Die wichtigsten Module der Python Standard Library

Einführung

Die Python Standard Library ist eine umfangreiche Sammlung von Modulen, die direkt mit Python ausgeliefert wird. Diese Module bieten Funktionen für eine Vielzahl von Aufgaben in Bereichen wie Dateioperationen, Systemverwaltung, Netzwerkkommunikation, Textverarbeitung und Datenanalyse.

Wichtige Module und ihre Anwendungsbereiche

`os` und `sys`

- `os`: Bietet eine Schnittstelle zum Betriebssystem, einschließlich Funktionen zum Interagieren mit dem Dateisystem, Prozessmanagement und Umgebungsvariablen.
- `sys`: Zugriff auf einige Variablen und Funktionen, die eng mit dem Python-Interpreter zusammenarbeiten.

Beispiel

```
import os
import sys

print(os.getcwd())  # Aktuelles Arbeitsverzeichnis
print(sys.argv)  # Kommandozeilenargumente
```

`datetime`

- `datetime`: Funktionen für die Handhabung von Datums- und Uhrzeitangaben, einschließlich der Berechnung von Differenzen zwischen Zeiten und der Formatierung von Datums-/Uhrzeitangaben.

Beispiel

```
from datetime import datetime

jetzt = datetime.now()
print(jetzt.strftime("%Y-%m-%d %H:%M:%S"))
```

`json`

- `json`: Lesen und Schreiben von JSON-Daten. Sehr nützlich für die Arbeit mit Web APIs oder Konfigurationsdateien.

Beispiel

```
import json

# Ein Python-Objekt in einen JSON-String konvertieren
daten = {"name": "Anna", "alter": 30}
daten_string = json.dumps(daten)
print(daten_string)
```

`re`

- `re`: Bietet reguläre Ausdrücke für fortgeschrittene String-Verarbeitung und -Analyse.

Beispiel

```
import re

# Überprüfen, ob der String eine E-Mail-Adresse enthält
if re.match(r"[^@]+@[^@]+\.[^@]+", "test@example.com"):
    print("Gültige E-Mail-Adresse")
```

Standardmodule `math`, `random`, `statistics` in Python

Einführung

Python bietet eine Vielzahl von Standardmodulen, die leistungsfähige Funktionen für mathematische Berechnungen, Zufallszahlen und statistische Analysen bereitstellen. Die Module `math`, `random` und `statistics` sind dabei besonders hervorzuheben, da sie grundlegende Werkzeuge für wissenschaftliche und technische Anwendungen bieten.

Modul `math`

Das Modul `math` enthält Funktionen für mathematische Operationen über den Bereich der Fließkommamathematik hinaus.

Wichtige Funktionen

- `math.sqrt(x)`: Berechnet die Quadratwurzel von x.
- `math.sin(x)`, `math.cos(x)`, `math.tan(x)`: Trigonometrische Funktionen.
- `math.log(x[, base])`: Berechnet den Logarithmus von x zur angegebenen Basis.
- `math.factorial(x)`: Berechnet die Fakultät von x.

Beispiel

```
import math
```

```
print(math.sqrt(16))  # Ausgabe: 4.0
print(math.factorial(5))  # Ausgabe: 120
```

Modul `random`

Das Modul `random` wird für die Generierung von Zufallszahlen verwendet und bietet Funktionen zur Auswahl von Elementen aus einer Sequenz zufällig.

Wichtige Funktionen

- `random.random()`: Gibt eine Zufallszahl zwischen 0 und 1 zurück.
- `random.randint(a, b)`: Gibt eine Zufallszahl zwischen den angegebenen ganzen Zahlen zurück.
- `random.choice(sequence)`: Wählt ein zufälliges Element aus einer Sequenz aus.

Beispiel

```
import random

print(random.random())
print(random.randint(1, 10))
```

Modul `statistics`

Das Modul `statistics` enthält Funktionen zur statistischen Analyse von numerischen Daten.

Wichtige Funktionen

- `statistics.mean(data)`: Berechnet den Mittelwert der Daten.
- `statistics.median(data)`: Berechnet den Median der Daten.
- `statistics.variance(data)`: Berechnet die Varianz der Daten.

Beispiel

```
import statistics

daten = [1, 2, 3, 4, 5]
print(statistics.mean(daten))  # Ausgabe: 3
```

Nutzung von `urllib`, `http`, `socket`, `subprocess` und `argparse` in Python

Einführung

Diese Module erweitern Pythons Fähigkeiten in den Bereichen Netzwerkkommunikation, Systeminteraktion und Befehlszeilenverarbeitung, was sie zu wertvollen Werkzeugen für eine breite Palette von Anwendungen macht.

Modul `urllib`

- `urllib`: Eine Sammlung von Modulen für das Arbeiten mit URLs, insbesondere für das Abrufen von Daten aus dem Internet.

Beispiel

```
from urllib.request import urlopen

url = "http://example.com"
response = urlopen(url)
content = response.read()
print(content)
```

Modul `http`

- `http`: Bietet Klassen und Funktionen für den Umgang mit HTTP, der grundlegenden Protokollschicht für das Web.

Beispiel

```
from http.server import HTTPServer, BaseHTTPRequestHandler

class SimpleHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.end_headers()
        self.wfile.write(b"Hello, world!")

server = HTTPServer(('localhost', 8000), SimpleHandler)
server.serve_forever()
```

Modul `socket`

- `socket`: Ermöglicht die niedrigere Netzwerkkommunikation über Sockets, sowohl im TCP als auch im UDP Protokoll.

Beispiel

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.connect(("example.com", 80))
s.sendall(b"GET / HTTP/1.1\r\nHost: example.com\r\n\r\n")
response = s.recv(1024)
print(response)
```

Modul `subprocess`

- `subprocess`: Zum Starten neuer Anwendungen oder Prozesse, Verbinden mit ihren Input-/Output-/Error-Pipes und Abrufen ihrer Ergebnisse.

Beispiel

```
import subprocess

result = subprocess.run(['echo', 'Hallo Welt'], capture_output=True, text=True)
print(result.stdout)
```

Modul `argparse`

- `argparse`: Zum Parsen von Befehlszeilenargumenten. Es ermöglicht die Erstellung von benutzerfreundlichen Befehlszeilenschnittstellen.

Beispiel

```
import argparse

parser = argparse.ArgumentParser(description='Ein Testprogramm.')
parser.add_argument('-n', '--name', help='Dein Name', required=True)

args = parser.parse_args()
print(f"Hallo, {args.name}!")
```