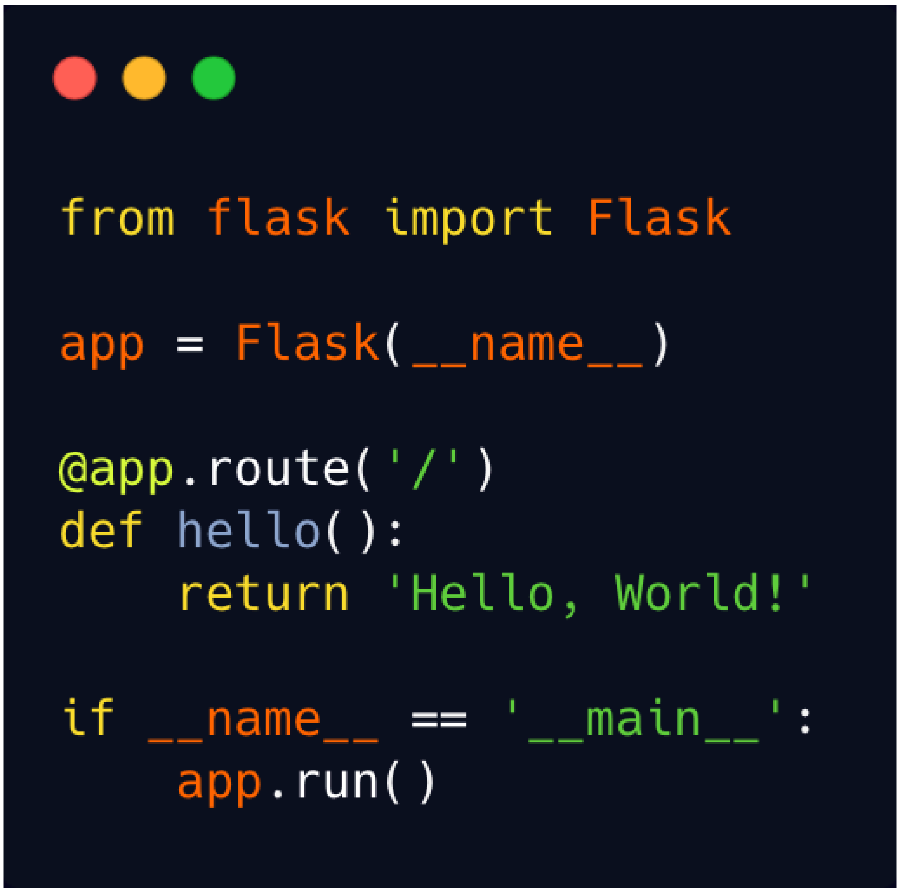
```python
import requests

# Sending a GET request
response = requests.get('https://api.example.com/data')

# Accessing the response
print(response.status_code)  # Output: 200
print(response.text)  # Output: JSON data from the API
```

Swipe next ⟶

save for later

Flask

Flask is a lightweight, flexible, and easy-to-use web framework for Python. It is designed to get started quickly and scale up to complex applications ranging from simple APIs to full-featured web applications.



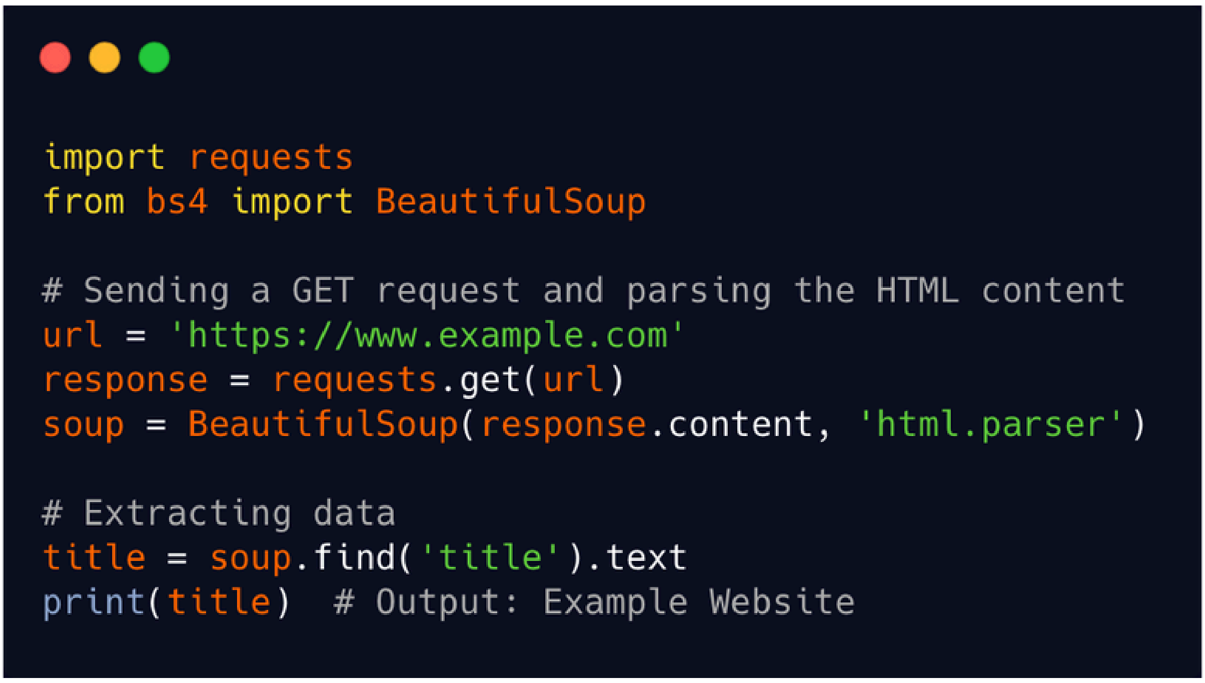
```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World!'

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

Swipe next ⟶

follow for more

Beautiful Soup

Beautiful Soup is a Python library for parsing structured data. It provides a simple way to navigate, search, and modify the tree-like structure of HTML and XML documents, making web scraping tasks easier.



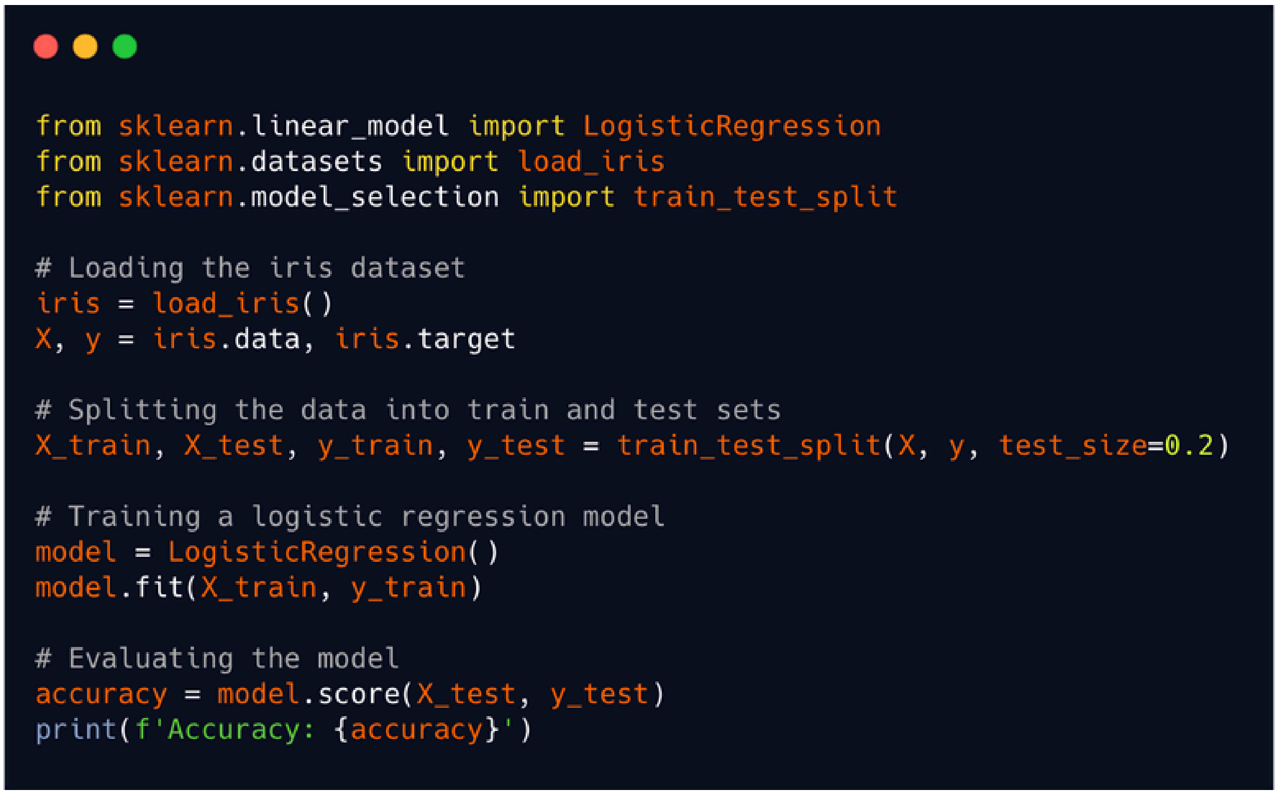
```python
import requests
from bs4 import BeautifulSoup

# Sending a GET request and parsing the HTML content
url = 'https://www.example.com'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')

# Extracting data
title = soup.find('title').text
print(title)  # Output: Example Website
```

Swipe next ⟶

save for later

Scikit-learn

Scikit-learn is a machine learning library for Python. It features various classification, regression, and clustering algorithms, including support vector machines, random forests, gradient boosting, and more. It also provides tools for model evaluation and data preprocessing.



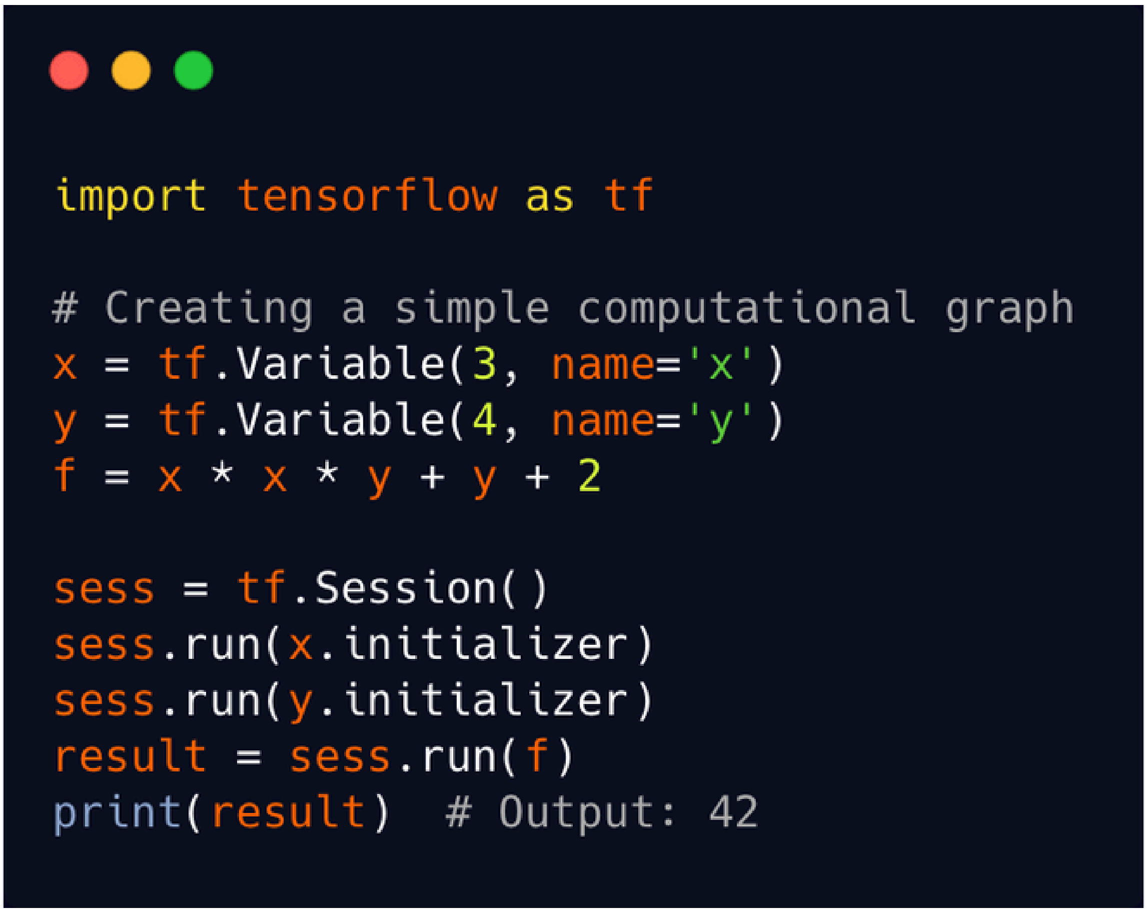
```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Loading the iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Training a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Evaluating the model
accuracy = model.score(X_test, y_test)
print(f'Accuracy: {accuracy}')
```

Swipe next ⟶

follow for more

TensorFlow

TensorFlow is an open-source library for machine learning and artificial intelligence. It provides a flexible ecosystem of tools, libraries, and community resources to help developers build and deploy machine learning models across various platforms and devices.



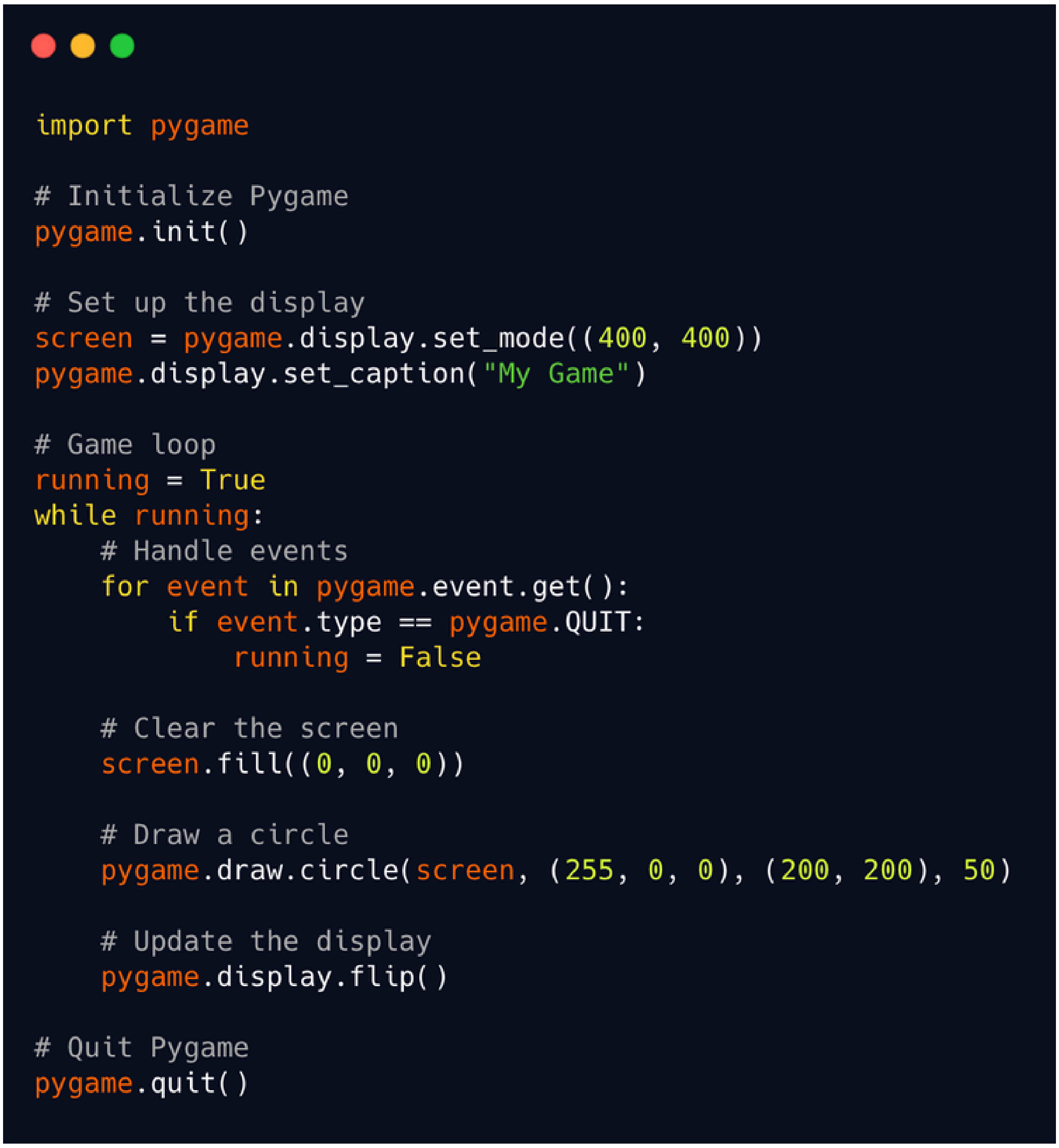
```python
import tensorflow as tf

# Creating a simple computational graph
x = tf.Variable(3, name='x')
y = tf.Variable(4, name='y')
f = x * x * y + y + 2

sess = tf.Session()
sess.run(x.initializer)
sess.run(y.initializer)
result = sess.run(f)
print(result)  # Output: 42
```

Swipe next ⟶

save for later

Pygame

Pygame is a set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.



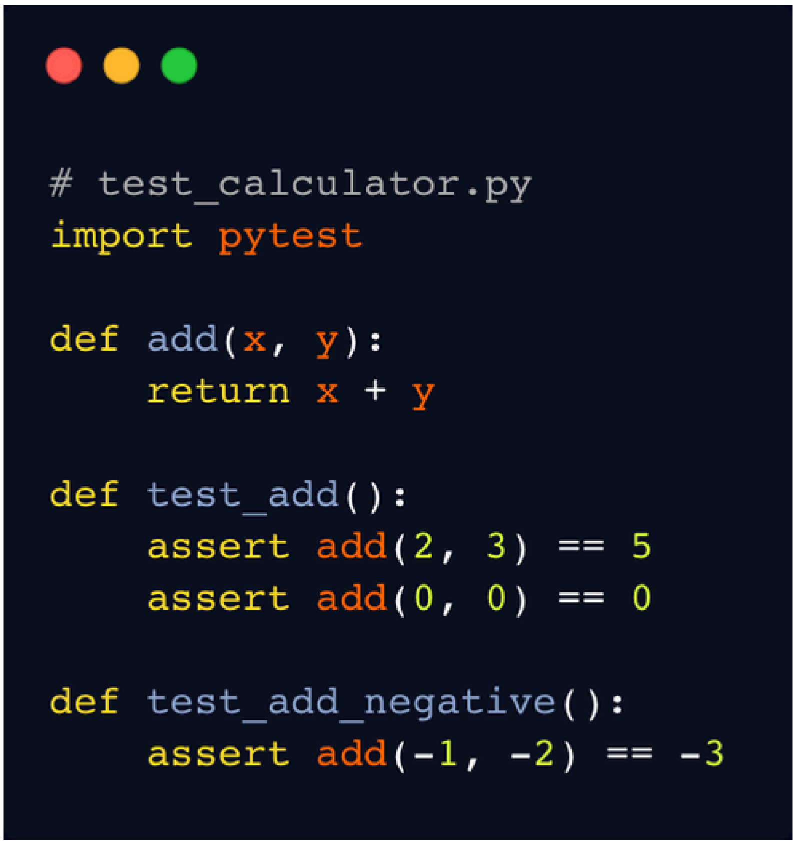
```python
import pygame

# Initialize Pygame
pygame.init()

# Set up the display
screen = pygame.display.set_mode((400, 400))
pygame.display.set_caption("My Game")

# Game loop
running = True
while running:
    # Handle events
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Clear the screen
    screen.fill((0, 0, 0))

    # Draw a circle
    pygame.draw.circle(screen, (255, 0, 0), (200, 200), 50)

    # Update the display
    pygame.display.flip()

# Quit Pygame
pygame.quit()
```

Swipe next ⟶

follow for more

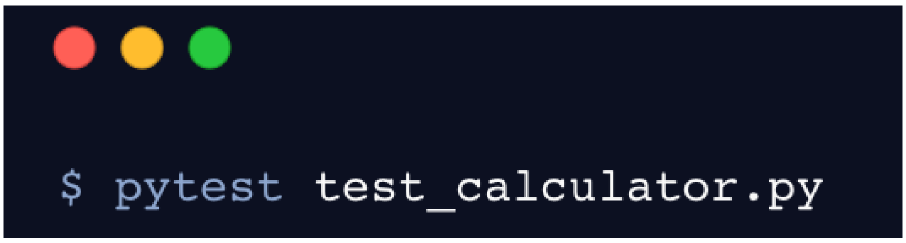
pytest

pytest is a popular testing framework for Python that makes it easy to write, organize, and run tests. It provides a simple and user-friendly syntax, supports fixtures for test setup and teardown, and offers powerful features for test parameterization and test reporting.



```python
# test_calculator.py
import pytest

def add(x, y):
    return x + y

def test_add():
    assert add(2, 3) == 5
    assert add(0, 0) == 0

def test_add_negative():
    assert add(-1, -2) == -3
```

To run the tests, you can use the **pytest** command in the terminal:



```
$ pytest test_calculator.py
```