{desafío} latam_

API

Lo que debes saber antes de comenzar esta unidad

Listas

- **Se definen como**
  - `lista = ["a", "b, "c"]

- **Tienen funciones que permiten manipular sus elementos:**
  - `append(x)`: Agrega un elemento al final de la lista.
  - `insert(i, x)`: Inserta un elemento en una posición específica.
  - `pop(i = n - 1)`: Saca el último elemento de la lista. Si se indica un índice como argumento, se saca el elemento de esa posición.
  - `remove(x)`: Elimina un elemento específico indicado como parámetro.
  - `sort()`: Ordena los elementos de la lista en forma ascendente.
  - `reverse()`: Invierte el orden de los elementos de la lista.

- **Comprensiones**:
  - Retornan una nueva lista. Se definen como `lista_nueva = [<salida> <condición> <iteración>]`

- **Iteración por índice y por elementos**
  - Se logra con `enumerate`:

```
for index, element in enumerate(["1", "gato", 40]):
      print("Indice iterado: ", index)
      print("Elemento iterado: ", element)
```

- **Operaciones funcionales:**
  - `map(f, list)`: Permite crear transformaciones sin aplicar un loop, en 1 línea de código. Recibe como argumentos una función y la lista sobre la que se va a operar. Devuelve una estructura `map` que se puede convertir a una lista.
  - `filter(f, list)`: Permite filtrar listas sin aplicar un loop, en 1 línea de código. Recibe como argumentos una función y la lista sobre la que se va a filtrar. Devuelve una estructura `filter` que se puede convertir a una lista.
  - `reduce(f, list)`: Requiere ser importado desde functools. Devuelve un único valor como resultado de una aplicación aplicada a todos los elementos de la lista. La función que se pasa como primer argumento requiere tener una variable acumuladora.

Pandas

- Librería orientada a la manipulación y limpieza de datos.
- Se importa como `import pandas as pd`.
- Sus principales estructuras son el DataFrame y las Series.
- Se puede crear un DataFrame a partir de un csv con la función pd.read_csv().

Numpy

- Librería para la computación numérica.
- Usado por muchos módulos de Python.
- Permiten crear arreglos n-dimensionales.
- Poseen funciones matemáticas orientadas a la velocidad de implementación.
- Permite trabajar con álgebra lineal.
- Se importa como `import numpy as pd`.

Glosario

Diccionario

- **Diccionario**: Contenedor o estructura de datos que permite guardar un conjunto no ordenado de pares clave-valor. Cada par se define como `"llave": "valor"`, y cada par va separado por comas. Los diccionarios se definen entre paréntesis de llaves `{}`.
- **Contenedor**: Objeto dentro de Python que implementa __contains__. Se les puede aplicar el operador **i**. En Python, los contenedores son las listas, los diccionarios, las tuplas y los sets.
- **Clave**: O "llave", o "key". Identificador para cada elemento dentro de un diccionario. Puede ser un objeto string, int, float, o bool, pero de manera general se usan `string` o `int`.
- **Valor**: Objeto hacia el cual apunta la llave de un diccionario. Pueden ser de cualquier tipo de dato de Python.

API

- **API**: **A**pplication **P**rogramming **I**nterface. Es una interfaz que permite a un programa de computador conversar con otro, ya sea dentro del mismo computador o en otro computador a través de internet.
- **REST**: **Re**presentation **S**tate **T**ransfer. Usualmente utilizado como API REST. Es un estilo de arquitectura de software para la construcción de servicios web.
- **Token**: A menudo son strings largos combinando distintos caracteres, que se utilizan como identificador único de autenticación para realizar una solicitud. Suelen expirar después de un tiempo.
- **Postman**: Herramienta gratuita para probar APIs Rest
- **Recurso**: En el contexto de API se refiere a los objetos que pueden ser consultados, por ejemplo posts, fotos, usuarios, u cualquier otro objeto disponible.
- **Request (pedido)**: Es el inicio de la comunición cliente - servidor.

Capítulo 1: Introducción a diccionarios

Objetivos

- Conocer la utilidad de un diccionario
- Crear un diccionario
- Conocer los conceptos de clave y valor
- Acceder a elementos dentro de un diccionario a través de la clave

Diccionarios

Los diccionarios son una estructura de datos compuesta por pares de `clave:valor`, donde cada clave se asocia con un elemento del diccionario. Como toda estructura de datos, permiten almacenar una gran cantidad de datos en una sola variable.

Reciben este nombre porque se leen igual que uno de la vida real. En un diccionario buscamos una palabra y esta nos lleva a la definición: la **clave** es equivalente a la palabra y el **valor** es equivalente a su definición.

Lista (array) vs Diccionario

Las listas (o arrays) y los diccionarios son estructuras bastante similares, con la diferencia de que para acceeder a los elementos de una lista, se hace a través de la posición o el índice asociado al elemento, mientras que en un diccionario, se hace por medio de la **clave** (o llave).

La otra diferencia, es que en una lista los índices se generan automáticamente para cada elemento, mientras que las claves de un diccionario no se generan automáticamente, sino que son definidas explícitamente al crear el diccionario, o al asignar un nuevo elemento a la estructura.

Por último, las claves del diccionario pueden ser un string, un número, o incluso un booleano, pero con frecuencia se utilizan los strings.

```
# En una lista usamos la posición para acceder a un elemento, y los índices se generan
en forma implícita
lista = [25, 31, "hola"]
lista[2] # "hola"

# En un diccionario usamos la clave, y se deben definir de forma explícita
diccionario = {"a": 25, "b": 31, "c": "hola"}
diccionario["c"] # "hola"
```

¿Para qué sirven los diccionarios?

Existen varios tipos de situaciones donde los diccionarios son mejores que las listas para resolver un determinado problema, principalmente cuando se requiere identificar rápidamente un elemento a través de un valor único, la **clave**.

En esta unidad, resolveremos algunos problemas utilizando ambas estrategias para poder compararlas e identificar cuándo una solución conviene por sobre la otra y por qué.

Crear un diccionario

En Python, un diccionario (vacío) se define con llaves: `{}`. Cada par de clave y valor se asocia mediante `:`, en la forma `llave: valor`.

```
notas = {"Camila": 7, "Antonio": 5, "Felipe": 6, "Antonia": 7}
```

Si al momento de definir un diccionario tenemos muchas llaves, podemos definirlo en múltiples líneas para facilitar la lectura de código.

```
notas = {
    "Camila": 7,
    "Antonio": 5,
    "Felipe": 6,
    "Daniela": 5,
    "Vicente": 7,
}
```

Acceder a un elemento dentro de un diccionario

Se accede a un elemento específico utilizando la **clave** de ese **valor**. Necesariamente se debe utilizar la misma clave en la que está almacenado el elemento, sino podríamos recibir un valor no deseado, o incluso tener un error si la clave no existe.

```
notas["Felipe"] # 6
notas["felipe"] # KeyError: 'felipe'
```

La clave tiene que ser única

En un diccionario, solo puede haber un valor asociado a una clave.

```
duplicados = {"clave": 1, "clave": 2}
duplicados # {"clave": 2}
```

Al usar dos veces la misma clave, Python se quedará con la última que definimos, ignorando completamente la existencia del primer valor.

Agregando un elemento a un diccionario

Tomemos un diccionario simple como el siguiente:

```
diccionario = {"llave 1": 5}
```

Para agregar un elemento a un diccionario, hace falta especificar una clave nueva, de forma que el nuevo valor ingresado sea identificado con dicha clave.

```
diccionario["llave 2"] = 9
diccionario # {"llave 1": 5, "llave 2": 9}
```

Ojo con la sintaxis: para definir un diccionario usamos llaves (`{}`), pero para acceder a sus elementos usamos corchetes `[]`.

Cambiando un elemento dentro de un diccionario

De forma similar a como agregamos un elemento a un diccionario, podemos actualizar el valor de uno. Para esto, simplemente tenemos que redefinir el valor asociado a la clave.

```
diccionario = {"llave 1": 5, "llave 2": 7}
diccionario["llave 2"] = 9
diccionario # {"llave 1": 5, "llave 2": 9}
```

Capítulo 2: Iterando diccionarios

Objetivos

- Iterar un diccionario con una sola variable
- Iterar un diccionario con dos variables
- Transformar los valores dentro de un diccionario
- Transformar los valores y generar un nuevo diciconario a partir de otro

Iterando un diccionario

Al iterar en un diccionario, la variable iteradora corresponde a **la clave**.

```
diccionario = {"nombre": "Juan", "apellido": "Pérez", "edad": 33, "altura": 1.75}

for clave in diccionario:
    print("Esta clave es: {}".format(clave))
```

```
Esta clave es: nombre
Esta clave es: apellido
Esta clave es: edad
Esta clave es: altura
```

Esto nos permite acceder al valor de cada clave en cada iteración:

```
diccionario = {"nombre": "Juan", "apellido": "Pérez", "edad": 33, "altura": 1.75}

for clave in diccionario:
    valor = diccionario[clave]
    print("La clave es {} y el valor es {}".format(clave, valor))
```

```
La clave es nombre y el valor es Juan
La clave es apellido y el valor es Pérez
La clave es edad y el valor es 33
La clave es altura y el valor es 1.75
```

Es por esto que podemos verificar si una clave está en un diccionario de una forma similar a como lo hacemos para saber si un elemento está en una lista: usando `in`

```
if "nombre" in diccionario:
    print("La clave 'nombre' está en el diccionario")
else:
    print("La clave 'nombre' no está en el diccionario")

if "peso" in diccionario:
    print("La clave '´peso' está en el diccionario")
else:
    print("La clave 'peso' no está en el diccionario")
```

```
La clave 'nombre' está en el diccionario
La clave 'peso' no está en el diccionario
```

Otra forma de iterar un diccionario, es accediendo a las claves y valores de forma simultánea, con una variable para la clave y otra para el valor de cada par. Para ello, se requiere que se itere sobre la función `items()` en el diccionario.

```
for clave, valor in diccionario.items():
    print("La clave es {} y el valor es {}".format(clave, valor))
```

```
La clave es nombre y el valor es Juan
La clave es apellido y el valor es Pérez
La clave es edad y el valor es 33
La clave es altura y el valor es 1.75
```

Normalmente, se utilizan las variables `k` para las claves (*key*) y `v` para los valores (*value*)

Desafío: Iteración de claves y valor

Tenemos un listado de países y la cantidad de usuarios por cada país en la siguiente tabla:

País	Cantidad de usuarios
México	70
Chile	50
Argentina	55

Requerimientos

- Mostrar solo los países
- Mostrar solo los países con más de 60 usuarios

```
usuarios_por_pais = {
    "México": 65,
    "Chile": 50,
    "Argentina": 55,
}
```

```
# Mostrar solo los países
for pais in usuarios_por_pais:
    print(pais)
```

```
México
Chile
Argentina
```

```
# Mostrar solo los países con más de 60 usuarios
for pais, usuarios in usuarios_por_pais.items():
    if usuarios > 60:
        print(pais)
```

```
México
```

Desafío: Transformación

Dada la información de ventas de 3 meses:

Mes	Ventas
Octubre	65000
Noviembre	68000
Diciembre	72000

Requerimientos

- Modificar el diccionario para incrementar las ventas en un 10%
- Hacer un nuevo diccionario pero con las ventas disminuidas un 20%

```
ventas_mensuales = {
    "Octubre": 65000,
    "Noviembre": 68000,
    "Diciembre": 72000,
}
```

```
for mes, venta in ventas_mensuales.items():
    ventas_mensuales[mes] = venta * 1.1

print(ventas_mensuales)
```

```
{'Octubre': 71500.0, 'Noviembre': 74800.0, 'Diciembre': 79200.0}
```

```
ventas_mensuales = {
    "Octubre": 65000,
    "Noviembre": 68000,
    "Diciembre": 72000,
}

nuevas_ventas_mensuales = {}
for mes, venta in ventas_mensuales.items():
    nuevas_ventas_mensuales[mes] = venta * 0.8

print(nuevas_ventas_mensuales)
print(ventas_mensuales)
```

```
{'Octubre': 52000.0, 'Noviembre': 54400.0, 'Diciembre': 57600.0}
{'Octubre': 65000, 'Noviembre': 68000, 'Diciembre': 72000}
```

Desafío: Filtrado

Dados los siguientes datos, crear un método que devuelva otro diccionario pero filtrando todos aquellos pares con un valor inferior a 70000.

Mes	Ventas
Octubre	65000
Noviembre	68000
Diciembre	72000

```
ventas_mensuales = {
    "Octubre": 65000,
    "Noviembre": 68000,
    "Diciembre": 72000,
}
```

```
def filtrar_valores_altos(diccionario):
    diccionario_filtrado = {}
    for clave, valor in diccionario.items():
        if valor > 70000:
            diccionario_filtrado[clave] = valor
    return diccionario_filtrado

print(filtrar_valores_altos(ventas_mensuales))
```

```
{'Diciembre': 72000}
```

Desafío: Integración

- Se tiene la siguiente lista de productos:

```
diccionario_productos = {"celular": 140000, "notebook": 489990, "tablet": 120000,
"cargador": 12400}
```

- Se solicita:
    - Si el producto tiene un valor menor a 120.000, se le debe aplicar un 10% de descuento (en el diccionario original).
    - En caso contrario, se debe asignar a un nuevo diccionario de filtrado.

Capítulo 3: Operaciones típicas en diccionarios

Objetivos

- Eliminar un elemento dentro de un diccionario
- Unir dos diccionarios en uno solo
- Invertir un diccionario
- Transformar las claves del diccionario en una lista
- Transformar los valores del diccionario en una lista

Introducción

En este capítulo aprenderemos varias funcionalidades de los diccionarios que facilitarán su uso. Todos los métodos que aprenderemos los podemos consultar en la documentación.

Eliminar elementos de un diccionario

Podemos eliminar una llave de un diccionario, junto a su valor, de dos formas: usando el método `pop` del diccionario o utilizando `del`. La principal diferencia entre ambas formas es que al utilizar `pop` obtendremos el valor del elemento eliminado.

```
diccionario = {"celular": 140000, "notebook": 489990, "tablet": 120000, "cargador":
12400}
```

```
del diccionario["celular"]

print(diccionario)
```

```
{'notebook': 489990, 'tablet': 120000, 'cargador': 12400}
```

```
eliminado = diccionario.pop("tablet")

print(eliminado)
print(diccionario)
```

```
120000
{'notebook': 489990, 'cargador': 12400}
```

Unir diccionarios

Una operación muy común es unir dos diccionarios, lo que se puede lograr a través de la siguiente expresión:

```
diccionario_a = {"nombre": "Alejandra", "apellido": "López", "edad": 33, "altura": 1.55}
diccionario_b = { "mascota":"miti", "ejercicio":"bicicleta"}


# Union de diccionario_a y diccionario_b
diccionario_a.update(diccionario_b)

# Notar que la unión queda en el primer diccionario
print(diccionario_a)
```

```
{'nombre': 'Alejandra', 'apellido': 'López', 'edad': 33, 'altura': 1.55, 'mascota':
'miti', 'ejercicio': 'bicicleta'}
```

Cuidado con las colisiones

Cuando ambos diccionarios tienen una clave en común, el valor del segundo diccionario sobreescribe al del primero

```
diccionario_a = {"nombre": "Alejandra", "apellido": "López", "edad": 33, "altura": 1.55}
diccionario_b = { "mascota":"miti", "ejercicio":"bicicleta", "altura": 155}

# Union de diccionario_a y diccionario_b
diccionario_a.update(diccionario_b)

# Se sobreescribió el valor de altura por el del diccionario_b
print(diccionario_a)
```

```
{'nombre': 'Alejandra', 'apellido': 'López', 'edad': 33, 'altura': 155, 'mascota':
'miti', 'ejercicio': 'bicicleta'}
```

Por lo tanto, no es lo mismo hacer `diccionario_a.update(diccionario_b)` que `diccionario_b.update(diccionario_a)` (ver el valor asociado a la clave `"edad"`:

```
diccionario_a = {"nombre": "Alejandra", "apellido": "López", "edad": 33, "altura": 1.55}
diccionario_b = { "mascota":"miti", "ejercicio":"bicicleta", "altura": 155}
diccionario_b.update(diccionario_a)
print(diccionario_b)
```

```
{'mascota': 'miti', 'ejercicio': 'bicicleta', 'altura': 1.55, 'nombre': 'Alejandra',
'apellido': 'López', 'edad': 33}
```

Invertir diccionario

A veces, resulta útil invertir la relación de un diccionario para poder acceder a la llave asociada a un valor en específico. Esto se puede lograr construyendo el diccionario nuevamente y almacenando el valor en el lugar de la llave y viceversa. Para esto podemos usar una variación de *lists comprehension* y hacerlo en una línea:

```
colors = {
    "red": "#cc0000",
    "green": "#00cc00",
    "blue": "#0000cc",
}
colors_inv = {v: k for k, v in colors.items()}
colors_inv["#cc0000"] # red
```

```
'red'
```

Obtener todas las claves y valores de un diccionario

Esto se puede lograr símplemente a través del método `keys` de un diccionario:

```
colors = {
    "red": "#cc0000",
    "green": "#00cc00",
    "blue": "#0000cc",
}

colors.keys()
```

```
dict_keys(['red', 'green', 'blue'])
```

Así como hay un método para las llaves, hay un método que permite obtener todos los valores de un diccionario, el método `values`:

```
colors = {
    "red": "#cc0000",
    "green": "#00cc00",
    "blue": "#0000cc",
}

colors.values()
```

```
dict_values(['#cc0000', '#00cc00', '#0000cc'])
```

Construir un diccionario a partir de claves y valores

Podemos usar la función `zip` para unir dos listas, una como key y la otra como value, y convertir esa unión en un diccionario:

```
dict(zip(["k1", "k2", "k3"], [1, 2, 3]))
```

```
{'k1': 1, 'k2': 2, 'k3': 3}
```

Desafío: Búsqueda de colores

Se tiene una base de datos de colores:

```
{
  "aliceblue": "#f0f8ff",
  "antiquewhite": "#faebd7",
  "aqua": "#00ffff",
  "aquamarine": "#7fffd4",
  "azure": "#f0ffff",
  "darkorchid": "#9932cc",
  "darkred": "#8b0000",
  "darksalmon": "#e9967a",
  "navajowhite": "#ffdead",
  "navy": "#000080",
  "orchid": "#da70d6",
  "palegoldenrod": "#eee8aa",
  "peachpuff": "#ffdab9",
  "peru": "#cd853f",
  "pink": "#ffc0cb",
  "purple": "#800080",
  "rebeccapurple": "#663399",
  "red": "#ff0000",
  "saddlebrown": "#8b4513",
  "seashell": "#fff5ee",
  "sienna": "#a0522d",
  "silver": "#c0c0c0",
  "skyblue": "#87ceeb",
  "slateblue": "#6a5acd",
  "teal": "#008080",
  "thistle": "#d8bfd8",
  "tomato": "#ff6347",
  "turquoise": "#40e0d0",
  "violet": "#ee82ee",
  "wheat": "#f5deb3",
  "white": "#ffffff",
  "whitesmoke": "#f5f5f5",
  "yellow": "#ffff00",
  "yellowgreen": "#9acd32",
}
```

Se pide crear un programa llamado `busqueda_colores.py`, donde el usuario ingrese un color en hexadecimal como argumento, y se muestre en pantalla el nombre del color. En caso de no encontrar el color, aparecerá el texto `no-no`

Uso:

```
$ busqueda_colores.py #6a5acd

slateblue
```

Soluciones

Para este ejercicio, hay dos soluciones posibles; Una es iterar el diccionario hasta encontrar la clave e imprimirla. Y la otra es invertir el diccionario y acceder al nombre del color utilizando el nuevo diccionario

```
colors = {
  "aliceblue": "#f0f8ff",
  "antiquewhite": "#faebd7",
  "aqua": "#00ffff",
  "aquamarine": "#7fffd4",
  "azure": "#f0ffff",
  "darkorchid": "#9932cc",
  "darkred": "#8b0000",
  "darksalmon": "#e9967a",
  "navajowhite": "#ffdead",
  "navy": "#000080",
  "orchid": "#da70d6",
  "palegoldenrod": "#eee8aa",
  "peachpuff": "#ffdab9",
  "peru": "#cd853f",
  "pink": "#ffc0cb",
  "purple": "#800080",
  "rebeccapurple": "#663399",
  "red": "#ff0000",
  "saddlebrown": "#8b4513",
  "seashell": "#fff5ee",
  "sienna": "#a0522d",
  "silver": "#c0c0c0",
  "skyblue": "#87ceeb",
  "slateblue": "#6a5acd",
  "teal": "#008080",
  "thistle": "#d8bfd8",
  "tomato": "#ff6347",
  "turquoise": "#40e0d0",
  "violet": "#ee82ee",
  "wheat": "#f5deb3",
  "white": "#ffffff",
  "whitesmoke": "#f5f5f5",
  "yellow": "#ffff00",
  "yellowgreen": "#9acd32",
}
```

```
# Solucion iterando
search = "#6a5acd"

for name, hexa in colors.items():
    if hexa == search:
        print(name)
```

```
slateblue
```

Si bien esta solución muestra en pantalla el valor correcto, no maneja la situación de cuando no encuentra el color. Podríamos estar tentandos a poner esto:

```
search = "#6a5acd"
for name, hexa in colors.items():
    if hexa == search:
        print(name)
    else:
        print("no-no")
```

El problema de este acercamiento es que vamos a mostrar muchas veces el texto de no encontrado y solo debemos mostrarlo una vez.

Para evitar esto, ocuparemos una variable para guardar cuando encontremos el valor y preguntaremos por ella al final. En la jerga, este tipo de variables se conoce como "*flag*", y se declaran fuera de un ciclo con valor `False`, y cambian su valor a `True` dada una condición. De esta forma, se permite también evitar seguir con el ciclo cuando ya se ha cumplido con la condición.

```
search = "#6a5acd"
found = False
for name, hexa in colors.items():
    if hexa == search and found == False:
        found = True
        print(name)

if not found:
    print("no-no")
```

```
slateblue
```

Como se mencionó, la otra forma de resolver el problema es invirtiendo el diccionario, y preguntando si la llave se encuentra en el diccionario invertido mediante el operador `in`.

```
# Solucion invirtiendo el diccionario y usando in
search = "#6a5acd"
colors_inv = {v: k for k, v in colors.items()}
if search in colors_inv:
    print(colors_inv[search])
else:
    print("no-no")
```

```
slateblue
```

Varias soluciones

¿Por qué se ven varias soluciones a un problema, si con una basta?

Esta pregunta es muy importante. Un programador no busca memorizar la respuesta a los problemas, de hecho muy rara vez nos toca resolver exactamente el mismo problemas dos veces. Pero el acercamiento a resolver un problema sí puede ser reutilizado en situaciones similares.

Lo importante es conocer diversas estrategias que nos permitan después enfrentar problemas nuevos.

En esta ocasión, aprendimos que invertir un diccionario puede ser mucho más fácil que iterarlo.

Desafío: Búsqueda de múltiples colores

Modificar el programa anterior para que el usuario pueda buscar múltiples colores.

Uso:

```
$ busqueda_colores.py #6a5acd #40e0d0

slateblue
turquoise
```

Tip: El usuario puede ingresar cuantos colores desee

Discutamos primerero la forma incorrecta de resolverlo

```
import sys

search1 = sys.argv[1]
search2 = sys.argv[2]
search3 = sys.argv[3]

colors_inv = {v: k for k, v in colors.items()}

if search1 in colors_inv:
    print(colors_inv[search1])
else:
    print("no-no")

if search2 in colors_inv:
    print(colors_inv[search2])
else:
    print("no-no")

if search3 in colors_inv:
    print(colors_inv[search3])
else:
    print("no-no")
```

Es una pésima solución porque si el usuario ingresa mas valores no funcionará. Otra forma de darse cuenta de que es mal código es porque estamos repitiendo el mismo código varias veces.

```
# Solución buena
import sys
colors_inv = {v: k for k, v in colors.items()}

# Simplemente tenemos que iterar la lista.
searched_hexas = sys.argv[1:]

for search in searched_hexas:
    if search in colors_inv:
        print(colors_inv[search])
    else:
        print("no-no")
```

Capítulo 4: Listas y diccionarios

Objetivos

- Convertir un diccionario en una lista
- Convertir una lista en un diccionario
- Transformar dos listas relacionadas en una única lista

¿Resolver un problema con una lista o con un diccionario?

Las listas y los diccionarios tienen distintas ventajas y desventajas. En algunos casos será más conveniente trabajar con diccionarios, mientras que en otras ocasiones será más conveniente hacerlo con listas.

Veamos un ejemplo

Supongamos que queremos almacenar alumnos y sus notas, perfectamente podríamos guardar la información en dos listas, una con los nombres y otra con las notas.

```
nombres = ["Alumno1", "Alumno2", "Alumno3"]
notas = [10, 3, 8]
```

Se solicita crear un programa que muestre la nota de un alumno dado su nombre.

Resolviendo el problema con listas

De esta forma, si quisieramos consultar la nota de un alumno, podríamos recuperar la posición del alumno en la lista de nombres, y luego consultar con ésta en la lista de las notas.

```
index = nombres.index('Alumno1')
nota_alumno = notas[index]
```

Resolviendo el problema con diccionarios

Mientras que resolver el mismo problema con un diccionario puede ser mucho mas sencillo.

```
notas_por_alumno = {
    "Alumno1": 10,
    "Alumno2": 3,
    "Alumno3": 8,
}
print(notas_por_alumno["Alumno1"])
```

Otro motivo importante

Algunas funciones reciben diccionarios como argumento, mientras que otras reciben listas. Por tanto, se requiere saber transformar los datos de un tipo al otro.

A lo largo del programa, no siempre se tiene control sobre cuál es la estructura que se está trabajando. Por ejemplo, puede ser que el resultado de una función sea un diccionario, pero se requiera una lista como argumento para otra función.

Convertir estructuras

Como vimos, es importante saber cómo transformar una lista en un diccionario y viceversa, para resolver distintos tipos de problemas.

Convertir un diccionario en una lista

Para lograr esto, se debe utilizar la función `items()`. Cada par (clave, valor) será una tupla:

```
list({"k1": 5, "k2": 7}.items()) # [('k1', 5), ('k2', 7)]
```

Una *tupla* es una estructura de datos similar a una lista, con la diferencia de que sus datos no son mutables. Se definen entre paréntesis redondos `()`.

Convertir una lista en un diccionario

Para invertir la transformación se utiliza la función `dict`

```
dict([('k1', 5), ('k2', 7)]) # {"k1": 5, "k2": 7}
```

Cruzando información de dos listas

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]
```

Si se tiene información en 2 listas separadas, y se quiere juntar ambas en un diccionario, las opciones son:

- Iterar las listas simultáneamente, con un índice
- Utilizar el método `zip`

Iterando las listas simultáneamente

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]

notas_por_alumno = {}
for i in range(len(nombres)):
    alumno = nombres[i]
    nota = notas[i]
    notas_por_alumno[alumno] = nota

print(notas_por_alumno)
```

```
{'Alumno1': 10, 'Alumno2': 3, 'Alumno3': 8}
```

Iterar el arreglo elemento a elemento también es posible, pero un poco más complejo.

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]

notas_por_alumno = {}
for alumno in nombres:
    i = nombres.index(alumno) # Obtenemos el indice
    alumno = nombres[i]
    nota = notas[i]
    notas_por_alumno[alumno] = nota

print(notas_por_alumno)
```

```
{'Alumno1': 10, 'Alumno2': 3, 'Alumno3': 8}
```

Utilizando `zip`

```
nombres = ['Alumno1', 'Alumno2', 'Alumno3']
notas = [10, 3, 8]

notas_por_alumno = dict(zip(nombres, notas))

print(notas_por_alumno)
```

```
{'Alumno1': 10, 'Alumno2': 3, 'Alumno3': 8}
```

Uso de `group by`

En el módulo `itertools` de Python, se tiene disponible la función `groupby`, la cual permite agrupar un conjunto de elementos bajo cualquier criterio.

Esta función, al ser iterada, retornará llaves consecutivas y sus grupos asociados. Cada grupo se creará según los datos que tengan igual valor. Es decir, cuando el valor iterado cambie, se creará un nuevo grupo. Por ello, se requiere que los datos estén ordenados antes de poder agruparlos.

Veremos algunos ejemplos.

```
from itertools import groupby
```

```
# Primero vemos qué viene en cada par de elementos de groupby
[(k, list(g)) for k, g in groupby('AAAABBBCCDAABBB')]
```

```
[('A', ['A', 'A', 'A', 'A']),
 ('B', ['B', 'B', 'B']),
 ('C', ['C', 'C']),
 ('D', ['D']),
 ('A', ['A', 'A']),
 ('B', ['B', 'B', 'B'])]
```

Por cada letra, se crea un grupo que tiene como clave la letra misma, y como valor una lista con la cantidad de ocurrencias de la letra.

```
# Podemos ocupar este output para generar un diccionario
{k: len(list(g)) for k, g in groupby('AAAABBBCCD')}
```

```
{'A': 4, 'B': 3, 'C': 2, 'D': 1}
```

Supongamos que tenemos una lista de palabras, que queremos agrupar según su largo.

Vimos que como primer argumento de `groupby` ingresamos una estructura iterable, en el caso anterior, un `string`, donde en cada iteración cada elemento corresponde a una letra. Pero también puede ser sobre una lista, donde cada grupo será cada elemento de la lista.

Esto por un lado, ya nos permite agrupar en base a una lista de palabras. Por otro lado, `groupby` recibe un segundo parámetro opcional, que corressponde cuál será la clave o `key` para la estructura retornada. Si no se especifica este parámetro, toma por defecto el valor mismo de la estructura agrupada. En caso de especificarse, debe corresponder a una función que se aplique al grupo y que retorne un valor.

Teniendo ya esta información, podemos crear el diccionario.

```
# Agrupar palabras por su largo
words = ["hola", "a", "todos", "y", "cada", "uno"]
# Luego agrupamos
{k: list(v) for k, v in groupby(words, key=len)}
```

```
{4: ['cada'], 1: ['y'], 5: ['todos'], 3: ['uno']}
```

Como se puede ver, el output no es el correcto. Esto sucede porque los elementos no están ordenados según el criterio por el que los queremos agrupar. Por lo tanto, primero ordenaremos las palabras según su largo y luego agruparemos.

```
# Primero ordenamos
words.sort(key=lambda x: len(x))
# Luego agrupamos
{k: list(v) for k, v in groupby(words, key=len)}
```

```
{1: ['a', 'y'], 3: ['uno'], 4: ['hola', 'cada'], 5: ['todos']}
```

Contando elementos dentro de un diccionario

Un uso relativamente frecuente de un diccionario es para contar elementos. Por ejemplo, tenemos la siguiente lista.

```
[1, 2, 6, 7, 2, 5, 8, 9, 1, 3, 6, 7]
```

Y queremos contar la cantidad de veces que aparece cada uno de los elementos.

Vamos a resolver el problema con dos estrategias distintas:

- Iterar y contar
- Agrupar y contar

Iterar y contar

Para resolver el problema por iteración, cada vez que se encuentre un elemento nuevo en el ciclo, éste se guardará en un diccionario con la cuenta en uno. Si se encuentra un elemento que ya está en el diccionario, se incrementará su cuenta en 1.

```
lista = [1, 2, 6, 7, 2, 5, 8, 9, 1, 2, 9, 7]
diccionario = {}
for i in lista:
    if i in diccionario:
        diccionario[i] += 1
    else:
        diccionario[i] = 1

print(diccionario)
```

```
{1: 2, 2: 3, 6: 1, 7: 2, 5: 1, 8: 1, 9: 2}
```

Agrupar y contar

Otra alternativa es usar `groupby`, recordando ordenar primero los datos y luego agruparlos, para finalmente contarlos midiendo el largo de la lista entregada por la función.

```
lista = [1, 2, 6, 7, 2, 5, 8, 9, 1, 2, 9, 7]

lista.sort()
diccionario = {k: len(list(v)) for k, v in groupby(lista)}

print(diccionario)
```

```
{1: 2, 2: 3, 5: 1, 6: 1, 7: 2, 8: 1, 9: 2}
```

Capítulo 5: Introducción a APIs

Objetivos

- Conocer el concepto de *request* y *response*
- Conocer la importancia de las API REST para comunicar programas por internet
- Utilizar Postman para realizar un *request* a una API

API y API REST

API, acrónimo de ***A**pplication **P**rogram **I**nterface*, es una interfaz de acceso que nos permite comunicar programas. Existen diversos tipos de APIs, pero particularmente hablaremos de **API REST** por lo que de ahora en adelante cuando utilicemos el concepto de API, nos estaremos refiriendo a una API REST.

En términos burdos, una API REST es una página web para un programa, es decir, una página web fácil de ser consultada desde cualquier programa. Esto es muy útil para integrar sistemas y comunicar distintas aplicaciones.

Al programa que consulta se le denomina **cliente**, mientras que al programa que entrega la respuesta se le suele llamar **servidor**.

Tipos de APIs existentes

Existen miles de APIs para distintos propósitos, por ejemplo:

- Clima y temperatura
- Cambio de monedas
- Indicadores económicos
- Servicios para subir archivos
- Compra y venta de criptomonedas
- Servicios de geolocalización, como Google Maps

Cualquier empresa puede construir una API y hacerla disponible a través de internet.

¿Cómo se usa una API?

Muy sencillo. A través de un cliente hacemos un ***request*** (pedido) a una dirección y obtendremos como resultado un ***response*** (respuesta).

Un *request* (o pedido) es información que el cliente envía a una página web aunque, para ser más precisos, desde ahora en adelante diremos **URL** (localización de un recurso). El *response* es la información que esa URL devuelve al cliente.

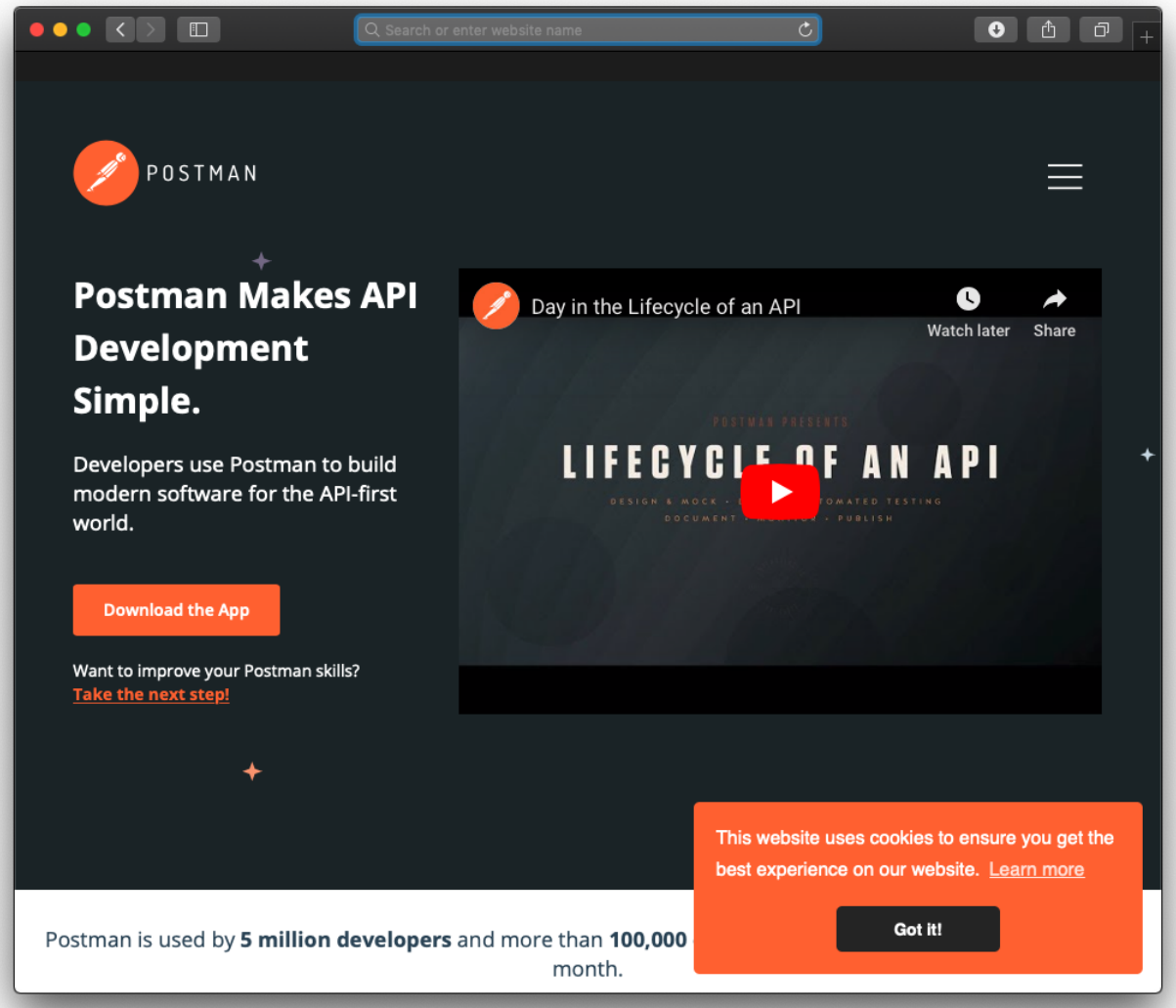
En este capítulo aprenderemos cómo hacer una request, y cómo analizar una response.

Probando una API con Postman

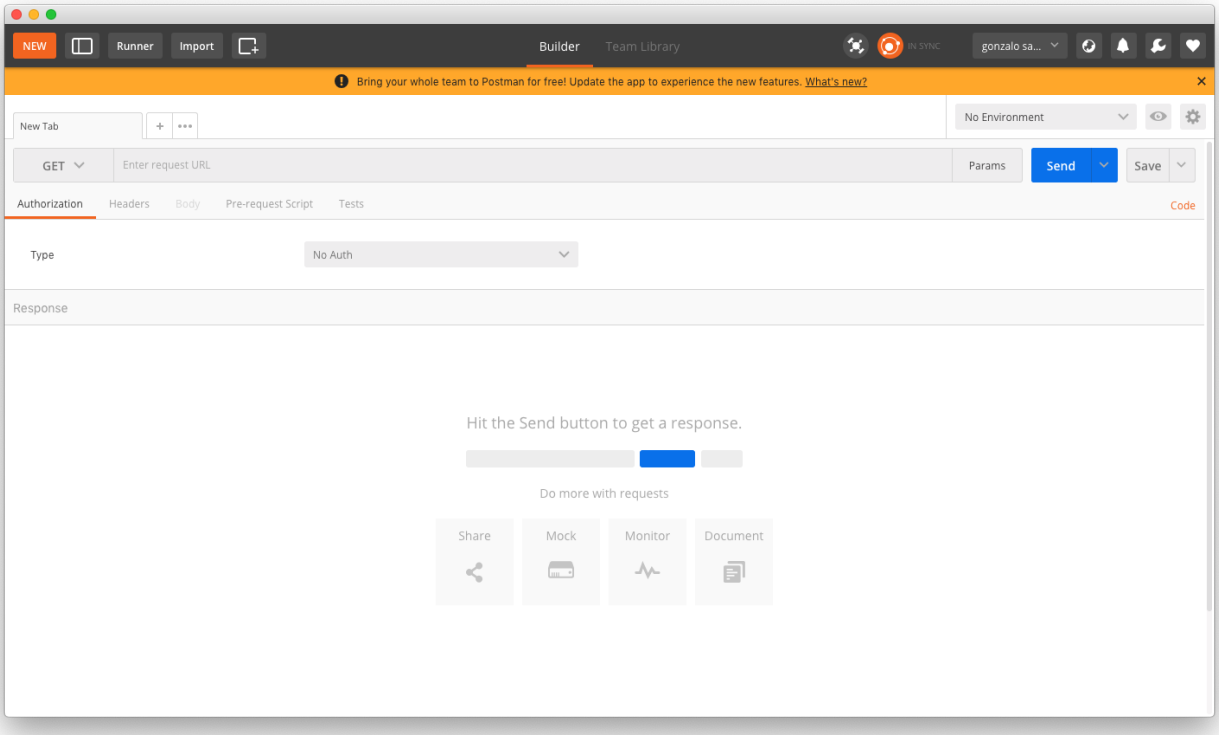
Existe una herramienta muy potente para probar APIs sin necesidad de programar, la que nos ayudará a comprender esta idea. Esta herramienta es **Postman**.

Descargando Postman

Podemos descargar Postman desde la página oficial. https://www.getpostman.com



Instalaremos y abriremos la aplicación descargada, donde deberíamos ver un panel como el siguiente.

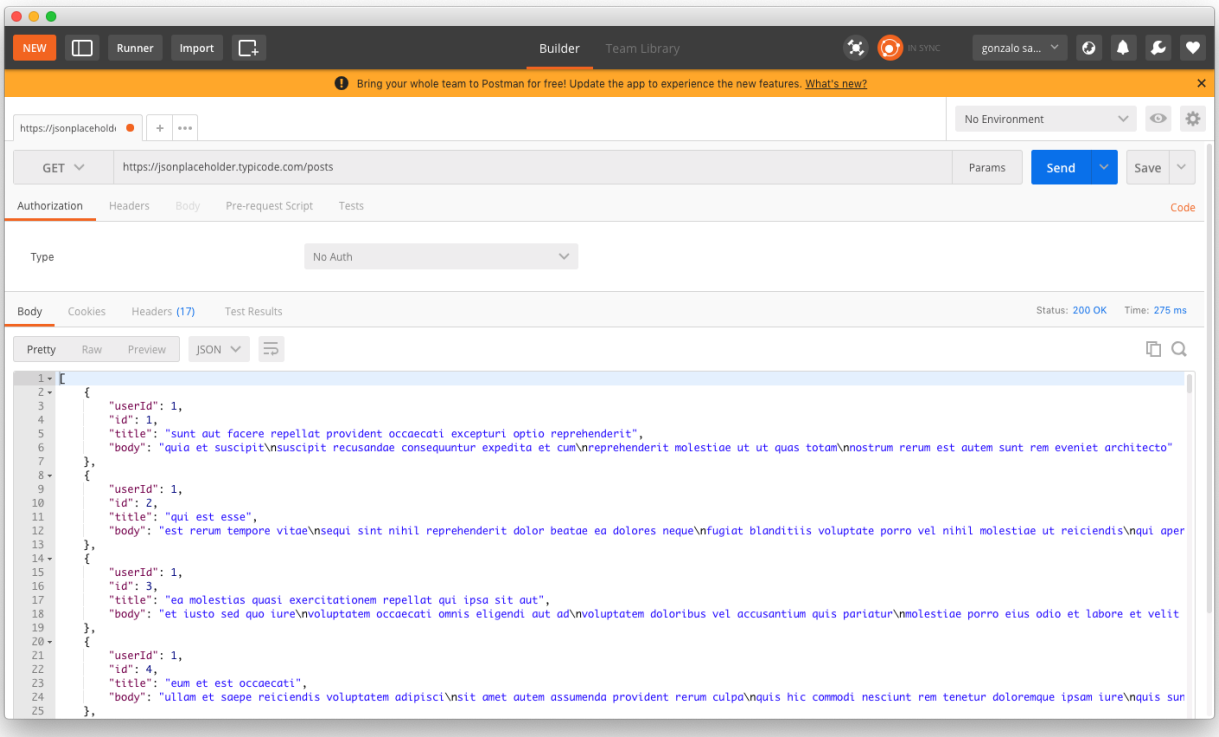


Nuestra primera consulta a una API

Nuestra primera consulta será a una API llamada jsonplaceholder, `https://jsonplaceholder.typicode.com/`. Esta API es una API falsa en el sentido de que no tiene información real, pero es una API en todo otro sentido.

Primer *request*

Donde dice "Request URL" pondremos la siguiente URL: `https://jsonplaceholder.typicode.com/posts`. Luego, haremos click donde dice "Send" y obtendremos nuestra primera respuesta.



Luego de algunos milisegundos (o segundos en el peor de los casos) el servidor nos enviará una respuesta: el texto que aparece en la parte inferior de la imagen.

La respuesta se parece mucho a cosas que hemos previamente: primero vemos la apertura de una lista y luego varios diccionarios. La respuesta está en un formato llamado **JSON**.

Se debe tener en cuenta eso sí, que si bien uno ve que la respuesta está organizada en una estructura como la de los diccionarios, en realidad corresponde a *un solo string* o cadena de texto (desde el inicio hasta el fin de la respuesta). Lo que ocurre es que Postman, y algunas extensiones de navegadores, permiten visualizar estos *string con estructura JSON* en una forma más "humana" y entendible.

JSON

JSON es acrónimo de JavaScript Object Notation. Es un formato para enviar información en texto plano, fácilmente leíble por humanos y fácilmente analizable por lenguajes de programación. Hoy en día es uno de los formato más utilizados para enviar información entre sistemas.

JSON no requiere de JavaScript

Que el acrónimo tenga la palabra JavaScript, no quiere decir que necesitemos saber JavaScript o utilizarlo. JSON es un formato plano que podemos utilizar desde cualquier lenguaje.

JSON a Python

En Python, transformar un string con un JSON a un diccionario es tan sencillo como:

```
import json

json.loads(string)
```

Esto lo veremos con detalle en el siguiente capítulo.

Capítulo 6: Consumiendo una API desde Python

Objetivos

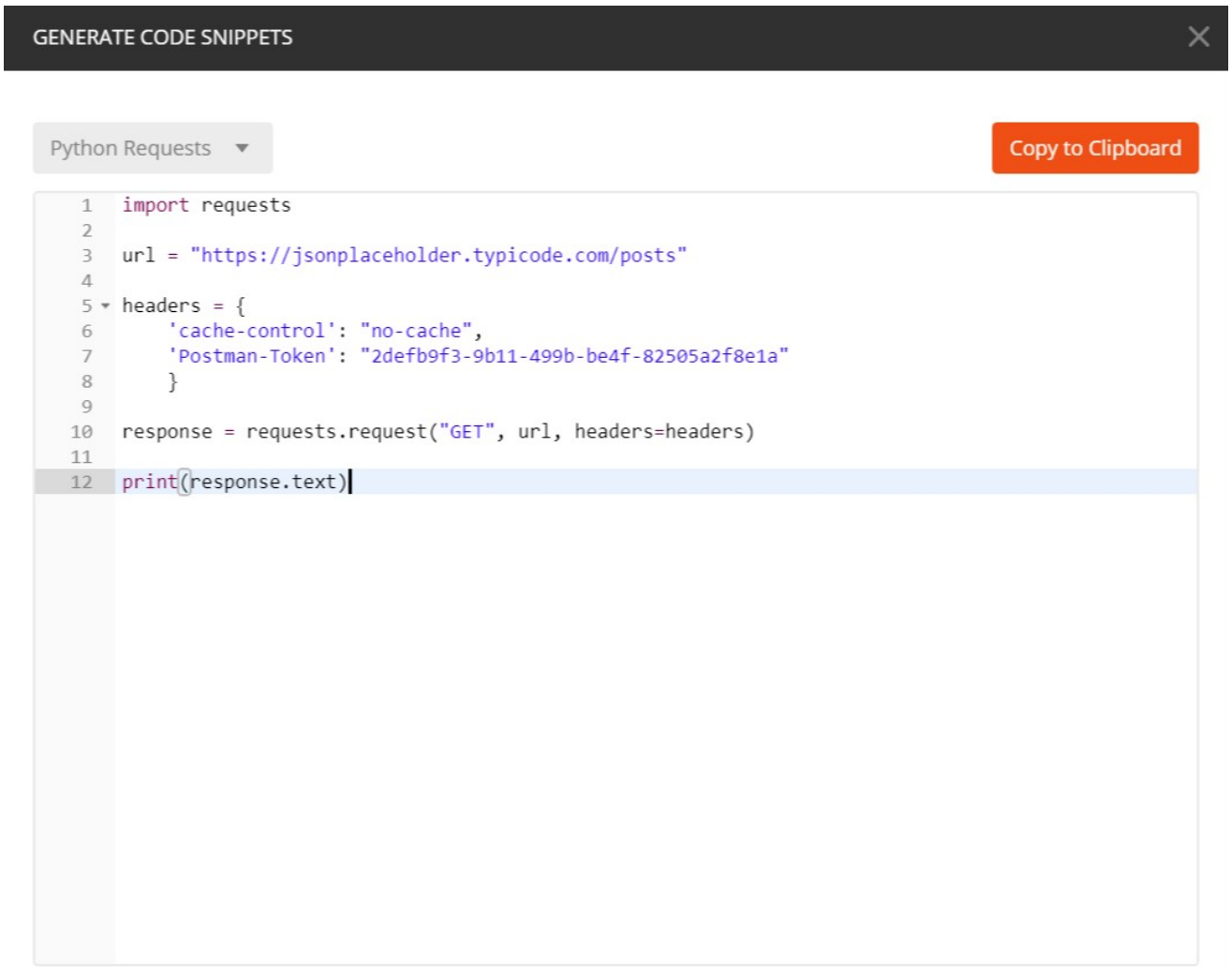
- Utilizar Postman para obtener el código Python necesario para hacer un *request*
- Consumir una API desde Python
- Transformar la respuesta de JSON a una estructura utilizable directamente en Python
- Conocer las implicaciones de SSL
- Aprender a nunca enviar información sensible sin SSL
- Entender la importancia de verificar el certificado

Crear un script que genere un request

En el capítulo anterior, aprendimos a realizar un *request* a una URL utilizando Postman. En este capítulo lo haremos desde Python.

Obteniendo el código desde Postman

Postman nos entrega automáticamente el código para hacer un *request* a la API.



Este código podemos pegarlo directamente en un script y probarlo.

```
import requests

url = "https://jsonplaceholder.typicode.com/posts"

headers = {
    'cache-control': "no-cache",
    'Postman-Token': "2defb9f3-9b11-499b-be4f-82505a2f8e1a",
}

response = requests.request("GET", url, headers=headers)

# Cuerpo de la respuesta o "body"
print(response.text)

# Código de la respuesta
print(response)

[
  {
```

```
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio
reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et
cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet
architecto"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea
dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui
aperiam non debitis possimus qui neque nisi nulla"
  },
  {
    "userId": 1,
    "id": 3,
    "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
    "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut
ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et
labore et velit aut"
  },
  {
    "userId": 1,
    "id": 4,
    "title": "eum et est occaecati",
    "body": "ullam et saepe reiciendis voluptatem adipisci\nsit amet autem assumenda
provident rerum culpa\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\nquis
sunt voluptatem rerum illo velit"
  },
  {
    "userId": 1,
    "id": 5,
    "title": "nesciunt quas odio",
    "body": "repudiandae veniam quaerat sunt sed\nalias aut fugiat sit autem sed
est\nvoluptatem omnis possimus esse voluptatibus quis\nest aut tenetur dolor neque"
  },
  {
    "userId": 1,
    "id": 6,
    "title": "dolorem eum magni eos aperiam quia",
    "body": "ut aspernatur corporis harum nihil quis provident sequi\nmollitia nobis
aliquid molestiae\nperspiciatis et ea nemo ab reprehenderit accusantium quas\nvoluptate
dolores velit et doloremque molestiae"
  },
  {
    "userId": 1,
    "id": 7,
    "title": "magnam facilis autem",
    "body": "dolore placeat quibusdam ea quo vitae\nmagni quis enim qui quis quo nemo
aut saepe\nquidem repellat excepturi ut quia\nsunt ut sequi eos ea sed quas"
```

```
  },
  {
    "userId": 1,
    "id": 8,
    "title": "dolorem dolore est ipsam",
    "body": "dignissimos aperiam dolorem qui eum\nfacilis quibusdam animi sint suscipit qui sint possimus cum\nquaerat magni maiores excepturi\nipsam ut commodi dolor voluptatum modi aut vitae"
  },
  {
    "userId": 1,
    "id": 9,
    "title": "nesciunt iure omnis dolorem tempora et accusantium",
    "body": "consectetur animi nesciunt iure dolore\nenim quia ad\nveniam autem ut quam aut nobis\net est aut quod aut provident voluptas autem voluptas"
  },
  {
    "userId": 1,
    "id": 10,
    "title": "optio molestias id quia eum",
    "body": "quo et expedita modi cum officia vel magni\ndoloribus qui repudiandae\nvero nisi sit\nquos veniam quod sed accusamus veritatis error"
  },
  {
    "userId": 2,
    "id": 11,
    "title": "et ea vero quia laudantium autem",
    "body": "delectus reiciendis molestiae occaecati non minima eveniet qui voluptatibus\naccusamus in eum beatae sit\nvel qui neque voluptates ut commodi qui incidunt\nut animi commodi"
  },
  {
    "userId": 2,
    "id": 12,
    "title": "in quibusdam tempore odit est dolorem",
    "body": "itaque id aut magnam\npraesentium quia et ea odit et ea voluptas et\nsapiente quia nihil amet occaecati quia id voluptatem\nincidunt ea est distinctio odio"
  },
  {
    "userId": 2,
    "id": 13,
    "title": "dolorum ut in voluptas mollitia et saepe quo animi",
    "body": "aut dicta possimus sint mollitia voluptas commodi quo doloremque\niste corrupti reiciendis voluptatem eius rerum\nsit cumque quod eligendi laborum minima\nperferendis recusandae assumenda consectetur porro architecto ipsum ipsam"
  },
  {
    "userId": 2,
    "id": 14,
    "title": "voluptatem eligendi optio",
    "body": "fuga et accusamus dolorum perferendis illo voluptas\nnon doloremque neque facere\nad qui dolorum molestiae beatae\nsed aut voluptas totam sit illum"
```

```
  },
  {
    "userId": 2,
    "id": 15,
    "title": "eveniet quod temporibus",
    "body": "reprehenderit quos placeat\nvelit minima officia dolores impedit
repudiandae molestiae nam\nvoluptas recusandae quis delectus\nofficiis harum fugiat
vitae"
  },
  {
    "userId": 2,
    "id": 16,
    "title": "sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio",
    "body": "suscipit nam nisi quo aperiam aut\nasperiores eos fugit maiores
voluptatibus quia\nvoluptatem quis ullam qui in alias quia est\nconsequatur magni
mollitia accusamus ea nisi voluptate dicta"
  },
  {
    "userId": 2,
    "id": 17,
    "title": "fugit voluptas sed molestias voluptatem provident",
    "body": "eos voluptas et aut odit natus earum\naspernatur fuga molestiae
ullam\ndeserunt ratione qui eos\nqui nihil ratione nemo velit ut aut id quo"
  },
  {
    "userId": 2,
    "id": 18,
    "title": "voluptate et itaque vero tempora molestiae",
    "body": "eveniet quo quis\nlaborum totam consequatur non dolor\nut et est
repudiandae\nest voluptatem vel debitis et magnam"
  },
  {
    "userId": 2,
    "id": 19,
    "title": "adipisci placeat illum aut reiciendis qui",
    "body": "illum quis cupiditate provident sit magnam\nea sed aut omnis\nveniam
maiores ullam consequatur atque\nadipisci quo iste expedita sit quos voluptas"
  },
  {
    "userId": 2,
    "id": 20,
    "title": "doloribus ad provident suscipit at",
    "body": "qui consequuntur ducimus possimus quisquam amet similique\nsuscipit porro
ipsam amet\neos veritatis officiis exercitationem vel fugit aut necessitatibus
totam\nomnis rerum consequatur expedita quidem cumque explicabo"
  },
  {
    "userId": 3,
    "id": 21,
    "title": "asperiores ea ipsam voluptatibus modi minima quia sint",
    "body": "repellat aliquid praesentium dolorem quo\nsed totam minus non itaque\nnihil
labore molestiae sunt dolor eveniet hic recusandae veniam\ntempora et tenetur expedita
sunt"
```

```
  },
  {
    "userId": 3,
    "id": 22,
    "title": "dolor sint quo a velit explicabo quia nam",
    "body": "eos qui et ipsum ipsam suscipit aut\nsed omnis non odio\nexpedita earum mollitia molestiae aut atque rem suscipit\nnam impedit esse"
  },
  {
    "userId": 3,
    "id": 23,
    "title": "maxime id vitae nihil numquam",
    "body": "veritatis unde neque eligendi\nquae quod architecto quo neque vitae\nest illo sit tempora doloremque fugit quod\net et vel beatae sequi ullam sed tenetur perspiciatis"
  },
  {
    "userId": 3,
    "id": 24,
    "title": "autem hic labore sunt dolores incidunt",
    "body": "enim et ex nulla\nomnis voluptas quia qui\nvoluptatem consequatur numquam aliquam sunt\ntotam recusandae id dignissimos aut sed asperiores deserunt"
  },
  {
    "userId": 3,
    "id": 25,
    "title": "rem alias distinctio quo quis",
    "body": "ullam consequatur ut\nomnis quis sit vel consequuntur\nipsa eligendi ipsum molestiae et omnis error nostrum\nmolestiae illo tempore quia et distinctio"
  },
  {
    "userId": 3,
    "id": 26,
    "title": "est et quae odit qui non",
    "body": "similique esse doloribus nihil accusamus\nomnis dolorem fuga consequuntur reprehenderit fugit recusandae temporibus\nperspiciatis cum ut laudantium\nomnis aut molestiae vel vero"
  },
  {
    "userId": 3,
    "id": 27,
    "title": "quasi id et eos tenetur aut quo autem",
    "body": "eum sed dolores ipsam sint possimus debitis occaecati\ndebitis qui qui et\nut placeat enim earum aut odit facilis\nconsequatur suscipit necessitatibus rerum sed inventore temporibus consequatur"
  },
  {
    "userId": 3,
    "id": 28,
    "title": "delectus ullam et corporis nulla voluptas sequi",
    "body": "non et quaerat ex quae ad maiores\nmaiores recusandae totam aut blanditiis mollitia quas illo\nut voluptatibus voluptatem\nsimilique nostrum eum"
  },
```

```
  {
    "userId": 3,
    "id": 29,
    "title": "iusto eius quod necessitatibus culpa ea",
    "body": "odit magnam ut saepe sed non qui\ntempora atque nihil\naccusamus illum
doloribus illo dolor\neligendi repudiandae odit magni similique sed cum maiores"
  },
  {
    "userId": 3,
    "id": 30,
    "title": "a quo magni similique perferendis",
    "body": "alias dolor cumque\nimpedit blanditiis non eveniet odio maxime\nblanditiis
amet eius quis tempora quia autem rem\na provident perspiciatis quia"
  },
  {
    "userId": 4,
    "id": 31,
    "title": "ullam ut quidem id aut vel consequuntur",
    "body": "debitis eius sed quibusdam non quis consectetur vitae\nimpedit ut qui
consequatur sed aut in\nquidem sit nostrum et maiores adipisci atque\nquaerat voluptatem
adipisci repudiandae"
  },
  {
    "userId": 4,
    "id": 32,
    "title": "doloremque illum aliquid sunt",
    "body": "deserunt eos nobis asperiores et hic\nest debitis repellat molestiae
optio\nnihil ratione ut eos beatae quibusdam distinctio maiores\nearum voluptates et aut
adipisci ea maiores voluptas maxime"
  },
  {
    "userId": 4,
    "id": 33,
    "title": "qui explicabo molestiae dolorem",
    "body": "rerum ut et numquam laborum odit est sit\nid qui sint in\nquasi tenetur
tempore aperiam et quaerat qui in\nrerum officiis sequi cumque quod"
  },
  {
    "userId": 4,
    "id": 34,
    "title": "magnam ut rerum iure",
    "body": "ea velit perferendis earum ut voluptatem voluptate itaque iusto\ntotam
pariatur in\nnemo voluptatem voluptatem autem magni tempora minima in\nest distinctio
qui assumenda accusamus dignissimos officia nesciunt nobis"
  },
  {
    "userId": 4,
    "id": 35,
    "title": "id nihil consequatur molestias animi provident",
    "body": "nisi error delectus possimus ut eligendi vitae\nplaceat eos harum
cupiditate facilis reprehenderit voluptatem beatae\nmodi ducimus quo illum voluptas
eligendi\net nobis quia fugit"
  },
```

```
  {
    "userId": 4,
    "id": 36,
    "title": "fuga nam accusamus voluptas reiciendis itaque",
    "body": "ad mollitia et omnis minus architecto odit\nvoluptas doloremque maxime aut non ipsa qui alias veniam\nblanditiis culpa aut quia nihil cumque facere et occaecati\nqui aspernatur quia eaque ut aperiam inventore"
  },
  {
    "userId": 4,
    "id": 37,
    "title": "provident vel ut sit ratione est",
    "body": "debitis et eaque non officia sed nesciunt pariatur vel\nvoluptatem iste vero et ea\nnumquam aut expedita ipsum nulla in\nvoluptates omnis consequatur aut enim officiis in quam qui"
  },
  {
    "userId": 4,
    "id": 38,
    "title": "explicabo et eos deleniti nostrum ab id repellendus",
    "body": "animi esse sit aut sit nesciunt assumenda eum voluptas\nquia voluptatibus provident quia necessitatibus ea\nrerum repudiandae quia voluptatem delectus fugit aut id quia\nratione optio eos iusto veniam iure"
  },
  {
    "userId": 4,
    "id": 39,
    "title": "eos dolorem iste accusantium est eaque quam",
    "body": "corporis rerum ducimus vel eum accusantium\nmaxime aspernatur a porro possimus iste omnis\nest in deleniti asperiores fuga aut\nvoluptas sapiente vel dolore minus voluptatem incidunt ex"
  },
  {
    "userId": 4,
    "id": 40,
    "title": "enim quo cumque",
    "body": "ut voluptatum aliquid illo tenetur nemo sequi quo facilis\nipsum rem optio mollitia quas\nvoluptatem eum voluptas qui\nunde omnis voluptatem iure quasi maxime voluptas nam"
  },
  {
    "userId": 5,
    "id": 41,
    "title": "non est facere",
    "body": "molestias id nostrum\nexcepturi molestiae dolore omnis repellendus quaerat saepe\nconsectetur iste quaerat tenetur asperiores accusamus ex ut\nnam quidem est ducimus sunt debitis saepe"
  },
  {
    "userId": 5,
    "id": 42,
    "title": "commodi ullam sint et excepturi error explicabo praesentium voluptas",
```

```
    "body": "odio fugit voluptatum ducimus earum autem est incidunt voluptatem\nodit
reiciendis aliquam sunt sequi nulla dolorem\nnon facere repellendus voluptates
quia\nratione harum vitae ut"
  },
  {
    "userId": 5,
    "id": 43,
    "title": "eligendi iste nostrum consequuntur adipisci praesentium sit beatae
perferendis",
    "body": "similique fugit est\nillum et dolorum harum et voluptate eaque
quidem\nexercitationem quos nam commodi possimus cum odio nihil nulla\ndolorum
exercitationem magnam ex et a et distinctio debitis"
  },
  {
    "userId": 5,
    "id": 44,
    "title": "optio dolor molestias sit",
    "body": "temporibus est consectetur dolore\net libero debitis vel velit laboriosam
quia\nipsum quibusdam qui itaque fuga rem aut\nea et iure quam sed maxime ut distinctio
quae"
  },
  {
    "userId": 5,
    "id": 45,
    "title": "ut numquam possimus omnis eius suscipit laudantium iure",
    "body": "est natus reiciendis nihil possimus aut provident\nex et dolor\nrepellat
pariatur est\nnobis rerum repellendus dolorem autem"
  },
  {
    "userId": 5,
    "id": 46,
    "title": "aut quo modi neque nostrum ducimus",
    "body": "voluptatem quisquam iste\nvoluptatibus natus officiis facilis dolorem\nquis
quas ipsam\nvel et voluptatum in aliquid"
  },
  {
    "userId": 5,
    "id": 47,
    "title": "quibusdam cumque rem aut deserunt",
    "body": "voluptatem assumenda ut qui ut cupiditate aut impedit veniam\noccaecati
nemo illum voluptatem laudantium\nmolestiae beatae rerum ea iure soluta
nostrum\neligendi et voluptate"
  },
  {
    "userId": 5,
    "id": 48,
    "title": "ut voluptatem illum ea doloribus itaque eos",
    "body": "voluptates quo voluptatem facilis iure occaecati\nvel assumenda rerum
officia et\nillum perspiciatis ab deleniti\nlaudantium repellat ad ut et autem
reprehenderit"
  },
  {
    "userId": 5,
```

```
    "id": 49,
    "title": "laborum non sunt aut ut assumenda perspiciatis voluptas",
    "body": "inventore ab sint\nnatus fugit id nulla sequi architecto nihil quaerat\neos
tenetur in in eum veritatis non\nquibusdam officiis aspernatur cumque aut commodi aut"
  },
  {
    "userId": 5,
    "id": 50,
    "title": "repellendus qui recusandae incidunt voluptates tenetur qui omnis
exercitationem",
    "body": "error suscipit maxime adipisci consequuntur recusandae\nvoluptas eligendi
et est et voluptates\nquia distinctio ab amet quaerat molestiae et vitae\nadipisci
impedit sequi nesciunt quis consectetur"
  },
  {
    "userId": 6,
    "id": 51,
    "title": "soluta aliquam aperiam consequatur illo quis voluptas",
    "body": "sunt dolores aut doloribus\ndolore doloribus voluptates tempora
et\ndoloremque et quo\ncum asperiores sit consectetur dolorem"
  },
  {
    "userId": 6,
    "id": 52,
    "title": "qui enim et consequuntur quia animi quis voluptate quibusdam",
    "body": "iusto est quibusdam fuga quas quaerat molestias\na enim ut sit accusamus
enim\ntemporibus iusto accusantium provident architecto\nsoluta esse reprehenderit qui
laborum"
  },
  {
    "userId": 6,
    "id": 53,
    "title": "ut quo aut ducimus alias",
    "body": "minima harum praesentium eum rerum illo dolore\nquasi exercitationem rerum
nam\nporro quis neque quo\nconsequatur minus dolor quidem veritatis sunt non explicabo
similique"
  },
  {
    "userId": 6,
    "id": 54,
    "title": "sit asperiores ipsam eveniet odio non quia",
    "body": "totam corporis dignissimos\nvitae dolorem ut occaecati accusamus\nex velit
deserunt\net exercitationem vero incidunt corrupti mollitia"
  },
  {
    "userId": 6,
    "id": 55,
    "title": "sit vel voluptatem et non libero",
    "body": "debitis excepturi ea perferendis harum libero optio\neos accusamus cum fuga
ut sapiente repudiandae\net ut incidunt omnis molestiae\nnihil ut eum odit"
  },
  {
    "userId": 6,
```

```
    "id": 56,
    "title": "qui et at rerum necessitatibus",
    "body": "aut est omnis dolores\nneque rerum quod ea rerum velit pariatur beatae excepturi\net provident voluptas corrupti\ncorporis harum reprehenderit dolores eligendi"
  },
  {
    "userId": 6,
    "id": 57,
    "title": "sed ab est est",
    "body": "at pariatur consequuntur earum quidem\nquo est laudantium soluta voluptatem\nqui ullam et est\net cum voluptas voluptatum repellat est"
  },
  {
    "userId": 6,
    "id": 58,
    "title": "voluptatum itaque dolores nisi et quasi",
    "body": "veniam voluptatum quae adipisci id\net id quia eos ad et dolorem\naliquam quo nisi sunt eos impedit error\nad similique veniam"
  },
  {
    "userId": 6,
    "id": 59,
    "title": "qui commodi dolor at maiores et quis id accusantium",
    "body": "perspiciatis et quam ea autem temporibus non voluptatibus qui\nbeatae a earum officia nesciunt dolores suscipit voluptas et\nanimi doloribus cum rerum quas et magni\net hic ut ut commodi expedita sunt"
  },
  {
    "userId": 6,
    "id": 60,
    "title": "consequatur placeat omnis quisquam quia reprehenderit fugit veritatis facere",
    "body": "asperiores sunt ab assumenda cumque modi velit\nqui esse omnis\nvoluptate et fuga perferendis voluptas\nillo ratione amet aut et omnis"
  },
  {
    "userId": 7,
    "id": 61,
    "title": "voluptatem doloribus consectetur est ut ducimus",
    "body": "ab nemo optio odio\ndelectus tenetur corporis similique nobis repellendus rerum omnis facilis\nvero blanditiis debitis in nesciunt doloribus dicta dolores\nmagnam minus velit"
  },
  {
    "userId": 7,
    "id": 62,
    "title": "beatae enim quia vel",
    "body": "enim aspernatur illo distinctio quae praesentium\nbeatae alias amet delectus qui voluptate distinctio\nodit sint accusantium autem omnis\nquo molestiae omnis ea eveniet optio"
  },
  {
```

```
    "userId": 7,
    "id": 63,
    "title": "voluptas blanditiis repellendus animi ducimus error sapiente et suscipit",
    "body": "enim adipisci aspernatur nemo\nnumquam omnis facere dolorem dolor ex quis temporibus incidunt\nab delectus culpa quo reprehenderit blanditiis asperiores\naccusantium ut quam in voluptatibus voluptas ipsam dicta"
  },
  {
    "userId": 7,
    "id": 64,
    "title": "et fugit quas eum in in aperiam quod",
    "body": "id velit blanditiis\neum ea voluptatem\nmolestiae sint occaecati est eos perspiciatis\nincidunt a error provident eaque aut aut qui"
  },
  {
    "userId": 7,
    "id": 65,
    "title": "consequatur id enim sunt et et",
    "body": "voluptatibus ex esse\nsint explicabo est aliquid cumque adipisci fuga repellat labore\nmolestiae corrupti ex saepe at asperiores et perferendis\nnatus id esse incidunt pariatur"
  },
  {
    "userId": 7,
    "id": 66,
    "title": "repudiandae ea animi iusto",
    "body": "officia veritatis tenetur vero qui itaque\nsint non ratione\nsed et ut asperiores iusto eos molestiae nostrum\nveritatis quibusdam et nemo iusto saepe"
  },
  {
    "userId": 7,
    "id": 67,
    "title": "aliquid eos sed fuga est maxime repellendus",
    "body": "reprehenderit id nostrum\nvoluptas doloremque pariatur sint et accusantium quia quod aspernatur\net fugiat amet\nnon sapiente et consequatur necessitatibus molestiae"
  },
  {
    "userId": 7,
    "id": 68,
    "title": "odio quis facere architecto reiciendis optio",
    "body": "magnam molestiae perferendis quisquam\nqui cum reiciendis\nquaerat animi amet hic inventore\nea quia deleniti quidem saepe porro velit"
  },
  {
    "userId": 7,
    "id": 69,
    "title": "fugiat quod pariatur odit minima",
    "body": "officiis error culpa consequatur modi asperiores et\ndolorum assumenda voluptas et vel qui aut vel rerum\nvoluptatum quisquam perspiciatis quia rerum consequatur totam quas\nsequi commodi repudiandae asperiores et saepe a"
  },
  {
```

```
    "userId": 7,
    "id": 70,
    "title": "voluptatem laborum magni",
    "body": "sunt repellendus quae\nest asperiores aut deleniti esse accusamus
repellendus quia aut\nquia dolorem unde\neum tempora esse dolore"
  },
  {
    "userId": 8,
    "id": 71,
    "title": "et iusto veniam et illum aut fuga",
    "body": "occaecati a doloribus\niste saepe consectetur placeat eum voluptate dolorem
et\nqui quo quia voluptas\nrerum ut id enim velit est perferendis"
  },
  {
    "userId": 8,
    "id": 72,
    "title": "sint hic doloribus consequatur eos non id",
    "body": "quam occaecati qui deleniti consectetur\nconsequatur aut facere quas
exercitationem aliquam hic voluptas\nneque id sunt ut aut accusamus\nsunt consectetur
expedita inventore velit"
  },
  {
    "userId": 8,
    "id": 73,
    "title": "consequuntur deleniti eos quia temporibus ab aliquid at",
    "body": "voluptatem cumque tenetur consequatur expedita ipsum nemo quia
explicabo\naut eum minima consequatur\ntempore cumque quae est et\net in consequuntur
voluptatem voluptates aut"
  },
  {
    "userId": 8,
    "id": 74,
    "title": "enim unde ratione doloribus quas enim ut sit sapiente",
    "body": "odit qui et et necessitatibus sint veniam\nmollitia amet doloremque
molestiae commodi similique magnam et quam\nblanditiis est itaque\nquo et tenetur
ratione occaecati molestiae tempora"
  },
  {
    "userId": 8,
    "id": 75,
    "title": "dignissimos eum dolor ut enim et delectus in",
    "body": "commodi non non omnis et voluptas sit\nautem aut nobis magnam et sapiente
voluptatem\net laborum repellat qui delectus facilis temporibus\nrerum amet et nemo
voluptate expedita adipisci error dolorem"
  },
  {
    "userId": 8,
    "id": 76,
    "title": "doloremque officiis ad et non perferendis",
    "body": "ut animi facere\ntotam iusto tempore\nmolestiae eum aut et dolorem
aperiam\nquaerat recusandae totam odio"
  },
  {
```

```
    "userId": 8,
    "id": 77,
    "title": "necessitatibus quasi exercitationem odio",
    "body": "modi ut in nulla repudiandae dolorum nostrum eos\naut consequatur omnis\nut incidunt est omnis iste et quam\nvoluptates sapiente aliquam asperiores nobis amet corrupti repudiandae provident"
  },
  {
    "userId": 8,
    "id": 78,
    "title": "quam voluptatibus rerum veritatis",
    "body": "nobis facilis odit tempore cupiditate quia\nassumenda doloribus rerum qui ea\nillum et qui totam\naut veniam repellendus"
  },
  {
    "userId": 8,
    "id": 79,
    "title": "pariatur consequatur quia magnam autem omnis non amet",
    "body": "libero accusantium et et facere incidunt sit dolorem\nnon excepturi qui quia sed laudantium\nquisquam molestiae ducimus est\nofficiis esse molestiae iste et quos"
  },
  {
    "userId": 8,
    "id": 80,
    "title": "labore in ex et explicabo corporis aut quas",
    "body": "ex quod dolorem ea eum iure qui provident amet\nquia qui facere excepturi et repudiandae\nasperiores molestias provident\nminus incidunt vero fugit rerum sint sunt excepturi provident"
  },
  {
    "userId": 9,
    "id": 81,
    "title": "tempora rem veritatis voluptas quo dolores vero",
    "body": "facere qui nesciunt est voluptatum voluptatem nisi\nsequi eligendi necessitatibus ea at rerum itaque\nharum non ratione velit laboriosam quis consequuntur\nex officiis minima doloremque voluptas ut aut"
  },
  {
    "userId": 9,
    "id": 82,
    "title": "laudantium voluptate suscipit sunt enim enim",
    "body": "ut libero sit aut totam inventore sunt\nporro sint qui sunt molestiae\nconsequatur cupiditate qui iste ducimus adipisci\ndolor enim assumenda soluta laboriosam amet iste delectus hic"
  },
  {
    "userId": 9,
    "id": 83,
    "title": "odit et voluptates doloribus alias odio et",
    "body": "est molestiae facilis quis tempora numquam nihil qui\nvoluptate sapiente consequatur est qui\nnecessitatibus autem aut ipsa aperiam modi dolore numquam\nreprehenderit eius rem quibusdam"
```

```
  },
  {
    "userId": 9,
    "id": 84,
    "title": "optio ipsam molestias necessitatibus occaecati facilis veritatis dolores aut",
    "body": "sint molestiae magni a et quos\neaque et quasi\nut rerum debitis similique veniam\nrecusandae dignissimos dolor incidunt consequatur odio"
  },
  {
    "userId": 9,
    "id": 85,
    "title": "dolore veritatis porro provident adipisci blanditiis et sunt",
    "body": "similique sed nisi voluptas iusto omnis\nmollitia et quo\nassumenda suscipit officia magnam sint sed tempora\nenim provident pariatur praesentium atque animi amet ratione"
  },
  {
    "userId": 9,
    "id": 86,
    "title": "placeat quia et porro iste",
    "body": "quasi excepturi consequatur iste autem temporibus sed molestiae beatae\net quaerat et esse ut\nvoluptatem occaecati et vel explicabo autem\nasperiores pariatur deserunt optio"
  },
  {
    "userId": 9,
    "id": 87,
    "title": "nostrum quis quasi placeat",
    "body": "eos et molestiae\nnesciunt ut a\ndolores perspiciatis repellendus repellat aliquid\nmagnam sint rem ipsum est"
  },
  {
    "userId": 9,
    "id": 88,
    "title": "sapiente omnis fugit eos",
    "body": "consequatur omnis est praesentium\nducimus non iste\neque hic deserunt\nvoluptatibus veniam cum et rerum sed"
  },
  {
    "userId": 9,
    "id": 89,
    "title": "sint soluta et vel magnam aut ut sed qui",
    "body": "repellat aut aperiam totam temporibus autem et\narchitecto magnam ut\nconsequatur qui cupiditate rerum quia soluta dignissimos nihil iure\ntempore quas est"
  },
  {
    "userId": 9,
    "id": 90,
    "title": "ad iusto omnis odit dolor voluptatibus",
```

```
    "body": "minus omnis soluta quia\nqui sed adipisci voluptates illum ipsam
voluptatem\neligendi officia ut in\neos soluta similique molestias praesentium
blanditiis"
  },
  {
    "userId": 10,
    "id": 91,
    "title": "aut amet sed",
    "body": "libero voluptate eveniet aperiam sed\nsunt placeat suscipit
molestias\nsimilique fugit nam natus\nexpedita consequatur consequatur dolores quia eos
et placeat"
  },
  {
    "userId": 10,
    "id": 92,
    "title": "ratione ex tenetur perferendis",
    "body": "aut et excepturi dicta laudantium sint rerum nihil\nlaudantium et at\na
neque minima officia et similique libero et\ncommodi voluptate qui"
  },
  {
    "userId": 10,
    "id": 93,
    "title": "beatae soluta recusandae",
    "body": "dolorem quibusdam ducimus consequuntur dicta aut quo laboriosam\nvoluptatem
quis enim recusandae ut sed sunt\nnostrum est odit totam\nsit error sed sunt eveniet
provident qui nulla"
  },
  {
    "userId": 10,
    "id": 94,
    "title": "qui qui voluptates illo iste minima",
    "body": "aspernatur expedita soluta quo ab ut similique\nexpedita dolores amet\nsed
temporibus distinctio magnam saepe deleniti\nomnis facilis nam ipsum natus sint
similique omnis"
  },
  {
    "userId": 10,
    "id": 95,
    "title": "id minus libero illum nam ad officiis",
    "body": "earum voluptatem facere provident blanditiis velit laboriosam\npariatur
accusamus odio saepe\ncumque dolor qui a dicta ab doloribus consequatur omnis\ncorporis
cupiditate eaque assumenda ad nesciunt"
  },
  {
    "userId": 10,
    "id": 96,
    "title": "quaerat velit veniam amet cupiditate aut numquam ut sequi",
    "body": "in non odio excepturi sint eum\nlabore voluptates vitae quia qui
et\ninventore itaque rerum\nveniam non exercitationem delectus aut"
  },
  {
    "userId": 10,
    "id": 97,
```

```
    "title": "quas fugiat ut perspiciatis vero provident",
    "body": "eum non blanditiis soluta porro quibusdam voluptas\nvel voluptatem qui
placeat dolores qui velit aut\nvel inventore aut cumque culpa explicabo aliquid
at\nperspiciatis est et voluptatem dignissimos dolor itaque sit nam"
  },
  {
    "userId": 10,
    "id": 98,
    "title": "laboriosam dolor voluptates",
    "body": "doloremque ex facilis sit sint culpa\nsoluta assumenda eligendi non ut
eius\nsequi ducimus vel quasi\nveritatis est dolores"
  },
  {
    "userId": 10,
    "id": 99,
    "title": "temporibus sit alias delectus eligendi possimus magni",
    "body": "quo deleniti praesentium dicta non quod\naut est molestias\nmolestias et
officia quis nihil\nitaque dolorem quia"
  },
  {
    "userId": 10,
    "id": 100,
    "title": "at nam consequatur ea labore ea harum",
    "body": "cupiditate quo est a modi nesciunt soluta\nipsa voluptas error itaque dicta
in\nautem qui minus magnam et distinctio eum\naccusamus ratione error aut"
  }
]
<Response [200]>
```


Analizando la respuesta

El métoodo `.request` devuelve un objeto que contiene dos elementos claves (entre otros): el código de la respuesta y el cuerpo *body*.

El código de la respuesta

En caso de haber obtenido exitosamente la respuesta, el código será 200 (OK). Existen varios códigos, pero no es necesario saberlos de memoria, ya que podemos encontrarlos de forma muy rápida en internet.

Según el número con el que se inicia la respuesta, será de distinto tipo

- 1xx: Información
- 2xx: Solicitudes exitosas
- 3xx: Redireccionamiento
- 4xx: Error del cliente (no hizo la request correctamente)

- 5xx: Error del servidor (no puede dar una respuesta)

Algunos de los códigos de respuesta más relevantes son:

- 200: Ok
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found
- 500: Server error

El contenido de la respuesta

La otra parte de la respuesta es el cuerpo, o *body*, el cual contiene la información que nos interesa. Esta información viene como un string, lo que hace muy complicado extraer información, por ejemplo, si solo queremos saber un detalle my específico de la respuesta, o seleccionar elementos específicos de ellas.


```
"[\n  {\n    \"userId\": 1,\n    \"id\": 1,\n    \"title\": \"sunt aut facere repellat
provident occaecati excepturi optio reprehenderit\",\n    \"body\": \"quia et
suscipit\\nsuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae ut
ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto\"\n  },\n  {\n
\"userId\": 1,\n    \"id\": 2,\n    \"title\": \"qui est esse\",\n    \"body\": \"est
rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatae ea dolores
neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui
aperiam non debitis possimus qui neque nisi nulla\"\n  },\n  {\n    \"userId\": 1,\n
\"id\": 3,\n    \"title\": \"ea molestias quasi exercitationem repellat qui ipsa sit
aut\",\n    \"body\": \"et iusto sed quo iure\\nvoluptatem occaecati omnis eligendi aut
ad\\nvoluptatem doloribus vel accusantium quis pariatur\\nmolestiae porro eius odio et
labore et velit aut\"\n  },\n  {\n    \"userId\": 1,\n    \"id\": 4,\n    \"title\":
\"eum et est occaecati\",\n    \"body\": \"ullam et saepe reiciendis voluptatem
adipisci\\nsit amet autem assumenda provident rerum culpa\\nquis hic commodi nesciunt
rem tenetur doloremque ipsam iure\\nquis sunt voluptatem rerum illo velit\"\n  },\n  {\n
   \"userId\": 1,\n    \"id\": 5,\n    \"title\": \"nesciunt quas odio\",\n
```


Sin embargo, el contenido dentro de este string tiene una estructura JSON, lo que hace muy fácil transformar los datos en un diccionario o una lista. A esto se le denomina *parsing*: transformar un input (ya sea un archivo o datos introducidos por el usuario, o la respuesta de una API) en datos que pueden ser fácilmente manipulados.

Transformando la respuesta a JSON

El cuerpo de la respuesta es un string que contiene un JSON. Podemos convertir esto en datos leíbles desde Python usando `json.loads`

```
import json


results = json.loads(response.text)
print(type(results)) # Veremos que el resultado es una lista
print(results[0]) # Mostramos el primer elemento
```

```
<class 'list'>
{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi
optio reprehenderit', 'body': 'quia et suscipit\nsuscipit recusandae consequuntur
expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt
rem eveniet architecto'}
```

Trabajando con la respuesta

El resto del trabajo dependerá de cómo venga estructurada la respuesta. Por ejemplo, en este caso, tenemos una lista donde cada elemento se compone de un diccionario y cada clave de este diccionario es un string.

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio
reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et
cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet
architecto"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea
dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui
aperiam non debitis possimus qui neque nisi nulla"
  }, ...
```

Accediendo a un solo elemento de la respuesta

Dentro del primer elemento, lo que tenemos es un diccionario y podemos accederlo como aprendimos.

```
print(results[0]["userId"])
print(results[0]["title"])
```

```
1
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
```

Iterando los resultados

Podríamos mostrar todos los títulos iterando la lista:

```
for post in results:
    print(post["title"])
```

```
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
qui est esse
ea molestias quasi exercitationem repellat qui ipsa sit aut
eum et est occaecati
nesciunt quas odio
dolorem eum magni eos aperiam quia
magnam facilis autem
dolorem dolore est ipsam
nesciunt iure omnis dolorem tempora et accusantium
optio molestias id quia eum
et ea vero quia laudantium autem
in quibusdam tempore odit est dolorem
dolorum ut in voluptas mollitia et saepe quo animi
voluptatem eligendi optio
eveniet quod temporibus
sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio
fugit voluptas sed molestias voluptatem provident
voluptate et itaque vero tempora molestiae
adipisci placeat illum aut reiciendis qui
doloribus ad provident suscipit at
asperiores ea ipsam voluptatibus modi minima quia sint
dolor sint quo a velit explicabo quia nam
maxime id vitae nihil numquam
autem hic labore sunt dolores incidunt
rem alias distinctio quo quis
est et quae odit qui non
quasi id et eos tenetur aut quo autem
delectus ullam et corporis nulla voluptas sequi
iusto eius quod necessitatibus culpa ea
a quo magni similique perferendis
ullam ut quidem id aut vel consequuntur
```

```
doloremque illum aliquid sunt
qui explicabo molestiae dolorem
magnam ut rerum iure
id nihil consequatur molestias animi provident
fuga nam accusamus voluptas reiciendis itaque
provident vel ut sit ratione est
explicabo et eos deleniti nostrum ab id repellendus
eos dolorem iste accusantium est eaque quam
enim quo cumque
non est facere
commodi ullam sint et excepturi error explicabo praesentium voluptas
eligendi iste nostrum consequuntur adipisci praesentium sit beatae perferendis
optio dolor molestias sit
ut numquam possimus omnis eius suscipit laudantium iure
aut quo modi neque nostrum ducimus
quibusdam cumque rem aut deserunt
ut voluptatem illum ea doloribus itaque eos
laborum non sunt aut ut assumenda perspiciatis voluptas
repellendus qui recusandae incidunt voluptates tenetur qui omnis exercitationem
soluta aliquam aperiam consequatur illo quis voluptas
qui enim et consequuntur quia animi quis voluptate quibusdam
ut quo aut ducimus alias
sit asperiores ipsam eveniet odio non quia
sit vel voluptatem et non libero
qui et at rerum necessitatibus
sed ab est est
voluptatum itaque dolores nisi et quasi
qui commodi dolor at maiores et quis id accusantium
consequatur placeat omnis quisquam quia reprehenderit fugit veritatis facere
voluptatem doloribus consectetur est ut ducimus
beatae enim quia vel
voluptas blanditiis repellendus animi ducimus error sapiente et suscipit
et fugit quas eum in in aperiam quod
consequatur id enim sunt et et
repudiandae ea animi iusto
aliquid eos sed fuga est maxime repellendus
odio quis facere architecto reiciendis optio
fugiat quod pariatur odit minima
voluptatem laborum magni
et iusto veniam et illum aut fuga
sint hic doloribus consequatur eos non id
consequuntur deleniti eos quia temporibus ab aliquid at
enim unde ratione doloribus quas enim ut sit sapiente
dignissimos eum dolor ut enim et delectus in
doloremque officiis ad et non perferendis
necessitatibus quasi exercitationem odio
quam voluptatibus rerum veritatis
pariatur consequatur quia magnam autem omnis non amet
labore in ex et explicabo corporis aut quas
tempora rem veritatis voluptas quo dolores vero
laudantium voluptate suscipit sunt enim enim
odit et voluptates doloribus alias odio et
optio ipsam molestias necessitatibus occaecati facilis veritatis dolores aut
```

```
dolore veritatis porro provident adipisci blanditiis et sunt
placeat quia et porro iste
nostrum quis quasi placeat
sapiente omnis fugit eos
sint soluta et vel magnam aut ut sed qui
ad iusto omnis odit dolor voluptatibus
aut amet sed
ratione ex tenetur perferendis
beatae soluta recusandae
qui qui voluptates illo iste minima
id minus libero illum nam ad officiis
quaerat velit veniam amet cupiditate aut numquam ut sequi
quas fugiat ut perspiciatis vero provident
laboriosam dolor voluptates
temporibus sit alias delectus eligendi possimus magni
at nam consequatur ea labore ea harum
```

Transformar el *request* en una función

Realizaremos múltiples *request* donde solo cambiaremos la URL, por lo que crearemos una función que nos permita replicar fácilmente esto.

```
import json
import requests


def request(requested_url):
    headers = {
        "cache-control": "no-cache",
        "Postman-Token": "2defb9f3-9b11-499b-be4f-82505a2f8e1a",
    }
    response = requests.request("GET", requested_url, headers=headers)
    return json.loads(response.text)
```

Luego para utilizar el método basta con:

```
request('http://jsonplaceholder.typicode.com/posts')
```

```
[{'userId': 1,
  'id': 1,
  'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit',
  'body': 'quia et suscipit\nsuscipit recusandae consequuntur expedita et
cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet
architecto'},
```

```
 {'userId': 1,
  'id': 2,
  'title': 'qui est esse',
  'body': 'est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea
dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui
aperiam non debitis possimus qui neque nisi nulla'},
 {'userId': 1,
  'id': 3,
  'title': 'ea molestias quasi exercitationem repellat qui ipsa sit aut',
  'body': 'et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem
doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit
aut'},
 {'userId': 1,
  'id': 4,
  'title': 'eum et est occaecati',
  'body': 'ullam et saepe reiciendis voluptatem adipisci\nsit amet autem assumenda
provident rerum culpa\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\nquis
sunt voluptatem rerum illo velit'},
 {'userId': 1,
  'id': 5,
  'title': 'nesciunt quas odio',
  'body': 'repudiandae veniam quaerat sunt sed\nalias aut fugiat sit autem sed
est\nvoluptatem omnis possimus esse voluptatibus quis\nest aut tenetur dolor neque'},
 {'userId': 1,
  'id': 6,
  'title': 'dolorem eum magni eos aperiam quia',
  'body': 'ut aspernatur corporis harum nihil quis provident sequi\nmollitia nobis
aliquid molestiae\nperspiciatis et ea nemo ab reprehenderit accusantium quas\nvoluptate
dolores velit et doloremque molestiae'},
 {'userId': 1,
  'id': 7,
  'title': 'magnam facilis autem',
  'body': 'dolore placeat quibusdam ea quo vitae\nmagni quis enim qui quis quo nemo aut
saepe\nquidem repellat excepturi ut quia\nsunt ut sequi eos ea sed quas'},
 {'userId': 1,
  'id': 8,
  'title': 'dolorem dolore est ipsam',
  'body': 'dignissimos aperiam dolorem qui eum\nfacilis quibusdam animi sint suscipit
qui sint possimus cum\nquaerat magni maiores excepturi\nipsam ut commodi dolor
voluptatum modi aut vitae'},
 {'userId': 1,
  'id': 9,
  'title': 'nesciunt iure omnis dolorem tempora et accusantium',
  'body': 'consectetur animi nesciunt iure dolore\nenim quia ad\nveniam autem ut quam
aut nobis\net est aut quod aut provident voluptas autem voluptas'},
 {'userId': 1,
  'id': 10,
  'title': 'optio molestias id quia eum',
  'body': 'quo et expedita modi cum officia vel magni\ndoloribus qui repudiandae\nvero
nisi sit\nquos veniam quod sed accusamus veritatis error'},
 {'userId': 2,
  'id': 11,
  'title': 'et ea vero quia laudantium autem',
```

```
  'body': 'delectus reiciendis molestiae occaecati non minima eveniet qui
voluptatibus\naccusamus in eum beatae sit\nvel qui neque voluptates ut commodi qui
incidunt\nut animi commodi'},
 {'userId': 2,
  'id': 12,
  'title': 'in quibusdam tempore odit est dolorem',
  'body': 'itaque id aut magnam\npraesentium quia et ea odit et ea voluptas et\nsapiente
quia nihil amet occaecati quia id voluptatem\nincidunt ea est distinctio odio'},
 {'userId': 2,
  'id': 13,
  'title': 'dolorum ut in voluptas mollitia et saepe quo animi',
  'body': 'aut dicta possimus sint mollitia voluptas commodi quo doloremque\niste
corrupti reiciendis voluptatem eius rerum\nsit cumque quod eligendi laborum
minima\nperferendis recusandae assumenda consectetur porro architecto ipsum ipsam'},
 {'userId': 2,
  'id': 14,
  'title': 'voluptatem eligendi optio',
  'body': 'fuga et accusamus dolorum perferendis illo voluptas\nnon doloremque neque
facere\nad qui dolorum molestiae beatae\nsed aut voluptas totam sit illum'},
 {'userId': 2,
  'id': 15,
  'title': 'eveniet quod temporibus',
  'body': 'reprehenderit quos placeat\nvelit minima officia dolores impedit repudiandae
molestiae nam\nvoluptas recusandae quis delectus\nofficiis harum fugiat vitae'},
 {'userId': 2,
  'id': 16,
  'title': 'sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio',
  'body': 'suscipit nam nisi quo aperiam aut\nasperiores eos fugit maiores voluptatibus
quia\nvoluptatem quis ullam qui in alias quia est\nconsequatur magni mollitia accusamus
ea nisi voluptate dicta'},
 {'userId': 2,
  'id': 17,
  'title': 'fugit voluptas sed molestias voluptatem provident',
  'body': 'eos voluptas et aut odit natus earum\naspernatur fuga molestiae
ullam\ndeserunt ratione qui eos\nqui nihil ratione nemo velit ut aut id quo'},
 {'userId': 2,
  'id': 18,
  'title': 'voluptate et itaque vero tempora molestiae',
  'body': 'eveniet quo quis\nlaborum totam consequatur non dolor\nut et est
repudiandae\nest voluptatem vel debitis et magnam'},
 {'userId': 2,
  'id': 19,
  'title': 'adipisci placeat illum aut reiciendis qui',
  'body': 'illum quis cupiditate provident sit magnam\nea sed aut omnis\nveniam maiores
ullam consequatur atque\nadipisci quo iste expedita sit quos voluptas'},
 {'userId': 2,
  'id': 20,
  'title': 'doloribus ad provident suscipit at',
  'body': 'qui consequuntur ducimus possimus quisquam amet similique\nsuscipit porro
ipsam amet\neos veritatis officiis exercitationem vel fugit aut necessitatibus
totam\nomnis rerum consequatur expedita quidem cumque explicabo'},
 {'userId': 3,
  'id': 21,
```

```
  'title': 'asperiores ea ipsam voluptatibus modi minima quia sint',
  'body': 'repellat aliquid praesentium dolorem quo\nsed totam minus non itaque\nnihil
labore molestiae sunt dolor eveniet hic recusandae veniam\ntempora et tenetur expedita
sunt'},
 {'userId': 3,
  'id': 22,
  'title': 'dolor sint quo a velit explicabo quia nam',
  'body': 'eos qui et ipsum ipsam suscipit aut\nsed omnis non odio\nexpedita earum
mollitia molestiae aut atque rem suscipit\nnam impedit esse'},
 {'userId': 3,
  'id': 23,
  'title': 'maxime id vitae nihil numquam',
  'body': 'veritatis unde neque eligendi\nquae quod architecto quo neque vitae\nest illo
sit tempora doloremque fugit quod\net et vel beatae sequi ullam sed tenetur
perspiciatis'},
 {'userId': 3,
  'id': 24,
  'title': 'autem hic labore sunt dolores incidunt',
  'body': 'enim et ex nulla\nomnis voluptas quia qui\nvoluptatem consequatur numquam
aliquam sunt\ntotam recusandae id dignissimos aut sed asperiores deserunt'},
 {'userId': 3,
  'id': 25,
  'title': 'rem alias distinctio quo quis',
  'body': 'ullam consequatur ut\nomnis quis sit vel consequuntur\nipsa eligendi ipsum
molestiae et omnis error nostrum\nmolestiae illo tempore quia et distinctio'},
 {'userId': 3,
  'id': 26,
  'title': 'est et quae odit qui non',
  'body': 'similique esse doloribus nihil accusamus\nomnis dolorem fuga consequuntur
reprehenderit fugit recusandae temporibus\nperspiciatis cum ut laudantium\nomnis aut
molestiae vel vero'},
 {'userId': 3,
  'id': 27,
  'title': 'quasi id et eos tenetur aut quo autem',
  'body': 'eum sed dolores ipsam sint possimus debitis occaecati\ndebitis qui qui et\nut
placeat enim earum aut odit facilis\nconsequatur suscipit necessitatibus rerum sed
inventore temporibus consequatur'},
 {'userId': 3,
  'id': 28,
  'title': 'delectus ullam et corporis nulla voluptas sequi',
  'body': 'non et quaerat ex quae ad maiores\nmaiores recusandae totam aut blanditiis
mollitia quas illo\nut voluptatibus voluptatem\nsimilique nostrum eum'},
 {'userId': 3,
  'id': 29,
  'title': 'iusto eius quod necessitatibus culpa ea',
  'body': 'odit magnam ut saepe sed non qui\ntempora atque nihil\naccusamus illum
doloribus illo dolor\neligendi repudiandae odit magni similique sed cum maiores'},
 {'userId': 3,
  'id': 30,
  'title': 'a quo magni similique perferendis',
  'body': 'alias dolor cumque\nimpedit blanditiis non eveniet odio maxime\nblanditiis
amet eius quis tempora quia autem rem\na provident perspiciatis quia'},
 {'userId': 4,
```

```
  'id': 31,
  'title': 'ullam ut quidem id aut vel consequuntur',
  'body': 'debitis eius sed quibusdam non quis consectetur vitae\nimpedit ut qui
consequatur sed aut in\nquidem sit nostrum et maiores adipisci atque\nquaerat voluptatem
adipisci repudiandae'},
 {'userId': 4,
  'id': 32,
  'title': 'doloremque illum aliquid sunt',
  'body': 'deserunt eos nobis asperiores et hic\nest debitis repellat molestiae
optio\nnihil ratione ut eos beatae quibusdam distinctio maiores\nearum voluptates et aut
adipisci ea maiores voluptas maxime'},
 {'userId': 4,
  'id': 33,
  'title': 'qui explicabo molestiae dolorem',
  'body': 'rerum ut et numquam laborum odit est sit\nid qui sint in\nquasi tenetur
tempore aperiam et quaerat qui in\nrerum officiis sequi cumque quod'},
 {'userId': 4,
  'id': 34,
  'title': 'magnam ut rerum iure',
  'body': 'ea velit perferendis earum ut voluptatem voluptate itaque iusto\ntotam
pariatur in\nnemo voluptatem voluptatem autem magni tempora minima in\nest distinctio
qui assumenda accusamus dignissimos officia nesciunt nobis'},
 {'userId': 4,
  'id': 35,
  'title': 'id nihil consequatur molestias animi provident',
  'body': 'nisi error delectus possimus ut eligendi vitae\nplaceat eos harum cupiditate
facilis reprehenderit voluptatem beatae\nmodi ducimus quo illum voluptas eligendi\net
nobis quia fugit'},
 {'userId': 4,
  'id': 36,
  'title': 'fuga nam accusamus voluptas reiciendis itaque',
  'body': 'ad mollitia et omnis minus architecto odit\nvoluptas doloremque maxime aut
non ipsa qui alias veniam\nblanditiis culpa aut quia nihil cumque facere et
occaecati\nqui aspernatur quia eaque ut aperiam inventore'},
 {'userId': 4,
  'id': 37,
  'title': 'provident vel ut sit ratione est',
  'body': 'debitis et eaque non officia sed nesciunt pariatur vel\nvoluptatem iste vero
et ea\nnumquam aut expedita ipsum nulla in\nvoluptates omnis consequatur aut enim
officiis in quam qui'},
 {'userId': 4,
  'id': 38,
  'title': 'explicabo et eos deleniti nostrum ab id repellendus',
  'body': 'animi esse sit aut sit nesciunt assumenda eum voluptas\nquia voluptatibus
provident quia necessitatibus ea\nrerum repudiandae quia voluptatem delectus fugit aut
id quia\nratione optio eos iusto veniam iure'},
 {'userId': 4,
  'id': 39,
  'title': 'eos dolorem iste accusantium est eaque quam',
  'body': 'corporis rerum ducimus vel eum accusantium\nmaxime aspernatur a porro
possimus iste omnis\nest in deleniti asperiores fuga aut\nvoluptas sapiente vel dolore
minus voluptatem incidunt ex'},
 {'userId': 4,
```

```
  'id': 40,
  'title': 'enim quo cumque',
  'body': 'ut voluptatum aliquid illo tenetur nemo sequi quo facilis\nipsum rem optio
mollitia quas\nvoluptatem eum voluptas qui\nunde omnis voluptatem iure quasi maxime
voluptas nam'},
 {'userId': 5,
  'id': 41,
  'title': 'non est facere',
  'body': 'molestias id nostrum\nexcepturi molestiae dolore omnis repellendus quaerat
saepe\nconsectetur iste quaerat tenetur asperiores accusamus ex ut\nnam quidem est
ducimus sunt debitis saepe'},
 {'userId': 5,
  'id': 42,
  'title': 'commodi ullam sint et excepturi error explicabo praesentium voluptas',
  'body': 'odio fugit voluptatum ducimus earum autem est incidunt voluptatem\nodit
reiciendis aliquam sunt sequi nulla dolorem\nnon facere repellendus voluptates
quia\nratione harum vitae ut'},
 {'userId': 5,
  'id': 43,
  'title': 'eligendi iste nostrum consequuntur adipisci praesentium sit beatae
perferendis',
  'body': 'similique fugit est\nillum et dolorum harum et voluptate eaque
quidem\nexercitationem quos nam commodi possimus cum odio nihil nulla\ndolorum
exercitationem magnam ex et a et distinctio debitis'},
 {'userId': 5,
  'id': 44,
  'title': 'optio dolor molestias sit',
  'body': 'temporibus est consectetur dolore\net libero debitis vel velit laboriosam
quia\nipsum quibusdam qui itaque fuga rem aut\nea et iure quam sed maxime ut distinctio
quae'},
 {'userId': 5,
  'id': 45,
  'title': 'ut numquam possimus omnis eius suscipit laudantium iure',
  'body': 'est natus reiciendis nihil possimus aut provident\nex et dolor\nrepellat
pariatur est\nnobis rerum repellendus dolorem autem'},
 {'userId': 5,
  'id': 46,
  'title': 'aut quo modi neque nostrum ducimus',
  'body': 'voluptatem quisquam iste\nvoluptatibus natus officiis facilis dolorem\nquis
quas ipsam\nvel et voluptatum in aliquid'},
 {'userId': 5,
  'id': 47,
  'title': 'quibusdam cumque rem aut deserunt',
  'body': 'voluptatem assumenda ut qui ut cupiditate aut impedit veniam\noccaecati nemo
illum voluptatem laudantium\nmolestiae beatae rerum ea iure soluta nostrum\neligendi et
voluptate'},
 {'userId': 5,
  'id': 48,
  'title': 'ut voluptatem illum ea doloribus itaque eos',
  'body': 'voluptates quo voluptatem facilis iure occaecati\nvel assumenda rerum officia
et\nillum perspiciatis ab deleniti\nlaudantium repellat ad ut et autem reprehenderit'},
 {'userId': 5,
  'id': 49,
```

```
  'title': 'laborum non sunt aut ut assumenda perspiciatis voluptas',
  'body': 'inventore ab sint\nnatus fugit id nulla sequi architecto nihil quaerat\neos
tenetur in in eum veritatis non\nquibusdam officiis aspernatur cumque aut commodi aut'},
 {'userId': 5,
  'id': 50,
  'title': 'repellendus qui recusandae incidunt voluptates tenetur qui omnis
exercitationem',
  'body': 'error suscipit maxime adipisci consequuntur recusandae\nvoluptas eligendi et
est et voluptates\nquia distinctio ab amet quaerat molestiae et vitae\nadipisci impedit
sequi nesciunt quis consectetur'},
 {'userId': 6,
  'id': 51,
  'title': 'soluta aliquam aperiam consequatur illo quis voluptas',
  'body': 'sunt dolores aut doloribus\ndolore doloribus voluptates tempora
et\ndoloremque et quo\ncum asperiores sit consectetur dolorem'},
 {'userId': 6,
  'id': 52,
  'title': 'qui enim et consequuntur quia animi quis voluptate quibusdam',
  'body': 'iusto est quibusdam fuga quas quaerat molestias\na enim ut sit accusamus
enim\ntemporibus iusto accusantium provident architecto\nsoluta esse reprehenderit qui
laborum'},
 {'userId': 6,
  'id': 53,
  'title': 'ut quo aut ducimus alias',
  'body': 'minima harum praesentium eum rerum illo dolore\nquasi exercitationem rerum
nam\nporro quis neque quo\nconsequatur minus dolor quidem veritatis sunt non explicabo
similique'},
 {'userId': 6,
  'id': 54,
  'title': 'sit asperiores ipsam eveniet odio non quia',
  'body': 'totam corporis dignissimos\nvitae dolorem ut occaecati accusamus\nex velit
deserunt\net exercitationem vero incidunt corrupti mollitia'},
 {'userId': 6,
  'id': 55,
  'title': 'sit vel voluptatem et non libero',
  'body': 'debitis excepturi ea perferendis harum libero optio\neos accusamus cum fuga
ut sapiente repudiandae\net ut incidunt omnis molestiae\nnihil ut eum odit'},
 {'userId': 6,
  'id': 56,
  'title': 'qui et at rerum necessitatibus',
  'body': 'aut est omnis dolores\nneque rerum quod ea rerum velit pariatur beatae
excepturi\net provident voluptas corrupti\ncorporis harum reprehenderit dolores
eligendi'},
 {'userId': 6,
  'id': 57,
  'title': 'sed ab est est',
  'body': 'at pariatur consequuntur earum quidem\nquo est laudantium soluta
voluptatem\nqui ullam et est\net cum voluptas voluptatum repellat est'},
 {'userId': 6,
  'id': 58,
  'title': 'voluptatum itaque dolores nisi et quasi',
  'body': 'veniam voluptatum quae adipisci id\net id quia eos ad et dolorem\naliquam quo
nisi sunt eos impedit error\nad similique veniam'},
```

```
 {'userId': 6,
  'id': 59,
  'title': 'qui commodi dolor at maiores et quis id accusantium',
  'body': 'perspiciatis et quam ea autem temporibus non voluptatibus qui\nbeatae a earum officia nesciunt dolores suscipit voluptas et\nanimi doloribus cum rerum quas et magni\net hic ut ut commodi expedita sunt'},
 {'userId': 6,
  'id': 60,
  'title': 'consequatur placeat omnis quisquam quia reprehenderit fugit veritatis facere',
  'body': 'asperiores sunt ab assumenda cumque modi velit\nqui esse omnis\nvoluptate et fuga perferendis voluptas\nillo ratione amet aut et omnis'},
 {'userId': 7,
  'id': 61,
  'title': 'voluptatem doloribus consectetur est ut ducimus',
  'body': 'ab nemo optio odio\ndelectus tenetur corporis similique nobis repellendus rerum omnis facilis\nvero blanditiis debitis in nesciunt doloribus dicta dolores\nmagnam minus velit'},
 {'userId': 7,
  'id': 62,
  'title': 'beatae enim quia vel',
  'body': 'enim aspernatur illo distinctio quae praesentium\nbeatae alias amet delectus qui voluptate distinctio\nodit sint accusantium autem omnis\nquo molestiae omnis ea eveniet optio'},
 {'userId': 7,
  'id': 63,
  'title': 'voluptas blanditiis repellendus animi ducimus error sapiente et suscipit',
  'body': 'enim adipisci aspernatur nemo\nnumquam omnis facere dolorem dolor ex quis temporibus incidunt\nab delectus culpa quo reprehenderit blanditiis asperiores\naccusantium ut quam in voluptatibus voluptas ipsam dicta'},
 {'userId': 7,
  'id': 64,
  'title': 'et fugit quas eum in in aperiam quod',
  'body': 'id velit blanditiis\neum ea voluptatem\nmolestiae sint occaecati est eos perspiciatis\nincidunt a error provident eaque aut aut qui'},
 {'userId': 7,
  'id': 65,
  'title': 'consequatur id enim sunt et et',
  'body': 'voluptatibus ex esse\nsint explicabo est aliquid cumque adipisci fuga repellat labore\nmolestiae corrupti ex saepe at asperiores et perferendis\nnatus id esse incidunt pariatur'},
 {'userId': 7,
  'id': 66,
  'title': 'repudiandae ea animi iusto',
  'body': 'officia veritatis tenetur vero qui itaque\nsint non ratione\nsed et ut asperiores iusto eos molestiae nostrum\nveritatis quibusdam et nemo iusto saepe'},
 {'userId': 7,
  'id': 67,
  'title': 'aliquid eos sed fuga est maxime repellendus',
  'body': 'reprehenderit id nostrum\nvoluptas doloremque pariatur sint et accusantium quia quod aspernatur\net fugiat amet\nnon sapiente et consequatur necessitatibus molestiae'},
 {'userId': 7,
```

```
  'id': 68,
  'title': 'odio quis facere architecto reiciendis optio',
  'body': 'magnam molestiae perferendis quisquam\nqui cum reiciendis\nquaerat animi amet
hic inventore\nea quia deleniti quidem saepe porro velit'},
 {'userId': 7,
  'id': 69,
  'title': 'fugiat quod pariatur odit minima',
  'body': 'officiis error culpa consequatur modi asperiores et\ndolorum assumenda
voluptas et vel qui aut vel rerum\nvoluptatum quisquam perspiciatis quia rerum
consequatur totam quas\nsequi commodi repudiandae asperiores et saepe a'},
 {'userId': 7,
  'id': 70,
  'title': 'voluptatem laborum magni',
  'body': 'sunt repellendus quae\nest asperiores aut deleniti esse accusamus repellendus
quia aut\nquia dolorem unde\neum tempora esse dolore'},
 {'userId': 8,
  'id': 71,
  'title': 'et iusto veniam et illum aut fuga',
  'body': 'occaecati a doloribus\niste saepe consectetur placeat eum voluptate dolorem
et\nqui quo quia voluptas\nrerum ut id enim velit est perferendis'},
 {'userId': 8,
  'id': 72,
  'title': 'sint hic doloribus consequatur eos non id',
  'body': 'quam occaecati qui deleniti consectetur\nconsequatur aut facere quas
exercitationem aliquam hic voluptas\nneque id sunt ut aut accusamus\nsunt consectetur
expedita inventore velit'},
 {'userId': 8,
  'id': 73,
  'title': 'consequuntur deleniti eos quia temporibus ab aliquid at',
  'body': 'voluptatem cumque tenetur consequatur expedita ipsum nemo quia explicabo\naut
eum minima consequatur\ntempore cumque quae est et\net in consequuntur voluptatem
voluptates aut'},
 {'userId': 8,
  'id': 74,
  'title': 'enim unde ratione doloribus quas enim ut sit sapiente',
  'body': 'odit qui et et necessitatibus sint veniam\nmollitia amet doloremque molestiae
commodi similique magnam et quam\nblanditiis est itaque\nquo et tenetur ratione
occaecati molestiae tempora'},
 {'userId': 8,
  'id': 75,
  'title': 'dignissimos eum dolor ut enim et delectus in',
  'body': 'commodi non non omnis et voluptas sit\nautem aut nobis magnam et sapiente
voluptatem\net laborum repellat qui delectus facilis temporibus\nrerum amet et nemo
voluptate expedita adipisci error dolorem'},
 {'userId': 8,
  'id': 76,
  'title': 'doloremque officiis ad et non perferendis',
  'body': 'ut animi facere\ntotam iusto tempore\nmolestiae eum aut et dolorem
aperiam\nquaerat recusandae totam odio'},
 {'userId': 8,
  'id': 77,
  'title': 'necessitatibus quasi exercitationem odio',
```

```
  'body': 'modi ut in nulla repudiandae dolorum nostrum eos\naut consequatur omnis\nut
incidunt est omnis iste et quam\nvoluptates sapiente aliquam asperiores nobis amet
corrupti repudiandae provident'},
 {'userId': 8,
  'id': 78,
  'title': 'quam voluptatibus rerum veritatis',
  'body': 'nobis facilis odit tempore cupiditate quia\nassumenda doloribus rerum qui
ea\nillum et qui totam\naut veniam repellendus'},
 {'userId': 8,
  'id': 79,
  'title': 'pariatur consequatur quia magnam autem omnis non amet',
  'body': 'libero accusantium et et facere incidunt sit dolorem\nnon excepturi qui quia
sed laudantium\nquisquam molestiae ducimus est\nofficiis esse molestiae iste et quos'},
 {'userId': 8,
  'id': 80,
  'title': 'labore in ex et explicabo corporis aut quas',
  'body': 'ex quod dolorem ea eum iure qui provident amet\nquia qui facere excepturi et
repudiandae\nasperiores molestias provident\nminus incidunt vero fugit rerum sint sunt
excepturi provident'},
 {'userId': 9,
  'id': 81,
  'title': 'tempora rem veritatis voluptas quo dolores vero',
  'body': 'facere qui nesciunt est voluptatum voluptatem nisi\nsequi eligendi
necessitatibus ea at rerum itaque\nharum non ratione velit laboriosam quis
consequuntur\nex officiis minima doloremque voluptas ut aut'},
 {'userId': 9,
  'id': 82,
  'title': 'laudantium voluptate suscipit sunt enim enim',
  'body': 'ut libero sit aut totam inventore sunt\nporro sint qui sunt
molestiae\nconsequatur cupiditate qui iste ducimus adipisci\ndolor enim assumenda soluta
laboriosam amet iste delectus hic'},
 {'userId': 9,
  'id': 83,
  'title': 'odit et voluptates doloribus alias odio et',
  'body': 'est molestiae facilis quis tempora numquam nihil qui\nvoluptate sapiente
consequatur est qui\nnecessitatibus autem aut ipsa aperiam modi dolore
numquam\nreprehenderit eius rem quibusdam'},
 {'userId': 9,
  'id': 84,
  'title': 'optio ipsam molestias necessitatibus occaecati facilis veritatis dolores
aut',
  'body': 'sint molestiae magni a et quos\neaque et quasi\nut rerum debitis similique
veniam\nrecusandae dignissimos dolor incidunt consequatur odio'},
 {'userId': 9,
  'id': 85,
  'title': 'dolore veritatis porro provident adipisci blanditiis et sunt',
  'body': 'similique sed nisi voluptas iusto omnis\nmollitia et quo\nassumenda suscipit
officia magnam sint sed tempora\nenim provident pariatur praesentium atque animi amet
ratione'},
 {'userId': 9,
  'id': 86,
  'title': 'placeat quia et porro iste',
```

```
  'body': 'quasi excepturi consequatur iste autem temporibus sed molestiae beatae\net quaerat et esse ut\nvoluptatem occaecati et vel explicabo autem\nasperiores pariatur deserunt optio'},
 {'userId': 9,
  'id': 87,
  'title': 'nostrum quis quasi placeat',
  'body': 'eos et molestiae\nnesciunt ut a\ndolores perspiciatis repellendus repellat aliquid\nmagnam sint rem ipsum est'},
 {'userId': 9,
  'id': 88,
  'title': 'sapiente omnis fugit eos',
  'body': 'consequatur omnis est praesentium\nducimus non iste\nneque hic deserunt\nvoluptatibus veniam cum et rerum sed'},
 {'userId': 9,
  'id': 89,
  'title': 'sint soluta et vel magnam aut ut sed qui',
  'body': 'repellat aut aperiam totam temporibus autem et\narchitecto magnam ut\nconsequatur qui cupiditate rerum quia soluta dignissimos nihil iure\ntempore quas est'},
 {'userId': 9,
  'id': 90,
  'title': 'ad iusto omnis odit dolor voluptatibus',
  'body': 'minus omnis soluta quia\nqui sed adipisci voluptates illum ipsam voluptatem\neligendi officia ut in\neos soluta similique molestias praesentium blanditiis'},
 {'userId': 10,
  'id': 91,
  'title': 'aut amet sed',
  'body': 'libero voluptate eveniet aperiam sed\nsunt placeat suscipit molestias\nsimilique fugit nam natus\nexpedita consequatur consequatur dolores quia eos et placeat'},
 {'userId': 10,
  'id': 92,
  'title': 'ratione ex tenetur perferendis',
  'body': 'aut et excepturi dicta laudantium sint rerum nihil\nlaudantium et at\na neque minima officia et similique libero et\ncommodi voluptate qui'},
 {'userId': 10,
  'id': 93,
  'title': 'beatae soluta recusandae',
  'body': 'dolorem quibusdam ducimus consequuntur dicta aut quo laboriosam\nvoluptatem quis enim recusandae ut sed sunt\nnostrum est odit totam\nsit error sed sunt eveniet provident qui nulla'},
 {'userId': 10,
  'id': 94,
  'title': 'qui qui voluptates illo iste minima',
  'body': 'aspernatur expedita soluta quo ab ut similique\nexpedita dolores amet\nsed temporibus distinctio magnam saepe deleniti\nomnis facilis nam ipsum natus sint similique omnis'},
 {'userId': 10,
  'id': 95,
  'title': 'id minus libero illum nam ad officiis',
```

```
  'body': 'earum voluptatem facere provident blanditiis velit laboriosam\npariatur
accusamus odio saepe\ncumque dolor qui a dicta ab doloribus consequatur omnis\ncorporis
cupiditate eaque assumenda ad nesciunt'},
 {'userId': 10,
  'id': 96,
  'title': 'quaerat velit veniam amet cupiditate aut numquam ut sequi',
  'body': 'in non odio excepturi sint eum\nlabore voluptates vitae quia qui
et\ninventore itaque rerum\nveniam non exercitationem delectus aut'},
 {'userId': 10,
  'id': 97,
  'title': 'quas fugiat ut perspiciatis vero provident',
  'body': 'eum non blanditiis soluta porro quibusdam voluptas\nvel voluptatem qui
placeat dolores qui velit aut\nvel inventore aut cumque culpa explicabo aliquid
at\nperspiciatis est et voluptatem dignissimos dolor itaque sit nam'},
 {'userId': 10,
  'id': 98,
  'title': 'laboriosam dolor voluptates',
  'body': 'doloremque ex facilis sit sint culpa\nsoluta assumenda eligendi non ut
eius\nsequi ducimus vel quasi\nveritatis est dolores'},
 {'userId': 10,
  'id': 99,
  'title': 'temporibus sit alias delectus eligendi possimus magni',
  'body': 'quo deleniti praesentium dicta non quod\naut est molestias\nmolestias et
officia quis nihil\nitaque dolorem quia'},
 {'userId': 10,
  'id': 100,
  'title': 'at nam consequatur ea labore ea harum',
  'body': 'cupiditate quo est a modi nesciunt soluta\nipsa voluptas error itaque dicta
in\nautem qui minus magnam et distinctio eum\naccusamus ratione error aut'}]
```

Tener este código como método nos permite realizar *requests* a otras direcciones con una sola línea de código.

Capítulo 7: API REST

Objetivos

- Entender la estructura tras una API REST
- Leer la documentación de una API REST
- Realizar *requests* a una API REST

Introducción

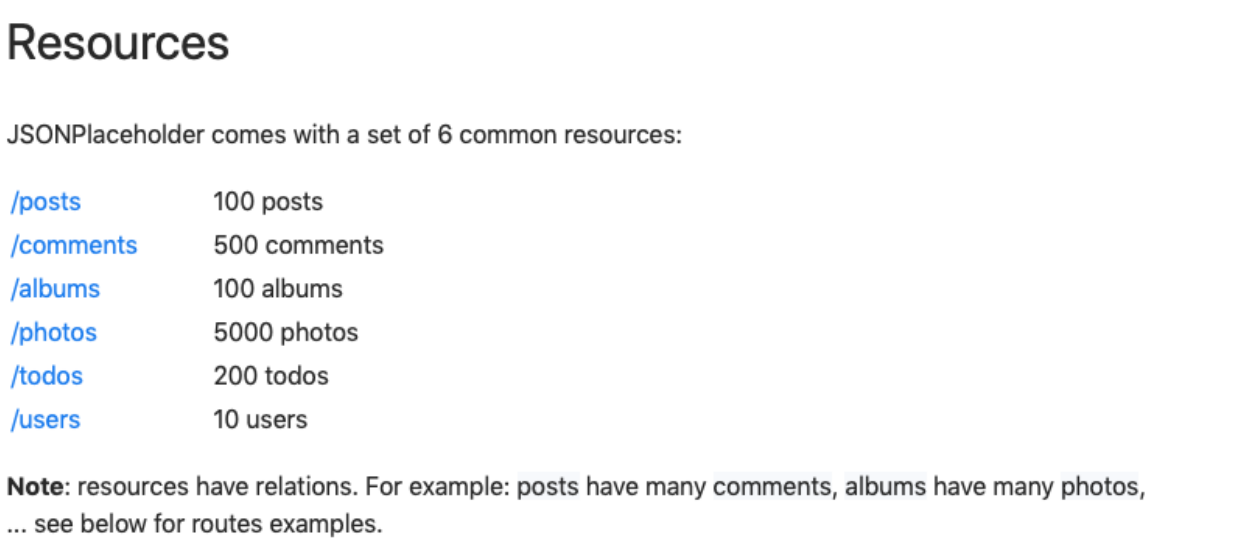
Las API REST son un estándar sobre cómo crear APIs, en base a recursos y acciones. Cada acción se realiza sobre un recurso en específico, y para esto tiene una ruta asignada.

Para saber cuáles recursos y rutas tenemos disponibles debemos leer la documentación de la API.

Leyendo la documentación

La documentación en cada página es distinta, pero particularmente lo que se necesita de cada una es saber qué recursos se tiene disponibles y cuáles son las rutas a esos recursos.

Los recursos



Resources

JSONPlaceholder comes with a set of 6 common resources:

/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	200 todos
/users	10 users

Note: resources have relations. For example: posts have many comments, albums have many photos, ... see below for routes examples.

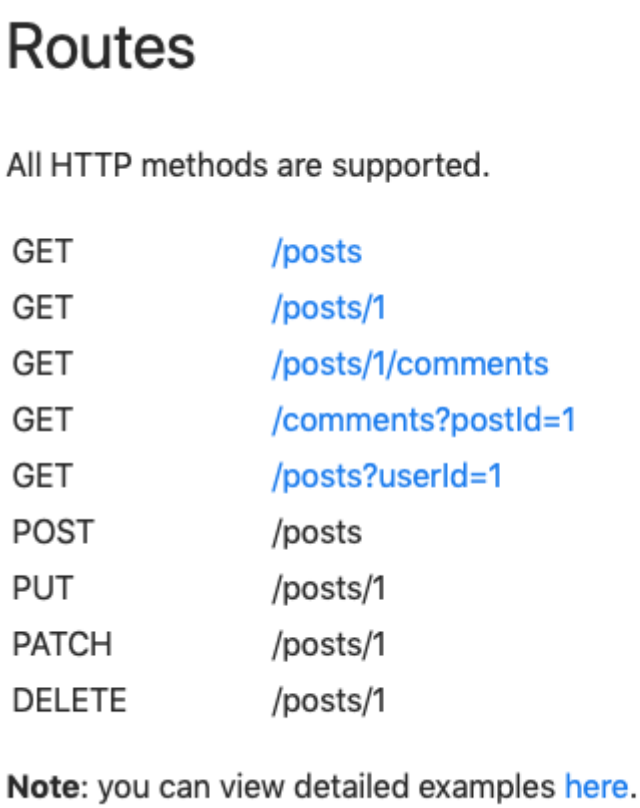
¿Qué son los recursos?

Los recursos son elementos que se puede obtener y mostrar, y además, en algunos casos, crear nuevos, actualizarlos y borrarlos.

Por ejemplo, en esta API tenemos los recursos "posts", "comentarios", "álbumes", "fotos", "listas de tareas (todos)" y "usuarios". Además hay relaciones entre estos recursos.

Para ver qué acciones se puede hacer en específico sobre estos recursos, se necesita conocer las rutas (o direcciones).

Las rutas



Routes

All HTTP methods are supported.

GET	/posts
GET	/posts/1
GET	/posts/1/comments
GET	/comments?postId=1
GET	/posts?userId=1
POST	/posts
PUT	/posts/1
PATCH	/posts/1
DELETE	/posts/1

Note: you can view detailed examples here.

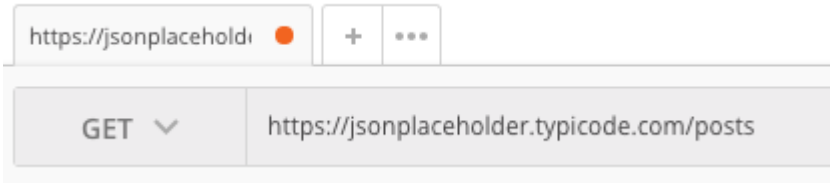
Las palabras que aparecen antes de cada ruta se conocen como métodos HTTP o verbos, y se debe usar el verbo especificado para cada ruta.

Request = URL + HTTP *method*

Se vio anteriormente que las rutas de una API tienen un **HTTP method**, también conocido como **verbo**, asociado.

Un *request* depende tanto de la dirección como del verbo. En el *request* anterior no lo mencionamos pero sí utilizamos uno: utilizamos el verbo GET.

Esto lo podemos ver en Postman:



También el verbo aparece especificado en el código.

```
response = requests.request("GET", url, headers=headers)
```

Por ahora seguiremos ocupando el verbo **GET**, pero haremos un *request* a otro recurso. Obtendremos las fotos.

Utilizaremos nuestro método `request` con otra URL.

```
data = request('https://jsonplaceholder.typicode.com/photos')[0:10] # Tomemos solo 10
data
```

```
[{'albumId': 1,
  'id': 1,
  'title': 'accusamus beatae ad facilis cum similique qui sunt',
  'url': 'https://via.placeholder.com/600/92c952',
  'thumbnailUrl': 'https://via.placeholder.com/150/92c952'},
 {'albumId': 1,
  'id': 2,
  'title': 'reprehenderit est deserunt velit ipsam',
  'url': 'https://via.placeholder.com/600/771796',
  'thumbnailUrl': 'https://via.placeholder.com/150/771796'},
 {'albumId': 1,
  'id': 3,
  'title': 'officia porro iure quia iusto qui ipsa ut modi',
  'url': 'https://via.placeholder.com/600/24f355',
  'thumbnailUrl': 'https://via.placeholder.com/150/24f355'},
 {'albumId': 1,
  'id': 4,
  'title': 'culpa odio esse rerum omnis laboriosam voluptate repudiandae',
  'url': 'https://via.placeholder.com/600/d32776',
  'thumbnailUrl': 'https://via.placeholder.com/150/d32776'},
 {'albumId': 1,
```

```
 'id': 5,
 'title': 'natus nisi omnis corporis facere molestiae rerum in',
 'url': 'https://via.placeholder.com/600/f66b97',
 'thumbnailUrl': 'https://via.placeholder.com/150/f66b97'},
{'albumId': 1,
 'id': 6,
 'title': 'accusamus ea aliquid et amet sequi nemo',
 'url': 'https://via.placeholder.com/600/56a8c2',
 'thumbnailUrl': 'https://via.placeholder.com/150/56a8c2'},
{'albumId': 1,
 'id': 7,
 'title': 'officia delectus consequatur vero aut veniam explicabo molestias',
 'url': 'https://via.placeholder.com/600/b0f7cc',
 'thumbnailUrl': 'https://via.placeholder.com/150/b0f7cc'},
{'albumId': 1,
 'id': 8,
 'title': 'aut porro officiis laborum odit ea laudantium corporis',
 'url': 'https://via.placeholder.com/600/54176f',
 'thumbnailUrl': 'https://via.placeholder.com/150/54176f'},
{'albumId': 1,
 'id': 9,
 'title': 'qui eius qui autem sed',
 'url': 'https://via.placeholder.com/600/51aa97',
 'thumbnailUrl': 'https://via.placeholder.com/150/51aa97'},
{'albumId': 1,
 'id': 10,
 'title': 'beatae et provident et ut vel',
 'url': 'https://via.placeholder.com/600/810b14',
 'thumbnailUrl': 'https://via.placeholder.com/150/810b14'}]
```

Analizando la respuesta

La estructura de la respuesta es la misma: una lista que contiene diccionarios donde cada uno representa una foto. El proceso de obtener resultados también es el mismo; iteramos la lista, y de cada diccionario obtenemos la información que necesitamos.

Como ejercicio de integración, vamos a obtener todas las direcciones de las fotos y generar una página web con una galería.

Desafío: Construir una página web a partir de las fotos

Primero idearemos el algoritmo. El secreto para resolver problemas es descomponerlos en partes que ya sabemos hacer.

1. Realizamos un request a `https://jsonplaceholder.typicode.com/photos` y obtenemos la lista con diccionarios de fotos.

2. Generamos una lista a partir de las fotos del diccionario.

3. Iteramos la lista con las fotos y guardamos sus resultados en un archivo.

   Al momento de guardar agregaremos las etiquetas de HTML.

```
# Solución
data = request("https://jsonplaceholder.typicode.com/photos")[0:10]

html = ""
for photo in data:
    html += "<img src=\"{}\">\n".format(photo["url"])

# Este recurso nos permite crear un archivo y escribir contenido en él
with open("output.html", "w") as f:
    f.write(html)
```

Se deja como tarea propuesta el agregar la información para generar de forma automática una página web válida (Doctype, y etiquetas "head" y "body").

Capítulo 8: Más allá de GET

Objetivos

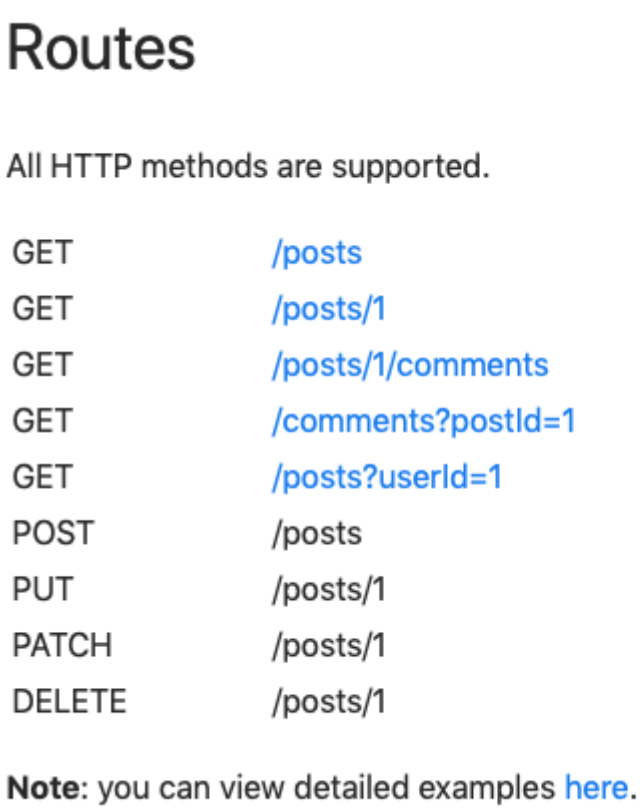
- Utilizar los verbos POST, PUT y DELETE
- Conocer el concepto de negociación de contenido
- Utilizar *headers* para negociar contenido
- Utilizar el body de un *request* para enviar contenido
- Crear un nuevo objeto uitlizando el método POST

Introducción

Las APIs no solo permiten obtener información, sino que también permiten subir (crear), actualizar y borrar información. En una API REST, estos pedidos suceden bajo otros verbos diferentes al de **GET**.

Creando un recurso

En una API REST hay un estandar definido, que es el método **Post**.



Routes

All HTTP methods are supported.

GET	/posts
GET	/posts/1
GET	/posts/1/comments
GET	/comments?postId=1
GET	/posts?userId=1
POST	/posts
PUT	/posts/1
PATCH	/posts/1
DELETE	/posts/1

Note: you can view detailed examples here.

Esta parte de la documentación no dice mucho, pero hay un link a ejemplos. Al seguir el link:

Creating a resource

```
// POST adds a random id to the object sent
fetch('https://jsonplaceholder.typicode.com/posts', {
    method: 'POST',
    body: JSON.stringify({
      title: 'foo',
      body: 'bar',
      userId: 1
    }),
    headers: {
      "Content-type": "application/json; charset=UTF-8"
    }
  })
  .then(response => response.json())
  .then(json => console.log(json))

/* will return
{
  id: 101,
  title: 'foo',
  body: 'bar',
  userId: 1
}
*/
```

El ejemplo de la documentación está en JavaScript, pero de todas formas podemos leerlo poniendo atención en algunos elementos específicos.

Estudiemos los elementos clave:

- El *request* indica el método POST
- El *request* indica un body con un userId, un body y un title
- El *request* indica además *headers* donde hay un content-type

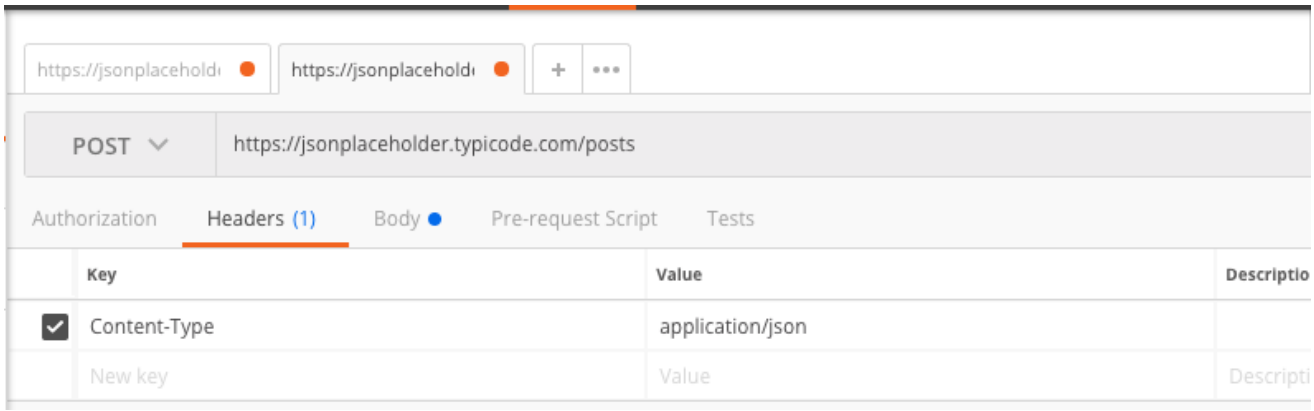
Utilicemos esto para agregar un artículo nuevo a la API

¿Qué son el *header* y el *body*?

Un request a una API REST tiene un *header* y un *body*. En el *header* especificamos información general, por ejemplo, codificación, o tipo de contenido que vamos a enviar. En el body enviaremos información específica, por ejemplo, si queremos subir un artículo, aquí agregaremos el título y el contenido.

Creando un recurso desde Postman

Para subir un artículo nuevo necesitamos seleccionar el verbo POST, e ingresar la URL de la documentación.



Negociación de contenido

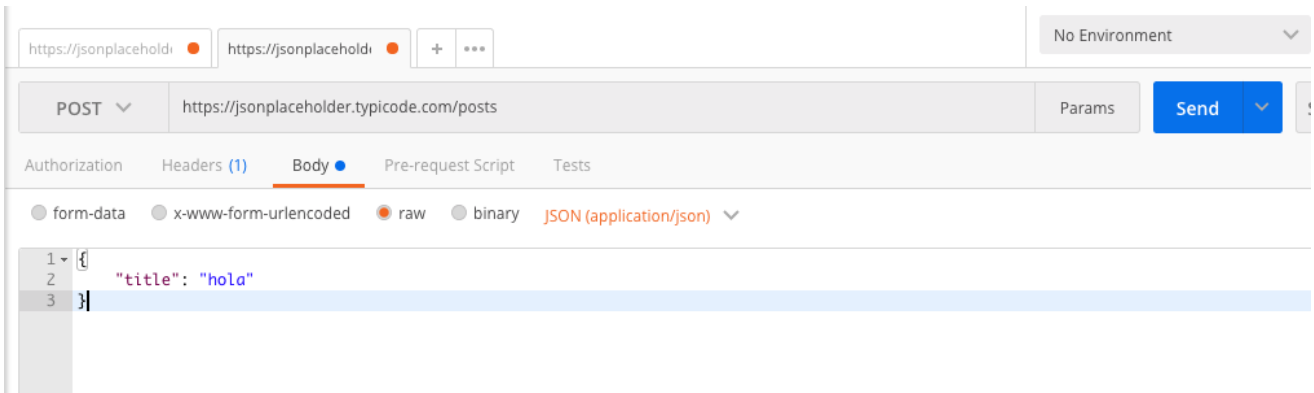
Además, necesitaremos agregar el header `Content-type: application/json`. Esto es necesario porque algunas APIs pueden recibir distintos tipos de contenido.

Negociar el contenido permite que, en un mismo endpoint, se puedan resolver pedidos de distintos clientes en distintos formatos. Los tipos de contenido que puede manejar una API se especifican en la documentación, en este caso, nosotros lo obtuvimos del ejemplo.

Agregando el contenido

Dentro de la API y del ejemplo, vimos que un artículo se compone de un título **title**, de un cuerpo **body** y de un dueño **userId**

Como primera prueba, subiremos un artículo con título, para lo que iremos al tab de body, seleccionaremos la opción **raw** y a la derecha marcaremos la opción JSON (application/json)



Aquí es muy importante indicar que se debe utilizar doble comilla alrededor de un string en un JSON para contenerlo correctamente.

Observando el resultado

En la sección inferior, bajo el tag body, encontraremos el resultado obtenido de la API indicando que se creó el artículo con id 101 y título "hola".

Intentemos guardar un nuevo artículo, ahora con `body` y `userId`

```
{
    "title": "Post 101",
    "body": "Este es nuestro primer post",
    "userId": 1
}
```

Obtendremos como respuesta:

```
{
    "title": "Post 101",
    "body": "Este es nuestro primer post",
    "userId": 1,
    "id": 101
}
```

Esta API en particular no es persistente

Para asegurar el funcionamiento con muchos usuarios distintos, esta API no guarda los cambios realizados, solo dice que los hizo. Trabajar con una API que los guarda es exactamente igual.

Creando un recurso desde Python

Para lograr esto, copiaremos el código que genera Postman.

```
import requests

url = "https://jsonplaceholder.typicode.com/posts"

payload = "{\r\n\t\"title\": \"Post 101\",\r\n\t\"body\": \"Este es nuestro primer
post\",\r\n\t\"userId\": 1\r\n}"
headers = {
    'Content-Type': "application/json",
    'cache-control': "no-cache",
    'Postman-Token': "ac158b42-ca0e-4ada-92f0-f362962775e7"
    }

response = requests.request("POST", url, data=payload, headers=headers)
```

```
print(response.text)
```

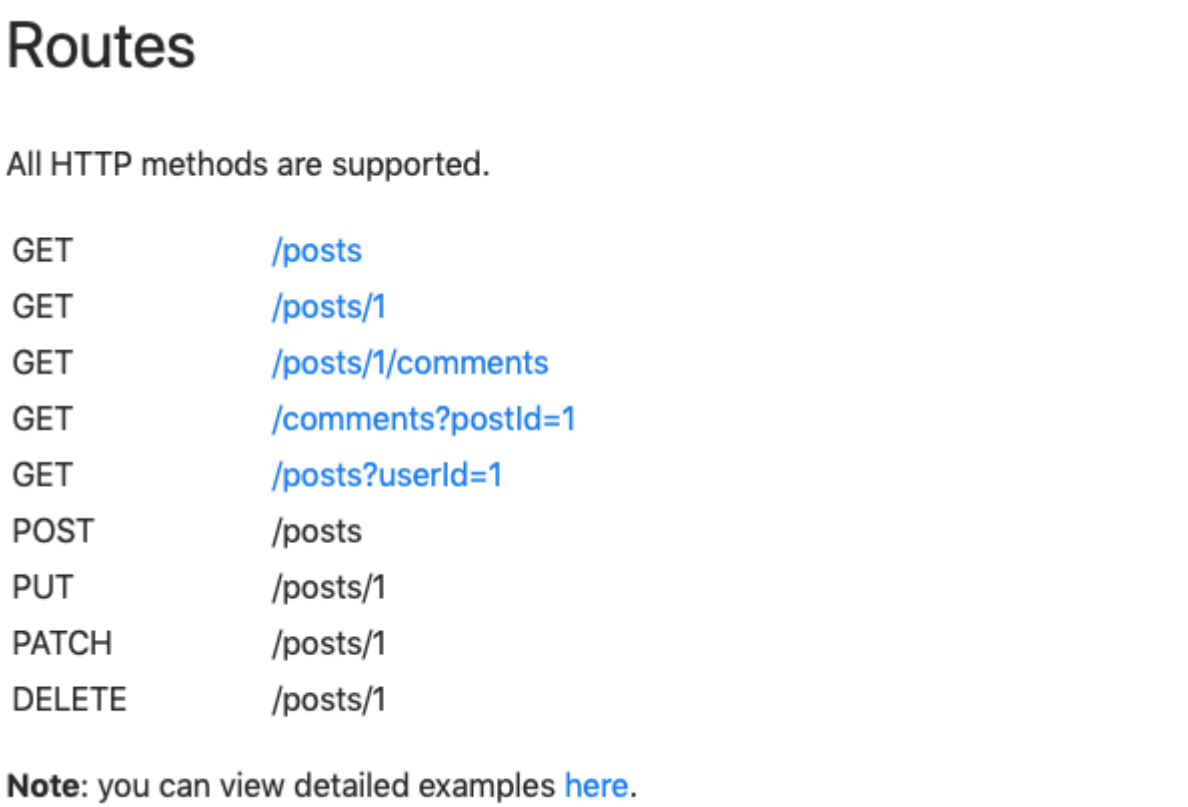
```
{
  "title": "Post 101",
  "body": "Este es nuestro primer post",
  "userId": 1,
  "id": 101
}
```

Actualizando un recurso

Los métodos para actualizar un recurso en una API REST son PUT o PATCH. Si bien existe una diferencia entre estos, la mayoría de las API no hace distinción.

Existen muchas APIs que para actualizar un recurso además aceptan el verbo POST. E incluso, algunas solo aceptan POST, y no PUT.

Lo otro que tenemos que saber además del verbo es el endpoint (o dirección). Para esto veremos de nuevo la documentación.



Routes

All HTTP methods are supported.

GET	/posts
GET	/posts/1
GET	/posts/1/comments
GET	/comments?postId=1
GET	/posts?userId=1
POST	/posts
PUT	/posts/1
PATCH	/posts/1
DELETE	/posts/1

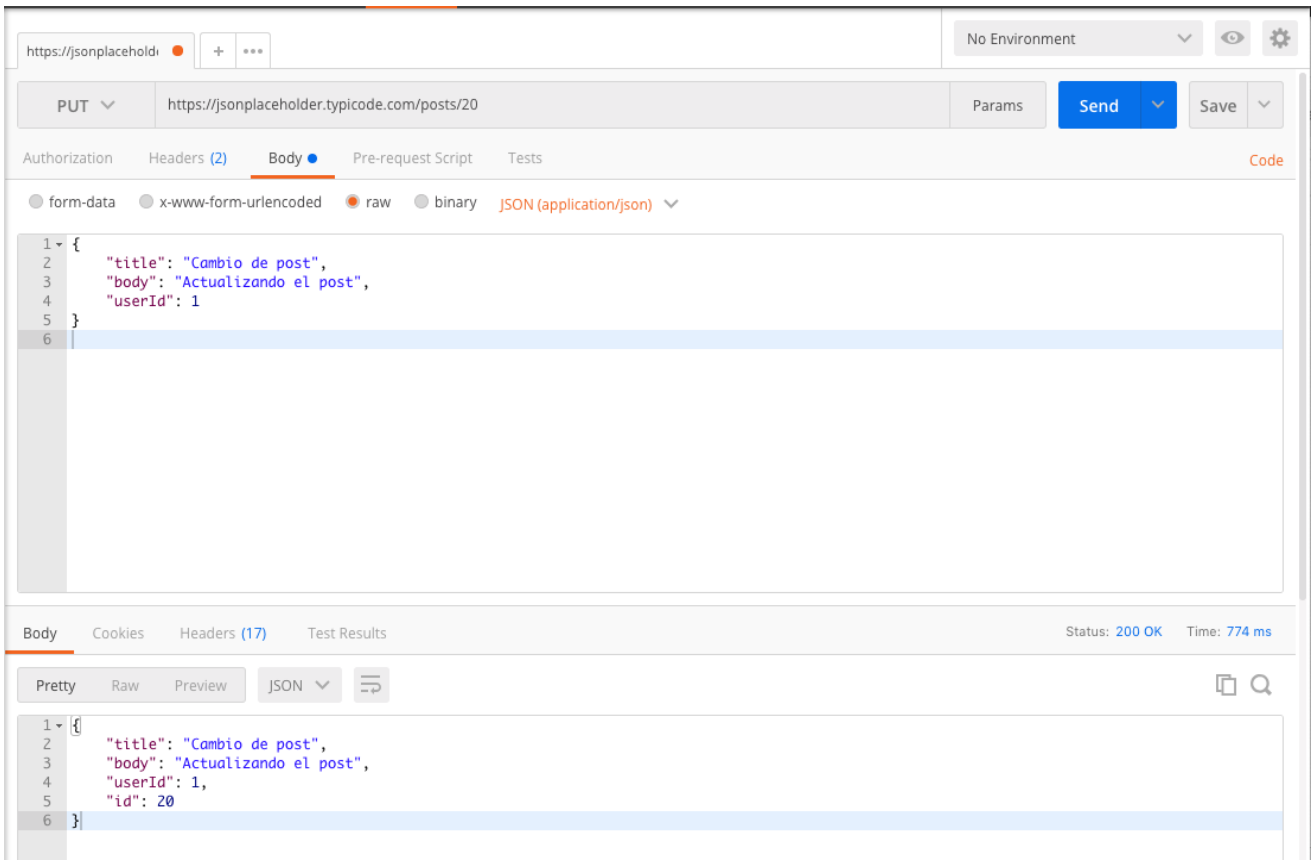
Note: you can view detailed examples here.

Actualizando un recurso desde Postman

Vemos en la documentación que, a diferencia de subir un artículo, que utiliza la ruta `/posts`, la actualización utiliza la ruta `/posts/1` donde el número no tiene que ser necesariamente 1, si no que es identificador del recurso que queremos acutalizar. Para saber cuáles son, tenemos que listar los recursos (como lo hicimos al momento de aprender a ocupar las APIs)

En este caso, los identificadores van desde 1 hasta 100, por lo tanto, si queremos cambiar el artículo con id 20 tendríamos que hacer un request con método put a `https://jsonplaceholder.typicode.com/posts/20`

Además, dentro del body de la request debemos que agregar los nuevos valores para el artículo.



Actualizando un artículo desde Python

Para estudiar cómo podemos modificar el artículo utilizando Python, lo que haremos es copiar el código generado por Postman y veremos que la idea es exactamente la misma.

```
import requests

url = "https://jsonplaceholder.typicode.com/posts/1"

payload = "{\n\t\"title\": \"Cambio de post\",\n\t\"body\": \"Actualizando el
post\",\n\t\"userId\": 1\n}\n"
headers = {
    'Content-Type': "application/json",
    'cache-control': "no-cache",
    'Postman-Token': "563a09ea-fdff-4a54-be88-7d8ec46bd01c"
    }

response = requests.request("PUT", url, data=payload, headers=headers)

print(response.text)
```

Borrar un recurso

Para borrar un artículo seguiremos la misma idea. Veremos de la documentación que el método es DELETE y que la dirección es `posts/id`.

Borrando un artículo desde Python

Nuevamente, utilizaremos el código generado por Postman

```
import requests

url = "https://jsonplaceholder.typicode.com/posts/1"

payload = ""
headers = {
    'Content-Type': "application/json",
    'cache-control': "no-cache",
    'Postman-Token': "fc3777ed-499a-4a9b-8250-43c1b1c8b6c3"
    }

response = requests.request("DELETE", url, data=payload, headers=headers)

print(response.text)
```

Resumen del capítulo:

En este capítulo aprendimos:

- Una API REST es un estandar para ver, crear, actualizar y borrar recursos.
- Para ver recursos utilizaremos el verbo (o http method) **GET**
- Para agregar un recurso nuevo utilizaremos el verbo **POST**
- Para actualizar un recurso utilizaremos **PUT** o **PATCH**
- Para borrar un recurso utilizaremos **DELETE**

Desafío: Integración con respuesta

Como ejercicio final vamos a consultar los últimos precios de bitcoin y generar una lista de fechas. Para esto ocuparemos los datos de cierre diario de la API de CoinDesk. Estos se encuentran en la siguiente URL

`https://api.coindesk.com/v1/bpi/historical/close.json`

Se pide obtener los precios y fechas del último periodo, y a partir de estos, obtener una lista de todas las fechas donde el valor ha sido menor a 5000 USD.

Solución

Primero dividiremos el problema en 2 grandes partes.

- Hacer el request a la API de CoinDesk y obtener los resultados
- Procesar los resultados para obtener una lista con las fechas

```
import json
import requests


def request(requested_url):
    headers = {
        "cache-control": "no-cache",
        "Postman-Token": "2defb9f3-9b11-499b-be4f-82505a2f8e1a",
    }
    response = requests.request("GET", requested_url, headers=headers)
    return json.loads(response.text)


prices = request("https://api.coindesk.com/v1/bpi/historical/close.json")["bpi"]
```

El segundo paso consiste en obtener la lista con las fechas, para lo que hay distintas posibilidades. La primera pregunta que deberíamos hacernos es si podemos utilizar una sola línea para seleccionar elementos en base a un valor.

```
# Solución 1

selected_data = [k for k, v in prices.items() if v < 5000]
selected_data
```

```
['2018-11-19',
 '2018-11-20',
 '2018-11-21',
 '2018-11-22',
 '2018-11-23',
 '2018-11-24',
 '2018-11-25',
 '2018-11-26',
 '2018-11-27',
 '2018-11-28',
 '2018-11-29',
 '2018-11-30',
 '2018-12-01',
 '2018-12-02',
 '2018-12-03',
 '2018-12-04',
 '2018-12-05',
 '2018-12-06',
 '2018-12-07',
 '2018-12-08',
 '2018-12-09',
 '2018-12-10',
 '2018-12-11']
```

Sin embargo hay muchas otras formas de resolver el problema, por ejemplo pudimos haberlo hecho utilizando un `for` y guardando en un nuevo diccionario todos los elementos que cumplan el criterio.

```
# Solución 2

selected_data = []
for date, value in prices.items():
    if value < 5000:
        selected_data.append(date)
selected_data
```

```
['2018-11-19',
 '2018-11-20',
 '2018-11-21',
 '2018-11-22',
```

```
 '2018-11-23',
 '2018-11-24',
 '2018-11-25',
 '2018-11-26',
 '2018-11-27',
 '2018-11-28',
 '2018-11-29',
 '2018-11-30',
 '2018-12-01',
 '2018-12-02',
 '2018-12-03',
 '2018-12-04',
 '2018-12-05',
 '2018-12-06',
 '2018-12-07',
 '2018-12-08',
 '2018-12-09',
 '2018-12-10',
 '2018-12-11']
```

Otra opción, bastante distinta, sería obtener una lista con únicamente los precios, filtrarla por los valores bajo 5000 y luego utilizar un diccionario invertido para buscarlos.

```
# Solución 3

under_5000 = [v for v in prices.values() if v < 5000]
prices_inv = {v: k for k, v in prices.items()}
selected_data = [prices_inv[k] for k in under_5000]
selected_data
```

```
['2018-11-19',
 '2018-11-20',
 '2018-11-21',
 '2018-11-22',
 '2018-11-23',
 '2018-11-24',
 '2018-11-25',
 '2018-11-26',
 '2018-11-27',
 '2018-11-28',
 '2018-11-29',
 '2018-11-30',
 '2018-12-01',
 '2018-12-02',
 '2018-12-03',
 '2018-12-04',
 '2018-12-05',
```

```
  '2018-12-06',
  '2018-12-07',
  '2018-12-08',
  '2018-12-09',
  '2018-12-10',
  '2018-12-11']
```

Existen muchas soluciones a un problema siempre deberíamos preferir las mas sencillas.

Capítulo 9: SSL

Objetivos

- Conocer las implicaciones de SSL.
- Conocer la importancia de **no** enviar información sensible sin SSL.
- Entender la importancia de verificar el certificado.

Introducción

SSL es un acrónimo de Secure Sockets Layer. Es una capa de seguridad que establece encriptación entre el navegador y el servidor, y evita que la información que enviamos o recibimos sea leída por un tercero.

Esto sucede ya sea conectándonos a una página web por el navegador o una API.

Cómo funciona la encriptación por SSL

Para entender el funcionamiento de la encriptación, hay que tener presente que existen dos claves: una para cifrar y otra descifrar. Explicaremos esto con un ejemplo.

Supongamos por un momento que la comunicación es entre dos personas, Alice y Bob. Para cifrar el mensaje, Alice tiene una clave y para descifrarlo tiene otra. Previo a comunicarse, Alice se junta con Bob y le traspasa la clave para descifrar.

Esto permite que Alice cifre su mensaje con su clave, le envía el mensaje a Bob, y Bob la descifra con la clave que la pasó previamente Alice. Esto tiene dos ventajas; Primero, el mensaje pasa cifrado por lo que nadie mas puede leerlo a menos que tenga la clave para descifrar. Pero además, la llave para descifrar solo sirve para descifrar mensajes de Alice, por lo que sabremos que el mensaje viene realmente de Alice y no de otra persona.

Clave pública, clave privada

Las dos claves son distintas; Una clave es para cifrar el mensaje, y la otra para descifrarlo. Por eso se dice que este sistema de cifrado es asimétrico.

La clave que firma el mensaje, pero jamás se comparte, recibe el nombre de clave privada, mientras que la clave que se comparte recibe el nombre de clave pública.

Ventajas de SSL

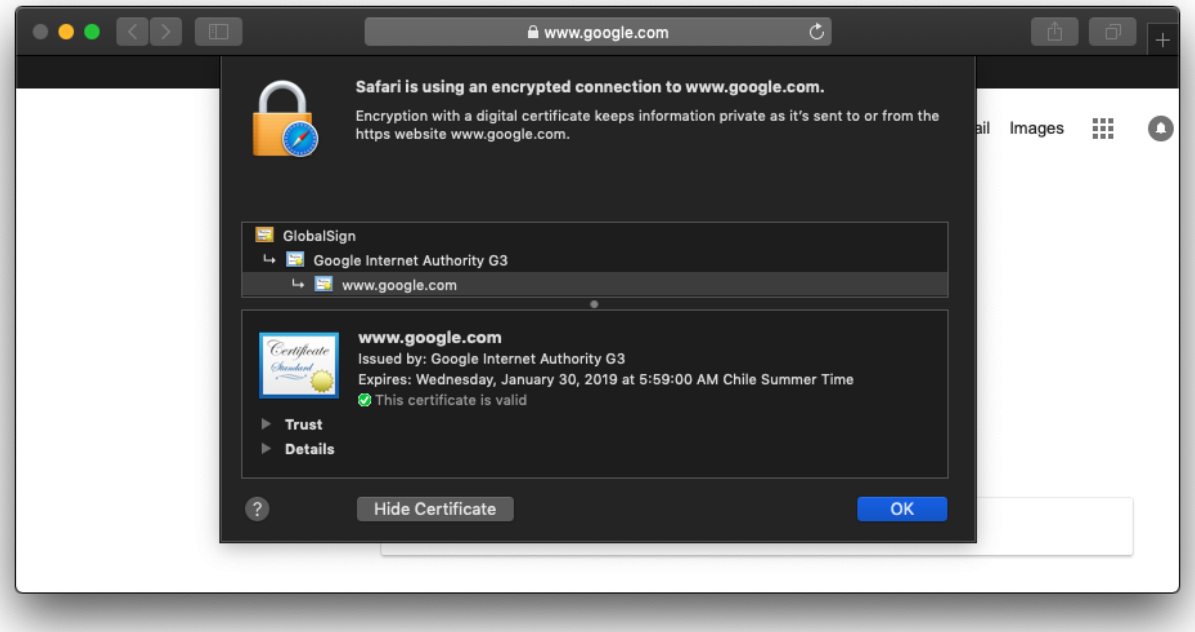
El sistema de juego entonces tiene dos ventajas.

- Cifra el mensaje impidiendo que terceros puedan leerlo.
- Asegura que el emisor es quien dice ser, porque si alguien mas cifró el mensaje con una llave distinao, el mensaje no tendrá sentido al desifrarlo.

La magia que sucede detrás

Al momento de conectarnos a un sitio web que utilice SSL con un navegador, se establece un acuerdo, llamado en inglés **handshake**, de forma automática. En este handshake el servidor envía un certificado que tiene el nombre del sitio y la clave pública al cliente.

Si nos conectamos con el navegador y hacemos click en el candado de la conexión segura podemos ver el certificado.



El cliente ocupa la clave pública que viene en el certificado para cifrar un mensaje y se la devuelve al servidor. Si el servidor puede descifrar el nuevo mensaje significa que el servidor tiene la clave privada correcta.

Dentro del proceso, cliente y servidor crearon una clave secreta especial utilizando los números que se enviaron durante el handshake. Durante el resto del proceso, se ocupa un sistema de cifrado simétrico utilizando esta única clave que nunca fue transmitida pero que ambos servidores tienen.

Conectandose a SSL

Postman identifica automáticamente si un *request* ocupa HTTPS y genera el código para conectarnos. La librería `requests`, por defecto, tiene la verificación de certificados SSL habilitada, por lo que si no puede verificar un certificado durante un *request*, ocurrirá un `SSLError`.

Hombre en el medio

Hombre en el medio (**Man in the middle**) es una estrategia de hacking que consiste en hacer de puente (**Proxy**) entre el cliente y el servidor.

El atacante se sitúa en el punto medio, captura la clave pública que envía el servidor y le envía una nueva clave pública al cliente. El cliente cree que se está conectando con el servidor directamente pero no es así y toda la comunicación queda comprometida.

Este ataque es muy fácil de lograr, pero solo si el certificado no es validado (o auto firmado). Al momento de conectarse, los certificados se revisan contra agentes certificadores. Un certificado validado (nombre de sitio + clave pública) complica la situación al atacante puesto que este difícilmente tendrá la clave privada asociada al certificado.

Capítulo 10: API con autenticación

Objetivos

- Obtener el id y la clave para autenticarse a una API
- Conectarse a una API que requiere autenticación
- Utilizar la API del diccionario de oxford
- Estudiar otros tipos de autenticación

Autenticación

Muchas APIs requieren de autenticación para poder acceder a sus servicios.

Para autenticarse con una API, nuestro primer paso consiste en conseguir una clave para conectarse. Las mismas APIs disponen de servicio de registro y al completarlo entregan un id.

Desafío: Consiguiendo la clave de la API de Oxford Dictionary

Para conseguir esta clave entraremos a https://developer.oxforddictionaries.com y llenaremos el formulario de registro (versión gratuita). También necesitaremos confirmar nuestro email.

En algunos casos se utilizan dos claves, un app_id y un app_key, en otros casos un key y un secret. Independiente de cuantas claves el sistema siempre es similar.

Una vez obtenido el key (o los keys) podremos acceder a la API

Autenticando desde Python

Existen distintos tipos de autenticación, la mayoría de los modelos consiste en pasar un token en los **header** del request.

```
import json
import requests


def request(requested_url):
    headers = {
        "app_id": "e51457d3",
        "app_key": "6cf09f1d0edbefd2381f937f396150a3",
    }
    response = requests.request("GET", requested_url, headers=headers)
    return json.loads(response.text)
```

```
word = "test"
request("https://od-api.oxforddictionaries.com:443/api/v1/entries/en/{}".format(word))
```

```
{'metadata': {'provider': 'Oxford University Press'},
 'results': [{'id': 'test',
   'language': 'en',
   'lexicalEntries': [{'derivatives': [{'id': 'testability',
       'text': 'testability'},
      {'id': 'testee', 'text': 'testee'}],
     'entries': [{'etymologies': ['late Middle English (denoting a cupel used to treat
gold or silver alloys or ore): via Old French from Latin testu, testum ‘earthen pot’,
variant of testa ‘jug, shell’. Compare with test. The verb dates from the early 17th
century'],
       'grammaticalFeatures': [{'text': 'Singular', 'type': 'Number'}],
       'homographNumber': '100',
       'senses': [{'definitions': ['a procedure intended to establish the quality,
performance, or reliability of something, especially before it is taken into widespread
use'],
         'examples': [{'text': 'four fax modems are on test'},
          {'text': 'both countries carried out nuclear tests in May'}],
         'id': 'm_en_gbus1042280.006',
         'short_definitions': ['procedure intended to establish quality, performance, or
reliability of something'],
         'subsenses': [{'definitions': ["a short written or spoken examination of a
person's proficiency or knowledge"],
           'examples': [{'text': 'a spelling test'}],
           'id': 'm_en_gbus1042280.009',
           'short_definitions': ["short written or spoken examination of person's
proficiency or knowledge"],
           'thesaurusLinks': [{'entry_id': 'test',
             'sense_id': 't_en_gb0014747.002'}]},
          {'definitions': ['an event or situation that reveals the strength or quality
of someone or something by putting them under strain'],
           'examples': [{'text': 'this is the first serious test of the peace
agreement'}],
           'id': 'm_en_gbus1042280.010',
           'short_definitions': ['event or situation that reveals strength or quality of
person or thing by putting them under strain']},
          {'definitions': ['an examination of part of the body or a body fluid for
medical purposes, especially by means of a chemical or mechanical procedure rather than
simple inspection'],
           'domains': ['Medicine'],
           'examples': [{'text': 'eye tests'},
            {'text': 'researchers developed a test for the virus'}],
           'id': 'm_en_gbus1042280.011',
           'short_definitions': ['examination of part of body or body fluid for medical
purposes']},
          {'definitions': ['a procedure employed to identify a substance or to reveal
the presence or absence of a constituent within a substance.'],
```

```
           'domains': ['Chemistry'],
           'id': 'm_en_gbus1042280.012',
           'short_definitions': ['procedure employed to identify substance or to reveal
presence or absence of constituent within substance']},
          {'definitions': ['the result of a medical examination or analytical
procedure'],
           'domains': ['Medicine'],
           'examples': [{'text': 'a positive test for protein'}],
           'id': 'm_en_gbus1042280.013',
           'short_definitions': ['result of medical examination or analytical
procedure']},
          {'definitions': ['a means of establishing whether an action, item, or
situation is an instance of a specified quality, especially one held to be
undesirable'],
           'examples': [{'text': 'a statutory test of obscenity'}],
           'id': 'm_en_gbus1042280.014',
           'short_definitions': ['means of establishing whether action, item, or
situation is instance of specified quality'],
           'thesaurusLinks': [{'entry_id': 'test',
             'sense_id': 't_en_gb0014747.003'}]}],
         'thesaurusLinks': [{'entry_id': 'test',
           'sense_id': 't_en_gb0014747.001'}]},
        {'crossReferenceMarkers': ['short for Test match'],
         'crossReferences': [{'id': 'test_match',
           'text': 'Test match',
           'type': 'abbreviation'}],
         'examples': [{'text': 'the first Test against New Zealand'}],
         'id': 'm_en_gbus1042280.016',
         'variantForms': [{'text': 'Test'}]},
        {'definitions': ['a movable hearth in a reverberating furnace, used for
separating gold or silver from lead.'],
         'domains': ['Metallurgy'],
         'id': 'm_en_gbus1042280.021',
         'short_definitions': ['movable hearth in reverberating furnace, used for
separating gold or silver from lead']}]},
      {'etymologies': ['mid 19th century: from Latin testa ‘tile, jug, shell’. Compare
with test'],
       'grammaticalFeatures': [{'text': 'Singular', 'type': 'Number'}],
       'homographNumber': '200',
       'senses': [{'definitions': ['the shell or integument of some invertebrates and
protozoans, especially the chalky shell of a foraminiferan or the tough outer layer of a
tunicate.'],
         'domains': ['Zoology'],
         'id': 'm_en_gbus1042290.005',
         'short_definitions': ['shell or integument of some invertebrates and
protozoans']}]}],
     'language': 'en',
     'lexicalCategory': 'Noun',
     'pronunciations': [{'audioFile':
'http://audio.oxforddictionaries.com/en/mp3/test_gb_1.mp3',
       'dialects': ['British English'],
       'phoneticNotation': 'IPA',
       'phoneticSpelling': 'tɛst'}],
```

```
     'text': 'test'},
    {'derivatives': [{'id': 'testee', 'text': 'testee'},
      {'id': 'testability', 'text': 'testability'}],
     'entries': [{'grammaticalFeatures': [{'text': 'Transitive',
         'type': 'Subcategorization'},
        {'text': 'Present', 'type': 'Tense'}],
       'homographNumber': '101',
       'senses': [{'definitions': ['take measures to check the quality, performance, or
reliability of (something), especially before putting it into widespread use or
practice'],
         'domains': ['Metallurgy'],
         'examples': [{'text': 'this range has not been tested on animals'},
          {'text': 'several trial runs were carried out to test the special brakes'}],
         'id': 'm_en_gbus1042280.023',
         'short_definitions': ['take measures to check quality, performance, or
reliability of something'],
         'subsenses': [{'definitions': ['give (someone) a short written or oral
examination of their proficiency or knowledge'],
           'examples': [{'text': 'all children are tested at eleven'}],
           'id': 'm_en_gbus1042280.029',
           'short_definitions': ['give someone short written or oral examination of
their proficiency or knowledge']},
          {'definitions': ["judge or measure (someone's proficiency or knowledge) by
means of a test"],
           'examples': [{'text': 'the exam will test accuracy and neatness'}],
           'id': 'm_en_gbus1042280.030',
           'short_definitions': ["judge or measure someone's proficiency or knowledge by
means of test"]},
          {'definitions': ['reveal the strengths or capabilities of (someone or
something) by putting them under strain'],
           'examples': [{'text': 'such behaviour would severely test any marriage'}],
           'id': 'm_en_gbus1042280.031',
           'short_definitions': ['reveal strengths or capabilities of person or thing by
putting them under strain'],
           'thesaurusLinks': [{'entry_id': 'testing',
             'sense_id': 't_en_gb0014753.001'},
            {'entry_id': 'test', 'sense_id': 't_en_gb0014747.005'}]},
          {'definitions': ['carry out a medical test on (a person, a part of the body,
or a body fluid)'],
           'domains': ['Medicine'],
           'examples': [{'text': "he's been tested for drugs"}],
           'id': 'm_en_gbus1042280.032',
           'short_definitions': ['carry out medical test on']},
          {'definitions': ['produce a specified result in a medical test, especially a
drugs test or AIDS test'],
           'domains': ['Medicine'],
           'examples': [{'text': 'he tested positive for steroids during the race'}],
           'id': 'm_en_gbus1042280.033',
           'notes': [{'text': 'no object, with complement',
             'type': 'grammaticalNote'}],
           'short_definitions': ['produce specified result in medical test']},
          {'definitions': ['examine (a substance) by means of a reagent.'],
           'domains': ['Chemistry'],
```

```
            'id': 'm_en_gbus1042280.034',
            'short_definitions': ['examine substance by means of reagent']},
           {'definitions': ['touch or taste (something) to check that it is acceptable
before proceeding further'],
            'examples': [{'text': 'she tested the water with the tip of her elbow'}],
            'id': 'm_en_gbus1042280.035',
            'short_definitions': ['touch or taste something to check that it is
acceptable before proceeding further']}],
          'thesaurusLinks': [{'entry_id': 'test',
            'sense_id': 't_en_gb0014747.004'}]}]}],
     'language': 'en',
     'lexicalCategory': 'Verb',
     'pronunciations': [{'audioFile':
'http://audio.oxforddictionaries.com/en/mp3/test_gb_1.mp3',
       'dialects': ['British English'],
       'phoneticNotation': 'IPA',
       'phoneticSpelling': 'tɛst'}],
     'text': 'test'}],
   'type': 'headword',
   'word': 'test'}]}
```

Dentro del resultado veremos muchos datos:

```
{"metadata"=>{"provider"=>"Oxford University Press"},
 "results"=>
  [{"id"=>"test",
    "language"=>"en",
    "lexicalEntries"=>
     [{"derivatives"=>
        [{"id"=>"testability", "text"=>"testability"},
         {"id"=>"testee", "text"=>"testee"}],
       "entries"=>
        [{"etymologies"=>
           ["late Middle English (denoting a cupel used to treat gold or silver alloys or
ore): via Old French from Latin testu, testum ‘earthen pot’, variant of testa ‘jug, shell
’. Compare with test. The verb dates from the early 17th century"],
          "grammaticalFeatures"=>[{"text"=>"Singular", "type"=>"Number"}],
          "homographNumber"=>"100",
          "senses"=>
           [{"definitions"=>
              ["a procedure intended to establish the quality, performance, or reliabilit
y of something, especially before it is taken into widespread use"],
             "examples"=>
              [{"text"=>"four fax modems are on test"},
```

La pregunta importante es qué información es la que queremos. Por ahora nos conformaremos con la definición **definitions**, así que iremos del final hasta el principio.

- Definitions es el key de un diccionario `['definitions']`
- El diccionario está dentro de una lista, es el primer elemento `[0]['definitions']`
- Esta lista es parte de un diccionario bajo la clave **senses** `['senses'][0]['definitions']`
- Senses es un key de entries `['entries']['senses'][0]['definitions']`

- El diccionario está dentro de una lista y es el primer elemento `[0]['entries']['senses'][0]['definitions']`
- La lista está asociada al key lexicalEntries `['lexicalEntries'][0]['entries']['senses'][0]['definitions']`
- El diccionario es el primer elemento de una lista `[0]['lexicalEntries'][0]['entries']['senses'][0]['definitions']`
- Todo está bajo la clave de results `['results'][0]['lexicalEntries'][0]['entries']['senses'][0]['definitions']`

```
result = request("https://od-api.oxforddictionaries.com:443/api/v1/entries/en/test")
result['results'][0]['lexicalEntries'][0]['entries'][0]['senses'][0]['definitions']
```

```
['a procedure intended to establish the quality, performance, or reliability of
something, especially before it is taken into widespread use']
```

No siempre es tan complicado, muchas veces los datos están con uno o dos niveles de profundidad, pero es un muy buen caso de estudio.

Otros tipos de autenticación

En este caso nos tocó ingresar la autenticación en el header. Esta es una de las opciones más frecuentes, pero algunas APIs tienen diferentes modelos de autenticación.

- En algunos casos nos tocará ingresar la clave en el body.
- En algunos casos, el token de autenticación cambiará después de cada request y tendremos que usar el último valor para rescatar el siguiente.

También existen otros modelos más complejos como el de Facebook y el de Twitter que al día de hoy ocupan OAuth2.

De todas formas, lo importante siempre es leer con calma la documentación y encontrar las pistas necesarias para lograr la conexión. En el peor de los casos, siempre existe la opción de comunicarse con los desarrolladores de la API para pedir ayuda o buscar otra API que logre el mismo propósito.

Precauciones sobre los token

Lo último y más importante, es que muchos servicios tienen límites de consumo y otros manejan información delicada. Los tokens, contraseñas y otros datos delicados entregados no deben ser subidos a repositorios públicos o ser compartidos con terceros.

¿Cómo podemos manejar los tokens?

Con `args`

Hay varias estrategias. Podemos utilizarar una muy simple que consiste en dejar los datos sensibles fuera del programa y entregarle los datos como parámetros (utilizando los `args`) al cargar el programa.

Con variables de entorno

Esta opción consiste en utilizar variables que se definen en el terminal y las podemos llamar desde nuestro programa.

Veamos un ejemplo básico (para Linux y OSX). En el terminal escribiremos `export a=5`. Luego entraremos a Python e importamos `os.environ` (`from os import environ`). Escribimos `environ["a"]` y observaremos el valor 5.

Persistiendo las variables de entorno

Siguiendo con Linux y OSX, este valor solo quedará en la sesión, es decir, al abrir un terminal nuevo, no lo tendremos. Para hacerlo persistente podemos escribir esta línea en el archivo `.bashrc` o `.bash_profile`. Este archivo que se encuentra en la carpeta de usuario. Lo podemos abrir con un editor de texto y en la última línea agregar nuestra variable.

```
export API_1_SECRET_KEY=23123u2139183
```

Finalmente, después de guardar, para recargar los cambios realizados dentro del archivo ocuparemos la siguiente línea:

```
source ~/.bash_profile
```

Si queremos probar si funcionó podemos hacerlo desde Python con `print(environ["API_1_SECRET_KEY"])`, o directamente desde el terminal con `echo $API_1_SECRET_KEY`. El signo peso es necesario en el echo para indicar que es una variable.

Cierre

A lo largo de este módulo aprendimos las bases de la programación, que consiste en la solución de problemas.

Revisemos algunos elementos claves que aprendimos.

- Un algoritmo es una serie de pasos finitos para resolver un problema
- Crear el algoritmo es la parte compleja de la programación
- Para crear algoritmos disponemos de herramientas como los diagramas de flujo y pseudocódigo
- Las funciones son importantes para ordenar el código y evitar repetir varias veces lo mismo
- Los ciclos son clave para evitar repetir código y hacer el programa escalable
- Un problema puede tener más de una solución, y cuál será la mejor dependerá de cada caso
- Las estructuras de datos como listas y diccionarios nos permiten almacenar y manejar grandes cantidades de datos en una sola variable
- Podemos comunicarnos con otros programas de forma segura mediante el uso de una API, y usar estos datos para nuestros propios programas

Palabas finales

Aprender a programar es un largo camino y requiere de mucha práctica. Hemos dado los primeros pasos a lo largo de estas cuatro unidades, donde estudiamos desde conceptos muy simples hasta protocolos de seguridad informática actuales. Sin embargo, todavía nos queda mucho por recorrer. Este es solo el punto de partida de la programación y existen muchas ramas interesantes donde podemos profundizar. Algunas interesantes para tenerlas en el radar.

- Estructuras de datos
- Complejidad algorítmica
- Almacenamiento y recuperación de información
- Bases de datos
- QA y Construcción de pruebas automatizadas
- Arquitectura de software
- Ciencia de datos
- Inteligencia artificial

Los campos son muy diversos y aunque están parcialmente entrelazados cada uno es un mundo de profunidad y la vida es muy corta para ser experto en todo.