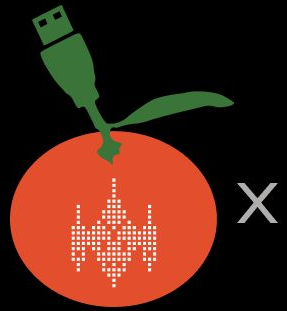


x

Welcome to

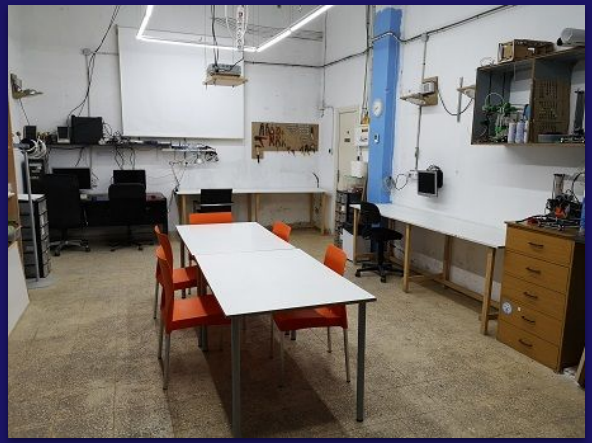
HACKERSPACE VALENCIA

We hope you enjoy!

Aprende Python básico creando un bot de Telegram

Ximo Catalá @ Hackerspace Valencia

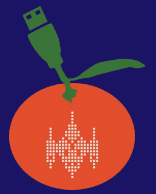
Hackerspace Valencia



Somos una asociación “maker” que promueve el uso de la tecnología libre (hardware y software).

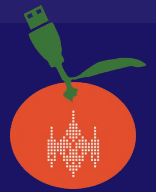
Si quieres formar parte de nuestra asociación preguntanos o visita https://hackvlc.es/

TABLA DE CONTENIDOS



01

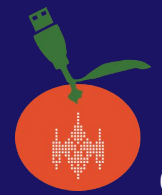
INTRODUCCIÓN



INTRODUCCIÓN

¿Qué es Python?

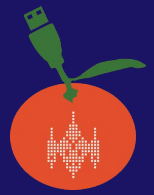
Python es un **lenguaje de programación** multiplataforma de código abierto. Es un lenguaje **interpretado** y se caracteriza por su **sencillez**. Es una buena opción para aprender a programar



INTRODUCCIÓN

¿Qué es un bot de Telegram?

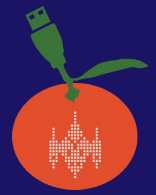
Un **bot** de Telegram es una **aplicación** escrita en un determinado lenguaje de programación y que permite crear automatizaciones dentro de chats, grupos o canales de Telegram. En nuestro caso usaremos **Python** y la librería **python-telegram-bot**



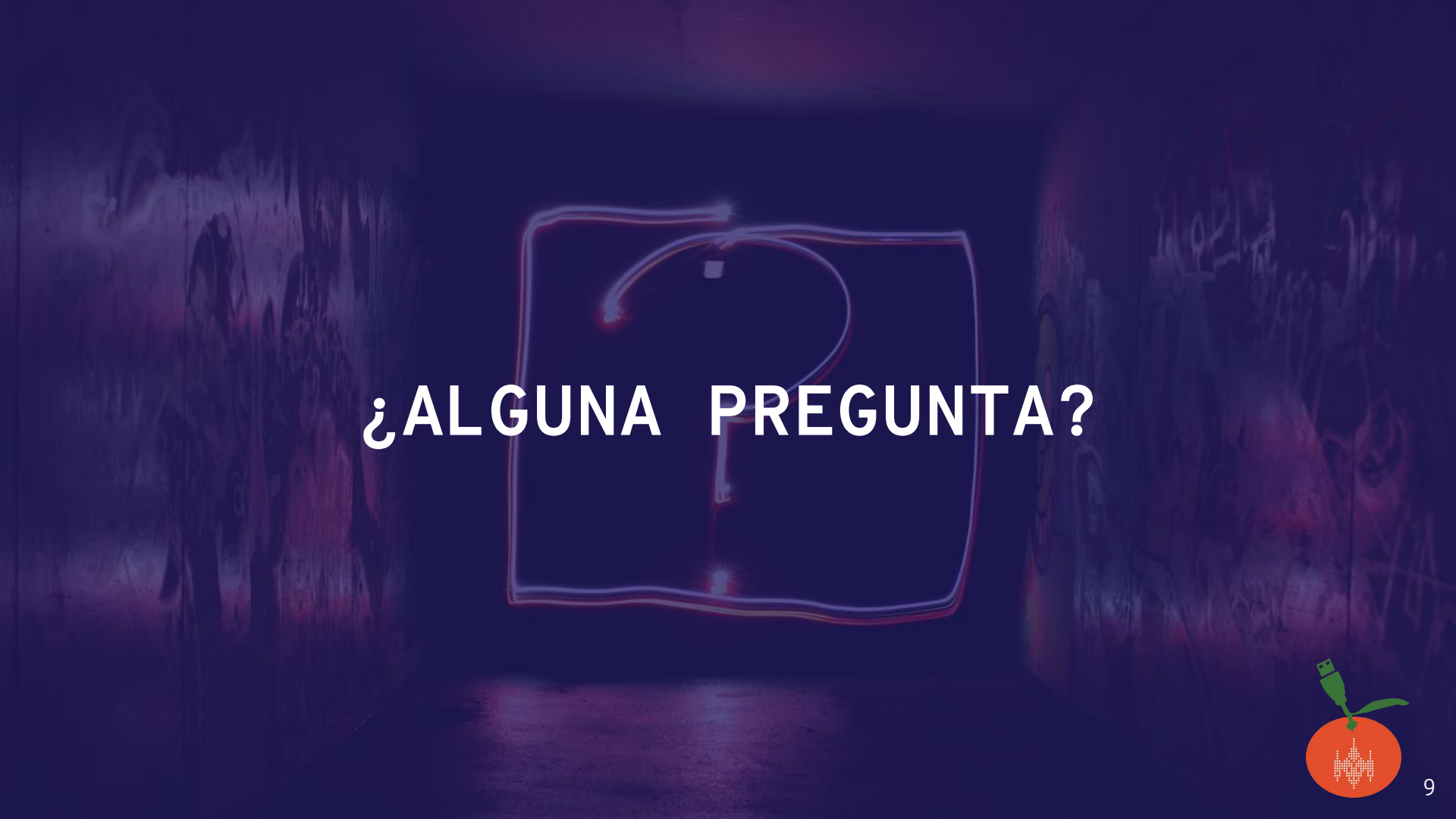
INTRODUCCIÓN

¿Qué vamos a necesitar?

1) El intérprete de Python (versión >3.7) https://www.python.org/downloads/
2) Un entorno de desarrollo: Visual Studio Code https://code.visualstudio.com/download
3) Una cuenta de Telegram

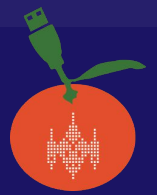


¿ALGUNA PREGUNTA?



02

CONCEPTOS BÁSICOS DE PROGRAMACIÓN

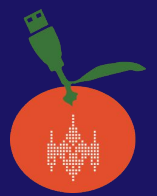


¿Qué es una variable?

Una variable es una "**caja**" donde puedo guardar **datos** para utilizar dentro de un programa. A una variable le ponemos un **nombre único** al que recurriremos para hacer uso de los datos que hay contenidos en ella. El contenido de esta "caja" (los datos) puede ir cambiando durante la ejecución del programa.

Casos "especiales":

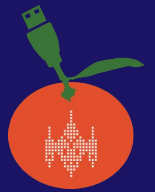
- **Constantes** (los datos de la variable no cambian durante la ejecución)
- **Vectores** o **arrays**. Son "cajas" que contienen otras "cajas" (conjunto ordenado de variables)



Tipos de datos

En Python básicamente los datos son de alguno de estos tipos:

- Números enteros (**int**). Ej. 4, -123, 12873, ...
- Números con decimales (**float**). Ej. -3.21, 123.75, ...
- Booleanos: True o False (**bool**)
- Cadena de caracteres (**str**). Ej. “hola2”, “True”, “12.34” (entre comillas)
- Otros (objetos...)



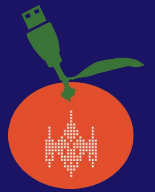
Asignación de datos

¿Cómo "meto" un dato en una variable? Mediante la **asignación** (símbolo **=**)

En Python no es necesario decir previamente de qué tipo va a ser una variable

```
# Los comentarios ayudan a entender el código. Todo lo que vaya detrás de
# una almohadilla no se tiene en cuenta a la hora de "ejecutar" el código

# Asignación Variables

peso = 75 #la variable peso es de tipo entero
mensaje = "Hola cómo estás" # variable tipo String (cadena de caracteres)
altura = 1.80 # variable float
encendido = True # variable booleana
```

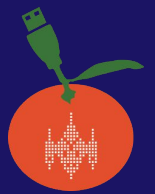
```
#Asignación arrays
notas_evaluaciones = [7,6,9]
```



Operaciones con datos

Disponemos de varios tipos de **operadores** que permiten hacer operaciones con los datos y las variables. Estos operadores son básicamente:

- **Aritméticos**: suma (+), resta (-), multiplicación (*), división (/)...
- **Lógicos**: AND (and), NOT (not), OR (or)...
- **Comparación**: Mayor (>), menor (<), igual (==), mayor o igual (>=), distinto (!=)...



Operaciones con datos

```python
# Operadores aritméticos
a = 1
b = 2
suma = a + b
multiplicacion = a * b
print("Suma:", suma)
print("Multiplicación:", multiplicacion)
```

Salida:

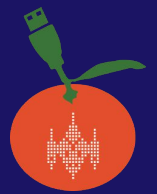
```
Suma: 3
Multiplicación: 2
```

```python
# Operadores Lógicos
a = True
b = False
resultado1 = a and b
resultado2 = not b
print("True AND False:", resultado1)
print("NOT True:", resultado2)
```

```
True AND False: False
NOT True: True
```

```python
#Operadores de comparación
a = 5
b = 7
resultado1 = a > b
resultado2 = a < b
print("¿ 5 > 7 ?: ", resultado1)
print("¿ 5 < 7 ?: ", resultado2)
```

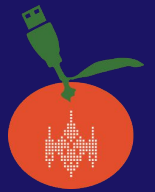
```
¿ 5 > 7 ?:  False
¿ 5 < 7 ?:  True
```



Control del flujo

Las **estructuras de control de flujo** permiten "desviar" el flujo normal de un programa. Es decir, permiten que la ejecución no sea secuencial línea tras línea de código. Las básicas son:

- **Bucles**: Sirven para ejecutar varias veces una parte del código
- **Condicionales**: Permiten hacer una u otra cosa dependiendo de que se dé o no una determinada condición



Control Flujo: bucles

Un bucle permite repetir una parte del código un número de veces. Básicamente en Python hay de dos tipos de bucles: **for** y **while**. Los bucles se pueden anidar, es decir, meter unos dentro de otros.

```
# Bucle que imprime los números del 0 al 4
# Los bucles empiezan a contar en 0
for i in range(5):
    print(f"i={i}")
```

Salida:

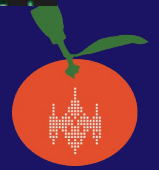
```
i=0
i=1
i=2
i=3
i=4
```

```
# Bucle anidado
# Los bucles empiezan a contar en 0
for i in range(5):
    for j in range(2):
        print(f"i={i} j={j}")
```

Salida:

```
i=0 j=0
i=0 j=1
i=1 j=0
i=1 j=1
i=2 j=0
i=2 j=1
i=3 j=0
i=3 j=1
i=4 j=0
i=4 j=1
```

¡¡¡ Mucho ojo con los **indentados** de línea !!!



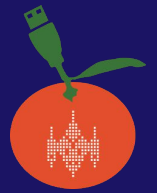
Control Flujo: condicionales

Las estructuras condicionales **if, if - else** permiten ejecutar una parte del código en función de si se da o no una **condición**. Las condicionales también pueden anidarse unas dentro de otras.

```
#Condicional if, array y for
numeros=[-1,2] #array de 2 números enteros
for n in numeros: #recorro el array
    if n >= 0:
        print(n, "es positivo")
    else:
        print(f"{n} es negativo")
```

```
-1 es negativo
2 es positivo
```

¡¡¡ Mucho ojo con los **indentados** de línea !!!



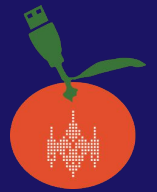
Funciones

Una función en una "una caja negra" a la que le paso unos **argumentos** de **entrada** y me devuelve uno o más datos de salida. Las ventajas son:

- Permiten dividir el código en partes más pequeñas (divide y vencerás)
- Pueden ser invocadas siempre que quieras, con lo que evitas escribir código de más

```
#Función que suma 3 números que paso como argumentos
# y devuelve la suma de dichos argumentos
def sumar(a,b,c):
    resultado=a+b+c
    return resultado

print("La suma vale:",sumar(5,2,4))
```

```
La suma vale: 11
```



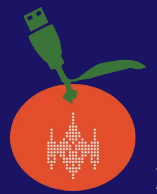
Librerías

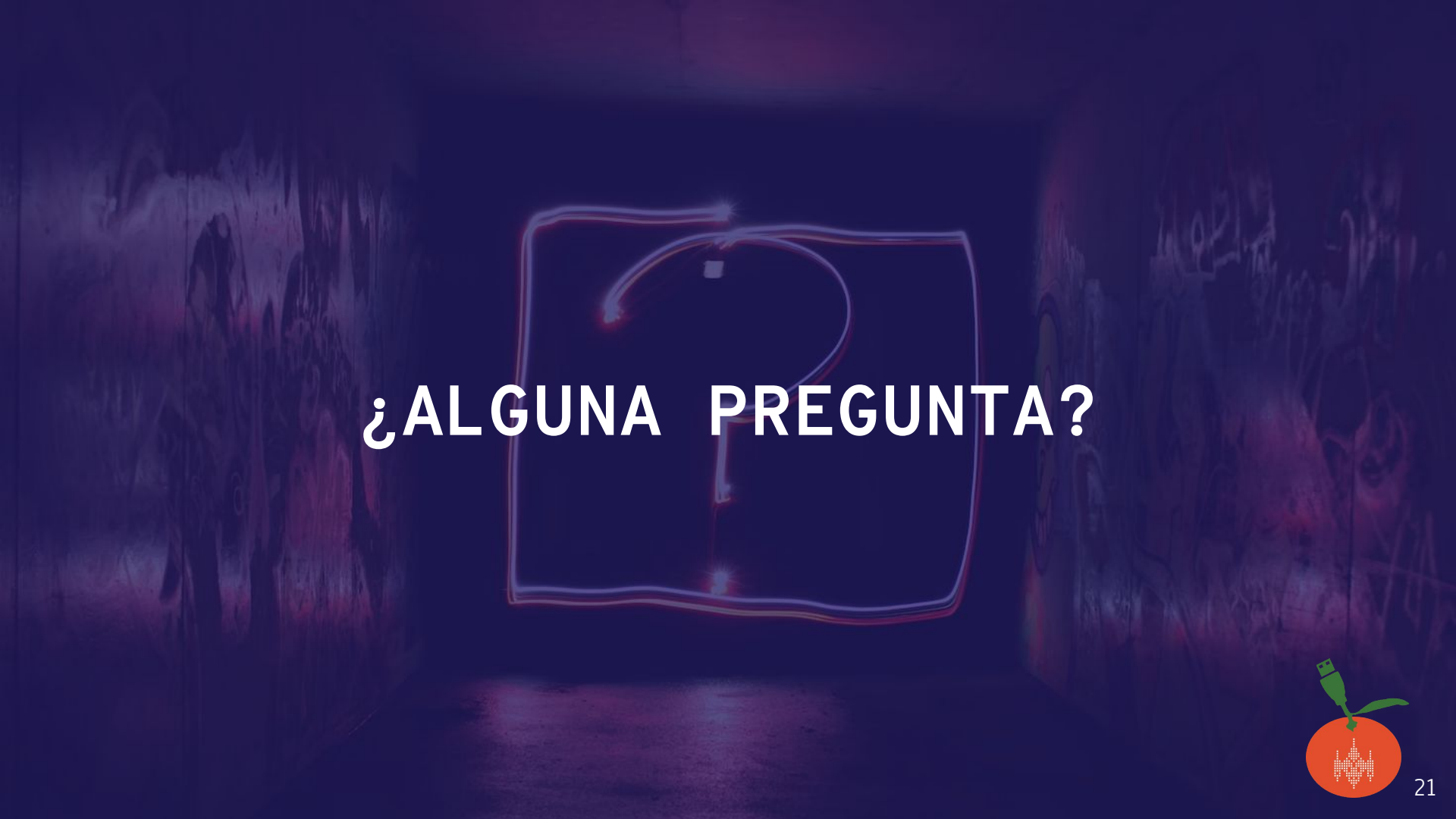
Las **librerías** son archivos de código, normalmente programadas por terceros, y que nos facilitan la vida puesto que incluyen multitud de funciones que resuelven problemas de manera transparente para nosotros. En nuestro caso usaremos las siguientes librerías:

- **Python-telegram-bot**. https://python-telegram-bot.org/
- Python-aemet https://github.com/pablo-moreno/python-aemet

```
import api
import private
from telegram import Update, InlineQueryResultArticle, InputTextMessageContent
from telegram.ext import InlineQueryHandler, filters, MessageHandler, ApplicationBuilder, CommandHandler, ContextTypes
```

```
from aemet import Aemet, Municipio
import private
```

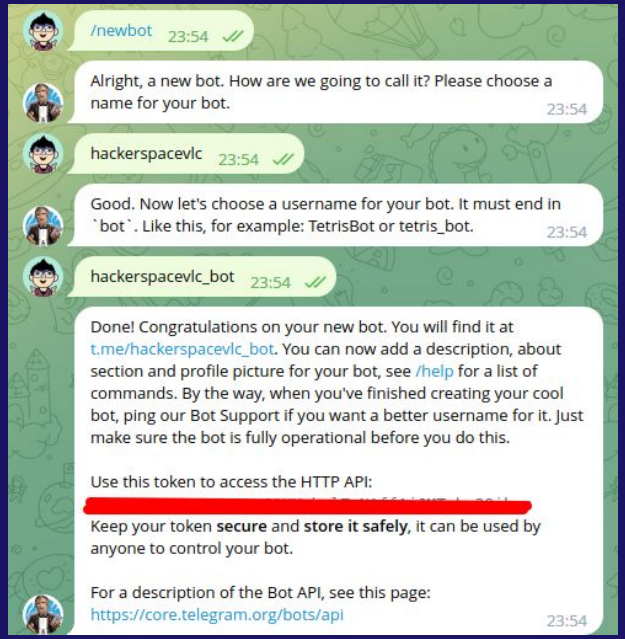



¿ALGUNA PREGUNTA?

03

Programando el bot

PROGRAMANDO EL BOT



BotFather

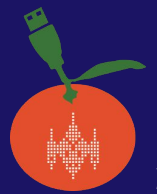
BotFather es el gestor de bots de Telegram. Se accede a él a través de un chat de Telegram y básicamente nos va a permitir:

- Crear un nuevo bot, proporcionándonos el **TOKEN** que lo identifica
- Gestionar nuestros bots

Nuestro bot meteorológico

El bot que vamos a construir nos ofrecerá el tiempo en los próximos 7 días en el municipio que le preguntemos. Obtendremos los datos de la API de **AEMET** (Agencia Estatal de Meteorología) que es abierta y lo único que necesitamos es una **API_KEY** (https://opendata.aemet.es/centrodedescargas/inicio)

La librería **python-aemet** nos facilitará el acceso a los datos

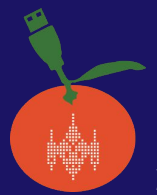


Estructura del programa

Para programar este bot, y ser más eficientes, tendremos nuestro código en 3 archivos diferentes

- **private.py** que contiene los datos personales y secretos (TOKEN del bot de Telegram y API_KEY de AEMET). Este archivo no debemos compartirlo.
- **api.py** que contiene las funciones para acceder a la API de AEMET
- **bot_telegram.py** que contiene el código del bot

Los archivos api.py y bot_telegram.py los puedes bajar de https://github.com/HackerspaceVLC/taller_bot_telegram.git



private.py

Los datos para escribir este archivo nos lo proporcionará:

- **TOKEN** Telegram: BotFather
- **API_KEY** AEMET. Nos llegará por correo al solicitarla en https://opendata.aemet.es/centrodedescargas/inicio

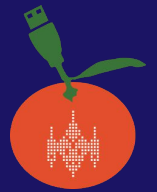
Recuerda **NO COMPARTIR** estos datos con nadie

```
1  TOKEN_TELEGRAM="[redacted]"
2  API_KEY_AEMET="[redacted]"
3
4
```

PROGRAMANDO EL BOT

api.py

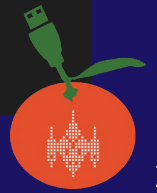
```python
from aemet import Aemet, Municipio
import private

aemet_client = Aemet(private.API_KEY_AEMET)

def seleccionar_municipio(nombre_municipio): #Entrada: municipio; Salida: array con los municipios que contienen dicho nombre
    lista_municipios=Municipio.buscar(nombre_municipio)
    return lista_municipios

def obtener_codigo_municipio(nombre_municipio): #Le paso un municipio y devuelve el código de AEMET
    municipios=seleccionar_municipio(nombre_municipio)
    return municipios[0].cpro+municipios[0].cmun

def componer_mensaje_prediccion(prediccion): #Construye el mensaje a mandar a partir un array con los datos meteorológicos
    mensaje=""
    for i in range(7):
        mensaje=mensaje + f"Día: {prediccion[0][i]} - Mín: {prediccion[2][i]:2.0f}ºC - \
            Máx: {prediccion[1][i]:2.0f}ºC - Prob.Prec.: {prediccion[3][i]:2.0f}% - Cielo: {prediccion[4][i]}\n"
    return mensaje

def prediccion(nombre_municipio): #Le paso el municipio y me devuelve el mensaje a enviar al usuario
    cod_municipio=obtener_codigo_municipio(nombre_municipio)
    prediccion=aemet_client.get_prediccion(codigo_municipio=cod_municipio, periodo='PERIODO_SEMANA',raw=True)
    dias=[]
    temperaturas_maximas=[]
    temperaturas_minimas=[]
    probabilidades_precipitacion=[]
    estados_cielo=[]
    for i in range(7):
        dias.append(prediccion['prediccion']['dia'][i]['fecha'][8:10])
        temperaturas_maximas.append(prediccion['prediccion']['dia'][i]['temperatura']['maxima'])
        temperaturas_minimas.append(prediccion['prediccion']['dia'][i]['temperatura']['minima'])
        probabilidades_precipitacion.append(prediccion['prediccion']['dia'][i]['probPrecipitacion'][0]['value'])
        estados_cielo.append(prediccion['prediccion']['dia'][i]['estadoCielo'][0]['descripcion'])
    resultado=[dias, temperaturas_maximas, temperaturas_minimas, probabilidades_precipitacion, estados_cielo]
    mensaje = componer_mensaje_prediccion(resultado)
    return mensaje
```



bot_telegram.py

PROGRAMANDO EL BOT

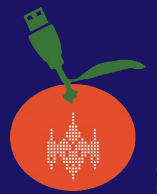
```python
import api
import private
from telegram import Update, InlineQueryResultArticle, InputTextMessageContent
from telegram.ext import InlineQueryHandler, filters, MessageHandler, ApplicationBuilder, CommandHandler, ContextTypes

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE): #Función asincrona para gestiona el comando de bot /start
    usuario= update.message.from_user
    await context.bot.send_message(chat_id=update.effective_chat.id,
            text=f"Hola {usuario.first_name}, para saber la predicción en una población manda /predict seguido de la población (primera letra en mayúsculas, ej. Valencia)")

async def prediccion_municipio(update: Update, context: ContextTypes.DEFAULT_TYPE): #Función asincrona que gestiona el comando de bot /predict
    #Obtenemos el nombre de la población que ha sido enviado en el mensaje tras el /predict
    poblacion=context.args[0]

    #Obtenemos el array con todas las poblaciones que coinciden con la que nos ha enviado
    # el usuario tras el /predict
    lista_poblaciones=api.seleccionar_municipio(poblacion)

    if len(lista_poblaciones)==0: #No hay ninguna población con ese nombre
        mensaje=f"Introduce otra población ya que '{poblacion}' no existe en la BD de AEMET"
    if len(lista_poblaciones)==1: #Solo hay una población en la BD de AEMET con ese nombre
        mensaje=api.prediccion(poblacion)
    if len(lista_poblaciones)>1: #Hay varias poblaciones con el mismo nombre
        # Esta parte no está programada, ¿te atreves?
        mensaje=f"Ups, parece que hay más de una población con el nombre '{poblacion}' y esta parte no está programada. ¿Te atreves a intentarlo?"

    await context.bot.send_message(chat_id=update.effective_chat.id, text=mensaje) #Mandamos el mensaje al usuario

if __name__ == '__main__':
    application = ApplicationBuilder().token(private.TOKEN_TELEGRAM).build()
    #/START
    start_handler = CommandHandler('start', start)
    application.add_handler(start_handler)
    #/PREDICT
    predict_handler = CommandHandler('predict', prediccion_municipio)
    application.add_handler(predict_handler)

    application.run_polling()
```

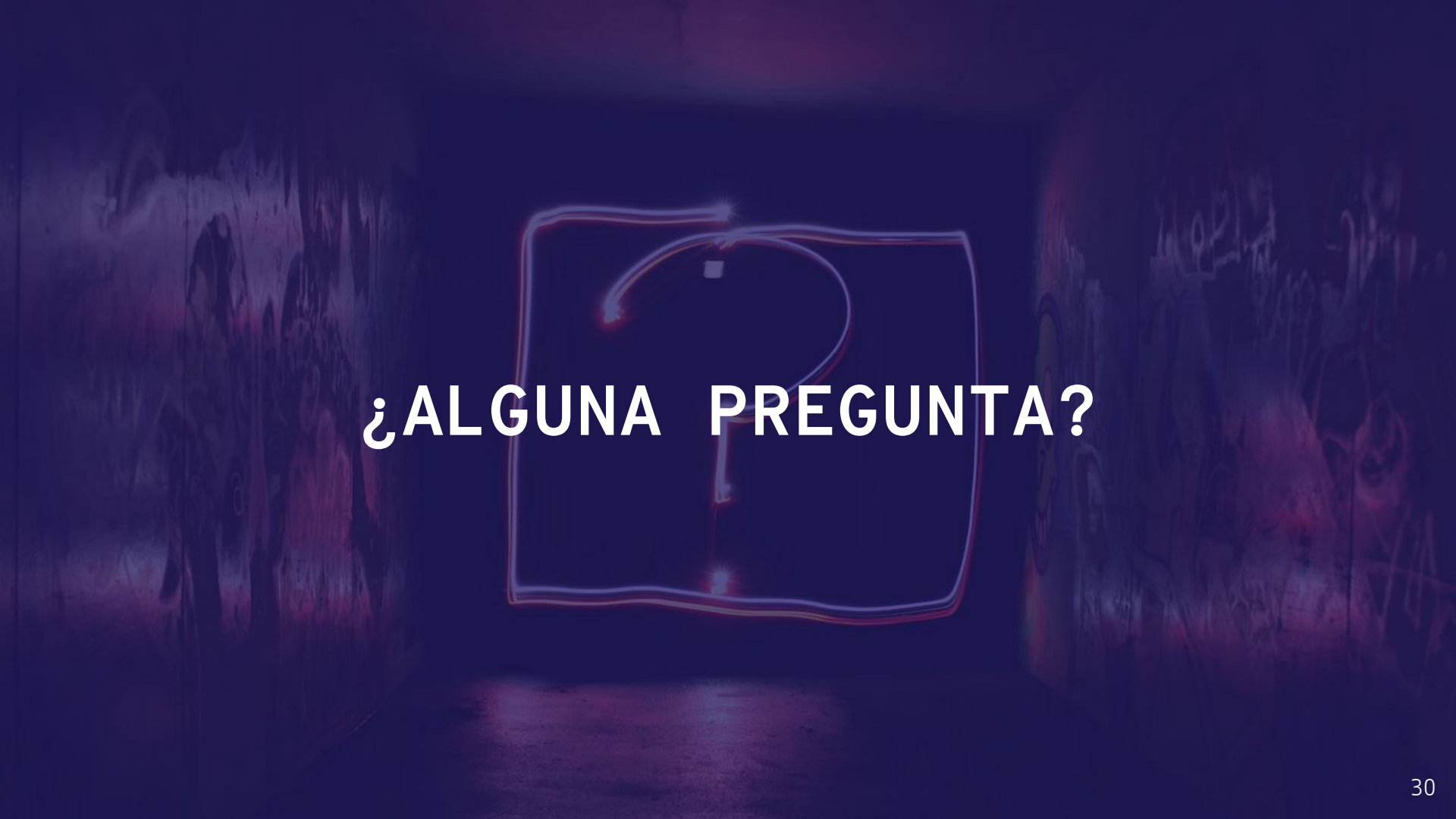


¿Seguro que ya está?

Prueba el bot y verás como **algo falta**.

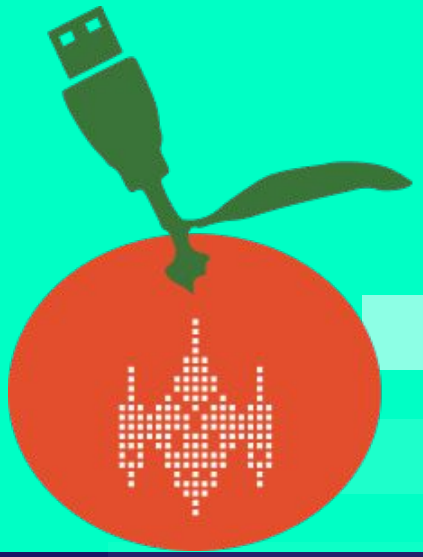
¿**Te atreves** a terminar el bot para que esté totalmente operativo?




¿ALGUNA PREGUNTA?

Gracias

https://hackvlc.es/
hackerspacevlc@gmail.com
Hackerspace Valencia



CREDITS: This presentation template was created by Slidesgo, incluiding icons by Flaticon, and infographics & images by Freepik.