Your Guide to the Innoverse Programming Challenges

49 6e 6e 6f 76 65 72 73 65 20 49 6e 76 65 6e 74 69 6f 6e 20 26 20 49 6e 6e 6f 76 61 74 69 6f 6e 20 45 78 70 6f

Version: 1.0.1
May 2024, GA

Overview

This guide provides detailed instructions for creating a programming challenge section for the Innoverse Invention & Innovation Expo. This section will include a downloadable PDF guide that outlines the challenges, instructions for participants, and guidelines for submitting their solutions via GitHub.

Introduction

Welcome to the Innoverse Invention & Innovation Expo Programming Challenge! This event aims to develop creativity, problem-solving, and technical skills among participants. In this guide, you will find everything you need to participate in our programming challenges, which are designed to be both fun and intellectually stimulating.

How to Participate

1. Register for the Innoverse Invention & Innovation Expo.
2. Download this PDF guide and read through the challenges.
3. Solve the challenge by writing code.
4. Upload your solution to GitHub.
5. Email us the link to your GitHub repository at [challenge@innoverse.world].

Submission Guidelines

1. **Create a GitHub Repository**

    - Create a new GitHub repository for your solution.
    - Name your repository in the format: ExpoChallenge_YourName.

2. **Upload Your Code**

    - Upload your code and any related files to the repository.
    - Ensure your code is well-structured and organized.

3. **Add a README File**

    - Include a README.md file in your repository for documentation.
    - The README should contain:
        - A brief description of your solution.
        - Instructions on how to run your code.
        - Any dependencies or libraries required.
        - Your contact information.

4. **Include a Payment Receipt**

    - Attach the receipt for your payment in the email when you submit your repository link.
    - Ensure the receipt is clearly legible and includes your name and the payment amount.

5. **Submit Your Solution**

    - Email the link to your GitHub repository to [challenge@innoverse.world] with the subject line "Programming Challenge Submission."
    - In the email body, include:
        - Your full name.
        - A brief summary of your approach to solving the challenge.
        - Attach the payment receipt.

Example Submission Email

Dear Innoverse Team,

I/we are submitting my/our solution for the [challenge name] challenge.

GitHub Repository Link: [Your Repository Link]

Summary of Approach:

[Briefly describe your approach to solving the challenge.]

If participating as a group, our team members include: (maximum 5)

[Member 1 Name]

[Member 2 Name]

[Member 3 Name]

[Additional Members]

Attached is the receipt for my/our payment.

Thank you for this opportunity. I/we look forward to participating in the Innoverse Invention & Innovation Expo.

Best regards,

[Your Name/Team Leader's Name]

[Your Contact Information/On behalf of [Team Name]]

Judging Criteria for Programming Challenge

1. Correctness (40%)

- **Accuracy**: The solution correctly addresses the problem statement and produces the expected results.
- **Completeness**: All parts of the problem are fully solved as per the requirements specified in the challenge.
- **Functionality**: The solution runs without errors and handles edge cases appropriately.

2. Efficiency (20%)

- **Performance**: The solution executes efficiently, using optimal algorithms and data structures to minimize time and space complexity.
- **Scalability**: The solution can handle large inputs and performs well under various conditions and constraints.

3. Code Quality (20%)

- **Readability**: Code is clean, well-organized, and easy to read. Proper use of indentation, naming conventions, and comments.
- **Modularity**: Code is modular, with functions and methods that have a single responsibility and can be reused.
- **Documentation**: Adequate comments and documentation are provided to explain the logic, purpose of functions, and how to run the code.

4. Innovation and Creativity (10%)

- **Uniqueness**: The approach or solution is unique or shows creative problem-solving techniques.
- **Innovative Use of Technologies**: Utilization of libraries, frameworks, or tools in an innovative way to enhance the solution.

5. Presentation and Submission (10%)

- **GitHub Repository**: The repository is well-structured, with a clear README file explaining the solution, instructions to run the code, and any dependencies.
- **Submission Process**: The solution is submitted correctly following the guidelines, including the proper format for filenames, documentation, and any required supplementary materials.

- **Professionalism**: The submission is professional and polished, reflecting a high level of effort and attention to detail.

Bonus Points

- **Extra Features**: Implementation of additional features or functionalities that go beyond the basic requirements of the challenge.
- **Testing**: Comprehensive test cases are provided, covering a wide range of scenarios and edge cases.

Disqualification Criteria

- **Plagiarism**: Any evidence of copying or using others' work without proper attribution.
- **Non-compliance**: Failure to follow the submission guidelines or meet the basic requirements of the challenge.

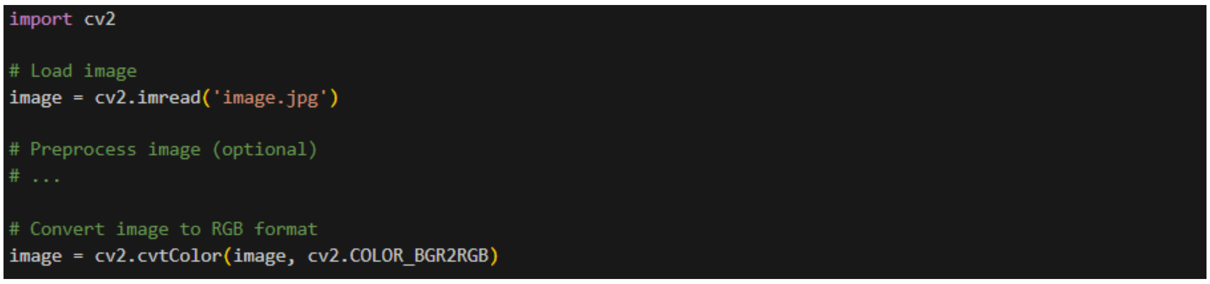
Challenge Questions

AI-Powered Crop Health Detection Challenge

Story: A large agricultural company is currently using traditional methods to assess the health of its crops across its vast farmlands. These methods are time-consuming, expensive, and sometimes inaccurate. The company is looking for an innovative solution to overcome these challenges and increase their productivity and profits.

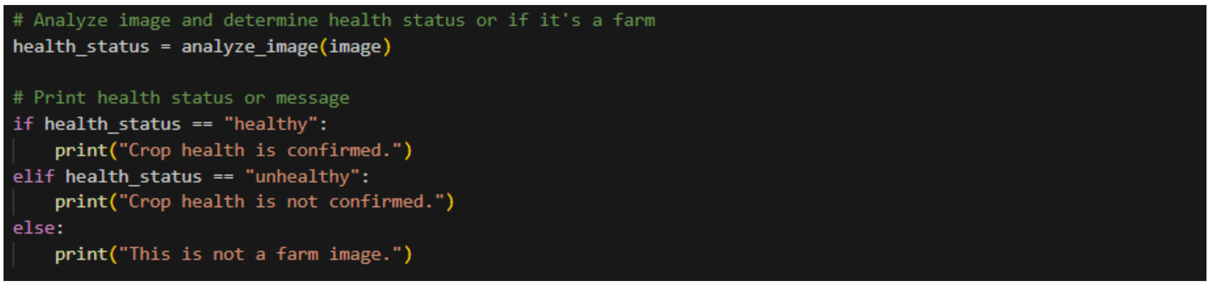
Challenge Topic and Objective: The goal of this challenge is to create an AI program using the Python programming language that can determine the health of crops in a given image. The program should be able to take various images of farms as input and provide an output for each image as "Healthy Crops Confirmed," "Unhealthy Crops Detected," or "This is not a farm image."

Sample Input:



```python
import cv2

# Load image
image = cv2.imread('image.jpg')

# Preprocess image (optional)
# ...

# Convert image to RGB format
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

Sample Output:



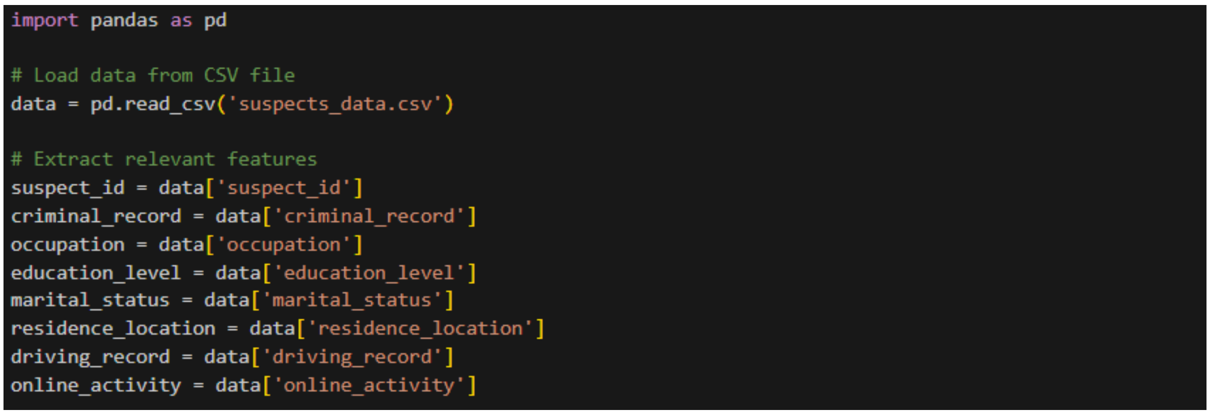
```python
# Analyze image and determine health status or if it's a farm
health_status = analyze_image(image)

# Print health status or message
if health_status == "healthy":
    print("Crop health is confirmed.")
elif health_status == "unhealthy":
    print("Crop health is not confirmed.")
else:
    print("This is not a farm image.")
```

AI-Powered Criminality Prediction Challenge for Robot Police

Story: In a world where robots and humans coexist, a city police department is upgrading its systems to combat crime using artificial intelligence. They are seeking a new AI-powered robot police officer to join their force and assist in predicting potential criminals.

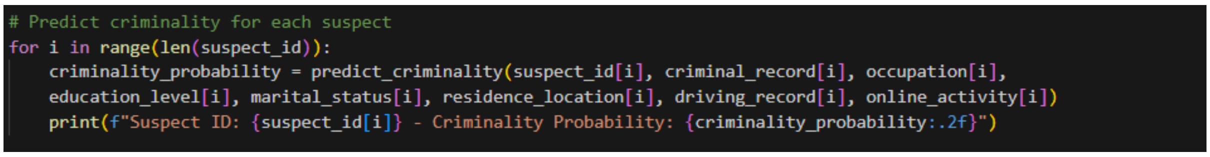
Challenge Topic and Objective: The goal of this challenge is to create an AI program using the Python programming language that can predict the likelihood of suspects being criminals based on their information. The program should be able to utilize various data such as criminal records, occupation, education level, marital status, residence location, driving history, and online activity to train its model and predict the criminality of new individuals.

Sample Input:



```python
import pandas as pd

# Load data from CSV file
data = pd.read_csv('suspects_data.csv')

# Extract relevant features
suspect_id = data['suspect_id']
criminal_record = data['criminal_record']
occupation = data['occupation']
education_level = data['education_level']
marital_status = data['marital_status']
residence_location = data['residence_location']
driving_record = data['driving_record']
online_activity = data['online_activity']
```

Sample Output:



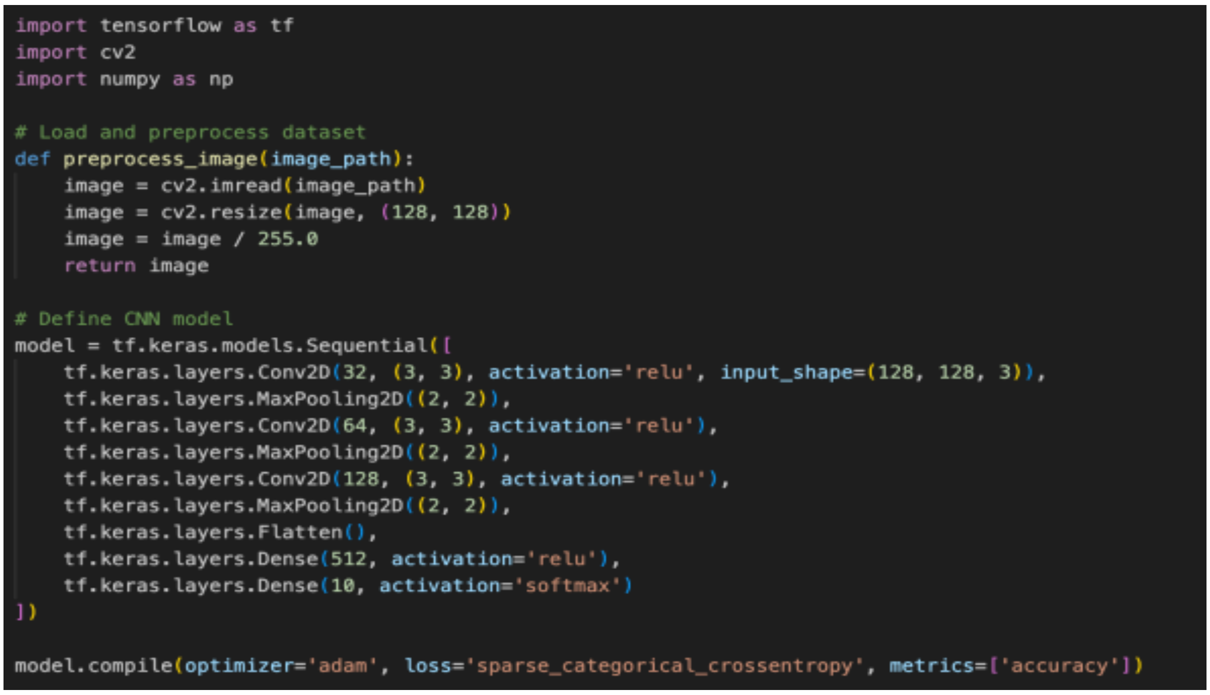
```python
# Predict criminality for each suspect
for i in range(len(suspect_id)):
    criminality_probability = predict_criminality(suspect_id[i], criminal_record[i], occupation[i],
    education_level[i], marital_status[i], residence_location[i], driving_record[i], online_activity[i])
    print(f"Suspect ID: {suspect_id[i]} - Criminality Probability: {criminality_probability:.2f}")
```

The Deep Learning Image Classifier

Story: You are developing a state-of-the-art image classifier to identify objects in space imagery. This involves building and training a deep learning model using convolutional neural networks (CNNs).

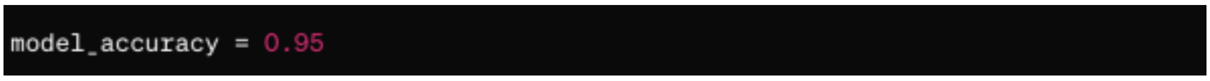
Challenge Topic and Objective: Write a program that uses TensorFlow or PyTorch to build, train, and evaluate a CNN model for image classification. Use additional libraries such as OpenCV for image preprocessing

Sample Input:



```python
import tensorflow as tf
import cv2
import numpy as np

# Load and preprocess dataset
def preprocess_image(image_path):
    image = cv2.imread(image_path)
    image = cv2.resize(image, (128, 128))
    image = image / 255.0
    return image

# Define CNN model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Sample Output:



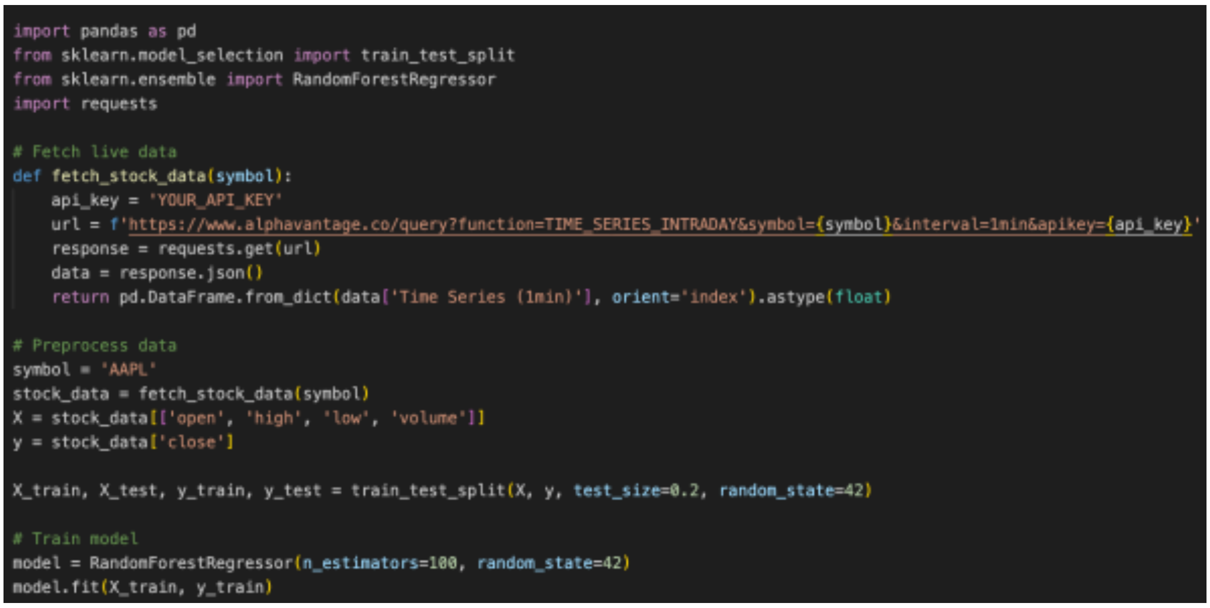
```
model_accuracy = 0.95
```

The Real-Time Stock Market Predictor

Story: You are developing a real-time stock market prediction system using machine learning. This involves fetching live data, processing it, and predicting future stock prices

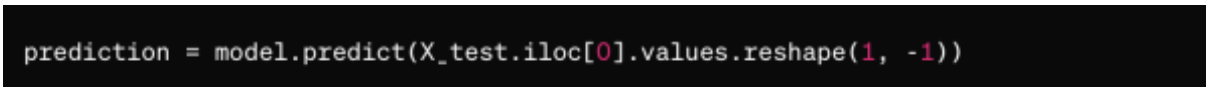
Challenge Topic and Objective: Write a program that uses libraries such as Scikit-Learn, Pandas, and Statsmodels to build a predictive model. Additionally, use the Alpha Vantage API to fetch live stock data

Sample Input:



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
import requests

# Fetch live data
def fetch_stock_data(symbol):
    api_key = 'YOUR_API_KEY'
    url = f'https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol={symbol}&interval=1min&apikey={api_key}'
    response = requests.get(url)
    data = response.json()
    return pd.DataFrame.from_dict(data['Time Series (1min)'], orient='index').astype(float)

# Preprocess data
symbol = 'AAPL'
stock_data = fetch_stock_data(symbol)
X = stock_data[['open', 'high', 'low', 'volume']]
y = stock_data['close']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Sample Output:



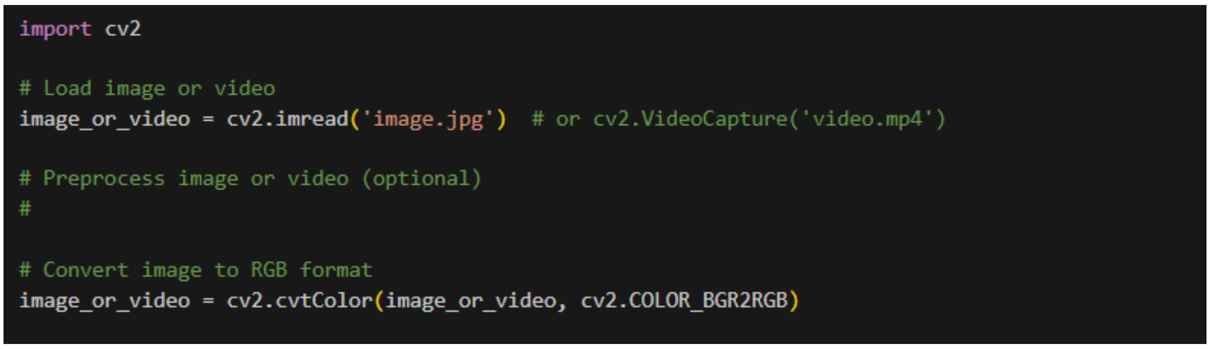
```python
prediction = model.predict(X_test.iloc[0].values.reshape(1, -1))
```

AI-Powered Facial Emotion Recognition Challenge

Story: Artificial intelligence (AI) is rapidly finding applications in various fields, including facial emotion recognition. This technology has the potential to revolutionize how we interact with machines and understand human behavior.

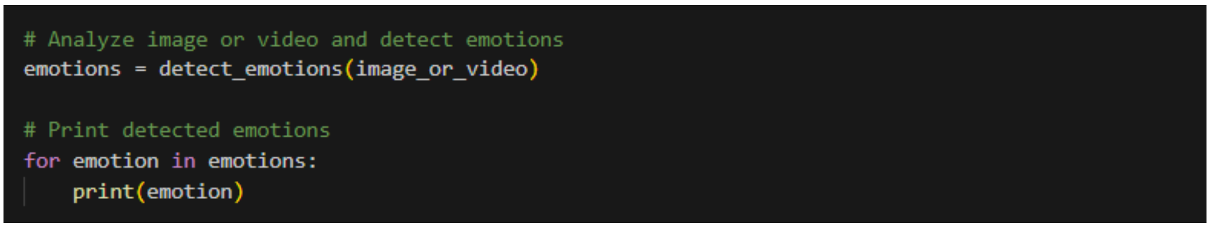
Challenge Topic and Objective: The goal of this challenge is to create an AI program using the Python programming language that can recognize emotions from human faces in images and videos. The program should be able to take various images and videos as input and provide an output for each image or video classifying the emotion.

Sample Input:



```python
import cv2

# Load image or video
image_or_video = cv2.imread('image.jpg')  # or cv2.VideoCapture('video.mp4')

# Preprocess image or video (optional)
#

# Convert image to RGB format
image_or_video = cv2.cvtColor(image_or_video, cv2.COLOR_BGR2RGB)
```

Sample Output:



```python
# Analyze image or video and detect emotions
emotions = detect_emotions(image_or_video)

# Print detected emotions
for emotion in emotions:
    print(emotion)
```

The Autonomous Drone Navigation

Story: You are tasked with developing a navigation system for an autonomous drone. The drone must navigate through an unknown terrain using sensor data and computer vision techniques.

Challenge Topic and Objective: Write a program that uses ROS (Robot Operating System) and OpenCV to process sensor data and control the drone. Implement SLAM (Simultaneous Localization and Mapping) using GMapping or Hector SLAM.

Sample Input:

```python
import rospy
from sensor_msgs.msg import LaserScan
import cv2

def callback(data):
    # Process laser scan data
    pass

def main():
    rospy.init_node('drone_navigation', anonymous=True)
    rospy.Subscriber('/scan', LaserScan, callback)
    rospy.spin()

if __name__ == '__main__':
    main()
```

Sample Output:

```python
drone_path = [(0, 0), (1, 1), (2, 2), (3, 3)]
```

Resources

To help you tackle these challenges, here is a list of recommended resources, including documentation, tutorials, and libraries:

General Resources

- Python Documentation: Comprehensive resource for Python programming.
- GitHub Guides: Tutorials on using GitHub effectively for version control and collaboration.

Algorithms and Data Structures

- GeeksforGeeks Algorithms: Extensive tutorials on various algorithms and data structures.
- Introduction to Algorithms: A classic textbook by Cormen, Leiserson, Rivest, and Stein.

Machine Learning and Deep Learning

- Scikit-Learn Documentation: User guide and API reference for the Scikit-Learn library.
- TensorFlow Documentation: Guides and tutorials for TensorFlow.
- PyTorch Documentation: Comprehensive resource for PyTorch.
- Kaggle: Platform for data science competitions and datasets.

Genetic Algorithms

- DEAP Documentation: User guide and API reference for the DEAP library.
- Genetic Algorithms Tutorial: Introduction to genetic algorithms and their applications.

Image Processing

- OpenCV Documentation: Tutorials and reference for the OpenCV library.
- Deep Learning for Computer Vision with Python: A book that covers deep learning techniques for image processing.

Real-Time Data Processing

- Pandas Documentation: User guide and reference for the Pandas library.

- Alpha Vantage API Documentation: Guide for using the Alpha Vantage API for fetching live financial data.
- Statsmodels Documentation: Resource for statistical modeling and econometrics in Python.

Robotics and Autonomous Systems

- ROS Documentation: Comprehensive resource for the Robot Operating System (ROS).
- SLAM Tutorials: Tutorials on implementing Simultaneous Localization and Mapping (SLAM) using various techniques.
- OpenCV for Robotics: A course on using OpenCV for robotics applications.