How to use Python to analyze the performance of a website

Measuring the response metrics of a web address is essential to understanding the performance and availability of a site. In this article, we will explore how to use Python to collect detailed data on the performance of a website, including latency, availability, and other key metrics. We will use several Python libraries to do this, including `requests`, `time`, `statistics`, and `matplotlib` for data visualization.

Before we begin, make sure you have Python and the following libraries installed:

```
pip install requests matplotlib
```

To measure the response latency of a website, we can use the `requests` library to send an HTTP request and measure the time it takes to get a response.

```
import requests
import time

def measure_latency(url, num_requests=10):
    latencies = []
    for _ in range(num_requests):
        start_time = time.time()
        response = requests.get(url)
        latency = time.time() - start_time
```

```
        latencies.append(latency)
        print(f'Response status:
{response.status_code}, Latency: {latency:.4f}
seconds')
    return latencies

url = "https://www.example.com"
latencies = measure_latency(url)
```

After collecting the latency data, we can analyze it to get useful information such as mean, standard deviation, and percentiles.

```
import statistics

def analyze_latencies(latencies):
    mean_latency = statistics.mean(latencies)
    median_latency = statistics.median(latencies)
    stdev_latency = statistics.stdev(latencies)
    percentile_90_latency =
statistics.quantiles(latencies, n=100)[89]

    print(f'Mean latency: {mean_latency:.4f}
seconds')
    print(f'Median latency: {median_latency:.4f}
seconds')
    print(f'Standard deviation: {stdev_latency:.4f}
seconds')
    print(f'90th percentile latency:
{percentile_90_latency:.4f} seconds')

analyze_latencies(latencies)
```

To better understand the performance of your website, it is useful to view the data collected. We will use `matplotlib` to create plots.

```
import matplotlib.pyplot as plt

def plot_latencies(latencies):
    plt.figure(figsize=(10, 6))
    plt.plot(latencies, marker='o')
    plt.title('Website Latency Over Time')
    plt.xlabel('Request Number')
    plt.ylabel('Latency (seconds)')
    plt.grid(True)
    plt.show()

plot_latencies(latencies)
```

In addition to latency, it is important to monitor the availability of your website. We can do this by logging the status of HTTP responses.

```
def monitor_availability(url, num_requests=10):
    availability = []
    for _ in range(num_requests):
        response = requests.get(url)
        availability.append(response.status_code ==
200)
        print(f'Response status:
{response.status_code}')
    uptime = sum(availability) / num_requests * 100
    print(f'Uptime: {uptime:.2f}%')
```

```
monitor_availability(url)
```

Conclusion

Measuring the response metrics of a web address is a crucial task to ensure that the website provides a good user experience. Using Python and the `requests`, `time`, `statistics` and `matplotlib` libraries, we can effectively collect, analyze and visualize website performance data. This allows us to identify and resolve any performance issues, improving the availability and speed of the site.

By following the steps in this article, you will be able to implement a performance monitoring system for any website, thus obtaining a detailed view of its response metrics.