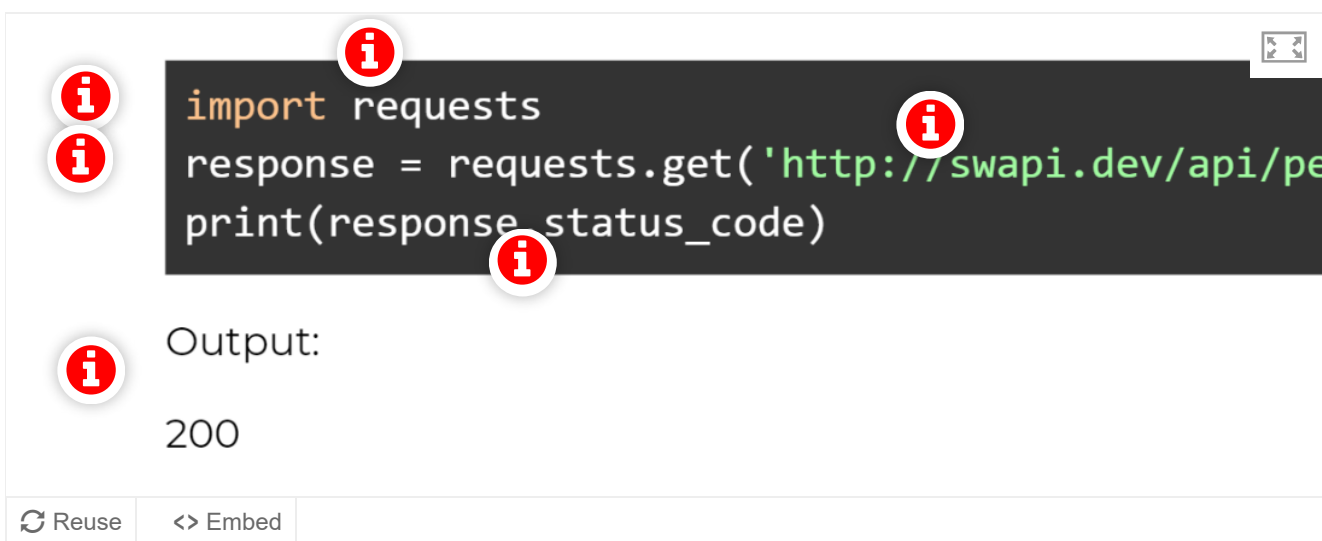


## 3.3.2 HTTP requests in Python

As your first experiment with HTTP requests in Python, you are going to call on an existing API. There are a huge number of APIs to choose from, each of them offering different data. For this exercise, you will use a completely free API, which doesn't require any registration. This API is called **Cat Facts** [\(https://alexwohlbruck.github.io/cat-facts/docs/\)](https://alexwohlbruck.github.io/cat-facts/docs/) (alexwohlbruck n.d.) and holds different facts about cats.

If you look at its documentation in the link provided, you will see that there are two endpoints: `/facts` and `/users` are two different routes with different purposes. We will be using the `/facts` endpoint in this unit.



```
import requests
response = requests.get('http://swapi.dev/api/pets')
print(response.status_code)
```

Output:

200

Reuse Embed

## Components of an HTTP request

So, what makes up an HTTP request? Generally, it contains two parts: commands and headers. The first line of the request is the command. It might look like this:

```
# GET is the HTTP verb or method.
# title.html is the resource identifier or path.
# HTTP/1.1 is the version of HTTP being used (protocol path).

GET /title.html HTTP/1.1
```

The headers make up the rest of the request. These are `key:value` pairs, usually `Name:Value`, present on each line. A `key:value` pair is a set of two linked items, namely a **key** (or **identifier**) and the **value** (data, or the **URI** where data is stored). This concept is similar to a dictionary that you explored earlier in this unit.

```
host: localhost:8000
accept-language: en
accept-encoding: gzip, compress
accept: text/*, */*;q=0.01
user-agent: Lynx/2.8.5rel.1 libwww-FM/2.14
cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
cache-control: no-cache
```

These `key:value` pairs are used to make smaller-scale decisions about the resource, such as what representation of it to show. For example, accept-encoding informs the server if your browser can accept compressed output, while user-agent shows you data on the browser version and the OS you are using. Headers give extra data in the response, such as the content type, and a time limit on cache response. They also regulate caching, transmit cookies, and provide authentication information (Norton et al. 2005).

## Status lines and headers

A **status line** is the first line in the response, indicating the status of the request. It's where you see the request's status code. The header provides extra data in the response, such as the content type of the payload, and a time limit on cache response. For example:

```
# Response status line.
Status Line : HTTP/1.1 200 OK

# Response header.
Request status code : 200 OK
Server : XXX
Date : XXX
Content-Type : XXX
Content-Length : XXX
Connection : XXX
Content-Type : text/html
```

Not all of the headers have to be there, and a request can be as simple as two lines. Sometimes the response header in the code block above, is divided into four headers:

### 1. HTTP **request** header:

- sent by the client or browser
- contains details about the request source, such as browser type, OS

- specifies the requested page
  - lists accepted outputs.
2. HTTP **response** header:
- sent by the web server
  - describes request success, or lack of
  - specifies the connection type
  - outlines file type, date, and size.
3. HTTP general or **representation** header:
- contains directives to be followed for both the client and server (such as Connection, Cache-Control, and Encoding)
  - do not describe the content itself.
4. HTTP **entity** or payload header:
- contains information on the body of the resource (such as language, length, location, and so on).

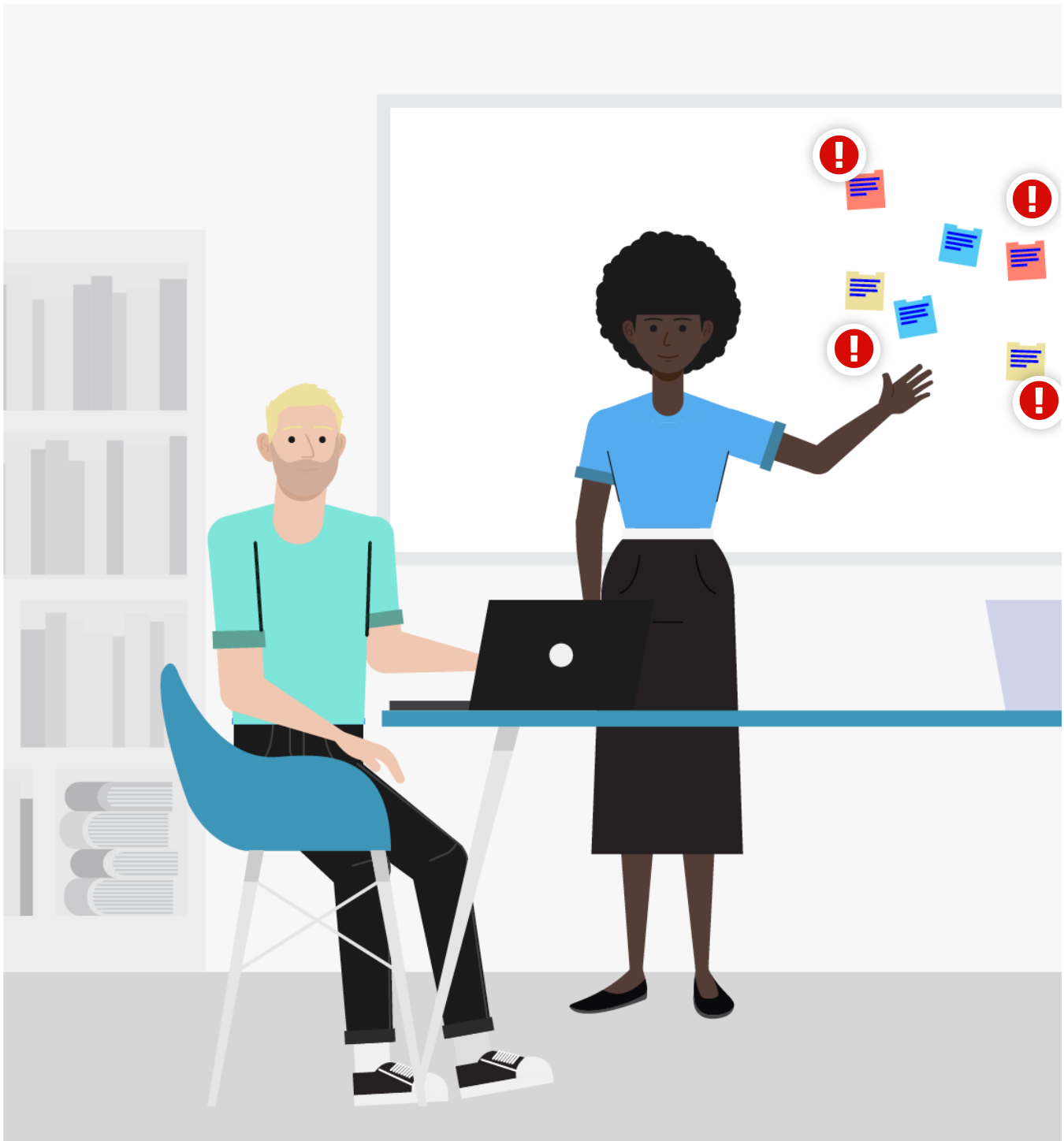
But, sometimes only three headers. For example:

```
# Response status line.  
Status Line : HTTP/1.1 200 OK  
  
# GENERAL HEADER:  
Date : XXX  
Connection : XXX  
  
# RESPONSE HEADER:  
Server : XXX  
Accept-Ranges : bytes  
  
# ENTITY HEADER:  
Content-Type : text/html  
Content-Length : XXX  
Last-Modified : XXX
```

Although there are a variety of headers, the important thing is to know how to read the headers and understand the HTTP response codes.

## Response HTTP codes



Whenever you run an HTTP request, you can check whether it was successful by referring to its HTTP code. You may be familiar with the code `404`, but there are a number of others. You don't need to memorise this, but as a general rule, you should know the categories that each of the first digits stand for. Select the icons to explore more.



### Additional reading

The Internet Engineering Task Force documentation contains helpful information on **Response status codes** → (<https://datatracker.ietf.org/doc/html/rfc7231#section-6>) (Fielding & Reschke 2014).

Refer to this resource that details what each of the HTTP codes means:

- **HTTP Status Codes**  (<https://httpstatuses.com/>) (HTTP Statuses n.d.)
- **HTTP headers**  (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>) (Mozilla 2021).

You will learn to create GET requests and perform other HTTP commands later this week.