API _

Diccionarios

Diferencias entre listas y diccionarios

Lista	Diccionario
• Se accede a los elementos por medio de la posición o índice. • Los índices se definen implícitamente. • Los índices siempre son int.	• Se accede a los elementos por medio de la clave. • Las claves se definen explícitamente. • Las claves suelen ser string.

```
lista = [25, 31, “hola”]
lista[2] # “hola”

diccionario = {“a”: 25, “b”:31, “c”: “hola”}
diccionario[“c”] # “hola”
```

Desafíos

Iteración, transformación y filtrado

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.

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

# Se crea diccionario vacío para el filtrado
diccionario_caros = {}

# Se itera por clave y valor utilizando items()
for key, value in diccionario_productos.items():

    if value < 120000:
        # Caso 1: Transformación
        diccionario_productos[key] = int(value * 0.9)
    else:
        # Caso 2: Filtrado
        diccionario_caros[key] = value

print(diccionario_productos)
print(diccionario_caros)
```

Desafío

Búsqueda de colores

(versión 1)

```python
import sys

colores = {
  "aliceblue": "#f0f8ff",
  "antiquewhite": "#faebd7",
  "aqua": "#00ffff",
  "aquamarine": "#7fffd4",
  "azure": "#f0ffff",
  "darkorchid": "#9932cc",
  "darkred": "#8b0000",
  "darksalmon": "#e9967a",
  "navajowhite": "#ffdead",
  "navy": "#000080",
  "orchid": "#da70d6"
}

# Opción del usuario
search = sys.argv[1]

# "Flag" que cambia a True si se encuentra el color
found = False

# Iteramos por clave (nombre) y valor (hexadecimal)
for name, hexa in colors.items():
    if hexa == search and found == False:
        # Si se encuentra, se imprime y se cambia el flag a True
        found = True
        print(name)

# Si flag permanece falso, se muesra mensaje de no encontrado
if not found:
    print("no-no")
```

Desafío

Búsqueda de colores

(versión 2)

```python
import sys

colores = {
  "aliceblue": "#f0f8ff",
  "antiquewhite": "#faebd7",
  "aqua": "#00ffff",
  "aquamarine": "#7fffd4",
  "azure": "#f0ffff",
  "darkorchid": "#9932cc",
  "darkred": "#8b0000",
  "darksalmon": "#e9967a",
  "navajowhite": "#ffdead",
  "navy": "#000080",
  "orchid": "#da70d6"
}

# Opción del usuario
search = sys.argv[1]

# Se invierte el diccionario
colors_inv = {v: k for k, v in colors.items()}

# Se consulta si el hexadecimal ingresado se encuentra en las claves
# del diccionario invertido. Si no es así, muestra el mensaje.
if search in colors_inv:
    print(colors_inv[search])
else:
    print("no-no")
```

Desafío

Búsqueda de colores

(versión 3)

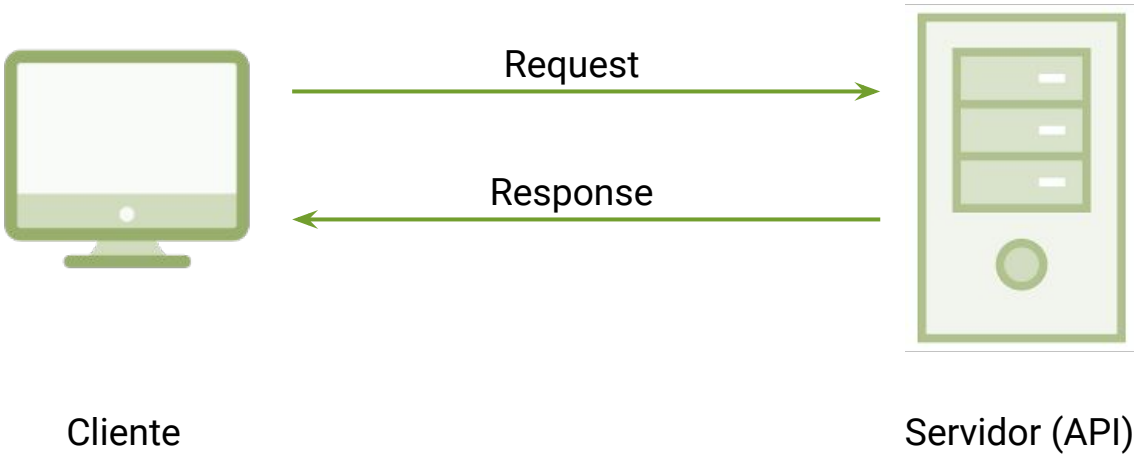
```python
import sys

colores = {
  "aliceblue": "#f0f8ff",
  "antiquewhite": "#faebd7",
  "aqua": "#00ffff",
  "aquamarine": "#7fffd4",
  "azure": "#f0ffff",
  "darkorchid": "#9932cc",
  "darkred": "#8b0000",
  "darksalmon": "#e9967a",
  "navajowhite": "#ffdead",
  "navy": "#000080",
  "orchid": "#da70d6"
}

# Opciones del usuario
searched_hexas = sys.argv[1:]

# Se invierte el diccionario
colors_inv = {v: k for k, v in colors.items()}

# Se itera por la lista de valores ingresados
for search in searched_hexas:
    # Se consulta por cada opción si existe esa clave en el diccionario invertido
    if search in colors_inv:
        print(colors_inv[search])
    else:
        print("no-no")

```

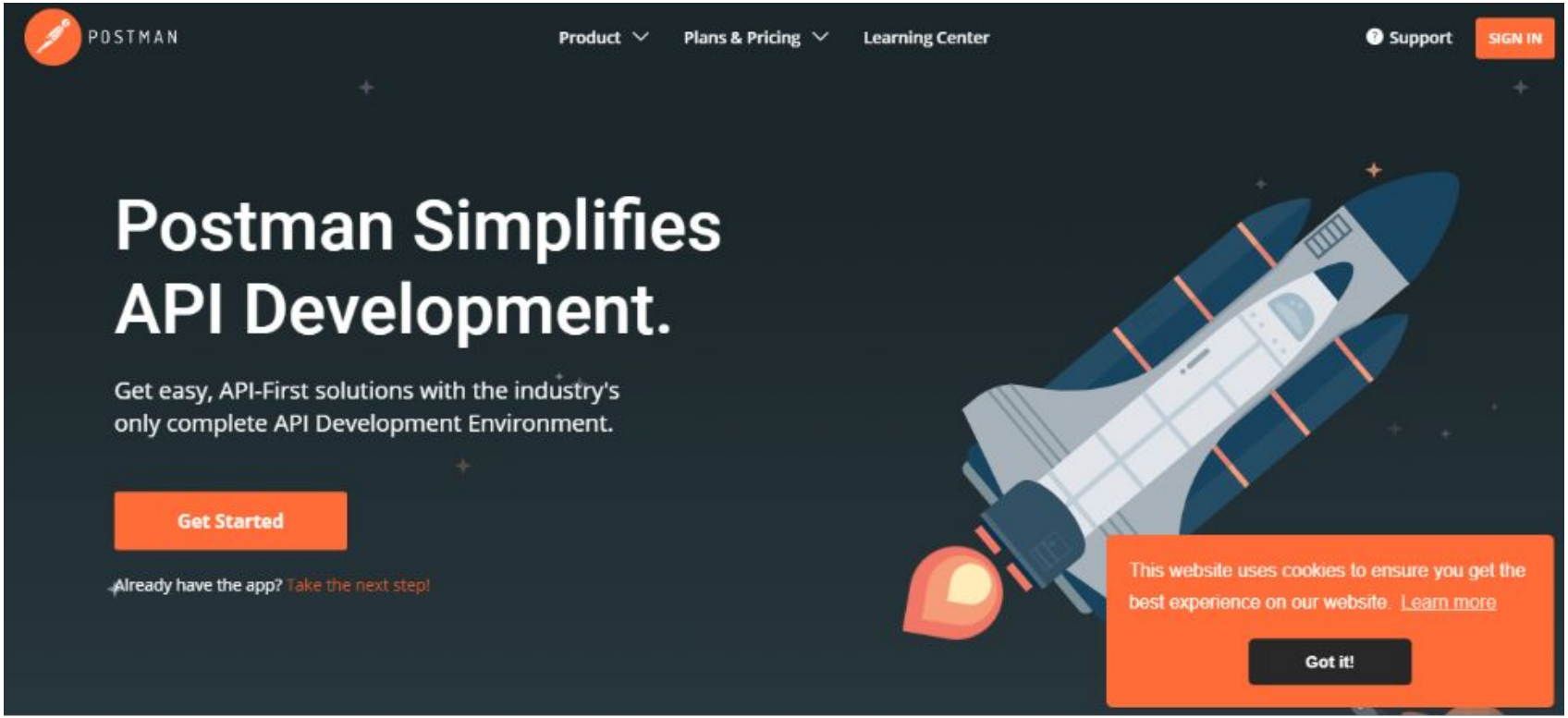
APIs

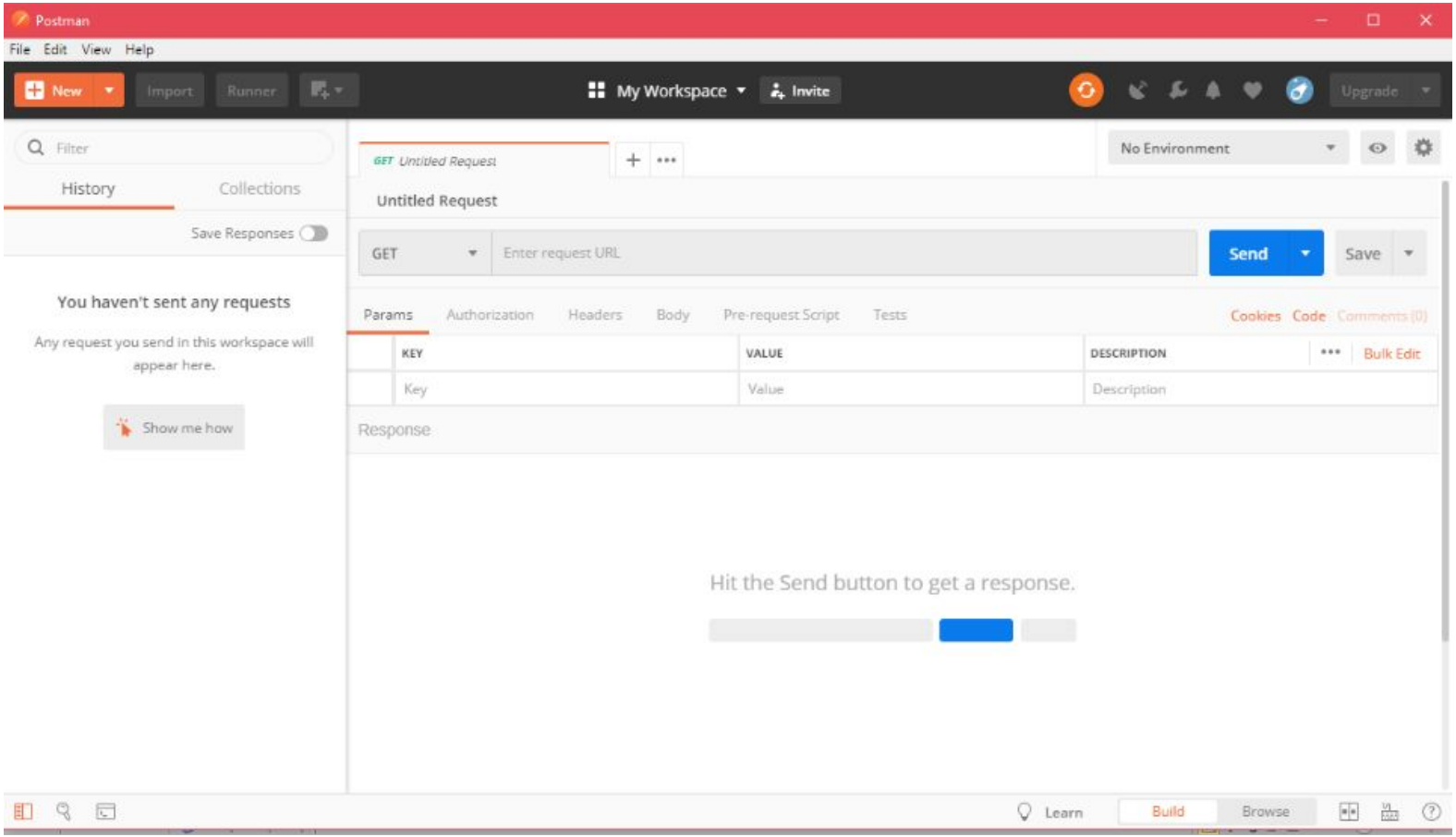
Tipos

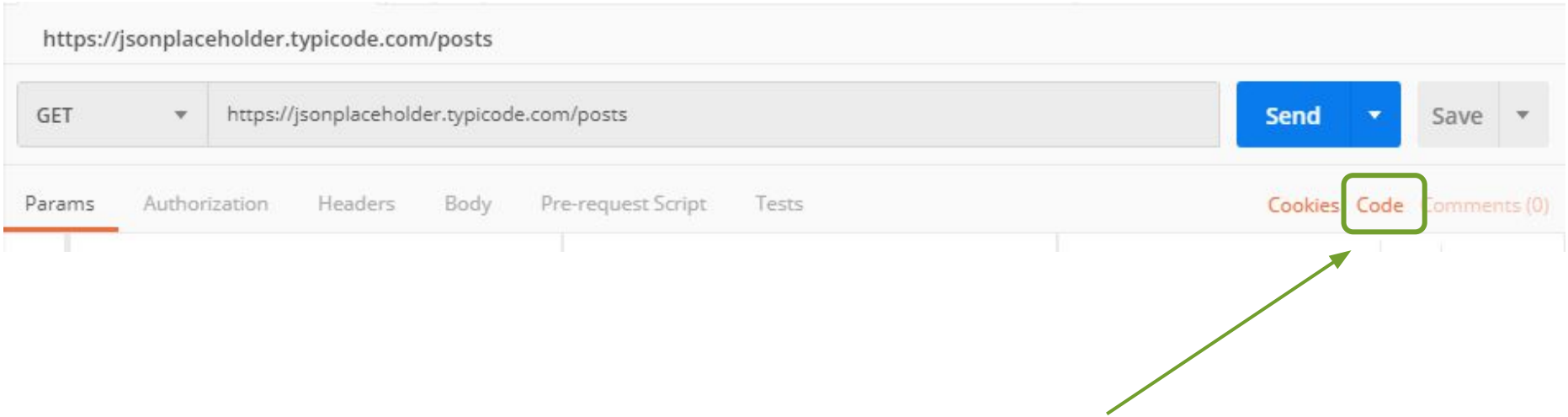
- Clima y temperatura
- Cambio de monedas
- Indicadores económicos
- Servicios para subir archivos
- Compra y venta de criptomonedas
- Servicios de geolocalización, como Google Maps

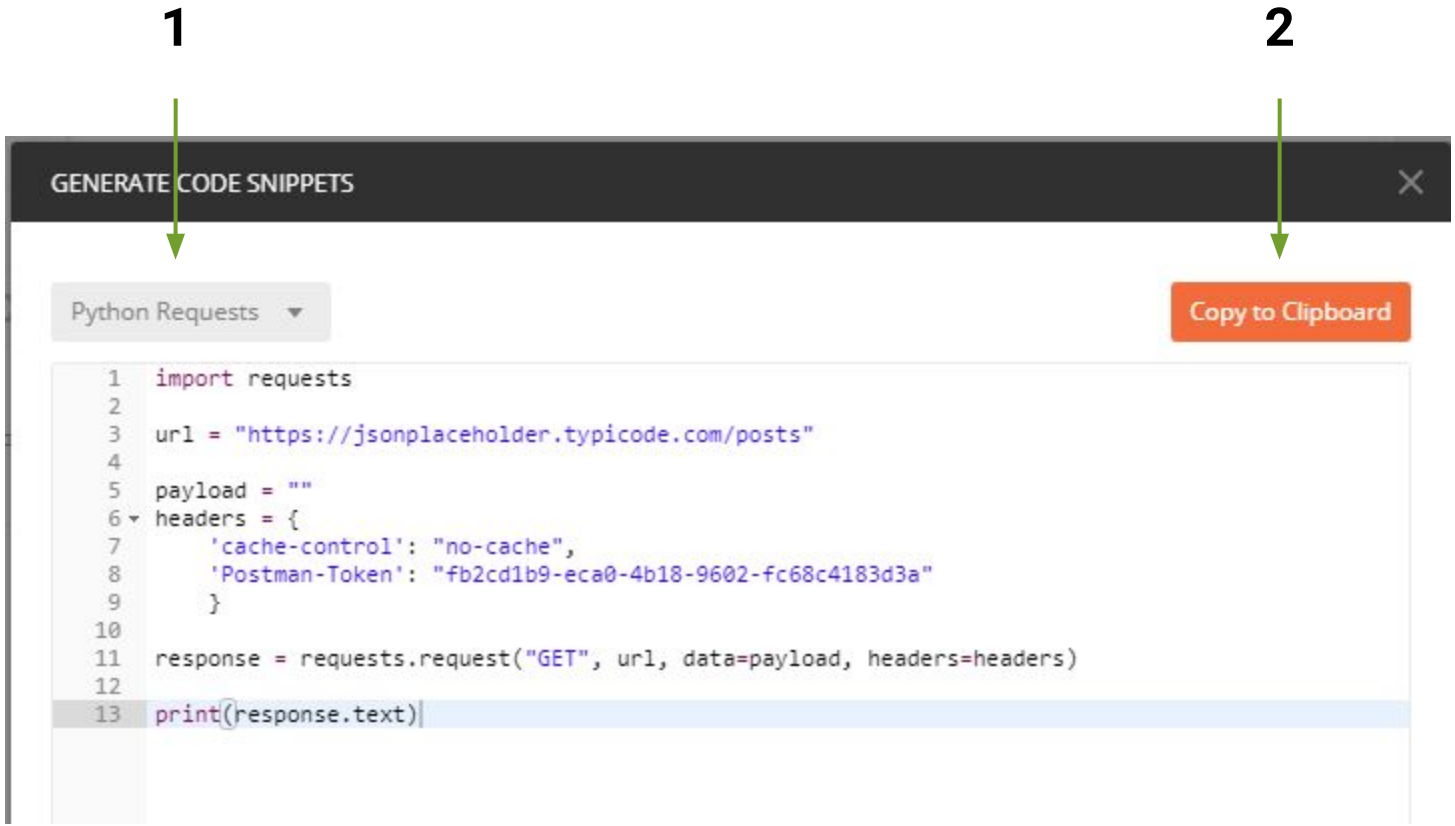
Funcionamiento




POSTMAN
Product
Plans & Pricing
Learning Center
Support
SIGN IN
Postman Simplifies API Development.
Get easy, API-First solutions with the industry's only complete API Development Environment.
Get Started
Already have the app? Take the next step!
This website uses cookies to ensure you get the best experience on our website. Learn more
Got it!


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
Save Responses
You haven't sent any requests
Any request you send in this workspace will appear here.
Show me how
GET Untitled Request
Untitled Request
No Environment
GET
Enter request URL
Send
Save
Params
Authorization
Headers
Body
Pre-request Script
Tests
Cookies
Code
Comments (0)
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Response
Hit the Send button to get a response.
Learn
Build
Browse


https://jsonplaceholder.typicode.com/posts
GET
https://jsonplaceholder.typicode.com/posts
Send
Save
Params
Authorization
Headers
Body
Pre-request Script
Tests
Cookies
Code
Comments (0)


1
2
GENERATE CODE SNIPPETS
Python Requests
Copy to Clipboard
1 import requests
2
3 url = "https://jsonplaceholder.typicode.com/posts"
4
5 payload = ""
6 headers = {
7 'cache-control': "no-cache",
8 'Postman-Token': "fb2cd1b9-eca0-4b18-9602-fc68c4183d3a"
9 }
10
11 response = requests.request("GET", url, data=payload, headers=headers)
12
13 print(response.text)

Códigos de respuesta

- **1xx**: Información (¡Espera!)
- **2xx**: Respuesta correcta (¡Todo bien!)
- **3xx**: Redirección (¡No es aquí!)
- **4xx**: Error del cliente (¡Lo hiciste mal!)
- **5xx**: Error del servidor (¡No eres tu!)

Analizar el body de la respuesta

```python
import requests
import json

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

payload = ""
headers = {
    'cache-control': "no-cache",
    'Postman-Token': "467ab332-ae53-499a-9ff2-5a7bb7ec1515"
    }

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

# Código de la respuesta
print(response)

#Cuerpo de la respuesta
body = response.text

# Convertir a estructura de python
results = json.loads(body)

# Es una lista
print(type(results))

# Vemos el primer elemento
print(results[0])

# Y su tipo (un diccionario)
print(type(results[0]))

# Accedemos a cada título para cada elemento de la respuesta
for post in results:
    print(post["title"])

```

Ejercicio de integración

```python
import requests
import json

def request(requested_url):
    headers = {
        "cache-control": "no-cache",
        "Postman-Token": "2467ab332-ae53-499a-9ff2-5a7bb7ec1515",
    }
    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"]

# Solución 1
selected_data = [k for k, v in prices.items() if v < 5000]
print(selected_data)

# Solución 2
selected_data = []
for date, value in prices.items():
    if value < 5000:
        selected_data.append(date)
print(selected_data)

# 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]
print(selected_data)
```

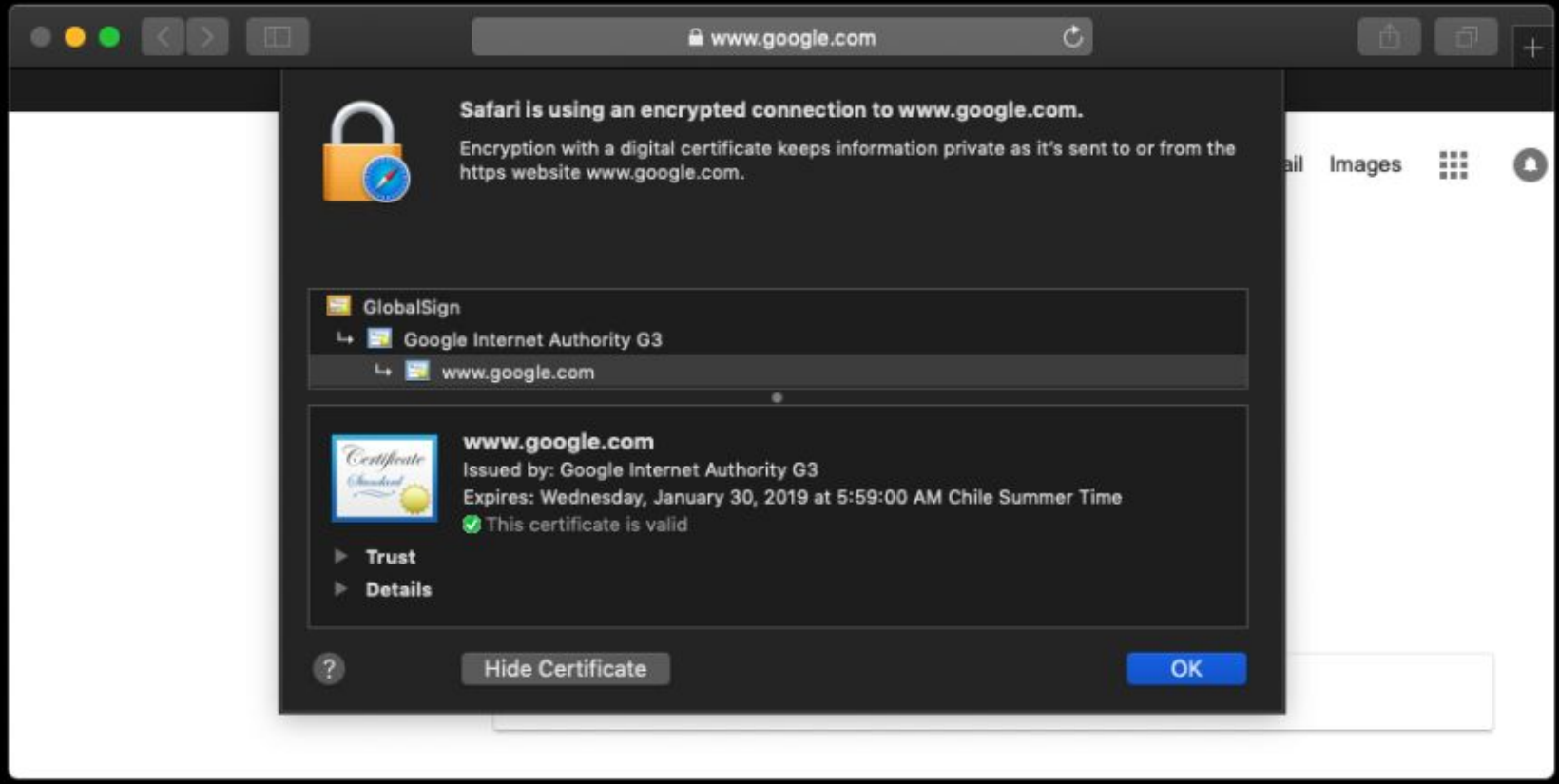
SSL

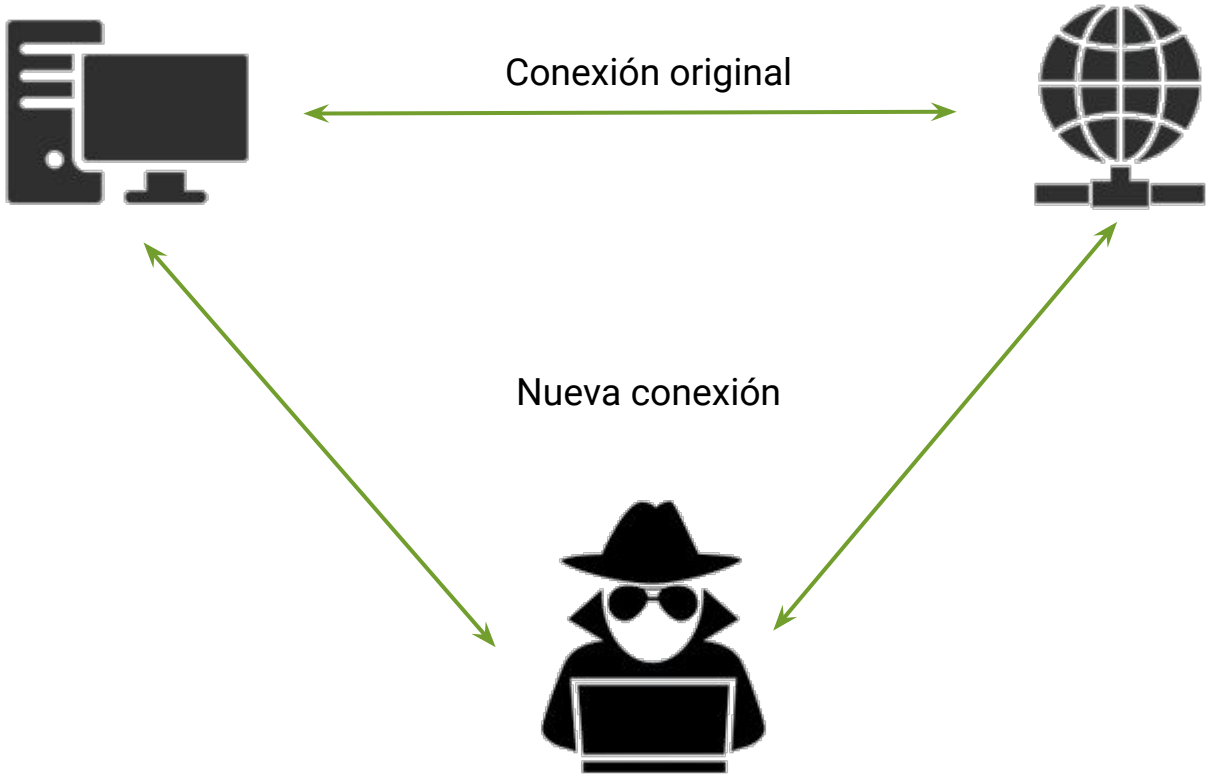
Alice tiene 2 claves:
Una privada para cifrar su mensaje: **AWETGT**
Una pública para descifrar su mensaje: **EGEUJU**

Alice se junta con Bob y le pasa su clave pública para descifrar

Alice envía a Bob un mensaje cifrado con su clave privada:

Bob utiliza la clave que Alice le pasó en el paso 2 para descifrar el mensaje


www.google.com
Safari is using an encrypted connection to www.google.com.
Encryption with a digital certificate keeps information private as it's sent to or from the https website www.google.com.
GlobalSign
Google Internet Authority G3
www.google.com
www.google.com
Issued by: Google Internet Authority G3
Expires: Wednesday, January 30, 2019 at 5:59:00 AM Chile Summer Time
This certificate is valid
Trust
Details
Hide Certificate
OK
Images


Conexión original
Nueva conexión
Hombre en el medio,
phisher o proxy

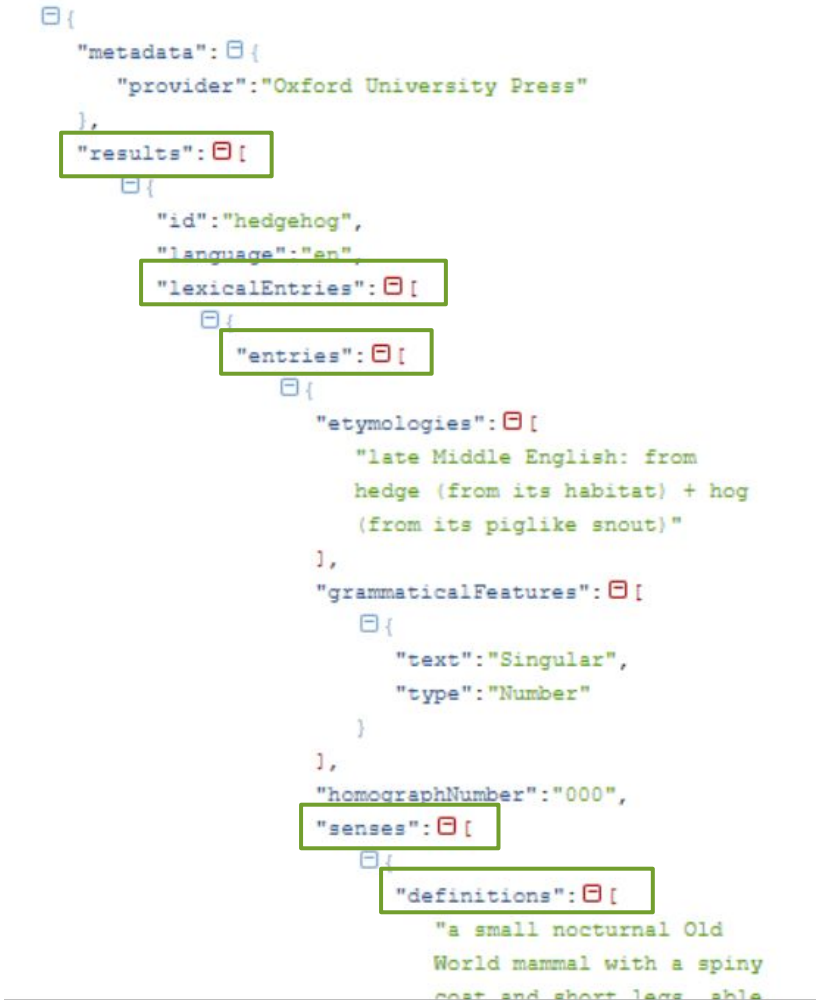
Desafío

Solicitar una palabra a la API del diccionario de Oxford

Código

```python
import json
import requests

def request(requested_url):
    headers = {
        "app_id": "[redacted]",
        "app_key": "[redacted]",
    }
    response = requests.request("GET", requested_url, headers=headers)
    return json.loads(response.text)


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

print(data['results'][0]['lexicalEntries'][0]['entries'][0]['senses'][0]['definitions'])
```

Salida


{
"metadata": {
"provider":"Oxford University Press"
},
"results": [
{
"id":"hedgehog",
"language":"en",
"lexicalEntries": [
{
"entries": [
{
"etymologies": [
"late Middle English: from
hedge (from its habitat) + hog
(from its piglike snout)"
],
"grammaticalFeatures": [
{
"text":"Singular",
"type":"Number"
}
],
"homographNumber":"000",
"senses": [
{
"definitions": [
"a small nocturnal Old
World mammal with a spiny

Elementos clave

- Un algoritmo es una serie de pasos finitos para resolver un problema.
- 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.

Otras cosas para explorar

- 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