# **Requests**

# **Datos desde la web:**

|  |
| --- |
| una biblioteca HTTP elegante y simple para Python, construida para seres humanos. |
| <https://requests.readthedocs.io/en/latest/>  https://docs.python-requests.org/ |

**.**

|  |
| --- |
| · |

**python -m pip install requests**

El Módulo requests realiza solicitudes HTTP

El protocolo HTTP se basa en dos conceptos principales:

petición (request) y respuesta (response).

Una petición está formada por:

● Un verbo HTTP que define la operación a realizar.

● Una cabecera o header que incluye información sobre la petición.

● Una ruta o path hacia un recurso.

● El cuerpo del mensaje o body con los datos de la petición.

|  |  |
| --- | --- |
| **Métodos** | **Descripción** |
| [Response](https://www.geeksforgeeks.org/response-headers-python-requests/)  [.headers](https://www.geeksforgeeks.org/response-headers-python-requests/) | Devuelve un diccionario de encabezados de respuesta. |
| [Response](https://www.geeksforgeeks.org/response-encoding-python-requests/)  [.encoding](https://www.geeksforgeeks.org/response-encoding-python-requests/) | Devuelve la codificación utilizada para decodificar response.content. |
| [Response](https://geeksforgeeks.org/response-elapsed-python-requests/)  [.elapsed](https://geeksforgeeks.org/response-elapsed-python-requests/) | Devuelve un objeto timedelta con el tiempo transcurrido desde que se envía la solicitud hasta la llegada de la respuesta. |
| [Response](https://www.geeksforgeeks.org/response-close-python-requests/)  [.close()](https://www.geeksforgeeks.org/response-close-python-requests/) | Cierra la conexión con el servidoresponse. |
| [Response](https://www.geeksforgeeks.org/response-content-python-requests/)  [.content](https://www.geeksforgeeks.org/response-content-python-requests/) | Devuelve el contenido de la respuesta, en bytes. |
| [Response](https://www.geeksforgeeks.org/response-cookies-python-requests/)  [.cookies](https://www.geeksforgeeks.org/response-cookies-python-requests/) | Devuelve un objeto CookieJar con las cookies enviadas desde el servidoresponse. |
| [Response](https://www.geeksforgeeks.org/response-history-python-requests/)  [.history](https://www.geeksforgeeks.org/response-history-python-requests/) | Devuelve una lista de objetos de respuesta que contienen el historial de solicitudes (URL). |
| [Response](https://www.geeksforgeeks.org/response-is_permanent_redirect-python-requests/)  [.is\_permanent\_redirect](https://www.geeksforgeeks.org/response-is_permanent_redirect-python-requests/) | Devuelve True si la respuesta es la URL redireccionada permanente.  En caso contrario, False. |
| [Response](https://www.geeksforgeeks.org/response-is_redirect-python-requests/)  [.is\_redirect](https://www.geeksforgeeks.org/response-is_redirect-python-requests/) | Devuelve True si la respuesta fue redirigida; en caso contrario, False. |
| [Response](https://www.geeksforgeeks.org/response-iter_content-python-requests/)  [.iter\_content()](https://www.geeksforgeeks.org/response-iter_content-python-requests/) | Itera sobre response.content. |
| [Response](https://www.geeksforgeeks.org/response-json-python-requests/)  [.json()](https://www.geeksforgeeks.org/response-json-python-requests/) | Devuelve un objeto JSON del resultado  (si el resultado se escribió en formato JSON, si no, genera un error). |
| [Response](https://www.geeksforgeeks.org/response-url-python-requests/)  [.url](https://www.geeksforgeeks.org/response-url-python-requests/) | Devuelve la URL de la respuesta. |
| [Response](https://www.geeksforgeeks.org/response-text-python-requests/)  [.text](https://www.geeksforgeeks.org/response-text-python-requests/) | Devuelve el contenido de la respuesta, en Unicode. |
| [Response](https://www.geeksforgeeks.org/response-status_code-python-requests/)  [.status\_code](https://www.geeksforgeeks.org/response-status_code-python-requests/) | Devuelve un número que indica el estado  (200 está bien, 404 es No encontrado). |
| [Response](https://www.geeksforgeeks.org/response-request-python-requests/)  [.request](https://www.geeksforgeeks.org/response-request-python-requests/) | Devuelve el objeto de solicitud que solicitó esta respuesta. |
| [Response](https://www.geeksforgeeks.org/response-reason-python-requests/)  [.reason](https://www.geeksforgeeks.org/response-reason-python-requests/) | Devuelve un texto correspondiente al código de estado. |
| [Response](https://www.geeksforgeeks.org/response-raise_for_status-python-requests/)  [.raise\_for\_status()](https://www.geeksforgeeks.org/response-raise_for_status-python-requests/) | Devuelve un objeto HTTPError si se ha producido un error durante el proceso. |
| [Response](https://www.geeksforgeeks.org/response-ok-python-requests/)  [.ok](https://www.geeksforgeeks.org/response-ok-python-requests/) | Devuelve True si status\_code es menor que 200; en caso contrario, False. |
| [Response](https://www.geeksforgeeks.org/response-links-python-requests/)  [.links](https://www.geeksforgeeks.org/response-links-python-requests/) | Devuelve los enlaces del encabezado. |

.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STEP 1:  Resource description→ | STEP 2:  Endpoints and methods→ | STEP 3:  Parameters→ | STEP 4:  Request example→ | STEP 5:  Response example and schema |

REST, REpresentational State Transfer, se trata de una arquitectura estándar para comunicaciones web entre sistemas, logrando que se entiendan mucho mejor entre ellos.

A los servicios que cumplen con este diseño se les llama RESTful API.

Cada una de estas “secciones” de un servicio web, en la terminología de la arquitectura REST, se denomina recurso.

Ahora bien, el protocolo HTTP define un conjunto de métodos para identificar qué tipo de operación se quiere ejecutar sobre un recurso determinado, a saber:

|  |  |
| --- | --- |
| GET | Esta solicitud se utiliza para obtener un recurso de un servidor. Si realiza una solicitud `GET`, el servidor busca los datos que solicitó y se los devuelve. En otras palabras, una solicitud `GET` realiza una operación `READ`. Este es el método de solicitud predeterminado. |
| POST | Esta solicitud se utiliza para crear un nuevo recurso en un servidor. Si realiza una solicitud `POST`, el servidor crea una nueva entrada en la base de datos y le informa si la creación se realizó correctamente. En otras palabras, una solicitud `POST` realiza una operación `CREATE`.  `PONE` y `PATCH` |
| PATCH  o  PUT | Estas dos solicitudes se utilizan para actualizar un recurso en un servidor. Si realiza una solicitud `PUT` o `PATCH`, el servidor actualiza una entrada en la base de datos y le dice si la actualización se realizó correctamente. En otras palabras, una solicitud `PUT` o `PATCH` realiza una operación `UPDATE`. |
| DELETE | Esta solicitud se utiliza para eliminar un recurso de un servidor. Si realiza una solicitud `DELETE`, el servidor elimina una entrada en la base de datos y le informa si la eliminación se realizó correctamente. En otras palabras, una solicitud `DELETE` realiza una operación `DELETE`. |

·

|  |
| --- |
| Solo veremos GET en este curso |

·

|  |
| --- |
| #windows  python -m pip install --upgrade pip wheel setuptools  python -m pip install requests  #mac o linux  python3 -m pip install --upgrade pip wheel setuptools  python3 -m pip install requests |

.

|  |
| --- |
| import requests  # Realizar una solicitud GET a una URL  response = requests.get('https://google.com/')  # Obtener el código de estado de la respuesta  status\_code = response.status\_code  print(f'Código de estado:{status\_code}') |

·

|  |
| --- |
| # Obtener el contenido de la respuesta como texto  content = response.text  # Imprimir el código de estado y el contenido de la respuesta  print(f'Contenido de la respuesta:{content}') |

·

|  |
| --- |
| url = "https://api.mercadolibre.com/sites/MLA/search?category=MLA5726"  res = requests.get(url)  print (f"{res=}")  print (f"{type(res)=}")  try:  for clave, valor in res.json().items():  print(clave, valor)  except:  pass |

·

|  |
| --- |
| for url in ("https://api.mercadolibre.com/sites/","https://api.mercadolibre.com/sites/MLA/search?category=MLA5726"):  print ("\*"\*100)  print (f"{url=}")  response = requests.get(url)  print (f"{response=}")  print (f"{type(response)=}")  print (f"{type(response.json())=}")  salida=response.json()  if "html" in salida:  print (f"html: {index}")  elif isinstance(salida,(list,tuple)):  if isinstance(salida[0],dict):  for clave, valor in salida[0].items():  print (f"{clave} {valor}")  else:  for index, valor in enumerate(salida):  print (f"{index} {valor}")  elif isinstance(salida,dict):  for clave, valor in salida.items():  print (f"{clave} {valor}")  else:  content = response.text  # Imprimir el código de estado y el contenido de la respuesta  print(f'Contenido de la respuesta:{content}') |

·

Utilizamos requests.get() para realizar una solicitud GET a la URL especificada.

Debemos ver que tipo de objeto nos devuelve - response.status\_code , generalmente json compuestos con listas y diccionarios anidados o texto plano a través de response.text.

|  |
| --- |
| **pip install requests** |
| La biblioteca requests en Python es una de las partes integrales para realizar solicitudes HTTP a una URL específica. Ya sean API REST o Web Scraping, es necesario aprender las solicitudes para seguir adelante con estas tecnologías. Cuando uno realiza una solicitud a un URI, devuelve una respuesta. Las solicitudes de Python proporcionan funcionalidades integradas para gestionar tanto la solicitud como la respuesta.  En este tutorial, exploraremos qué es la biblioteca de solicitudes de Python , cómo realizar solicitudes GET a través de solicitudes de Python, objetos y métodos de respuesta, autenticación mediante solicitudes de Python, etc. |

·

|  |
| --- |
| **Make requests** |
| El módulo requests tiene varios métodos integrados para realizar solicitudes Http a un URI específico mediante solicitudes GET, POST, PUT, PATCH o HEAD. Una solicitud Http está destinada a recuperar datos de un URI específico o enviar datos a un servidor. Funciona como un protocolo de solicitud-respuesta entre un cliente y un servidor. Demostremos cómo realizar una solicitud GET a un punto final. El método GET se utiliza para recuperar información del servidor determinado utilizando un URI determinado. El método GET envía la información de usuario codificada adjunta a la solicitud de página. La página y la información codificada están separadas por el símbolo '?' personaje.  https://www.google.com/search?q=hola  Cómo realizar una solicitud GET a través de solicitudes de Python  El módulo de solicitudes de Python proporciona un método integrado llamado get() para realizar una solicitud GET a un URI específico.  **Sin Autenticación**   |  | | --- | | Sintaxis: | | request.get(url, parámetros={clave: valor}, argumentos) | | import requests  r = requests.get('https://httpbin.org/basic-auth/user/pass', auth=('user', 'pass'))  print(f"{r=}")  print(f"{r.status\_code=}")  print(f"{r.headers['content-type']=}")#'application/json; charset=utf8'  print(f"{r.encoding=}")#'utf-8'  print(f"{r.text=}")#'{"authenticated": true, ...'  print(f"{r.json()=}")#{'authenticated': True, ...} | | r=<Response [200]>  r.status\_code=200  r.headers['content-type']='application/json'  r.encoding='utf-8'  r.text='{\n "authenticated": true, \n "user": "user"\n}\n'  r.json()={'authenticated': True, 'user': 'user'} |   **.** |
| **Autenticación mediante solicitudes de Python**  La autenticación se refiere a otorgar permisos a un usuario para acceder a un recurso en particular. Dado que no se puede permitir que todos accedan a los datos de cada URL, se requeriría principalmente autenticación. Para lograr esta autenticación, normalmente se proporcionan datos de autenticación a través del encabezado de Autorización o un encabezado personalizado definido por el servidor.   |  | | --- | | Sintaxis: | | import requests  from requests.auth import HTTPBasicAuth  r = requests.get('https://api.github.com/user, ', auth = HTTPBasicAuth('user', 'pass'))  print(response) | | <Response [404]> |   . |

·

.

|  |
| --- |
| Así es como realiza una solicitud HTTP POST: |
| .   |  | | --- | | Sintaxis: | | import requests  r = requests.post('https://httpbin.org/post', data={'key': 'value'})  print(f"{r=}")  r = requests.put('https://httpbin.org/put', data={'key': 'value'})  print(f"{r=}")  r = requests.delete('https://httpbin.org/delete')  print(f"{r=}")  r = requests.head('https://httpbin.org/get')  print(f"{r=}")  r = requests.options('https://httpbin.org/get')  print(f"{r=}") |   . |

.

|  |
| --- |
| **Ejemplo de Solicitud GET:** |
| .   |  | | --- | | import requests  **url = "https://jsonplaceholder.typicode.com/posts/1"**  response = requests.get(url)  **# Acceder al código de estado**  print("Código de estado:", response.status\_code)  **# Acceder al contenido de la respuesta en formato JSON**  print("Contenido de la respuesta:",response.json()) |   . |

.

|  |
| --- |
| **Funcionalidades Adicionales:** |
| .   |  | | --- | | **Parámetros de URL:**   1. Puedes incluir parámetros en tus solicitudes GET usando el parámetro params.   payload = {'key1': 'value1', 'key2': 'value2'}  response = requests.get(url, params=payload) | | **Encabezados Personalizados:**   * Puedes enviar encabezados personalizados en tu solicitud utilizando el parámetro headers.   headers = {'Content-Type': 'application/json'}  response = requests.post(url, data=data, headers=headers) | | **Manejo de Errores:**   * Puedes verificar el código de estado de la respuesta para manejar errores.   if response.status\_code == 200:  # Procesar la respuesta exitosa  else:  # Manejar el error | | **Autenticación:**   * requests admite diferentes métodos de autenticación, como HTTP básica o token.   from requests.auth import HTTPBasicAuth  response = requests.get(url, auth=HTTPBasicAuth('username', 'password')) | | **Gestión de Sesiones:**   * Puedes utilizar sesiones para persistir ciertos parámetros entre múltiples solicitudes.   session = requests.Session()  response = session.get(url) | | Estos son solo algunos aspectos de la biblioteca requests.  La documentación oficial de requests (https://docs.python-requests.org/) proporciona detalles completos sobre todas las características y opciones disponibles. |   . |

.

|  |
| --- |
| .Conceptos Básicos:  1. Solicitudes HTTP:  ◦ requests.get(url): Realiza una solicitud HTTP GET a la URL especificada.  ◦ requests.post(url, data): Realiza una solicitud HTTP POST a la URL especificada con los datos proporcionados en el cuerpo de la solicitud.  2. Objeto de Respuesta (Response):  ◦ Después de realizar una solicitud, obtenemos un objeto Response.  ◦ Contiene información sobre la respuesta del servidor, como el código de estado, encabezados y el contenido de la respuesta. |
| .   |  | | --- | | Sintaxis:  Ejemplo de solicitud GET:  Aquí se utiliza la API de JSONPlaceholder, que proporciona datos ficticios para realizar pruebas. La solicitud GET se realiza a un recurso específico y se imprime la respuesta. | | import requests  **# URL de la API a la que haremos la solicitud**  url = "https://jsonplaceholder.typicode.com/posts/1"  **# Realizamos una solicitud GET**  response = requests.get(url)  **# Imprimimos el código de estado de la respuesta (200 significa éxito)**  print("Código de estado:", response.status\_code)  **# Imprimimos el contenido de la respuesta en formato JSON**  print("Contenido de la respuesta:",response.json()) |   . |
| .   |  | | --- | | Sintaxis:  Ejemplo de solicitud POST:  Aquí, estamos enviando datos al servidor mediante una solicitud POST a la misma API de JSONPlaceholder. El servidor responderá con un nuevo recurso creado. | | **# URL de la API a la que haremos la solicitud POST**  url = "https://jsonplaceholder.typicode.com/posts"  **# Datos que enviaremos en el cuerpo de la solicitud POST**  data = {  'title': 'foo',  'body': 'bar',  'userId': 1  }  **# Realizamos una solicitud POST con los datos**  response = requests.post(url, data=data)  **# Imprimimos el código de estado de la respuesta (201 significa creado**)  print("Código de estado:", response.status\_code)  **# Imprimimos el contenido de la respuesta en formato JSON**  print("Contenido de la respuesta:",response.json()) |   . |

-.

|  |
| --- |
| params |
| .   |  | | --- | | params: | | import requests  payload = {'key1': 'value1', 'key2': 'value2'}  r = requests.get('https://httpbin.org/get', params=payload)  print(f"{r=}")  print(f"{r.status\_code=}")  print(f"{r.headers['content-type']=}")  print(f"{r.encoding=}")  print(f"{r.text=}")  print(f"{r.json()=}") | | r=<Response [200]>  r.status\_code=200  r.headers['content-type']='application/json'  r.encoding='utf-8'  {  "args": {  "key1": "value1",  "key2": "value2"  },  "headers": {  "Accept": "\*/\*",  "Accept-Encoding": "gzip, deflate",  "Host": "httpbin.org",  "User-Agent": "python-requests/2.31.0",  "X-Amzn-Trace-Id":"Root=1-65bd3764-798d0ec52852556f67ffec76"  },  "origin": "181.117.10.90",  "url": "https://httpbin.org/get?key1=value1&key2=value2"  }  r.json()={'args': {'key1': 'value1', 'key2': 'value2'}, 'headers': {'Accept': '\*/\*', 'Accept-Encoding': 'gzip, deflate', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.31.0', 'X-Amzn-Trace-Id': 'Root=1-65bd3764-798d0ec52852556f67ffec76'}, 'origin': '181.117.10.90', 'url': 'https://httpbin.org/get?key1=value1&key2=value2'} |   . |

.

|  |
| --- |
| **headers** |
| **Nota:** Los encabezados personalizados tienen menos prioridad que las fuentes de información más específicas.  Encabezados de autorización establecidos con headers= se anulará si las credenciales se especifican en .netrc, que a su vez será anulado por el auth= parámetro. Las solicitudes buscarán el archivo netrc en ~/.netrc, ~/\_netrc, o en la ruta especificada por el NETRC variable de entorno.  Los encabezados de autorización se eliminarán si se le redirige fuera del host.  Los encabezados de autorización de proxy serán anulados por las credenciales proxy proporcionadas en la URL.  Los encabezados de longitud de contenido se anularán cuando podamos determinar la longitud del contenido.  Además, las solicitudes no cambian su comportamiento en absoluto en función de qué encabezados personalizados se especifican. Los encabezados simplemente se pasan a la solicitud final.  Nota: Todos los valores de encabezado deben ser a string, bytestring o unicode. Si bien está permitido, se recomienda evitar pasar valores de encabezado unicode. |
| .   |  | | --- | | headers: | | url = 'https://api.github.com/some/endpoint'  headers = {'user-agent': 'my-app/0.0.1'}  r = requests.get(url, headers=headers)  print(f"{r=}")  print(f"{r.status\_code=}")  print(f"{r.headers['content-type']=}")  print(f"{r.encoding=}")  print(f"{r.text}")  print(f"{r.json()=}")  print ("\*"\*50) |   . |

.

|  |
| --- |
| **API Endpoints** |
| **Nota: En general, existen 3 métodos diferentes para usar la API de ipstack:**  **Búsqueda estándar:**  **busca los datos detrás de una dirección IP.**  **Búsqueda masiva: busque los datos detrás de varias direcciones IP a la vez. B**  **úsqueda del solicitante: busque los datos detrás de la dirección IP de la que proviene su solicitud de API.**  **URL base: cualquiera que sea el método de API que elija utilizar, todas las solicitudes de API a la API de ipstack comienzan con la siguiente URL base: http://api.ipstack.com/** |
| .   |  | | --- | | url = 'https://api.github.com/some/endpoint'  headers = {'user-agent': 'my-app/0.0.1'}  r = requests.get(url, headers=headers)  print(f"{r=}")  print(f"{r.status\_code=}")  print(f"{r.headers['content-type']=}")  print(f"{r.encoding=}")  print(f"{r.text}")  print(f"{r.json()=}")  print ("\*"\*50) | | r=<Response [404]>  r.status\_code=404  r.headers['content-type']='application/json; charset=utf-8'  r.encoding='utf-8'  {"message":"Not Found","documentation\_url":"https://docs.github.com/rest"}  r.json()={'message': 'Not Found', 'documentation\_url': 'https://docs.github.com/rest'} |   . |

.

|  |
| --- |
| Solicitudes POST más complicadas |
| Por lo general, desea enviar algunos datos codificados por formulario — como un formulario HTML. Para hacer esto, simplemente pase un diccionario al data argumento. Tu el diccionario de datos se codificará automáticamente cuando se realice la solicitud: |
| .   |  | | --- | | El data el argumento también puede tener múltiples valores para cada clave. Esto puede ser hecho haciendo data ya sea una lista de tuplas o un diccionario con listas como valores. Esto es particularmente útil cuando la forma tiene varios elementos que usa la misma clave: | | import requests  import json  print ("\*"\*50)  payload\_tuples = [('key1', 'value1'), ('key1', 'value2')]  r1 = requests.post('https://httpbin.org/post', data=payload\_tuples)  print(f"{r1.json()['form']=}")  print ("\*"\*50)  payload\_dict = {'key1': ['value1', 'value2']}  r2 = requests.post('https://httpbin.org/post', data=payload\_dict)  print(f"{r2.json()['form']=}")  print ("\*"\*50)  print(f"{r1==r2=}")  print(f"{r1.json()['form']==r2.json()['form']=}")  print ("\*"\*50) | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  r1.json()['form']={'key1': ['value1', 'value2']}  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  r2.json()['form']={'key1': ['value1', 'value2']}  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  r1==r2= False  r1.json()['form']==r2.json()['form']= True  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |   . |

.

|  |
| --- |
| Para obtener el pronóstico de tu ciudad ingresa a  [https://openweathermap.org](https://openweathermap.org/)  en el menú busca Sign in    crea un usuario si lo deseas y te llegara un mal para confirmar el usuario y password    Luego ingresa y busca el apikey |

·

|  |
| --- |
| import json  import requests  from pprint import \*  apiKey = "12345678901234567890123456789012"#32 caracteres  #cityId = "3435910";  #city="Esquel"  listaciudades = [  "Buenos Aires",  "Chaco",  "Resistencia",  "Esquel",  "Cordoba",  "Oslo",  "Haedo"  ]  for city in listaciudades:  print ("registrarse en https://home.openweathermap.or ")  url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={apiKey}&units=metric"  res = requests.get(url)  print (f"{res=}")  for clave, valor in res.json().items():  print(clave, valor)  print(f"{city}") |

·

|  |
| --- |
| **Ejemplos reales** (con api key) |
| Crea una cuenta y a trabajar **https://home.openweathermap.org/users/sign\_in** |
| .   |  |  | | --- | --- | |  |  | |  |  |   . |
| import json  import requests  def clima():  # Clima weather map  # api con respuesta con api key  #https://openweathermap.org/current  #https://openweathermap.org/find  **apiKey** = "8f7731766827fc49ebeb1991c47baa91"#32 caracteres  **city** = "Buenos Aires"  **url** = f"http://api.openweathermap.org/data/2.5/weather?q={**city**}&appid={**apiKey**}&units=metric"  **res** = requests.get(**url**)  print (f"{**res**=}")  if **res**.status\_code != 200:  exit()  elif **res**.status\_code==200:  print(f"{**city**}")  dicc= **res**.json()  for clave, valor in dicc.items():  print(f"{clave=}:")  if isinstance(valor, dict):  for sub\_clave, sub\_valor in valor.items():  print (f" { sub\_clave=} {sub\_valor=}")  elif isinstance(valor, (list,tuple)):  for index\_, sub\_valor in enumerate(valor):  print (f" { sub\_clave=} {sub\_valor=}")  else:  print (f" {valor=}")  clima() |
| Buenos Aires  clave='coord':  sub\_clave='lon' sub\_valor=-58.3772  sub\_clave='lat' sub\_valor=-34.6132  clave='weather':  sub\_clave='lat' sub\_valor={'id': 800, 'main': 'Clear', 'description': 'clear sky', 'icon': '01d'}  clave='base':  valor='stations'  clave='main':  sub\_clave='temp' sub\_valor=34.33  sub\_clave='feels\_like' sub\_valor=35.48  sub\_clave='temp\_min' sub\_valor=33.28  sub\_clave='temp\_max' sub\_valor=34.88  sub\_clave='pressure' sub\_valor=1011  sub\_clave='humidity' sub\_valor=38  sub\_clave='sea\_level' sub\_valor=1011  sub\_clave='grnd\_level' sub\_valor=1008  clave='visibility':  valor=10000  clave='wind':  sub\_clave='speed' sub\_valor=5.31  sub\_clave='deg' sub\_valor=87  sub\_clave='gust' sub\_valor=7.27  clave='clouds':  sub\_clave='all' sub\_valor=0  clave='dt':  valor=1706904698  clave='sys':  sub\_clave='type' sub\_valor=2  sub\_clave='id' sub\_valor=2008409  sub\_clave='country' sub\_valor='AR'  sub\_clave='sunrise' sub\_valor=1706865257  sub\_clave='sunset' sub\_valor=1706914796  clave='timezone':  valor=-10800  clave='id':  valor=3435910  clave='name':  valor='Buenos Aires'  clave='cod':  valor=200 |

..

|  |
| --- |
| Se que las estadisticas económicas de la argentina no son precisamente para mostrar al mundo.  Pero te recomiendo ingresar a  <http://estadisticasbcra.com/api/documentacion>  Allí a registrarse o directamente al link https://estadisticasbcra.com/api/registracion    Te llegara por mail la confirmación.  Sin importar que te llegue pasaras a la siguiente pantallas |

·

|  |
| --- |
| import json  import requests  from pprint import \*  token="BEARER xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"  endpoint = "usd"  puntosfinales = [ "usd",  "usd\_of",  "usd\_of\_minorista" ]  for endpoint in puntosfinales:  url = "https://api.estadisticasbcra.com/"+endpoint  headers = {"Authorization": token}  res = requests.get(url, headers=headers)  print(f"{res.status\_code=} {type(res.status\_code)=}")  if res.status\_code != 200:  exit()  elif res.status\_code==200:  print(f"{res.json()[-1]=}")  pprint(f"{res.json()[-1]=}")  #----------------------------------------------------------------------------------------------  if res:  print('Respuesta OK')  else:  print('Respuesta Error') |
| ENDPOINTS : Descripcion  https://api.estadisticasbcra.com/milestones : eventos relevantes (presidencia, ministros de economía, presidentes del BCRA, cepo al dólar)  https://api.estadisticasbcra.com/base : base monetaria  https://api.estadisticasbcra.com/base\_usd: base monetaria dividida USD  https://api.estadisticasbcra.com/base\_usd\_of: base monetaria dividida USD Oficial  https://api.estadisticasbcra.com/reservas : reservas internacionales  https://api.estadisticasbcra.com/base\_div\_res : base monetaria dividida reservas internacionales  https://api.estadisticasbcra.com/usd : cotización del USD  https://api.estadisticasbcra.com/usd\_of : cotización del USD Oficial  https://api.estadisticasbcra.com/usd\_of\_minorista : cotización del USD Oficial (Minorista)  https://api.estadisticasbcra.com/var\_usd\_vs\_usd\_of : porcentaje de variación entre la cotización del USD y el USD oficial  https://api.estadisticasbcra.com/circulacion\_monetaria : circulación monetaria  https://api.estadisticasbcra.com/billetes\_y\_monedas : billetes y monedas  https://api.estadisticasbcra.com/efectivo\_en\_ent\_fin : efectivo en entidades financieras  https://api.estadisticasbcra.com/depositos\_cuenta\_ent\_fin : depositos de entidades financieras en cuenta del BCRA  https://api.estadisticasbcra.com/depositos : depósitos  https://api.estadisticasbcra.com/cuentas\_corrientes : cuentas corrientes  https://api.estadisticasbcra.com/cajas\_ahorro : cajas de ahorro  https://api.estadisticasbcra.com/plazo\_fijo : plazos fijos  https://api.estadisticasbcra.com/tasa\_depositos\_30\_dias : tasa de interés por depósitos  /////  https://api.estadisticasbcra.com/prestamos : prestamos  https://api.estadisticasbcra.com/tasa\_prestamos\_personales : tasa préstamos personales  https://api.estadisticasbcra.com/tasa\_adelantos\_cuenta\_corriente : tasa adelantos cuenta corriente  https://api.estadisticasbcra.com/porc\_prestamos\_vs\_depositos : porcentaje de prestamos en relación a depósitos  https://api.estadisticasbcra.com/lebac : LEBACs  https://api.estadisticasbcra.com/leliq : LELIQs  https://api.estadisticasbcra.com/lebac\_usd : LEBACs en USD  https://api.estadisticasbcra.com/leliq\_usd : LELIQs en USD  https://api.estadisticasbcra.com/leliq\_usd\_of : LELIQs en USD Oficial  https://api.estadisticasbcra.com/tasa\_leliq : Tasa de LELIQs  https://api.estadisticasbcra.com/m2\_privado\_variacion\_mensual : M2 privado variación mensual  https://api.estadisticasbcra.com/cer : CER  https://api.estadisticasbcra.com/uva : UVA  https://api.estadisticasbcra.com/uvi : UVI  https://api.estadisticasbcra.com/tasa\_badlar : tasa BADLAR  https://api.estadisticasbcra.com/tasa\_baibar : tasa BAIBAR  https://api.estadisticasbcra.com/tasa\_tm20 : tasa TM20  https://api.estadisticasbcra.com/tasa\_pase\_activas\_1\_dia : tasa pase activas a 1 día  https://api.estadisticasbcra.com/tasa\_pase\_pasivas\_1\_dia : tasa pase pasivas a 1 día  https://api.estadisticasbcra.com/inflacion\_mensual\_oficial : inflación mensual oficial  https://api.estadisticasbcra.com/inflacion\_interanual\_oficial : inflación inteanual oficial  https://api.estadisticasbcra.com/inflacion\_esperada\_oficial : inflación esperada oficial  https://api.estadisticasbcra.com/dif\_inflacion\_esperada\_vs\_interanual : diferencia entre inflación interanual oficial y esperada  https://api.estadisticasbcra.com/var\_base\_monetaria\_interanual : variación base monetaria interanual  https://api.estadisticasbcra.com/var\_usd\_interanual : variación USD interanual  https://api.estadisticasbcra.com/var\_usd\_oficial\_interanual : variación USD (Oficial) interanual  https://api.estadisticasbcra.com/var\_merval\_interanual : variación merval interanual  https://api.estadisticasbcra.com/var\_usd\_anual : variación anual del dólar (porcentaje de variación de la cotización del dólar un año despues a la cotización de la fecha indicada)  https://api.estadisticasbcra.com/var\_usd\_of\_anual : variación anual del dólar oficial (porcentaje de variación de la cotización del dólar oficial un año despues a la cotización de la fecha indicada)  https://api.estadisticasbcra.com/var\_merval\_anual : variación anual del MERVAL (porcentaje de variación del MERVAL un año despues al la cotización de la fecha indicada)  https://api.estadisticasbcra.com/merval : MERVAL  https://api.estadisticasbcra.com/merval\_usd : MERVAL dividido cotización del USD |

·

|  |
| --- |
| **Ejemplos reales** (con token) |
| Crea una cuenta y a trabajar **https://estadisticasbcra.com/api/registracion** |
| .   |  |  | | --- | --- | |  |  | |  |  | |  |  |   . |
| import json  import requests  def bcra():  # Banco Central de la Republica Argentina  # api con respuesta con token  #http://estadisticasbcra.com/api/documentacion")  token="""BEARER xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"""  puntosfinales = [ **"usd", "usd\_of", "usd\_of\_minorista", "var\_usd\_vs\_usd\_of"**  ]  for **endpoint** in puntosfinales:  print ("\*"\*50)  **url** = f"https://api.estadisticasbcra.com/{**endpoint**}"  print (f"{**url**.center(50)}")  **headers** = {"Authorization": token}  **res** = requests.get(**url**, headers=**headers**)  print(f"{**res**.status\_code=} {type(**res**.status\_code)=}")  if **res**.status\_code != 200:  print('Respuesta Error de coneccion (no importa cual)')  exit()  elif **res**.status\_code==200:  print(f"{**res**.json()[0]=}")  print(f"{**res**.json()[-1]=}") |
| Requests  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  https://api.estadisticasbcra.com/usd  res.status\_code=200 type(res.status\_code)=<class 'int'>  res.json()[0]={'d': '2000-05-24', 'v': 1.0005}  res.json()[-1]={'d': '2024-02-01', 'v': 1195}  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  https://api.estadisticasbcra.com/usd\_of  res.status\_code=200 type(res.status\_code)=<class 'int'>  res.json()[0]={'d': '2002-03-04', 'v': 2.01}  res.json()[-1]={'d': '2024-02-02', 'v': 827.35}  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  https://api.estadisticasbcra.com/usd\_of\_minorista  res.status\_code=200 type(res.status\_code)=<class 'int'>  res.json()[0]={'d': '2010-06-01', 'v': 3.96}  res.json()[-1]={'d': '2024-02-02', 'v': 876.62}  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  https://api.estadisticasbcra.com/var\_usd\_vs\_usd\_of  res.status\_code=200 type(res.status\_code)=<class 'int'>  res.json()[0]={'d': '2002-03-04', 'v': 5.4726}  res.json()[-1]={'d': '2024-02-01', 'v': 44.5244} |

.