Οδηγός Χρήσης της Βιβλιοθήκης Python Requests

Python Requests Library Guide

Εισαγωγή / Introduction

Η βιβλιοθήκη Requests είναι η πιο δημοφιλής βιβλιοθήκη Python για την αποστολή HTTP αιτημάτων. Είναι απλή στη χρήση και πολύ ισχυρή.

The Requests library is the most popular Python library for sending HTTP requests. It's simple to use and very powerful.

```python
import requests
```

Βασικές Μέθοδοι HTTP / Basic HTTP Methods

GET Request

```python
# Λήψη δεδομένων από ένα API / Fetching data from an API
response = requests.get('https://api.example.com/data')
print(f'Κατάσταση / Status: {response.status_code}')
print(f'Περιεχόμενο / Content: {response.text}')
```

POST Request

```python
# Αποστολή δεδομένων σε ένα API / Sending data to an API
data = {
    'username': 'student',
    'password': 'password123'
}
response = requests.post('https://api.example.com/login', json=data)
```

PUT Request

```python
# Ενημέρωση υπάρχοντων δεδομένων / Updating existing data
data = {'name': 'New Name'}
response = requests.put('https://api.example.com/user/1', json=data)
```

DELETE Request

```python
# Διαγραφή δεδομένων / Deleting data
response = requests.delete('https://api.example.com/user/1')
```

Κωδικοί Κατάστασης HTTP / HTTP Status Codes

- 2xx: Επιτυχία / Success
  - 200: OK (Επιτυχής αίτημα / Successful request)
  - 201: Created (Επιτυχής δημιουργία / Successful creation)
  - 204: No Content (Επιτυχία χωρίς περιεχόμενο / Success with no content)

- 3xx: Ανακατεύθυνση / Redirection
  - 301: Moved Permanently (Μόνιμη ανακατεύθυνση / Permanent redirect)
  - 302: Found (Προσωρινή ανακατεύθυνση / Temporary redirect)
- 4xx: Σφάλματα Πελάτη / Client Errors
  - 400: Bad Request (Εσφαλμένο αίτημα / Invalid request)
  - 401: Unauthorized (Μη εξουσιοδοτημένο / Not authenticated)
  - 403: Forbidden (Απαγορευμένο / Access denied)
  - 404: Not Found (Δεν βρέθηκε / Resource not found)
- 5xx: Σφάλματα Διακομιστή / Server Errors
  - 500: Internal Server Error (Εσωτερικό σφάλμα διακομιστή / Server error)
  - 503: Service Unavailable (Μη διαθέσιμη υπηρεσία / Service down)

Χειρισμός JSON Απαντήσεων / Handling JSON Responses

```
# Λήψη και επεξεργασία JSON δεδομένων / Getting and processing JSON data
response = requests.get('https://api.example.com/users')
if response.status_code == 200:
    data = response.json()  # Αυτόματη μετατροπή JSON σε Python dictionary / Automatic JSON to Python dictionary conversion
    for user in data['users']:
        print(f"Όνομα / Name: {user['name']}")
        print(f"Email: {user['email']}")
```

Κατέβασμα Αρχείων / Downloading Files

```
# Κατέβασμα αρχείου με διαχείριση μνήμης / Memory-efficient file download
def download_file(url, filename):
    # Αποστολή αιτήματος με streaming / Send request with streaming
    response = requests.get(url, stream=True)

    # Έλεγχος επιτυχίας / Check if successful
    if response.status_code == 200:
        # Άνοιγμα αρχείου σε λειτουργία εγγραφής / Open file in write mode
        with open(filename, 'wb') as file:
            # Εγγραφή κατά τμήματα / Write in chunks
            for chunk in response.iter_content(chunk_size=8192):
                if chunk:
                    file.write(chunk)
        return True
    return False

# Παράδειγμα χρήσης / Usage example
success = download_file('https://example.com/file.pdf',
'downloaded_file.pdf')
print('Επιτυχία / Success' if success else 'Αποτυχία / Failure')
```

Παράμετροι URL και Headers / URL Parameters and Headers

```
# Προσθήκη παραμέτρων URL / Adding URL parameters
params = {
    'page': 1,
    'limit': 10,
```

```
    'sort': 'name'
}
response = requests.get('https://api.example.com/users', params=params)

# Προσθήκη headers / Adding headers
headers = {
    'Authorization': 'Bearer your-token-here',
    'Content-Type': 'application/json'
}
response = requests.get('https://api.example.com/protected',
headers=headers)
```

Χειρισμός Σφαλμάτων / Error Handling

```
try:
    response = requests.get('https://api.example.com/data')
    response.raise_for_status()  # Εγείρει εξαίρεση για κωδικούς σφάλματος /
Raises exception for error codes
    data = response.json()
except requests.exceptions.RequestException as e:
    print(f'Σφάλμα στο αίτημα / Request error: {e}')
except ValueError as e:
    print(f'Σφάλμα στην ανάλυση JSON / JSON parsing error: {e}')
```

Συνεδρίες / Sessions

```
# Χρήση συνεδριών για πολλαπλά αιτήματα / Using sessions for multiple
requests
with requests.Session() as session:
    # Το session διατηρεί cookies και headers / Session maintains cookies
and headers
    session.headers.update({'Authorization': 'Bearer token'})

    # Πολλαπλά αιτήματα με την ίδια συνεδρία / Multiple requests with same
session
    response1 = session.get('https://api.example.com/data1')
    response2 = session.get('https://api.example.com/data2')
```

Καλές Πρακτικές / Best Practices

1. Πάντα ελέγχετε τους κωδικούς κατάστασης / Always check status codes
2. Χρησιμοποιείτε το `raise_for_status()` για αυτόματο έλεγχο σφαλμάτων / Use `raise_for_status()` for automatic error checking
3. Χρησιμοποιείτε συνεδρίες για πολλαπλά αιτήματα / Use sessions for multiple requests
4. Διαχειριστείτε τους πόρους σωστά με το `with` statement / Manage resources properly with `with` statement
5. Χρησιμοποιείτε streaming για μεγάλα αρχεία / Use streaming for large files

In [ ]: