AI_Simulation_Infrastructure_ShaneciaHolden

December 9, 2024

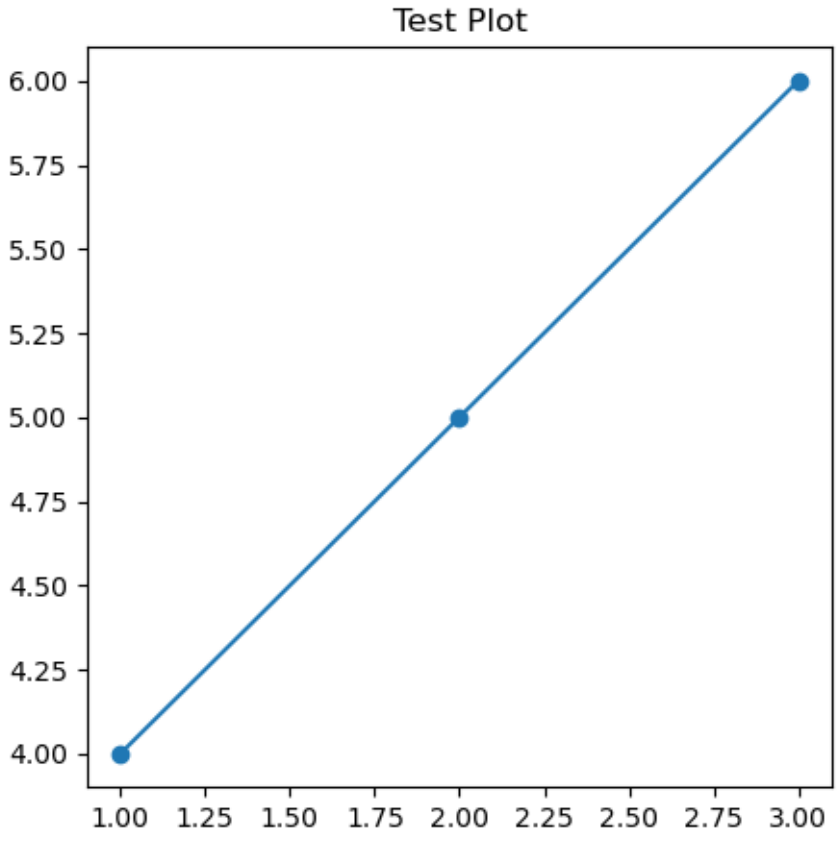
[154]:
```
pip install matplotlib
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: matplotlib in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (1.4.4)
Requirement already satisfied: numpy<2,>=1.21 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[155]:
```
import matplotlib.pyplot as plt
print("Matplotlib imported successfully!")
```

```
Matplotlib imported successfully!
```

[156]:
```
import matplotlib.pyplot as plt

%matplotlib inline

# Test plot
plt.figure(figsize=(5, 5))
plt.plot([1, 2, 3], [4, 5, 6], marker='o')
plt.title("Test Plot")
plt.show()
```



[157]:
```
import matplotlib as plt
```

[158]:
```
import numpy as np
```

[159]:
```
import pandas as pd
```

[160]:
```
import random
```

[161]:
```
import matplotlib as plt
```

[162]:
```
import psutil
```

[163]:
```
import time
```

[164]:
```
def generate_sensor(num_sensors, num_readings): return np.ran(num_sensors,␣
 ↪num_readings)
```

[165]:
```
def process_data(data): return np.mean(data, axis=1)
```

[166]:
```
# Function to generate sensor data
def generate_sensor_data(num_sensors, num_readings):
    return [[random.random() for _ in range(num_readings)] for _ in␣
 ↪range(num_sensors)]

# Function to process the sensor data
def process_data(data):
    # Simulate processing by calculating averages
    return [sum(readings) / len(readings) for readings in data]
```

[167]:
```
# Parameters
num_sensors = 1000
num_readings = 100
num_iterations = 50
processing_times = []  # Initialize an empty list
```

[168]:
```
import random

# Generate random sensor data
def generate_sensor_data(num_sensors, num_readings):
    return [[random.random() for _ in range(num_readings)] for _ in␣
 ↪range(num_sensors)]

# Process the sensor data (e.g., calculate averages)
def process_data(data):
    return [sum(readings) / len(readings) for readings in data]
```

[185]:
```
# Simulate data processing and measure time
import time

for _ in range(num_iterations):
    data = generate_sensor_data(num_sensors, num_readings)
    start_time = time.time()
    processed_data = process_data(data)
```

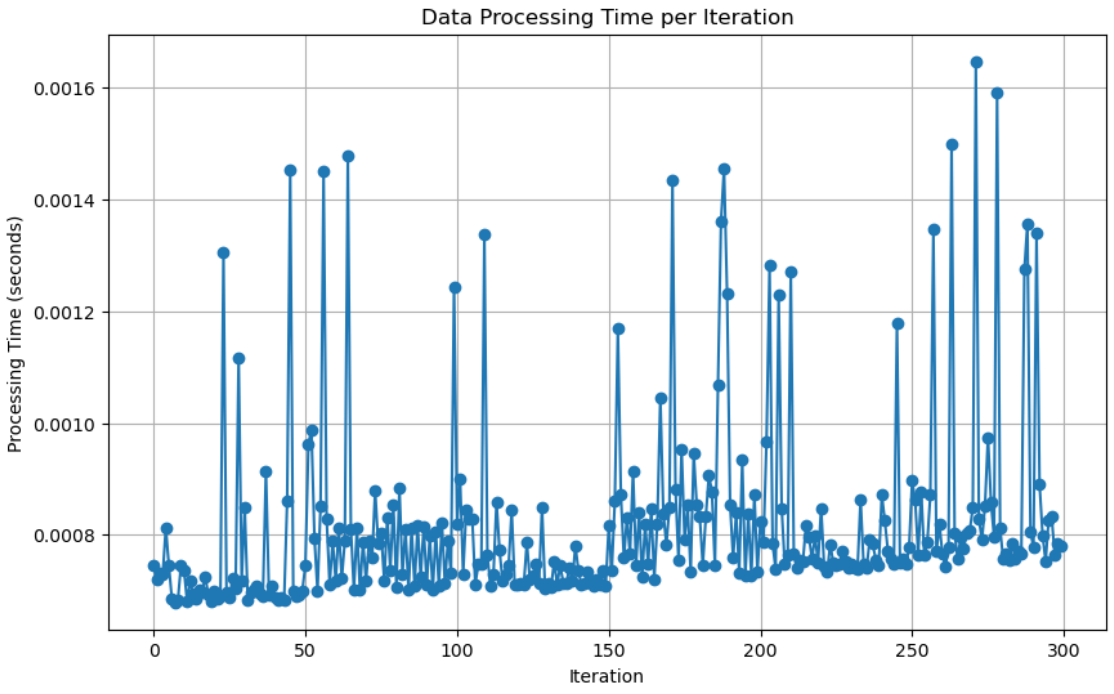
```
    end_time = time.time()
    processing_times.append(end_time - start_time)

print("Simulation completed successfully!")
```

```
Simulation completed successfully!
```

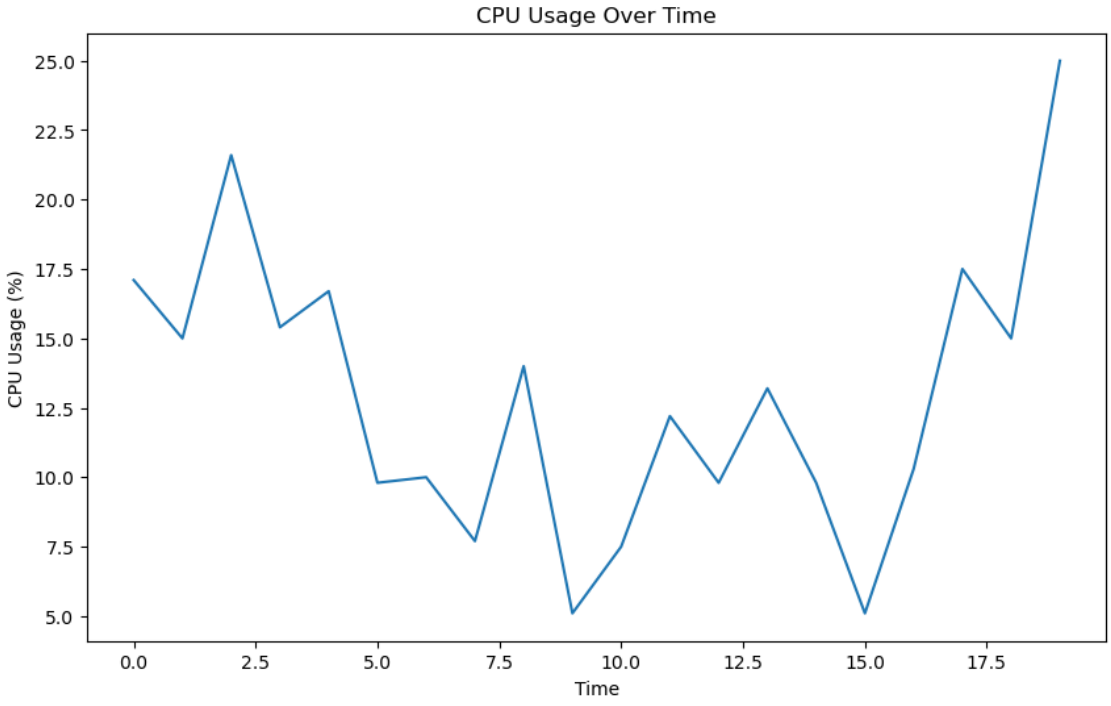
[119]:
```
import matplotlib as plt
```

[197]:
```
for _ in range(num_iterations):
    data = generate_sensor_data(num_sensors, num_readings)
    start_time = time.time()
    processed_data = process_data(data)
    end_time = time.time()
    processing_times.append(end_time - start_time)
```

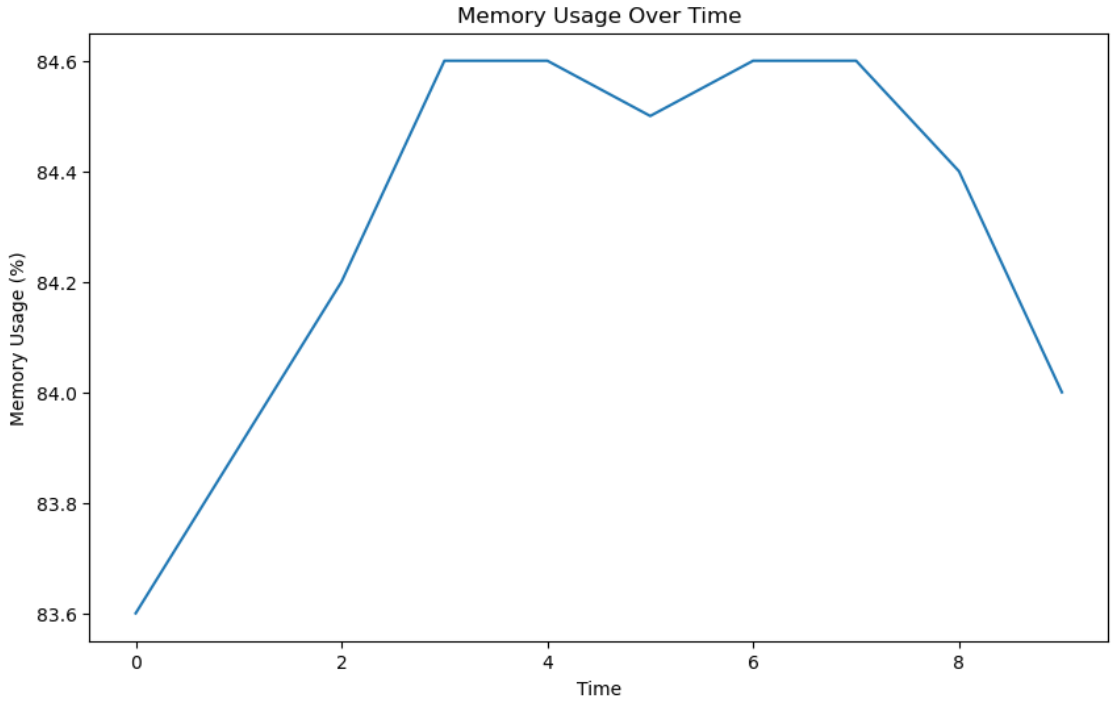
[203]:
```
import matplotlib.pyplot as plt

%matplotlib inline

plt.figure(figsize=(10, 6))
plt.plot(processing_times, marker='o')
plt.title('Data Processing Time per Iteration')
plt.xlabel('Iteration')
plt.ylabel('Processing Time (seconds)')
plt.grid(True)
plt.show()
```



[205]:
```
import psutil

%matplotlib inline

def simulate_cpu_usage(duration):
    start_time = time.time()
    while time.time() - start_time < duration:
        pass

cpu_usage = []
# Simulate CPU activity and monitor usage
for _ in range(20):
    simulate_cpu_usage(0.1)  # Keep CPU busy for 0.1 seconds
    cpu_usage.append(psutil.cpu_percent(interval=0.1))

# Plot CPU usage
plt.figure(figsize=(10, 6))
plt.plot(cpu_usage)
plt.title('CPU Usage Over Time')
plt.xlabel('Time')
plt.ylabel('CPU Usage (%)')
plt.show()
```



[207]:
```python
def simulate_memory_usage(size_mb):
    # Simulate memory usage by allocating a block of memory
    return ' ' * (size_mb * 1024 * 1024)

memory_usage = []
# Simulate increasing memory allocation
for i in range(10):
    _ = simulate_memory_usage(100 * i)  # Allocate memory in 100MB increments
    memory_usage.append(psutil.virtual_memory().percent)

# Plot memory usage
plt.figure(figsize=(10, 6))
plt.plot(memory_usage)
plt.title('Memory Usage Over Time')
plt.xlabel('Time')
plt.ylabel('Memory Usage (%)')
plt.show()
```



[ ]: