

Tema	WIDGETS DE ESCRITORIO	
Descripción de la clase	Comprensión de las categorías de las apps de software, los niños aprenden a reestructurar el código para crear widgets como una extensión de los softwares de apps, centrándose más en el comportamiento del usuario.	
Clase	AVANZADO-C183	
Duración de la clase	55 minutos.	
Objetivo	<ul style="list-style-type: none"> Desarrollar un widget meteorológico basado en una Interfaz Gráfica de Usuario (GUI) utilizando el concepto de interfaz de programación de apps (API). 	
Recursos requeridos	<ul style="list-style-type: none"> Recursos para las maestras: <ul style="list-style-type: none"> Usa tus credenciales de inicio de sesión de Gmail. Audífonos con micrófono. Libreta y pluma. Recursos para los alumnos. <ul style="list-style-type: none"> Usa tus credenciales de inicio de sesión de Gmail. Audífonos con micrófono (opcional). Libreta y pluma. 	
Estructura de la clase	Rompiendo el hielo. Actividad dirigida por la maestra. Actividad dirigida por el alumno. Conclusión.	5 minutos 15 minutos 35 minutos 5 minutos
Pasos de la clase	Decir	Hacer
Paso 1: Rompiendo el hielo (5 minutos)	En la clase anterior, continuamos creando la app de asistencia de escritorio y agregamos la funcionalidad de abrir sitios web como "google.com" y "youtube.com";	Abre la Actividad de la maestra 2 , descarga el archivo weather_application.py y ejecútalo en Spyder antes

también abrir apps de escritorio como el editor de texto (bloc de notas en Windows y el editor de texto en MAC) a través de un comando de voz y también a agregar la GUI a la app.

¡Tengo una pregunta emocionante para ti! ¿Estás listo para responderla?

P: ¿Qué función usamos para abrir la app del bloc de notas?

R: usamos la función **subprocess.Popen()** para abrir la app de bloc de notas (para Windows).

Usamos la función **subprocess.call()** para abrir la app de bloc de notas (para Mac).

de la clase para verificar que todo funcione bien.

Haz clic en el botón



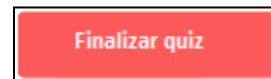
en la esquina inferior derecha de la pantalla para iniciar el quiz en clase.

Un quiz será visible tanto para ti como para el alumno.

Anima al alumno a responder la pregunta del quiz.

El alumno puede elegir la opción incorrecta, ayuda al alumno a pensar correctamente sobre la pregunta y luego responder nuevamente.

Después de que el alumno selecciona la opción correcta, el botón



comenzará a aparecer en tu pantalla.

	<p>P: ¿Cuál era la funcionalidad de la función Speech() que habíamos creado en el código?</p> <p>R: la funcionalidad de la función speech() era convertir los datos de texto a voz.</p> <p>Ahora que respondiste las preguntas y tienes claros los temas que aprendimos en la última clase, pasaremos a nuestro siguiente tema.</p>	<p>Haz clic en Finalizar quiz para cerrar la ventana emergente del quiz y continuar con la clase.</p> <p>Sigue el flujo de la clase:</p> <ul style="list-style-type: none"> - Explica al alumno todo según el documento. - Permite que el alumno copie el código predefinido que se le proporcionó desde la Actividad del alumno 2 - Comienza a agregar el código para agregar la funcionalidad para enviar la solicitud de API y obtener los informes del clima y la humedad con formato JSON de la API.
La maestra comienza a compartir pantalla.		
<p>Paso 2: Actividad dirigida por la maestra (15 minutos)</p>	<p>En la clase de hoy, crearemos un widget meteorológico en vivo utilizando el concepto de API, que nos brindará los detalles del clima y la humedad de una ciudad en particular.</p> <p>Entonces, un widget es básicamente una app de software simple y fácil de usar que podemos encontrar en</p>	

nuestra computadora de escritorio o dispositivos móviles.

Un widget es diferente a una app porque se usa principalmente para mostrar algún tipo de información al usuario. Al igual que un widget de calendario, solo muestra la fecha de hoy, pero para usar más funcionalidades, debe abrir la app del calendario y luego configurar los eventos del día o configurar recordatorios, etc.

Por lo tanto, el widget es como un banner o una etiqueta que muestra información.

¿Recuerdas que en JavaScript habíamos creado la app para desarrollar el clima y la app usando la API?

Ya que obtuvimos la API del sitio web <https://openweathermap.org/> para obtener la temperatura de la ciudad ingresada y también habíamos ubicado esa ciudad en el mapa de Google.

Hoy vamos a construir la misma app usando el concepto de API en Python. Pero generaremos el informe meteorológico que brinda la información meteorológica de una ciudad junto con la humedad en porcentaje.

Pero antes de comenzar a programar la app, revisemos qué es una API.

P: ¿Puedes decirme qué es una API?

R: API significa interfaz de programación de apps. Básicamente, es un conjunto de funciones que consta de procedimientos o métodos para interactuar con otra computadora o sistema para obtener básicamente los datos requeridos.

Entendamos esto con un ejemplo:

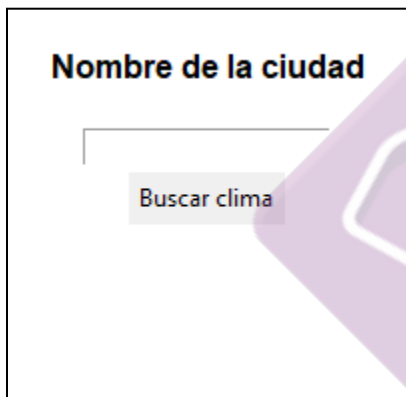
Imagina que estás sentado en una mesa en un restaurante con una carta de menú. La cocina es una parte del “sistema” que preparará un pedido para ti. Pero, ¿cómo te conectarás a este sistema, el cual es la cocina, para conseguir tu comida?

Ahí es donde el mesero, o podemos decir que la API, entra en escena. Entonces, esta API o mesero es el enlace más importante, o podemos decir un mensajero, que toma tu solicitud u orden y le dice a la cocina, o al sistema que prepare la comida según tu orden, o procese los datos según tu solicitud y le entregue la comida ordenada, o datos.

NOTA: abre la [Actividad de la maestra 2](#), descarga el archivo [weather_application.py](#), ejecútalo en Spyder y demuestra el output según las instrucciones a continuación.

Permíteme enseñarte la demostración de la app que crearemos hoy.

Esta es la GUI del widget del clima. Puedes ver esto en la esquina superior izquierda de la pantalla.

A screenshot of a weather application's GUI. It features a title "Nombre de la ciudad" in bold black text. Below the title is a text input field with a light gray border. To the right of the input field is a button with the text "Buscar clima" in a light gray font. The entire GUI is enclosed in a black rectangular border. A large, faint, diagonal watermark reading "BYJU'S FUTURE SCHOOL" is visible across the background of the screenshot.

Tenemos que ingresar cualquier nombre de una ciudad en el cuadro de entrada. De esta manera:

Nombre de la ciudad

Toronto

Buscar clima

Y hacemos clic en el botón "Buscar clima".

Toronto

Clima: Clear

Humedad: 62%

Podemos ver que obtenemos el informe de clima y humedad de Toronto, que es una ciudad en Canadá. Y si observamos bien, podemos ver que el widget de entrada y el botón se vuelven invisibles al mostrar solo el informe del clima y la humedad.

De la misma manera, también podemos buscar diferentes lugares en todo el mundo.

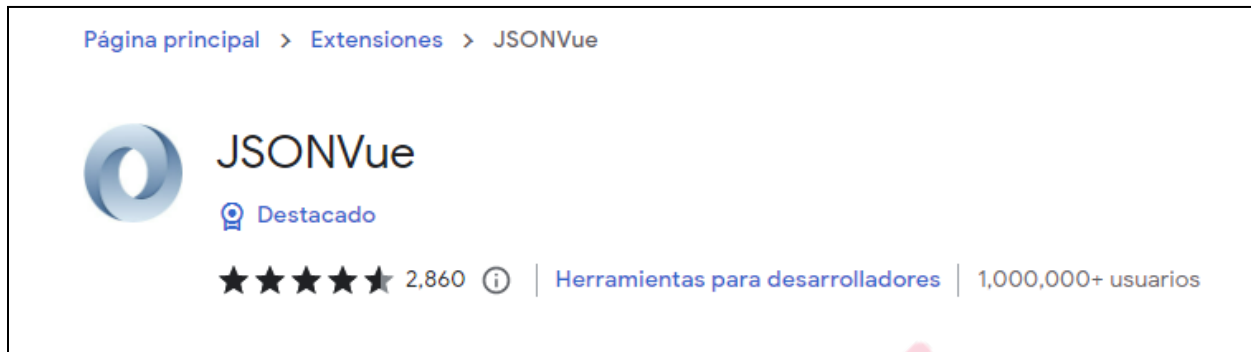
Entonces, para obtener el informe meteorológico, usaremos la API de <https://openweathermap.org/> y este enlace también se te proporcionó en la **Actividad del alumno 3**.

Llamamos a la API, obtenemos una respuesta en formato JSON. JSON es una sintaxis para almacenar e intercambiar datos. JSON es un texto, escrito con notación de objetos JavaScript.

Comprender y obtener los datos del formato JSON:

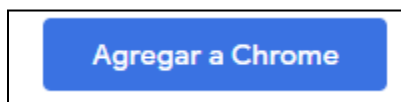
Paso 1.

- Primero abrimos la **Actividad del alumno 5**, tenemos un enlace de referencia para el visor JSON.



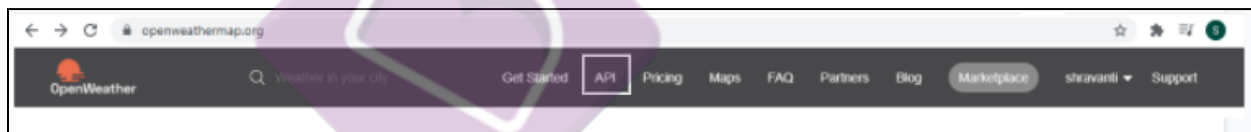
Paso 2:

- Luego, agregamos el visor JSON a Chrome simplemente haciendo clic en el botón **Agregar a Chrome**.



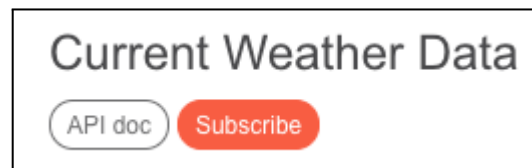
Paso 3.

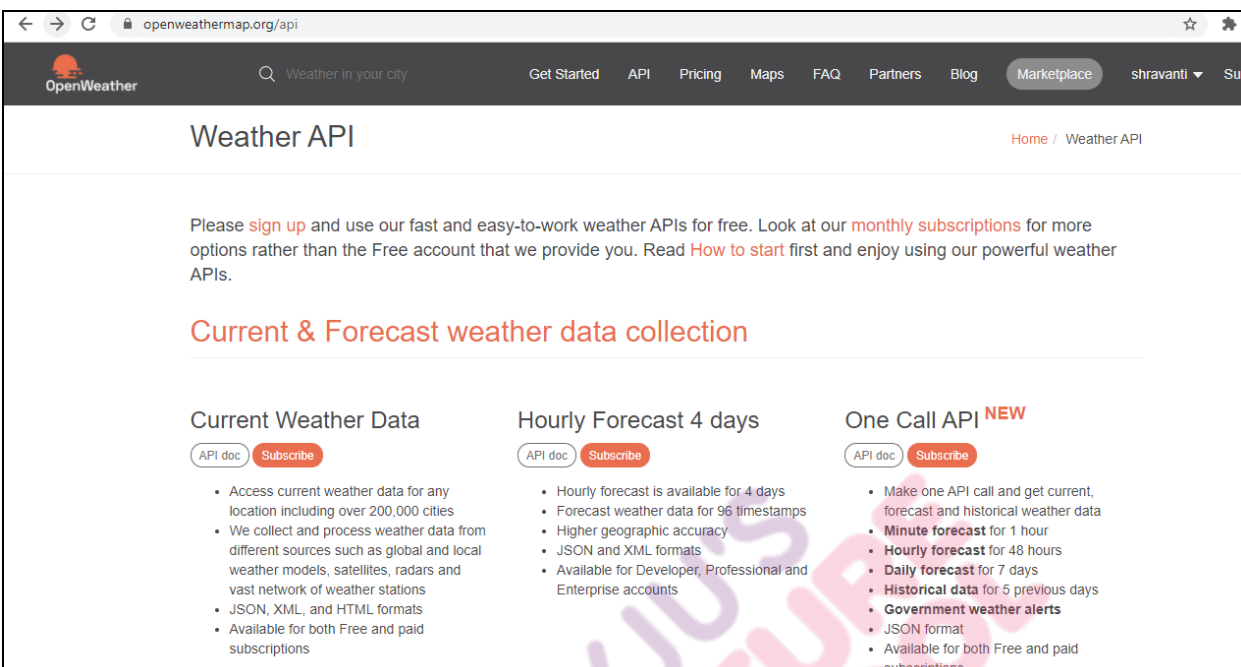
- Vamos al sitio web <https://openweathermap.org/> y hacemos clic en la pestaña API.



Paso 4.

- Luego presionamos el botón API doc debajo de Current Weather Data - (*datos meteorológicos actuales*).





Weather API

Please [sign up](#) and use our fast and easy-to-work weather APIs for free. Look at our [monthly subscriptions](#) for more options rather than the Free account that we provide you. Read [How to start](#) first and enjoy using our powerful weather APIs.

Current & Forecast weather data collection

Current Weather Data

[API doc](#) [Subscribe](#)

- Access current weather data for any location including over 200,000 cities
- We collect and process weather data from different sources such as global and local weather models, satellites, radars and vast network of weather stations
- JSON, XML, and HTML formats
- Available for both Free and paid subscriptions

Hourly Forecast 4 days

[API doc](#) [Subscribe](#)

- Hourly forecast is available for 4 days
- Forecast weather data for 96 timestamps
- Higher geographic accuracy
- JSON and XML formats
- Available for Developer, Professional and Enterprise accounts

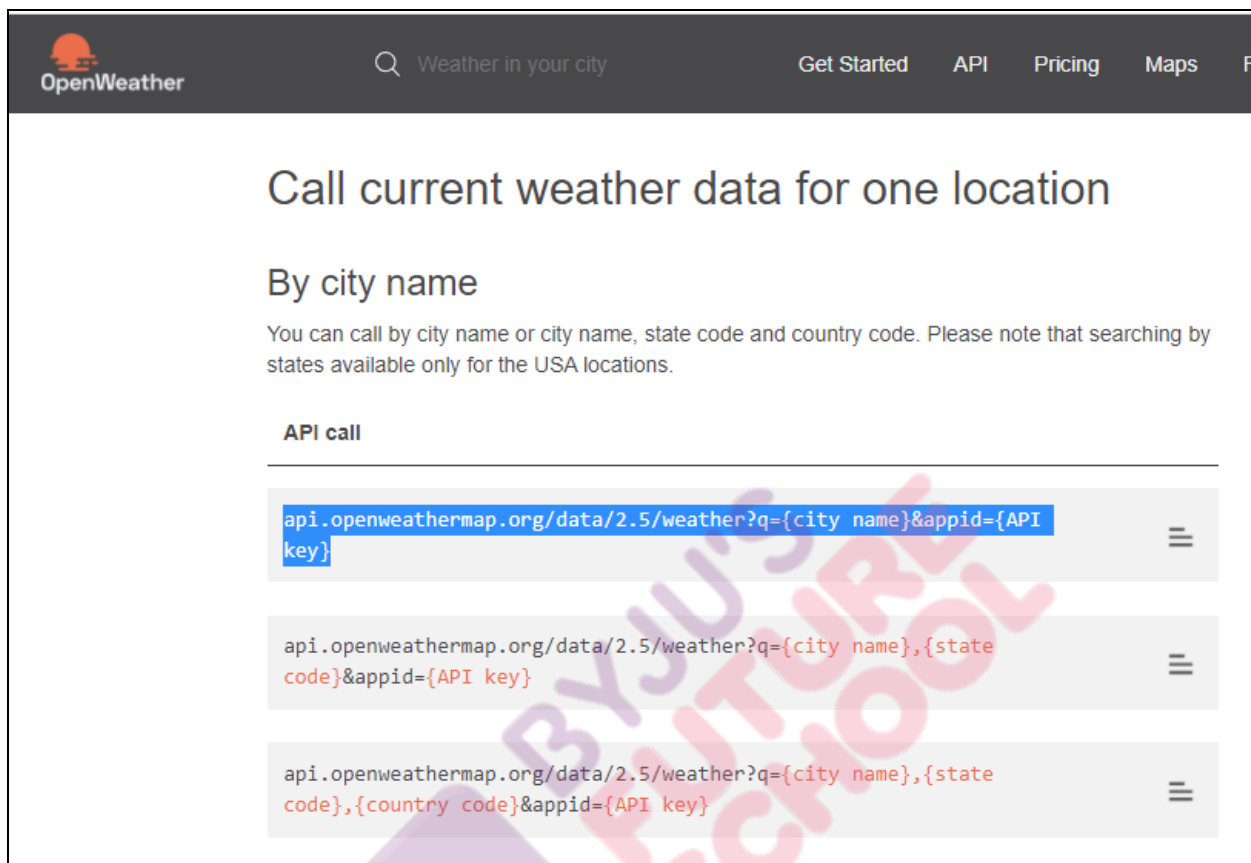
One Call API **NEW**

[API doc](#) [Subscribe](#)

- Make one API call and get current, forecast and historical weather data
- Minute forecast** for 1 hour
- Hourly forecast** for 48 hours
- Daily forecast** for 7 days
- Historical data** for 5 previous days
- Government weather alerts**
- JSON format
- Available for both Free and paid subscriptions

Paso 5.

- En esta página, nos desplazamos hacia abajo hasta las API call, donde obtendremos ciertos formatos en los que podemos llamar a la API usando los nombres de las ciudades. Así que seleccionamos el primer formato.



The screenshot shows the OpenWeather API documentation page. At the top, there's a navigation bar with the OpenWeather logo, a search bar, and links for 'Get Started', 'API', 'Pricing', and 'Maps'. The main heading is 'Call current weather data for one location'. Below it, the section 'By city name' explains that you can call by city name or city name, state code, and country code. It notes that searching by states is only available for USA locations. Under 'API call', three example URLs are listed in a code block, each with a copy icon to its right:

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
```

```
api.openweathermap.org/data/2.5/weather?q={city name},{state code}&appid={API key}
```

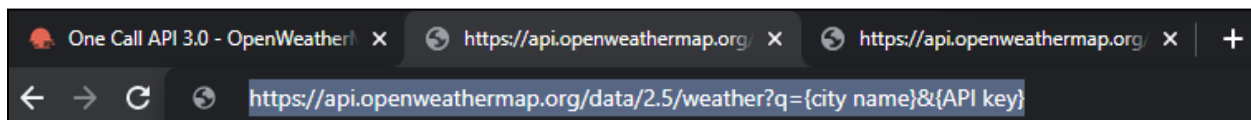
```
api.openweathermap.org/data/2.5/weather?q={city name},{state code},{country code}&appid={API key}
```

El enlace que se copia debe ser así:

["https://api.openweathermap.org/data/2.5/weather?q={city name}&appid{API key}"](https://api.openweathermap.org/data/2.5/weather?q={city name}&appid{API key})

Paso 6.

- Ahora abrimos una nueva pestaña y pasamos el enlace que copiamos.

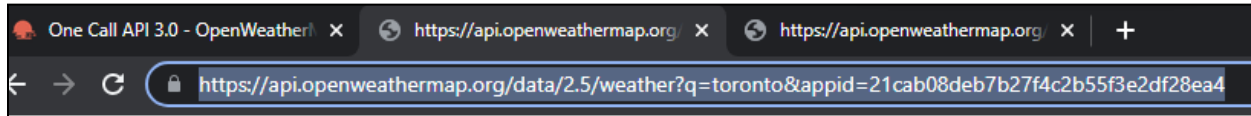


Paso 7.

- Ahora reemplazamos donde está escrito **{city_name}** con cualquier nombre de ciudad. Por ejemplo, Toronto y también reemplazamos **{api key}** con la clave API

real sin incluir los corchetes. Y presionamos enter.

La **clave API** se te da en la **Actividad del alumno 4**.



Paso 8.

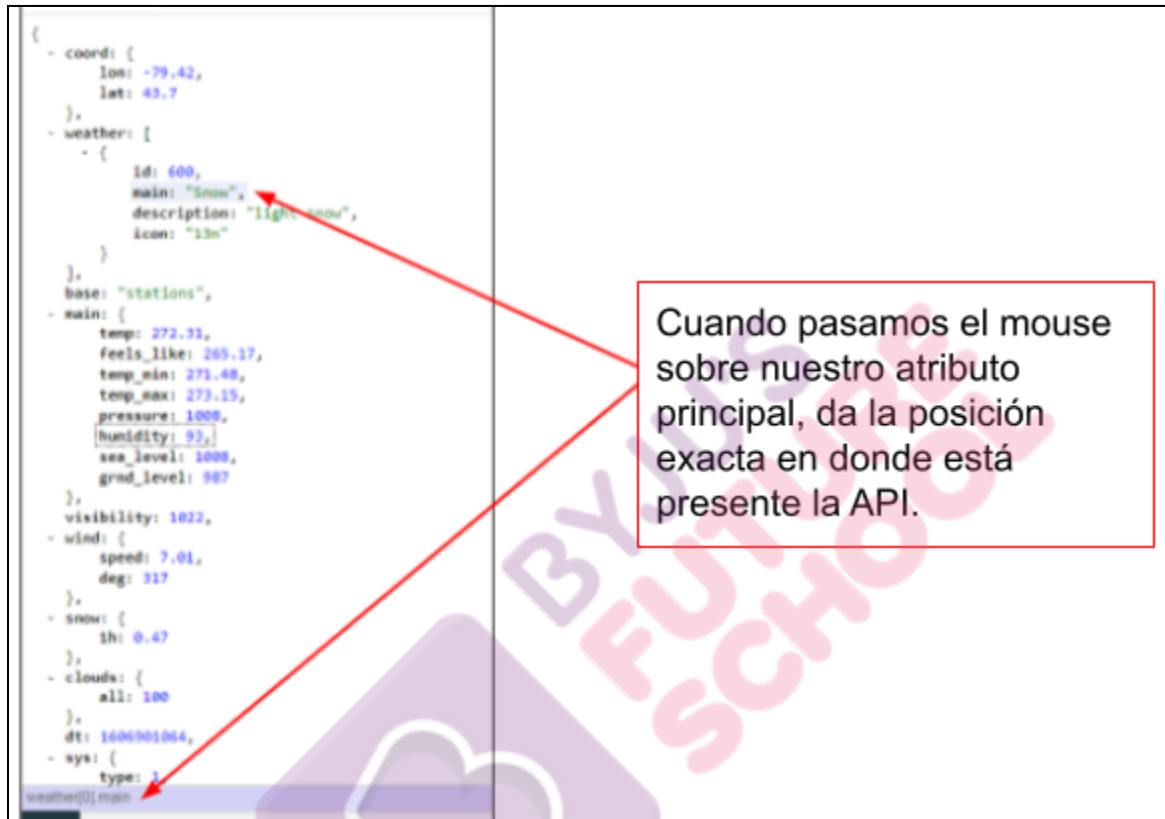
- Encontraremos la página que muestra los datos en algún diccionario y formato de lista, por lo que estos son básicamente los datos en formato JSON del clima en la ciudad de Toronto.

```
{
  - coord: {
    lon: -79.42,
    lat: 43.7
  },
  - weather: [
    - {
      id: 600,
      main: "Snow",
      description: "light snow",
      icon: "13n"
    }
  ],
  base: "stations",
  - main: {
    temp: 272.31,
    feels_like: 265.17,
    temp_min: 271.48,
    temp_max: 273.15,
    pressure: 1008,
    humidity: 93,
    sea_level: 1008,
    grnd_level: 987
  },
  visibility: 1022,
  - wind: {
    speed: 7.01,
    deg: 317
  },
  - snow: {
    1h: 0.47
  },
  - clouds: {
    all: 100
  },
  dt: 1606901064,
  - sys: {
    type: 1,
    id: 718,
```

Paso 9.

- Simplemente, desplazamos el mouse sobre los datos que deseamos, supongamos que pasamos el mouse sobre la API main - (*principal*) dentro del objeto weather - (*clima*). Luego, en la esquina inferior izquierda, encontraremos

que se muestran algunos datos que no son más que la posición de la API principal dentro del objeto weather. Entonces, al usar esto, podemos obtener las posiciones de las API y usarlas en nuestro programa.



NOTA: TODAS LAS IMÁGENES A CONTINUACIÓN ESTÁN PRESENTES EN LA ACTIVIDAD DE LA MAESTRA 1. POR FAVOR, ABRE LA ACTIVIDAD DE LA MAESTRA 1 MIENTRAS EXPLICAS.

Antes de comenzar a programar, hay dos nuevos módulos que usaremos en nuestra app.

1. **json:** Python tiene un paquete integrado llamado **json**, que se puede usar para trabajar con datos JSON.

JSON es una sintaxis para almacenar e intercambiar datos. JSON es un texto, escrito con notación de objetos JavaScript.

Como Python viene con el módulo json incorporado, no hay necesidad de instalar este módulo externamente, solo tenemos que importarlo para usar este módulo en nuestro programa.

2. **Requests** - (*solicitudes*): la biblioteca requests de Python nos permite enviar la solicitud HTTP al servidor para obtener la información requerida.

- La solicitud HTTP devuelve un objeto de respuesta con todos los datos de respuesta.

Por ejemplo:

Si envías una carta a tu amigo a través de la oficina de correos, esto se conocerá como envío de una solicitud HTTP.

Luego, cuando tu amigo responda tu carta, esto se conocerá como `.response` - (*respuesta*).

La biblioteca requests no es la biblioteca incorporada, tenemos que instalar esta biblioteca para usarla en nuestro programa. Explicaremos cómo instalarla más adelante en la clase.

Código predefinido:

```
1  from tkinter import *
2  root=Tk()
3  root.title("Mi app meteorológica")
4  root.geometry("350x300")
5
6  root.configure(background="white")
7  #Etiquetas de configuración
8  city_name_label=Label(root, text="Nombre de la ciudad",font=("Helvetica", 12,'bold'),bg="white")
9  city_name_label.place(relx=0.28, rely=0.15, anchor=CENTER)
10
11  city_entry=Entry(root)
12  city_entry.place(relx=0.28, rely=0.35, anchor=CENTER)
13
14  weather_info_label = Label(root, bg="white", font=("bold", 10))
15  weather_info_label.place(relx=0.23, rely=0.6, anchor=CENTER)
16
17  humidity_info_label = Label(roo, bg="white", font=( "bold",10))
18  humidity_info_label.place(relx=0.22, rely=0.7, anchor=CENTER)
19
20  root.mainloop()
```

Nota: el código predefinido anterior se proporciona al alumno en la actividad del alumno 2. El alumno debe descargarlo, guardarlo como `weather_application.py` y comenzar a agregar el código resaltado en el código predefinido.

Código completo:

```

1  from tkinter import *
2  import requests
3  import json
4
5  root=Tk()
6  root.overrideredirect(True)
7  root.title("Mi app meteorologica")
8
9  root.geometry("200x200")
10 root.configure(background="white")
11 #Etiquetas de configuración
12 city_name_label=Label(root, text="Nombre de la ciudad",font=("Helvetica", 12,'bold'),bg="white")
13 city_name_label.place(relx=0.5,relx=0.15,anchor=CENTER)
14
15 city_entry=Entry(root)
16 city_entry.place(relx=0.5,relx=0.35,anchor=CENTER)
17
18 weather_info_label = Label(root, bg="white", font=("bold", 10))
19 weather_info_label.place(relx=0.5,relx=0.38,anchor=CENTER)
20
21 humidity_info_label = Label(root, bg="white", font=("bold",10))
22 humidity_info_label.place(relx=0.5,relx=0.5,anchor=CENTER)
23
24 #Función principal
25 def city_name():
26     api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" +
27     city_entry.get() + "&appid=" + "21cab08deb7b27f4c2b55f3e2df28ea4")
28
29     api_output_json = json.loads(api_request.content)
30
31     weather_info=api_output_json['weather'][0]['main']
32     print(weather_info)
33
34     humidity=api_output_json['main']['humidity']
35     print(str(humidity) + "%")
36
37     weather_info_label["text"]= "Clima: " + str(weather_info)
38     humidity_info_label["text"]="Humedad: " + str(humidity) + "%"
39
40     city_name_label["text"]= city_entry.get()
41     city_entry.destroy()
42     search_btn.destroy()
43
44     search_btn=Button(root, text="Buscar clima", command=city_name, relief=FLAT)
45     search_btn.place(relx=0.5,relx=0.48,anchor=CENTER)
46     root.mainloop()
47

```

1) Luego de escribir la plantilla básica del módulo tkinter y de importar el módulo tkinter, en el código predefinido vamos a importar dos módulos más requeridos para nuestra app. Ese es el módulo **requests** y el módulo **json**.

- Primero importaremos `import requests`: como tenemos que obtener los datos del sitio web, primero debemos enviar la solicitud a ese sitio web para acceder a su contenido requerido para generar el informe meteorológico. Entonces, como requerimos los datos del clima y la humedad de la API, <https://openweathermap.org/>, enviaremos la solicitud a este sitio web. Entonces, para enviar la solicitud a este sitio web, hay un

módulo llamado **request** que necesitamos instalar en nuestro sistema, el cual te mostraré más adelante, por ahora solo necesitamos importar este módulo para acceder a él en nuestro programa.

- A continuación, importamos el módulo `import json`: json es el paquete integrado de Python que se utiliza para obtener y formatear los datos, lo que también ayuda a almacenar e intercambiarlos. Como la información que vamos a utilizar en nuestra app está presente en formato JSON en el sitio web, y como tenemos que acceder y obtener esos datos básicamente para cargarlos en nuestro programa, necesitamos importar el módulo **json** que es la notación de objetos JavaScript.

- 2) Luego, después de crear la ventana root - (*ventana raíz: nombre propio de la interfaz principal de un programa*), agregamos este código

`root.overrideRedirect(True)`. Llegaremos a este código después de completar la programación de nuestra app.

- 3) Luego, después del código predefinido escrito para colocar la etiqueta de humedad en la ventana root, definiremos una función **city_name()** usando la palabra clave **def** `def city_name():`. La funcionalidad de esta función es acceder a los informes meteorológicos desde el sitio web <https://openweathermap.org/> presente en la API con formato JSON, utilizando la clave de API única del usuario y mostrar estos contenidos API en nuestra app GUI.

- 4) Así que dentro de la función **city_name()** `def city_name():`, primero hacemos una solicitud al sitio web desde el que tenemos que acceder a los datos. Entonces escribimos el código para:

1. Acceder a los datos del informe meteorológico desde el sitio web <https://openweathermap.org/>, primero debemos realizar una solicitud a este sitio web.

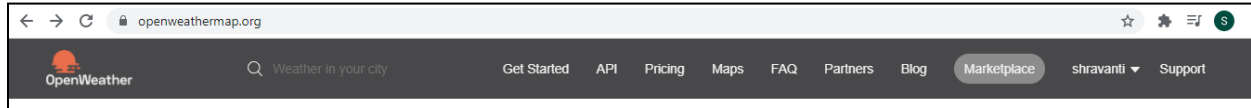
P: ¿Qué quiere decir enviar una solicitud al sitio web?

R: cuando queremos acceder a un dato particular presente en cualquier servidor, tenemos que hacer una solicitud a ese sitio web para acceder a la información. Y esa información procesada la obtenemos en forma de respuesta.

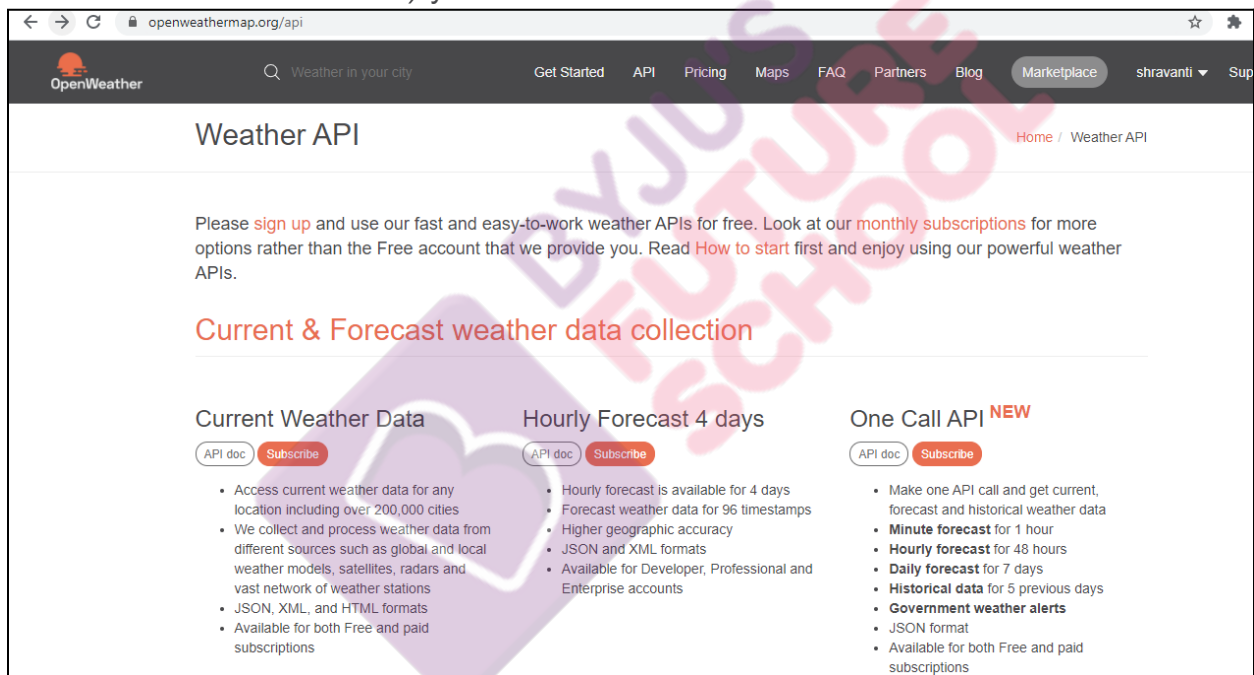
Entonces, para hacer una solicitud a este sitio web, usamos la función **request.get()** del módulo **request**.

Ahora que tenemos que hacer la solicitud al sitio web, primero obtendremos la URL. Entonces, para obtener la URL,

- Primero visitaremos el sitio web <https://openweathermap.org/>, el enlace del sitio web se te proporciona en la **Actividad del alumno 3**.
- Después de iniciar sesión, hacemos clic en la pestaña de la **API** desde la barra de navegación.



- Luego vamos a Current Weather Data - (*Datos meteorológicos actuales*) y hacemos clic en API doc.



- Luego, de los formatos enumerados de la llamada API, seleccionamos el primer formato de la URL, de esta manera:

URL de formato de llamada API:

`api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}`

- Ahora, si vemos esta URL tiene un formato determinado que nos permite ingresar el nombre de una ciudad y la clave API.
- Así que hacemos una solicitud a esta URL, escribiéndola dentro de la función **request.get()**, de esta manera:

```
requests.get(api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key})
```

- Ahora desglosamos este enlace:
 - Y primero agregamos **https://** al comienzo de este enlace. Así:

```
requests.get("https://api.openweathermap.org/data/2.5/weather?q="
```

- Como tenemos que encontrar el informe meteorológico para varios países tomando la entrada del usuario, reemplazaremos **{city**

name} - (nombre de la ciudad) con **city_entry.get()**, que es el elemento de entrada que ya está programado en el código predefinido. Y concatenarlo con el signo "+" al enlace que obtuvimos. Así:

```
requests.get("https://api.openweathermap.org/data/2.5/weather?q=" + city_entry.get())
```

- Ahora nuevamente usando el signo "+" concatenamos la cadena de caracteres "&appid=" a la URL de esta manera:

```
requests.get("https://api.openweathermap.org/data/2.5/weather?q=" + city_entry.get() + "&appid="+
```

- Y nuevamente usando el símbolo "+" agregamos la clave API que obtuvimos cuando iniciamos sesión en nuestra cuenta de openweathermap y que se te proporciona en la **Actividad del alumno 4**. De esta manera:

```
requests.get("https://api.openweathermap.org/data/2.5/weather?q=" + city_entry.get() + "&appid="+  
"21cab08deb7b27f4c2b55f3e2df28ea4")
```

- Y luego almacenamos esta solicitud realizada a la URL en la variable **api_request**. De esta manera:

```
def city_name():  
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" + city_entry.get() +  
"&appid="+ "21cab08deb7b27f4c2b55f3e2df28ea4")
```

2. Ahora que realizamos la solicitud al sitio web para que nos dé acceso a los datos requeridos. Carguemos los datos específicos que se requieren en nuestra app.

- Entonces, como vimos que estos datos están presentes en el formato JSON, necesitamos recuperar esta información de ese formato. Para esto cargaremos el contenido de la API usando la función **loads()** del módulo

JSON accediendo a ella, así `json.loads()` y dentro de la misma pasamos la variable **api_request** que contiene la URL de la API

```
json.loads(api_request.content)
```

Y la almacenamos en la variable **api_output_json**.

```
#Función principal  
def city_name():  
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" +  
city_entry.get() + "&appid="+ "21cab08deb7b27f4c2b55f3e2df28ea4")  
  
    api_output_json = json.loads(api_request.content)
```

3. Obtenemos el clima con los datos JSON.

```
#Función principal
def city_name():
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" +
    city_entry.get() + "&appid=" + "21cab08deb7b27f4c2b55f3e2df28ea4")

    api_output_json = json.loads(api_request.content)

    weather_info=api_output_json['weather'][0]["main"]
```

- Como sabemos, las respuestas de API están en formato JSON. De esta manera:



```
{
  - coord: {
    lon: -79.42,
    lat: 43.7
  },
  - weather: [
    - {
      id: 600,
      main: "Snow",
      description: "light snow",
      icon: "13n"
    }
  ],
  base: "stations",
  - main: {
    temp: 272.31,
    feels_like: 265.17,
    temp_min: 271.48,
    temp_max: 273.15,
    pressure: 1008,
    humidity: 93,
    sea_level: 1008,
    grnd_level: 987
  },
  visibility: 1022,
  - wind: {
    speed: 7.01,
    deg: 317
  },
  - snow: {
    1h: 0.47
  },
  - clouds: {
    all: 100
  },
  dt: 1606901064,
  - sys: {
    type: 1,
    id: 718,
```

- Entonces, a partir de esto, tenemos que obtener el atributo main, ya que contiene la condición del clima.
- Entonces, para obtener esta información, la escribiremos en el formato como `api_output_json['weather'][0]["main"]`.
 - Primero escribiremos `api_output_json` porque esta variable contiene la respuesta JSON completa de la API.
 - Luego escribiremos `["weather"]` - (*clima*) como atributo principal

dentro del objeto weather.

- Como el clima está en formato de lista, y el atributo main está en el primer índice de la lista, para acceder al primer índice de la lista escribimos 0, por lo tanto, escribiremos **[“weather”][0]**.
- Ahora solo escribimos **[“main”]** - (*principal*) después de [0] y esto nos dará el contenido del clima presente en main.
- Y lo guardamos dentro de una variable, de esta manera:

```
weather_info=api_output_json['weather'][0]['main']
```

- También imprimiremos el valor del clima en la consola usando la función print(), pasando la variable weather_info dentro de la función print(), así:

```
print(weather_info)
```

```
def city_name():  
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" + city_entry.get() +  
    "&appid="+ "21cab08deb7b27f4c2b55f3e2df28ea4")  
  
    api_output_json = json.loads(api_request.content)  
  
    weather_info=api_output_json['weather'][0]['main']  
    print(weather_info)
```

4. Ahora tenemos que obtener la humedad de la respuesta de JSON.

```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 282.55,
    "feels_like": 281.86,
    "temp_min": 280.37,
    "temp_max": 284.26,
    "pressure": 1023,
    "humidity": 100
  },
  "visibility": 16093,
  "wind": {
    "speed": 1.5,
    "deg": 350
  }
}
```

Entonces, para obtener esta información, la escribiremos en el formato como: `api_output_json['main']['humidity']`.

- Primero escribiremos `api_output_json` ya que esta variable contiene la respuesta JSON completa de la API.
- Luego escribiremos `["main"]` puesto que el atributo **'humidity'** - (*humedad*) está dentro del objeto **main**.
- Como **'main'** está en formato de diccionario, todos los atributos dentro de **'main'** están presentes en pares clave-valor. Entonces podemos acceder al valor del atributo **'humidity'** pasando directamente la clave **'humidity'** como una clave para el objeto **'main'**, por lo tanto, lo escribiremos como `['main']['humidity']`.
- Ahora solo accedemos a este atributo **'humidity'** presente en el objeto **'main'** mediante el uso de la variable `api_output_json` y esto nos dará el contenido de la humedad presente en el objeto **main**.

- e. También imprimimos la humedad en la consola usando la función `print()` y concatenamos la humedad obtenida con el signo de porcentaje "%", de esta manera:

```
print(str(humidity) + "%")
```

```
def city_name():
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" + city_entry.get() +
    "&appid="+ "21cab08deb7b27f4c2b55f3e2df28ea4")

    api_output_json = json.loads(api_request.content)

    weather_info=api_output_json['weather'][0]["main"]
    print(weather_info)

    humidity=api_output_json['main']['humidity']
    print(str(humidity) + "%")
```

5. Ahora actualizamos el parámetro "text" de la etiqueta **weather_info_label** con el informe meteorológico que obtuvimos de la API. Para eso, primero escribimos una cadena de caracteres "Clima: ", damos un símbolo "+" para concatenarlo con el valor del clima escribiendo así `str(weather_info)`. De esta manera:

```
weather_info_label["text"] = "Clima: " + str(weather_info)
```

```
def city_name():
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" +
    city_entry.get() + "&appid="+ "21cab08deb7b27f4c2b55f3e2df28ea4")

    api_output_json = json.loads(api_request.content)

    weather_info=api_output_json['weather'][0]["main"]
    print(weather_info)

    humidity=api_output_json['main']['humidity']
    print(str(humidity) + "%")

    weather_info_label["text"] = "Clima: " + str(weather_info)
```

6. Ahora actualizamos el parámetro "text" de la etiqueta **humidity_info_label** con la humedad que recuperamos de la API. Entonces, para eso, primero escribimos una cadena de caracteres "Humedad: ", damos un signo "+" para concatenarla con el valor de humedad, escribiendo así `str(humidity)` y convertimos esta humedad en cadena de caracteres escribiéndola dentro de la función `str()` y nuevamente concatenamos con "%", ya que la humedad se da en

porcentajes. De esta manera:

```
humidity_info_label["text"]="Humedad: "+ str(humidity) + "%"
```

```
def city_name():
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" +
    city_entry.get() + "&appid="+ "21cab08deb7b27f4c2b55f3e2df28ea4")

    api_output_json = json.loads(api_request.content)

    weather_info=api_output_json['weather'][0]['main']
    print(weather_info)

    humidity=api_output_json['main']['humidity']
    print(str(humidity) + "%")

    weather_info_label["text"]= "Clima: " + str(weather_info)
    humidity_info_label["text"]="Humedad: "+ str(humidity) + "%"
```

7. Como estamos creando un widget, podemos ver en la demostración que cuando ingresamos el nombre de una ciudad en el campo de entrada, este y el botón desaparecen y solo obtenemos el nombre de la ciudad que ingresamos, el clima y el informe de humedad de la ciudad ingresada.

Básicamente, aquí estamos eliminando o removiendo el campo de entrada y el botón antes de mostrar el output.

La razón para eliminar el campo de entrada y el botón es que normalmente en los widgets los vemos, solo tenemos algo de información. Es por eso que estamos eliminándolos, después de configurar el nombre de la ciudad. Entonces, para esto, simplemente agregamos el siguiente código en la función city_name

- Actualizamos el parámetro de texto de city_name_label con la ciudad que ingresamos en el campo de entrada. Entonces

escribimos: `city_name_label["text"]= city_entry.get()`. Esto elimina el texto **"Nombre de la ciudad"** de la etiqueta **city_name_label** y lo actualiza con el nombre de la ciudad ingresado en el campo de entrada.

- A continuación, como queremos ocultar el campo de entrada de la ventana root antes de mostrar el informe meteorológico, usaremos la función **destroy()** para eliminar el campo de entrada **city_entry**

de esa manera: `city_entry.destroy()`, esto reemplazará el informe meteorológico.

- También queremos ocultar el elemento del botón de la ventana root, tan pronto como hagamos clic en él y muestre el resultado, por lo que volveremos a usar la función **destroy()** para eliminar el elemento del botón de la ventana root, de esta manera:

```
search_btn.destroy()
```

```
def city_name():
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" +
    city_entry.get() + "&appid=" + "21cab08deb7b27f4c2b55f3e2df28ea4")

    api_output_json = json.loads(api_request.content)

    weather_info=api_output_json['weather'][0]["main"]
    print(weather_info)

    humidity=api_output_json['main']['humidity']
    print(str(humidity) + "%")

    weather_info_label["text"] = "Clima: " + str(weather_info)
    humidity_info_label["text"] = "Humedad: " + str(humidity) + "%"

    city_name_label["text"] = city_entry.get()
    city_entry.destroy()
    search_btn.destroy()
```

Ahora crearemos un elemento de botón usando la clase **Button()** en la ventana root

```
def city_name():
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" +
    city_entry.get() + "&appid=" + "21cab08deb7b27f4c2b55f3e2df28ea4")

    api_output_json = json.loads(api_request.content)

    weather_info=api_output_json['weather'][0]["main"]
    print(weather_info)

    humidity=api_output_json['main']['humidity']
    print(str(humidity) + "%")

    weather_info_label["text"] = "Clima: " + str(weather_info)
    humidity_info_label["text"] = "Humedad: " + str(humidity) + "%"

    city_name_label["text"] = city_entry.get()
    city_entry.destroy()
    search_btn.destroy()

search_btn=Button(root, text="Buscar clima", command=city_name, relief=FLAT)
```

Establecemos los parámetros del botón en el siguiente orden:

- **text = "Buscar clima"** para que aparezca en el botón.
- **command = city_name** para llamar a la función que creamos para acceder a las API.
- **relief=FLAT** para que el botón se vea plano.
- Y almacenamos este botón en una variable **search_btn**.

```
search_btn=Button(root, text="Buscar clima", command=city_name, relief=FLAT)
```


Ahora colocamos este botón en la ventana root usando la función **place()**.

Tenemos que colocar este **search_btn** exactamente debajo del elemento de entrada.

P: ¿Puedes adivinar el valor **relx** para colocar el botón en la ventana root?

R: colocaremos el elemento del botón en **relx= 0.5**, ya que debería colocarse en la misma alineación que el elemento de entrada.

P: ¿Cuál debería ser el valor de **rely** para colocar el elemento de botón en la ventana root?

R: podemos colocar el elemento del botón en **rely = 0.48**, que está exactamente debajo del elemento de entrada.

Y para que este botón aparezca en la alineación adecuada, establecemos **anchor** en

CENTER `search_btn.place(relx=0.5, rely=0.48, anchor=CENTER)`.

```
def city_name():
    api_request = requests.get("https://api.openweathermap.org/data/2.5/weather?q=" +
    city_entry.get() + "&appid="+ "21cab08deb7b27f4c2b55f3e2df28ea4")

    api_output_json = json.loads(api_request.content)

    weather_info=api_output_json['weather'][0]["main"]
    print(weather_info)

    humidity=api_output_json['main']['humidity']
    print(str(humidity) + "%")

    weather_info_label["text"] = "Clima: " + str(weather_info)
    humidity_info_label["text"] = "Humedad: " + str(humidity) + "%"

    city_name_label["text"] = city_entry.get()
    city_entry.destroy()
    search_btn.destroy()

search_btn=Button(root, text="Buscar clima", command=city_name, relief=FLAT)
search_btn.place(relx=0.5, rely=0.48, anchor=CENTER)
```

Ahora que completamos la programación de nuestra app meteorológica. ¿Qué hay de convertirlo en un widget?

P: ¿Sabes qué es un widget?

R: un widget es una app de software simple y fácil de usar, que podemos encontrar en el escritorio de nuestra computadora o en el celular en forma de un reloj, una calculadora o una brújula, una app meteorológica o una nota adhesiva que resulta útil siempre que lo necesitemos.

Entonces, un widget es una pequeña ventana que tiene ciertas funcionalidades y no viene con ninguna opción de cierre, maximización o minimización como otras apps.

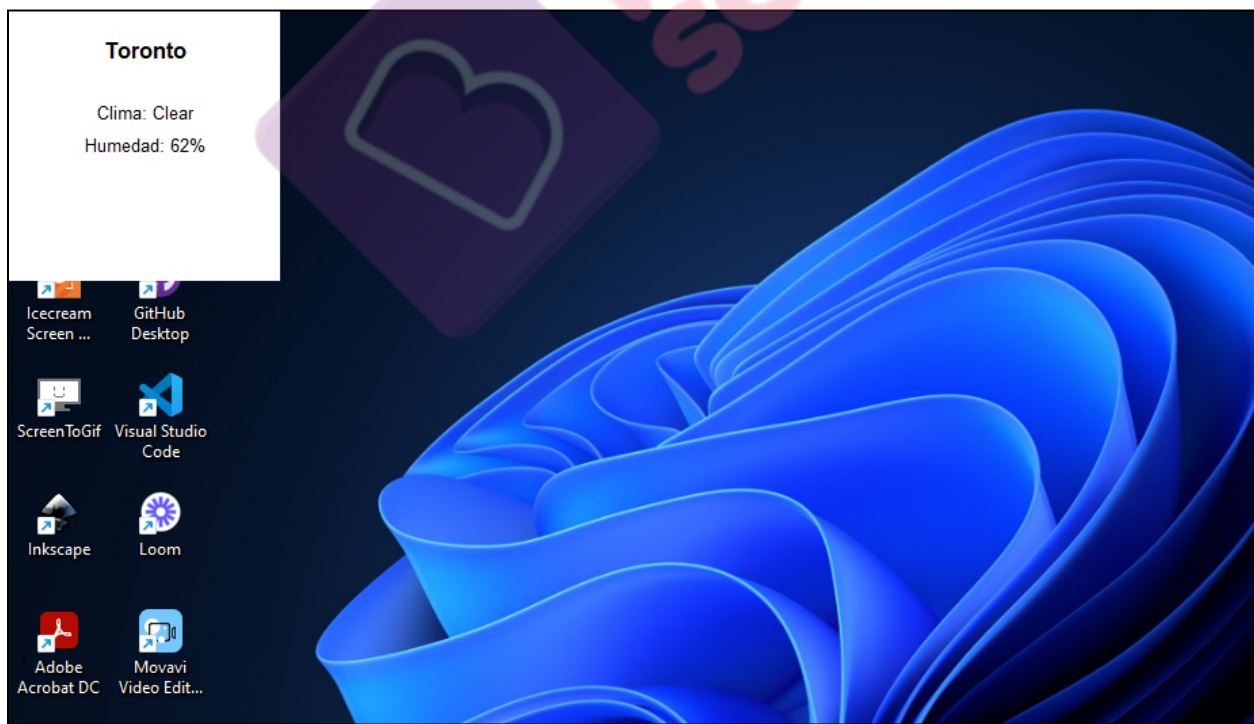
Entonces, convertiremos esta app en un widget simplemente agregando un código de una sola línea que es `root.overrideRedirect(True)`.

```
1 from tkinter import *
2 import requests
3 import json
4
5 root=Tk()
6 root.overrideRedirect(True)
7 root.title("Mi app meteorológica")
8
9 root.geometry("200x200")
10 root.configure(background="white")
11 #Etiquetas de configuración
12 city_name_label=Label(root, text="Nombre de la ciudad",font=("Helvetica", 12,'bold'),bg="white")
13 city_name_label.place(relx=0.5,rely=0.15,anchor=CENTER)
14
15 city_entry=Entry(root)
16 city_entry.place(relx=0.5,rely=0.35,anchor=CENTER)
```

Básicamente, estamos usando la **función** `root.overrideRedirect()` para deshabilitar los botones de minimizar, maximizar y cerrar la ventana, básicamente toda la barra de título de la ventana root.

Entonces, después de agregar `root.overrideRedirect()`, lo configuramos como **True** - (verdadero), simplemente pasando **True** dentro de la función, de esta manera:

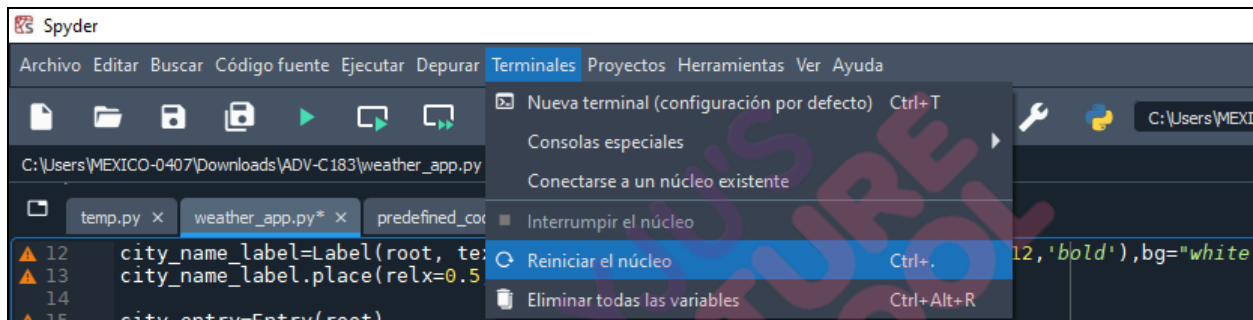
`root.overrideRedirect(True)`, obtenemos el siguiente output.



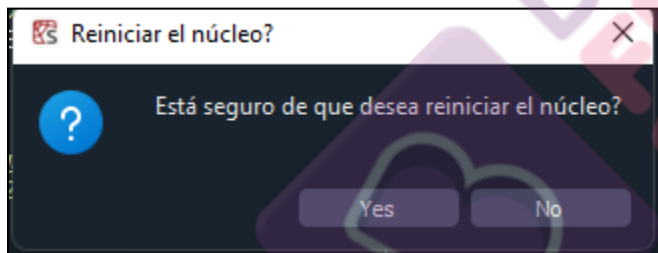
Podemos ver cómo se muestra la app meteorológica en la esquina superior izquierda de la ventana del escritorio.

Entonces, como no hay un botón de cierre, cuando tengamos que cerrar esta app, simplemente iremos a terminales y reiniciaremos el núcleo, de esta manera la app se cerrará cuando esté encendida.

Vamos a terminales >> Reiniciar el núcleo.



Hacemos clic en yes:



Se cerrará el widget.

La maestra deja de compartir pantalla.

Ahora es tu turno.

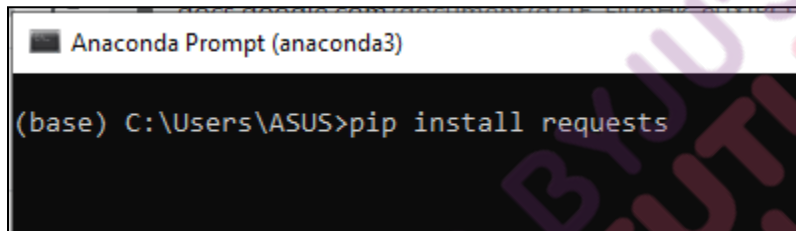
- Pide al alumno que presione ESC para regresar al panel.
- Guía al alumno para que comience a compartir pantalla.
- La maestra entra a pantalla completa.

Paso 3: Actividad dirigida por el alumno (10 minutos)	Pide al alumno que: <ul style="list-style-type: none"> - Instale requests. <p>Las instrucciones para la instalación se mencionan a continuación.</p>	<p>Cuando el alumno comparta su pantalla, la maestra debe guiar al alumno para que use la biblioteca según las instrucciones mencionadas.</p>
--	---	--

Instalación de **requests** en Anaconda prompt.

Para usuarios de Windows:

Ve a Inicio >> Anaconda prompt >> escribe **pip install requests** >> enter

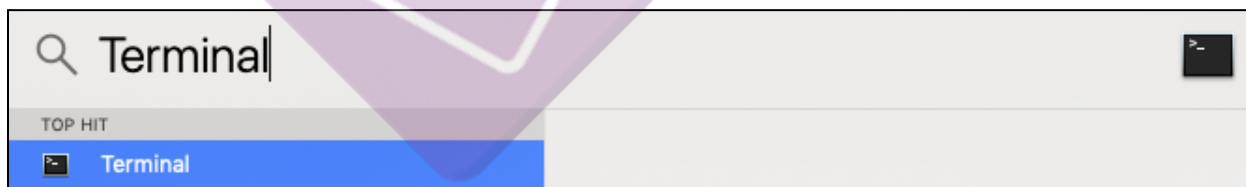


```
Anaconda Prompt (anaconda3)
(base) C:\Users\ASUS>pip install requests
```

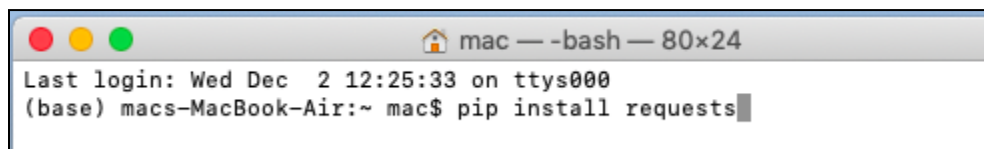
Recibirás el mensaje **"successfully installed"** - *(instalado exitosamente)*.

Para usuarios de Mac:

Presiona comando + espacio para abrir Spotlight, escribe **"Terminal"** y abre la Terminal.



Y luego escribe **"pip install requests"** y presiona Enter.



```
mac — -bash — 80x24
Last login: Wed Dec 2 12:25:33 on ttys000
(base) macs-MacBook-Air:~ mac$ pip install requests
```

	<p>Ahora, es tu turno. Muéstrame cómo crearás esta app.</p> <p>El alumno tiene que descargar la forma del código predefinido de la Actividad del alumno 2 y comenzar a programar para el widget meteorológico.</p> <p>Una vez que hayas completado la programación, comienza a grabar la pantalla, graba el output usando Loom y envía el enlace del video en el panel del alumno.</p>	<p>Actividad del alumno 1- DIAGRAMA DE CÓDIGO</p> <p>Actividad del alumno 2- CÓDIGO PREDEFINIDO</p> <p>Actividad del alumno 3- ENLACE OPENWEATHERMAP</p> <p>Actividad del alumno 4- LLAVE API</p> <p>Actividad del alumno 5- JSON VIEW</p> <p>Después de que el alumno haya completado la programación, guía al alumno para que inicie la grabación de pantalla, grabe el output usando Loom según la Actividad de referencia de la maestra 1, y envía el enlace de video al panel del alumno.</p>
<p>Guía al alumno para que deje de compartir pantalla.</p>		
<p>Paso 4: Conclusión (3 minutos)</p>	<p>También lo hiciste muy bien hoy. ¡Excelente! Te mereces dos felicitaciones.</p>	<p><i>(Da dos felicitaciones como mínimo).</i></p> <p>Presiona el icono de Felicitaciones por</p>

P: ¿Por qué importamos el módulo request?

R: como tenemos que acceder a las APIs requeridas desde el sitio web www.openweathermap.org, primero debemos enviar la solicitud a estos módulos. Para estos efectos tenemos que importar el módulo request.

P: ¿Qué función se utiliza para cargar el contenido de la API desde la URL?

R: mediante el uso de la función `loads()`.

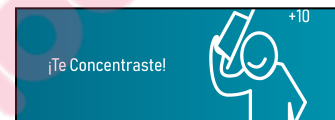
Resolver creativamente las actividades.



Presiona el icono de Felicitaciones por **Muy buena pregunta.**



Presiona el icono de felicitaciones por **Te concentraste.**



Si no tienes tiempo de llevar a cabo actividades adicionales, pide al alumno que haga todas las actividades adicionales después de la clase. Las actividades adicionales

		<p>son MUY importantes para los niños, para que estén listos para el siguiente módulo. Muchos conceptos desafiantes se aproximan.</p> <p>También recuerda al alumno que consulte la actividad de referencia para incrementar su conocimiento. Esto también debe hacerse.</p>
<p>Para la solución de toda la Actividad Adicional abre la Actividad de la maestra 5 y navega hasta el número de clase C183.</p> <p>Actividad adicional 1: Ejecuta la actividad del alumno 6 desde el panel. La TAREA y las PISTAS están en la página web.</p> <p>Actividad adicional 2: Ejecuta la actividad del alumno 7 desde el panel. La TAREA y las PISTAS están en la página web.</p> <p>Actividad adicional 3: Ejecuta la actividad del alumno 8 desde el panel. La TAREA y las PISTAS están en la página web.</p> <p>Actividad adicional 4: Ejecuta la actividad del alumno 9 desde el panel. La TAREA y las PISTAS están en la página web.</p> <p>Actividad adicional 5: Ejecuta la actividad del alumno 10 desde el panel. La TAREA y las PISTAS están en la página web.</p>		
<p>La maestra hace clic en</p>		<p>✕ Finalizar Clase</p>

Actividad	Nombre de la actividad	Enlaces
Actividad de la maestra 1	DIAGRAMA DE CÓDIGO	https://s3-whjr-curriculum-uploads.whjr.online/5b82e2c9-4aaf-45e7-b412-31ad07f514c5.pdf
Actividad de la maestra 2	CÓDIGO FUENTE COMPLETO	https://s3-whjr-curriculum-uploads.whjr.online/03606ea5-85aa-4dca-8aaa-bc1589eabafb.zip
Actividad de la maestra 3	API DE OPEN WEATHER MAP	https://openweathermap.org
Actividad de la maestra 4	SOLUCIONES DE LAS ACTIVIDADES ADICIONALES	https://s3-whjr-curriculum-uploads.whjr.online/aa7bbd68-7243-4f61-8df5-cbd66bffdada.xlsx
Actividad de referencia de la maestra 1	REFERENCIA PARA GRABAR LA PANTALLA	https://s3-whjr-curriculum-uploads.whjr.online/bd83cb5f-59ae-4245-9ce7-9eb2d07099b9.pdf
Actividad del alumno 1	DIAGRAMA DE CÓDIGO	https://s3-whjr-curriculum-uploads.whjr.online/5b82e2c9-4aaf-45e7-b412-31ad07f514c5.pdf
Actividad del alumno 2	CÓDIGO PREDEFINIDO	https://s3-whjr-curriculum-uploads.whjr.online/87c995cd-ec4a-4b1c-b9f9-9431d391e244.zip
Actividad del alumno 3	API DE OPEN WEATHER MAP	https://openweathermap.org
Actividad del alumno 4	LLAVE API	21cab08deb7b27f4c2b55f3e2df28ea4
Actividad del alumno 5	JSON VIEW	https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc
Actividad de	REFERENCIA PARA	https://s3-whjr-curriculum-uploads.whjr.online

referencia del alumno 1	GRABAR PANTALLA	ne/bd83cb5f-59ae-4245-9ce7-9eb2d07099b9.pdf
Actividad de referencia del alumno 2	PAQUETE JSON	https://www.w3schools.com/python/module_requests.asp
Actividad de referencia del alumno 3	MÓDULO REQUEST	https://www.w3schools.com/python/module_requests.asp

