

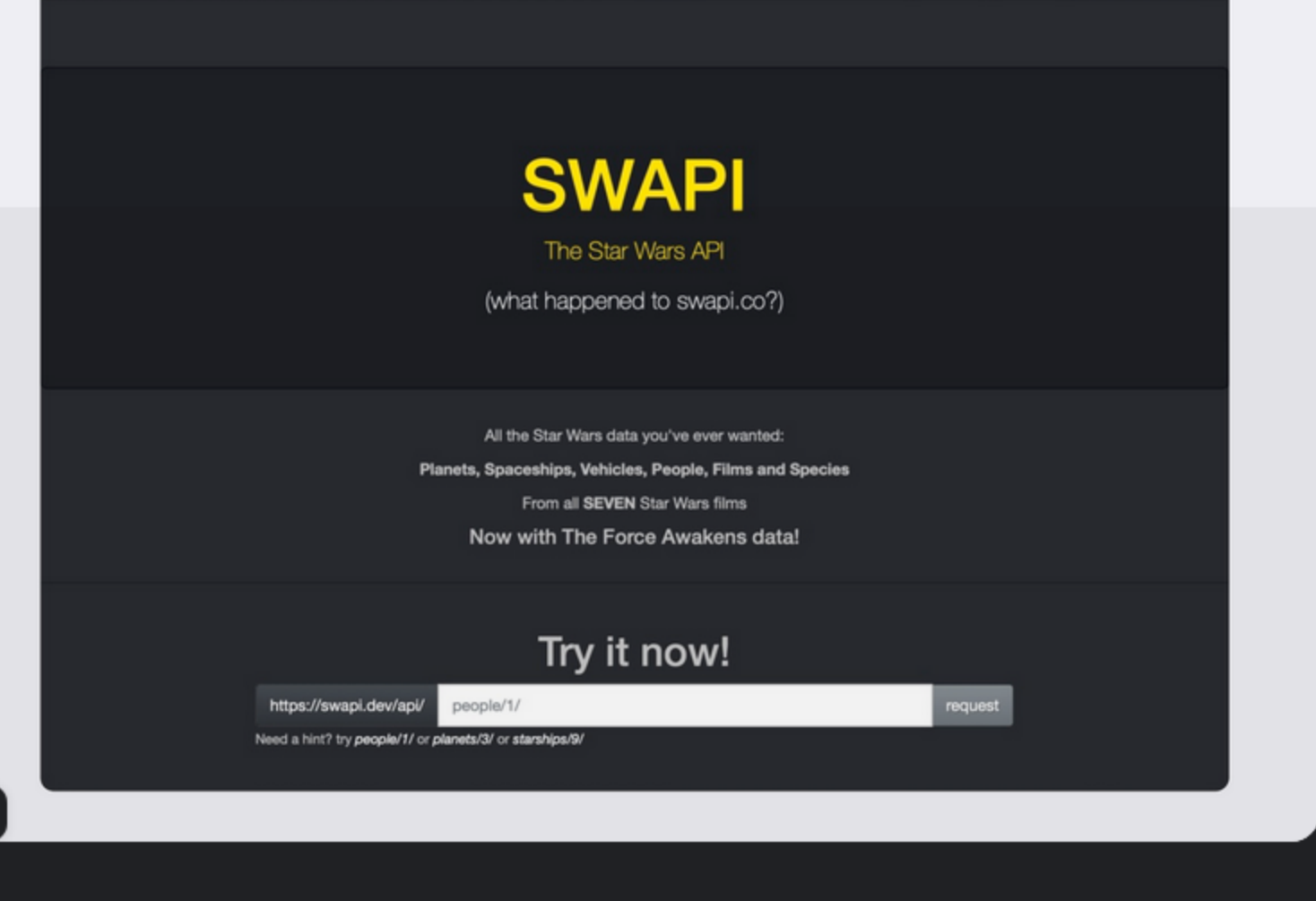
## Исследование запросов

В одной далёкой-далёкой галактике тоже есть API, созданный по принципам REST: SWAPI (Star Wars API), он хранит информацию о персонажах, космических кораблях и планетах, на которых происходит события эпопеи Star Wars. Поработаем с этим API, отправим к нему несколько запросов и разберём его ответы.

Как и всякая техника, сервер SWAPI иногда ломается — и тогда на главной странице сервиса вместо заставки вы увидите сообщение об ошибке. В этом случае обращайтесь к другому ресурсу: <https://swapi.py4e.com/>.

Он полностью аналогичен основному.

Документация сервиса — на странице <https://swapi.py4e.com/documentation>.



### JSON vs типы данных Python

Делать запросы к API будем из кода Python — этот опыт пригодится при выполнении финального задания.

Создайте директорию для экспериментального проекта, создайте и активируйте виртуальное окружение, установите модуль `requests`:

```
pip install requests
```

С этим модулем вы уже работали, но полистать его документацию никогда не вредно.

Создайте файл `swapi.py`, импортируйте в код модуль `requests`.

Отправьте GET-запрос к адресу `https://swapi.dev/api/starships/9/`, чтобы получить информацию о космическом корабле с ID = 9

### Метод `get()`: отправляем запросы из кода

Метод `get()` модуля `requests` отправляет GET-запрос к указанному адресу и возвращает объект класса `Response`. Этот объект включает в себя полную информацию об ответе сервера; получить доступ к этим данным можно через свойства и методы класса `Response`. Например, содержимое ответа можно получить из свойства `response.text`

Скопируйте этот код в файл и выполните его:

```
import requests

response = requests.get('https://swapi.dev/api/starships/9/')
response = response.text

# Напечатаем в терминале содержимое ответа сервера...
print(response)

# ...и его тип
print(type(response))
```

SWAPI возвращает ответы в формате JSON. В терминале будет напечатано:

```
{
  "name": "Death Star",
  "model": "DS-1 Orbital Battle Station",
  "manufacturer": "Imperial Depa
<class 'str'>
```

Это структурированные данные, похожие на словарь, но Python воспринимает ответ API как обычную строку: `<class 'str'>`. Python не умеет напрямую работать с JSON.

Преобразовать JSON-строку в тип данных, понятный Python, можно методом `json()` класса `Response`:

```
import requests

response = requests.get('https://swapi.dev/api/starships/9/')

# Приведём ответ сервера к типам данных Python...
response = response.json()
# ...и напечатаем его.
print(response)

# Напечатаем и тип данных объекта, полученного в результате преобразования:
print(type(response))
```

Из обычной строки получили словарь, который содержит строки, списки и другие типы данных Python.

```
{
  "name": "Death Star",
  "model": "DS-1 Orbital Battle Station",
  "manufacturer": "Imperial
<class 'dict'>
```

Теперь ответ API можно обрабатывать встроенными методами и функциями!

### Работа со словарём: другой метод `get()`

Чтобы обработать данные, хранящиеся в полученном словаре, нужно извлечь из него значения; к значениям словаря обращаются по ключу.

Начинающие разработчики обычно вызывают значения словаря с помощью синтаксиса с квадратными скобками: `имя_словаря[ключ]`.

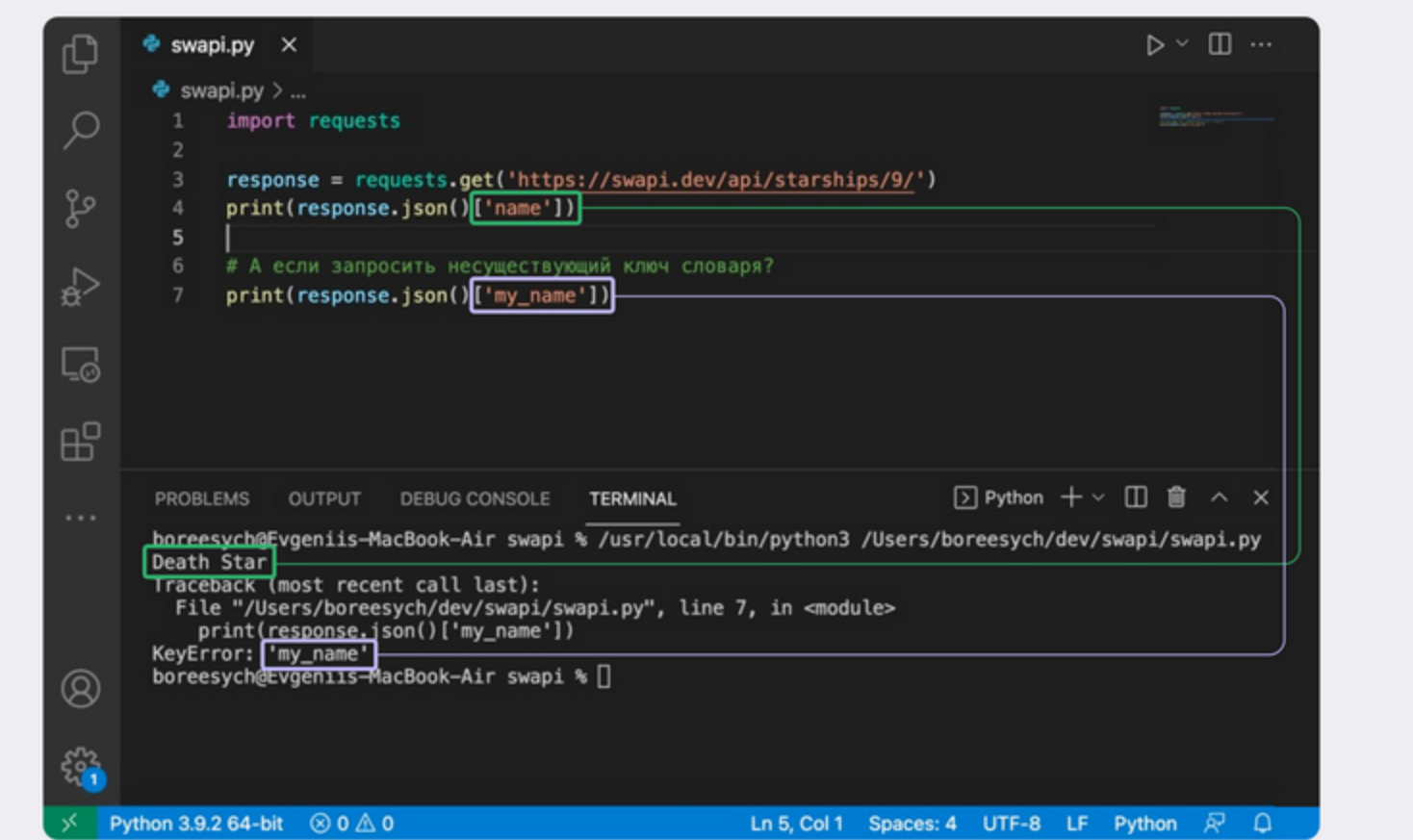
Если запрошенный ключ существует — вернётся значение, соответствующее этому ключу. А если такого ключа в словаре нет — выполнение программы будет прерушено, а в терминал будет выведена информация о причинах остановки.

Выполните код и посмотрите на результат в терминале.

```
import requests

response = requests.get('https://swapi.dev/api/starships/9/')
print(response.json()['name'])

# А если запросить несуществующий ключ словаря?
print(response.json()['my_name'])
```



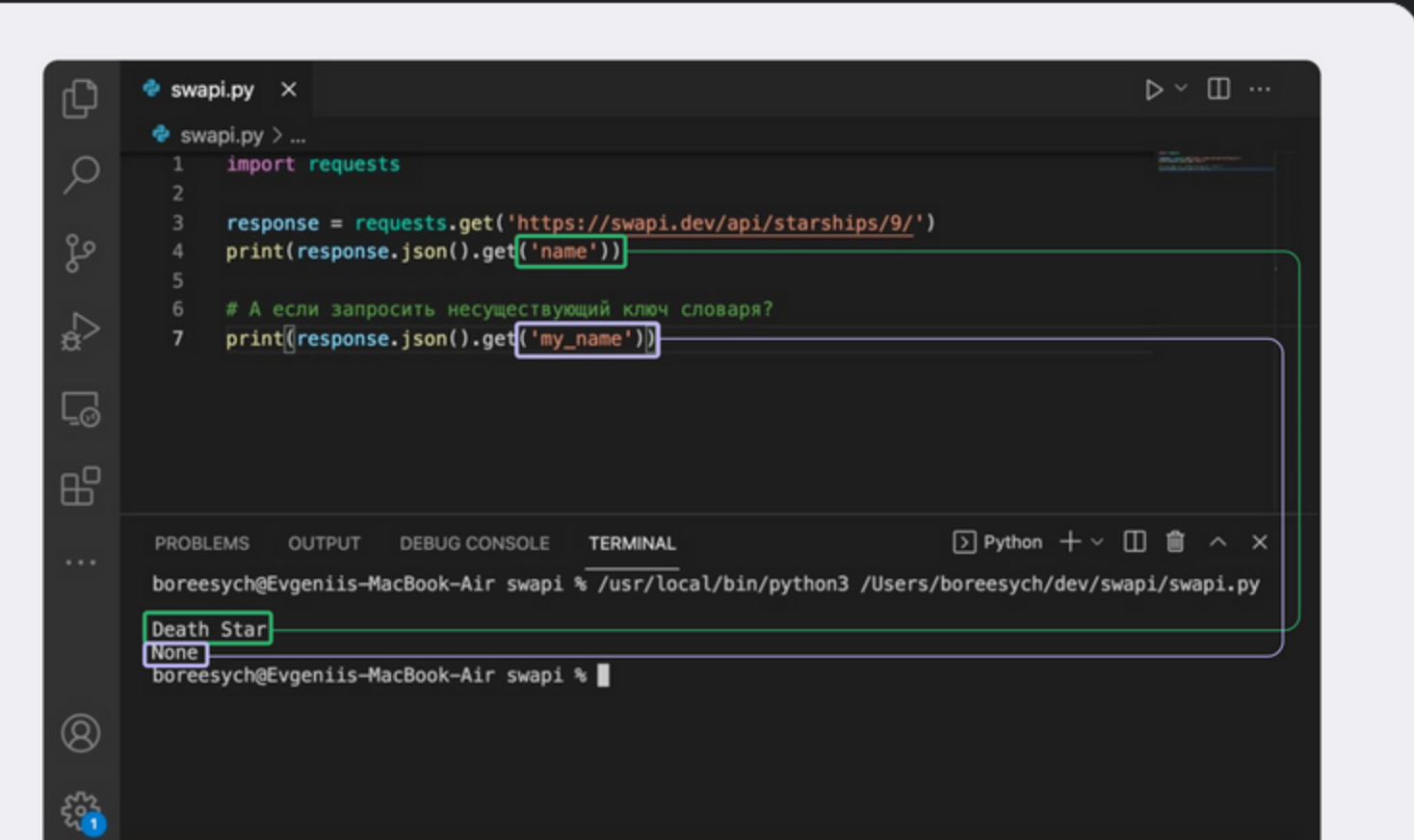
К значениям словаря можно обратиться иначе. У словарей Python есть встроенный метод `get()`, он возвращает из словаря значение по ключу. У этого метода есть два параметра:

- Ключ, значение которого нужно получить. Это обязательный параметр.
- Дефолтное значение: метод `get()` вернёт его, если запрошенного ключа нет в словаре. Это необязательный параметр: если его не указать — при отсутствии запрошенного ключа метод вернёт `None`.

```
import requests

response = requests.get('https://swapi.dev/api/starships/9/')
print(response.json().get('name'))

# А если запросить несуществующий ключ словаря?
print(response.json().get('my_name'))
```



Даже при отсутствии запрошенного ключа программа продолжит работу.

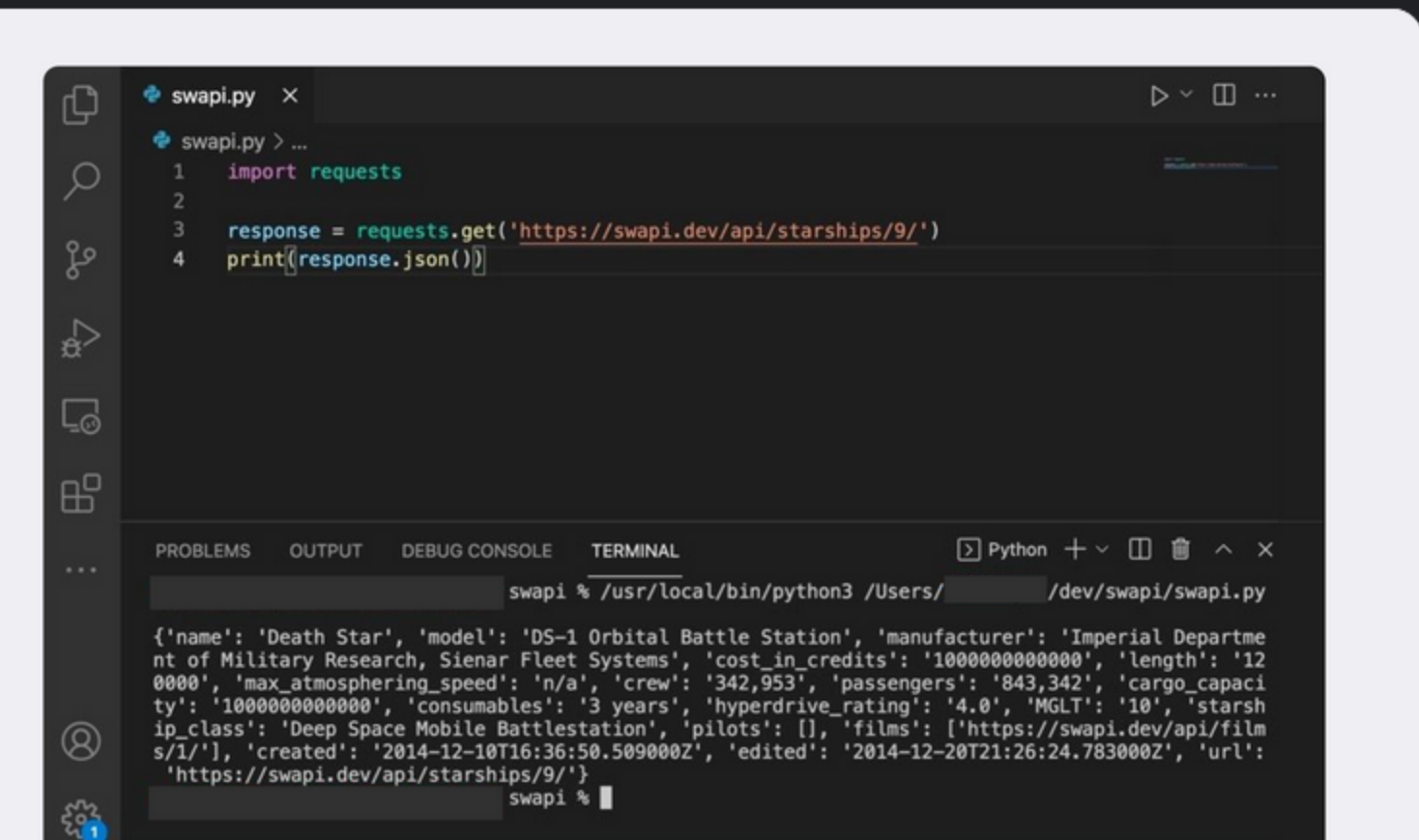
Этот метод будет часто встречаться в дальнейших примерах при работе с внешними API.

### Красивый вывод на печать

При исследовании запросов бывает необходимо вывести в терминал содержимое списков, словарей или других объектов Python. С простыми объектами проблем не возникает, но при выводе большого массива информации результат будет нечитаемым.

```
import requests

response = requests.get('https://swapi.dev/api/starships/9/')
print(response.json())
```



Разобраться в такой строке будет трудно.

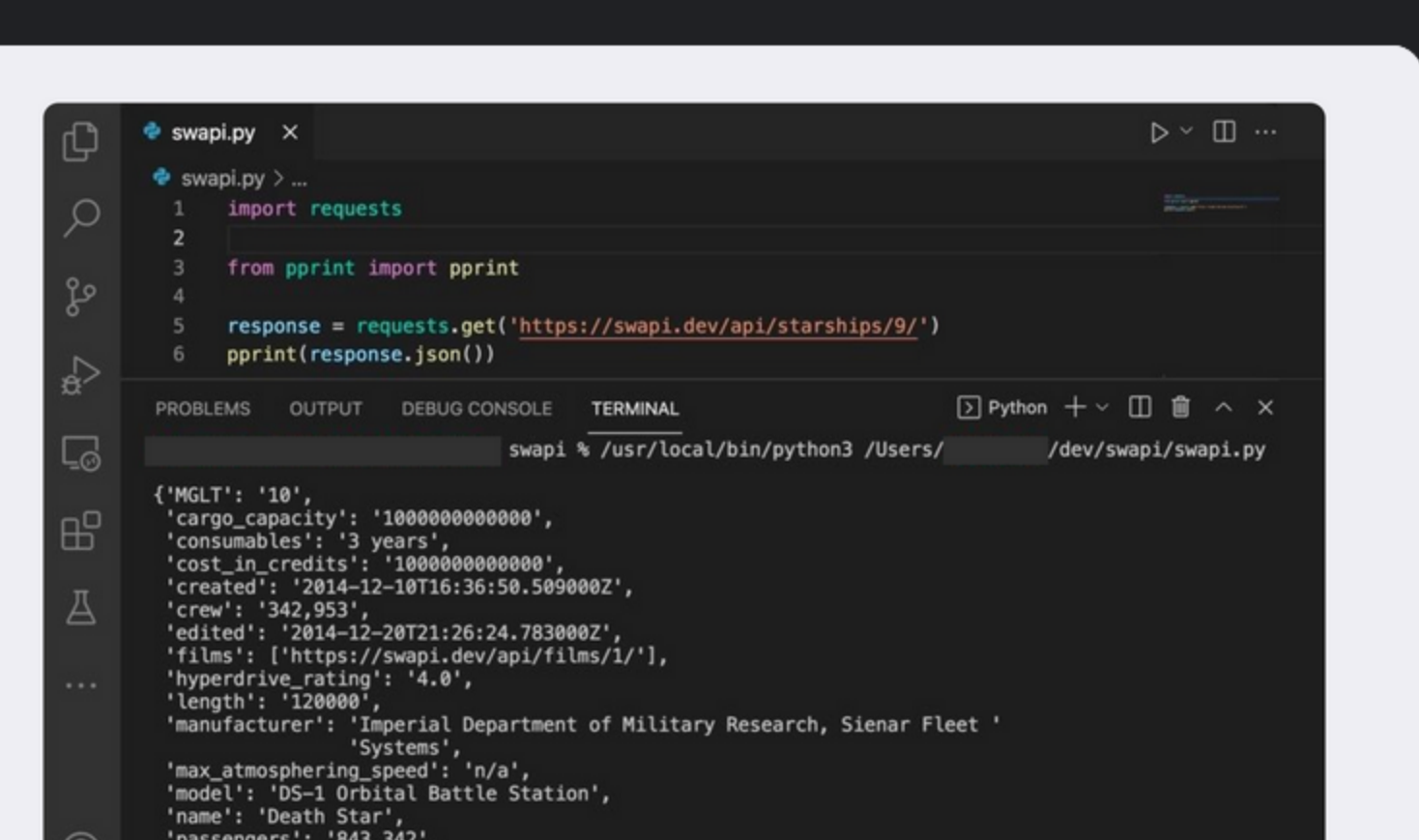
Решение есть: функция `pprint()` одноимённого модуля стандартной библиотеки Python отформатирует вывод, отбьет отступами вложенные элементы и сделает всё, чтобы читать было легко и приятно. При этом меняется только отображение объекта, а не его структура.

Действительно, «красивая печать» — `pretty print`!

```
import requests

from pprint import pprint

response = requests.get('https://swapi.dev/api/starships/9/')
pprint(response.json())
```



### Да пребудет с вами Сила

Практика работы небольшой с запросами к API ждёт сейчас вас. Задача финальная ваша: диаметр планеты родной Люка Скайуокера узнать. Предварительных испытаний несколько пройти потребуется вам прежде.

Да пребудет с вами Сила и официальная документация сервиса SWAPI! Сайт недоступен если, альтернативной ссылкой воспользуйтесь.

