

# API Best Practices

Instructor

Prashant Sahu  
Manager Data Science, Analytics Vidhya



# Best Practices for Working with APIs



## Read the Documentation

Understand endpoints, parameters, and authentication



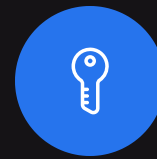
## Handle Errors Gracefully

Implement robust error handling



## Respect Rate Limits

Avoid exceeding limits to prevent security storage



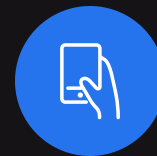
## Secure Your API keys

Use environment variables or secure storage



## Validate Inputs and Outputs

Ensure data integrity and prevent security risks



## Use Logging

Keep logs of API requests and responses for debugging

# Error Handling in API Requests

```
# Handling Exceptions:
try:
    response = requests.get(url)
    response.raise_for_status()
except requests.exceptions.HTTPError as err:
    print(f'HTTP error occurred: {err}')
except Exception as err:
    print(f'An error occurred: {err}')
```

- 400: Bad Request
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found
- 500: Internal Server error

Best Practices:

Always check `status_code`

Use `raise_for_status()` to raise exceptions

# API Rate Limiting



## Definition

Limiting the number of API request in a given time frame



## Purpose

Prevents abuse and overloading the server  
Ensures fair usage among clients



## How to Handle

**Check Headers:** APIs often include rate limit info in response headers

**Implement Delays:** Use `time.sleep()` to wait before making new requests

**Exponential Backoff:** Gradually increase wait time after each retry



## Best Practices

Respect the API's rate limit  
Monitor your application's request rate

# Authentication in APIs



## Why Authentication is Needed:

Protects sensitive data

Controls access to resources



## Common Authentication Methods

API Keys

OAuth 2.0

Bearer Tokens



## Including Authentication in Requests

**Headers:** Use the [Authorization](#) header

**Parameters:** Include in query parameters (less secure)

# APIs Keys

```
## Include the key in your request ##

# Create the Headers and send the get request:
headers = {'Authorization': 'Bearer YOUR_API_KEY'}
response = requests.get(url, headers=headers)

# OR Use the Query Parameters:
params = {'api_key': 'YOUR_API_KEY'}
response = requests.get(url, params=params)
```

- Definition: Unique identifiers used to authenticate requests
- How to use:
  - Obtain an API key by registering with the API provider
  - Include the key in your request
- Security Tips
  - Never expose your API keys in public code repositories
  - Store keys securely using environment variables

# OAuth Authentication



## OAuth 2.0

An open standard for access delegation  
Allow users to grant access to their resources  
Used by platforms like Google, Facebook,  
Twitter



## Flow

Authorisation Request: Redirect user to  
authorisation server  
User authenticates and authorise access  
Server returns an authorisation code  
Exchange code for access token  
Use token in API requests



## Implementation

More complex but offers enhanced security

# Summary



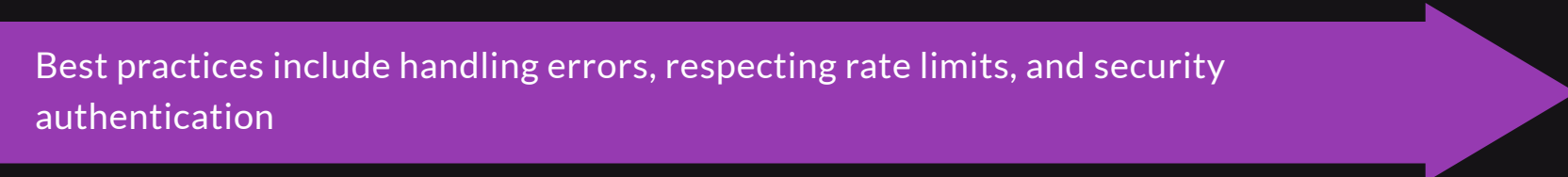
APIs enable communication between software systems



Understanding REST and JSON is crucial for working with web APIs



Python's request library simplifies API interactions



Best practices include handling errors, respecting rate limits, and security authentication