Star Wars API Explorer

A Python-based command-line tool that fetches and displays data from the Star Wars API (SWAPI). Explore characters, planets, starships, vehicles, species, and films from the Star Wars universe through an interactive interface.

Features

- **Interactive Data Retrieval**: Fetch Star Wars data by category
- **Multiple Categories**: Access people, planets, films, species, vehicles, and starships
- **Error Handling**: Robust HTTP error handling and user feedback
- **Clean Display**: Formatted output of entity names
- **RESTful API Integration**: Real-world API consumption example
- **User-Friendly**: Simple command-line interface

Installation

Prerequisites

- Python 3.7 or higher
- pip package manager
- Internet connection (to access the API)

Setup

1. Clone the repository:

bash

```
git clone https://github.com/YOUR-USERNAME/star-wars-api-explorer.git
cd star-wars-api-explorer
```

2. Install required dependencies:

bash

```
pip install -r requirements.txt
```

Usage

Running the Jupyter Notebook

bash

```
jupyter notebook "Star War API Project.ipynb"
```

Running as Python Script

bash

```
python star_wars_explorer.py
```

Interactive Usage

When you run the program, you'll be prompted to enter a category:

```
Enter an option (e.g 'people' or 'planets'): people
```

Available Categories:

- `people` - Characters from Star Wars
- `planets` - Planets in the Star Wars universe
- `films` - Star Wars movies
- `species` - Different species
- `vehicles` - Vehicles used in Star Wars
- `starships` - Starships and spacecraft

Project Structure

```
star-wars-api-explorer/
│
├── Star War API Project.ipynb    # Main Jupyter Notebook
├── star_wars_explorer.py         # Standalone Python script
├── README.md                     # This file
├── requirements.txt              # Python dependencies
└── LICENSE                       # License file
```

Code Explanation

Main Function: `fetch_data(option)`

This function handles all API communication and data retrieval.

Parameters:

- `option` (str): The category of data to fetch (e.g., 'people', 'planets')

Returns:

- `list`: List of dictionaries containing entity data

- `None`: If an error occurs

Process Flow:

1. **URL Construction**

python

```
url = f"https://swapi.mimo.dev/api/{option}/"
```

Dynamically builds the API endpoint based on user input

2. **HTTP Request**

python

```
response = requests.get(url)
response.raise_for_status()
```

- Sends GET request to the API
- `raise_for_status()` raises an exception for 4xx/5xx status codes

3. **JSON Parsing**

python

```
data = response.json()
```

Converts JSON response to Python data structures

4. **Error Handling**

python

```
except requests.HTTPError as e:
    print(f"Error message: {e}")
    return None
```

Catches HTTP errors and provides user feedback

User Input & Output

python

```
option = input("Enter an option (e.g 'people' or 'animal')").strip().lower()
```

- `strip()`: Removes leading/trailing whitespace
- `lower()`: Converts to lowercase for case-insensitive matching

python

```python
if data:
    for item in data:
        print(item["name"])
```

Iterates through results and displays entity names

Examples

Example 1: Fetching People (Characters)

Enter an option (e.g 'people' or 'planets'): people

The number of entities are 82
Luke Skywalker
C-3PO
R2-D2
Darth Vader
Leia Organa
Owen Lars
Beru Whitesun lars
R5-D4
Biggs Darklighter
Obi-Wan Kenobi
...

Example 2: Fetching Planets

Enter an option (e.g 'people' or 'planets'): planets

The number of entities are 60
Tatooine
Alderaan
Yavin IV
Hoth
Dagobah
Bespin
Endor
Naboo

Coruscant

Kamino

...

Example 3: Fetching Films

Enter an option (e.g 'people' or 'planets'): films

The number of entities are 6

A New Hope

The Empire Strikes Back

Return of the Jedi

The Phantom Menace

Attack of the Clones

Revenge of the Sith

Example 4: Error Handling

Enter an option (e.g 'people' or 'planets'): invalid

Error message: 404 Client Error: Not Found for url: https://swapi.mimo.dev/api/invalid/

Unable to download data

API Categories

Available Endpoints

Category	Description	Example Entities
people	Characters	Luke Skywalker, Darth Vader, Leia Organa
films	Movies	A New Hope, Empire Strikes Back
starships	Spacecraft	Millennium Falcon, X-wing, TIE Fighter
vehicles	Ground/Air vehicles	AT-AT, Speeder bike, Sandcrawler
species	Species/Races	Human, Wookiee, Droid
planets	Planets	Tatooine, Alderaan, Coruscant

Data Structure

Each entity returns different fields. For example, **people** returns:

json

{

```
 "name": "Luke Skywalker",
 "height": "172",
 "mass": "77",
 "hair_color": "blond",
 "skin_color": "fair",
 "eye_color": "blue",
 "birth_year": "19BBY",
 "gender": "male",
 "homeworld": "https://swapi.mimo.dev/api/planets/1/",
 "films": [...],
 "species": [...],
 "vehicles": [...],
 "starships": [...],
 "created": "2014-12-09T13:50:51.644000Z",
 "edited": "2014-12-20T21:17:56.891000Z",
 "url": "https://swapi.mimo.dev/api/people/1/"
}
```


API Documentation

Base URL

https://swapi.mimo.dev/api/

Endpoint Format

https://swapi.mimo.dev/api/{category}/

Response Format

The API returns a list of objects, each containing:

- `name` or `title`: Primary identifier
- Additional properties specific to the category
- Related resource URLs

HTTP Methods

- **GET**: Retrieve data (only method used in this project)

Status Codes

- `200 OK`: Successful request
- `404 Not Found`: Invalid category or resource
- `500 Internal Server Error`: Server-side issues

Key Programming Concepts

This project demonstrates:

1. **API Integration**: Making HTTP requests to external APIs
2. **JSON Parsing**: Converting JSON to Python data structures
3. **Error Handling**: Using try-except blocks for robust code
4. **HTTP Status Codes**: Understanding and handling different response codes
5. **User Input**: Processing and validating user input
6. **String Manipulation**: Using `.strip()` and `.lower()` for data cleaning
7. **List Iteration**: Looping through API responses
8. **F-Strings**: Modern Python string formatting

Learning Outcomes

Perfect for learning:

- RESTful API consumption
- HTTP request/response cycle
- JSON data handling
- Error handling in Python
- User input validation
- Working with external services
- Data parsing and display

Contributing

Contributions are welcome! Here's how you can help:

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit your changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

Contribution Ideas:

- Add pagination support
- Implement search functionality
- Create detailed entity views

- Add unit tests
- Improve error messages
- Add data export features

Resources

- **SWAPI Documentation**: https://swapi.dev/documentation
- **Requests Library**: https://requests.readthedocs.io/
- **Star Wars Wiki**: https://starwars.fandom.com/

License

Acknowledgments

- **SWAPI**: The Star Wars API for providing free access to Star Wars data
- **Requests Library**: For simplifying HTTP requests in Python
- **Star Wars**: Created by George Lucas

Use Cases

This project is useful for:

- **Learning API Integration**: Understanding how to work with external APIs
- **Star Wars Fans**: Exploring the Star Wars universe programmatically
- **Data Analysis**: Analyzing patterns in Star Wars data
- **Educational Projects**: Teaching HTTP requests and JSON parsing
- **Portfolio Projects**: Demonstrating API consumption skills

Contact

For questions, suggestions, or feedback, please open an issue on GitHub.