SISTEMAS WEB

1ª Práctica: Cliente IoT

OBJETIVO

Usando PyCharm IDE, se tiene que programar en Python un cliente web que suba los datos de %CPU y %RAM actuales del ordenador a ThingSpeak (https://thingspeak.com/). Este cliente web deberá cumplir los siguientes requisitos:

- Cuando el cliente se ponga en marcha, la primera acción que realizará será crear un canal con dos campos de datos. Las etiquetas de los campos de datos serán **%CPU** y **%RAM**.
- Una vez creado el canal, el cliente subirá los datos **%CPU** y **%RAM** del ordenador cada **15 segundos**, utilizando la librería **psutil**.
- Para finalizar la subida de datos y cerrar el cliente, el usuario pulsará **Ctrl+C**. El cliente gestionará esta señal de interrupción vaciando el canal y cerrando la conexión TCP (Nota: el vaciado del canal hace referencia a la eliminación de los datos existentes en el canal, por lo que no es necesario eliminar el canal)

Aspectos que deberéis desarrollar por vuestra cuenta:

- Cargar una cadena JSON en un diccionario Python.
- Lectura de datos desde un diccionario Python.
- Programar excepciones de Python.

ENTREGABLES

El resultado de la práctica será el programa Python que implementa el cliente web. El programa debe subirse a la tarea disponible en eGela al finalizar la sesión de práctica:

GRUPO 1 - https://egela.ehu.eus/mod/assign/view.php?id=5039184

GRUPO 2 - https://egela.ehu.eus/mod/assign/view.php?id=5212829

PASOS E INSTRUCCIONES PARA LA PRÁCTICA

1. *Crear un proyecto de PyCharm*
2. *Instalar la librería psutil en el entorno virtual del proyecto*
3. *Crear un fichero Python en el proyecto*
4. *Estructura de un programa Python*
5. *Utilizando el API de la librería psutil, desarrollar un código que imprima el %CPU y %RAM del ordenador en el terminal de datos cada 15 segundos.*
6. *Ejecución de un programa Python*
7. *Programar la excepción que gestiona la salida con Ctrl+C*
8. *Juntar los códigos de los apartados 5 y 7: lectura de datos %CPU y %RAM cada 15 segundos y desarrollo del programa que finaliza la ejecución pulsando Ctrl+C*
9. *Primera aproximación a ThingSpeak*
10. *Programar el envío de una solicitud HTTP y la lectura de la respuesta*
11. *Programar la solicitud HTTP para crear un canal con dos campos de datos de ThingSpeak*
12. *Parsear la respuesta HTTP con los datos del canal recién creado en formato JSON*
13. *Desarrollar un código que suba cada 15 segundos a ThinkSpeak dos datos*
14. *Juntar los códigos 7, 12 y 13 para crear un canal de ThingSpeak y desarrollar un programa que suba dos datos cada 15 segundos.*
15. *Desarrollar un código que elimine datos de un canal de ThinkSpeak*
16. *Juntar los códigos de los apartados 14 y 15: se crea un canal de ThingSpeak y se desarrolla un programa que suba los datos %CPU y %RAM cada 15 segundos, se vacía el canal pulsando Ctrl+C y finaliza la ejecución del programa.*

1. Crear un proyecto de PyCharm

Las explicaciones para la ejecución de este paso se dieron en la clase **M 28-01-2022.**

2. Instalar la librería psutil en el entorno virtual del proyecto

Las explicaciones para la ejecución de este paso se dieron en la clase **M 28-01-2022.**

3. Crear un fichero Python en el proyecto

File → New → Python file

4. Estructura de un programa Python

Las explicaciones para la ejecución de este paso se dieron en la clase **M 28-01-2022.**

5. Utilizando el API de la librería psutil, desarrollar un código que imprima el %CPU y %RAM del ordenador en el terminal de datos cada 15 segundos

Documentación de la librería: *https://psutil.readthedocs.io/en/latest/#system-related-functions*

Recordar que la llamada al método para extraer el dato de %CPU deberá ser no bloqueante. Para calcular el porcentaje de memoria utilizado, utilizar como guía el código https://github.com/giampaolo/psutil/blob/master/scripts/meminfo.py

El programa debe ejecutarse de forma continua, imprimiendo los datos %CPU y %RAM actuales del ordenador cada 5 segundos en el terminal.

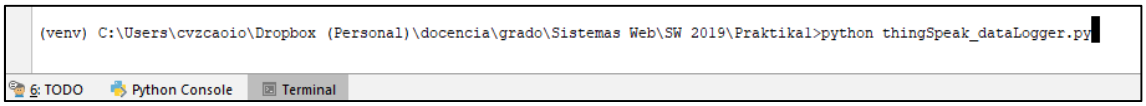
A continuación, se presenta una plantilla del programa que se solicita:

```
import psutil
import time


def cpu_ram():
    while True:
        # CODIGO: utilizando la libreria psutil, obtener %CPU y %RAM
        cpu =
        ram =
        print("CPU: %" + str(cpu) + "\tRAM: %" + str(ram))
        time.sleep(5)


if __name__ == "__main__":
    cpu_ram()
```

6. Ejecución de un programa Python

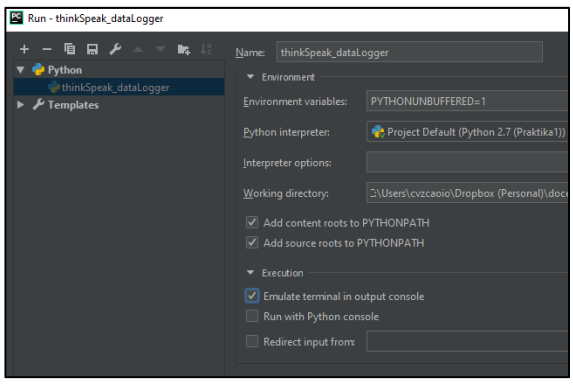
Para ejecutar un programa Python hay dos formas: a) Terminal b) PyCharm.

a) Terminal: pulsar el botón *Terminal*, inferior de la ventana de PyCharm, y llamar al intérprete de Python pasando el nombre del programa como parámetro.



Para finalizar la ejecución del programa hay que pulsar Ctrl+C.

b) PyCharm: **Run → Run…**: En este caso, al pulsar Ctrl+C **no** se finalizará la ejecución del programa. ¿Por qué? Porque PyCharm entiende Ctrl+C como "copia". Cuando un programa se ejecuta en la **ventana Run**, para que Ctrl+C se entienda como terminación, hay que habilitar en **la configuración Run** de PyCharm "**Emulate terrminal in output console**"



7. Programar la excepción que gestiona la suspensión Ctrl+C

Ctrl+C es la combinación de teclas que produce la señal INT. ¿Qué es una señal? En el contexto de los sistemas operativos, las señales son un tipo de comunicación entre procesos (IPC) https://en.wikipedia.org/wiki/Signal_(IPC). La señal INT está pensada para interrumpir un proceso desde una terminal. En general, esta suspensión supone la finalización ordenada del proceso.

A continuación, se presenta el programa que se solicita. Escribir y ejecutar el programa en PyCharm:

```
import signal
import sys

def handler(sig_num, frame):
    # Gestionar evento
    print('\nSignal handler called with signal ' + str(sig_num))
    print('Check signal number on '
          'https://en.wikipedia.org/wiki/Signal_%28IPC%29#Default_action')

    print('\nExiting gracefully')
    sys.exit(0)

if __name__ == '__main__':
    # Cuando se recibe SIGINT se ejecutará el método "handler"
    signal.signal(signal.SIGINT, handler)

    print('Running. Press CTRL-C to exit.')
    while True:
        pass # No hacer nada
```

8. Juntar los códigos de los apartados 5 y 7

Es decir, realizar la lectura de los datos %CPU y %RAM cada 15 segundos y desarrollar el programa que finaliza la ejecución pulsando Ctrl+C.

9. Primera aproximación a ThingSpeak

Revisar las transparencias que están junto con este guión:
https://egela.ehu.eus/mod/resource/view.php?id=5039182

Y utilizando la herramienta Burp, realizar los ejemplos que en ella se plantean.

10. Programar el envío de una solicitud HTTP y la lectura de la respuesta

Las explicaciones para la ejecución de este paso se dieron en la clase **M 04-02-2022**

En esta práctica el cliente web debe realizar tres solicitudes HTTP:

1. Creación del canal 1 (una sola vez en la puesta en marcha del cliente)
2. Subida de datos (continua, cada 15 segundos, hasta que el usuario pulsa Ctrl+C)
3. Vaciado del canal (cuando el usuario pulsa Ctrl+C)

En el siguiente enlace se indican las características que deben cumplir las siguientes solicitudes: https://es.mathworks.com/help/thingspeak/rest-api.html

Recomendación: programar y comprobar cada solicitud por separado. Cuando se compruebe que el funcionamiento de las tres solicitudes es correcto, entonces suma las tres.

11. Programar la solicitud HTTP para crear un canal con dos campos de datos de ThingSpeak

https://es.mathworks.com/help/thingspeak/rest-api.html

→ REST API reference → Create and Delete → Create Channel

https://es.mathworks.com/help/thingspeak/createchannel.html

Las explicaciones para la ejecución de este paso se dieron en la clase **M 12-02-2021.**

```
POST /channels.json HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 70

api_key=67PRF2TFLS11MXW3&name=Nire+kanala&field1=%25CPU&field2=%25RAM
```

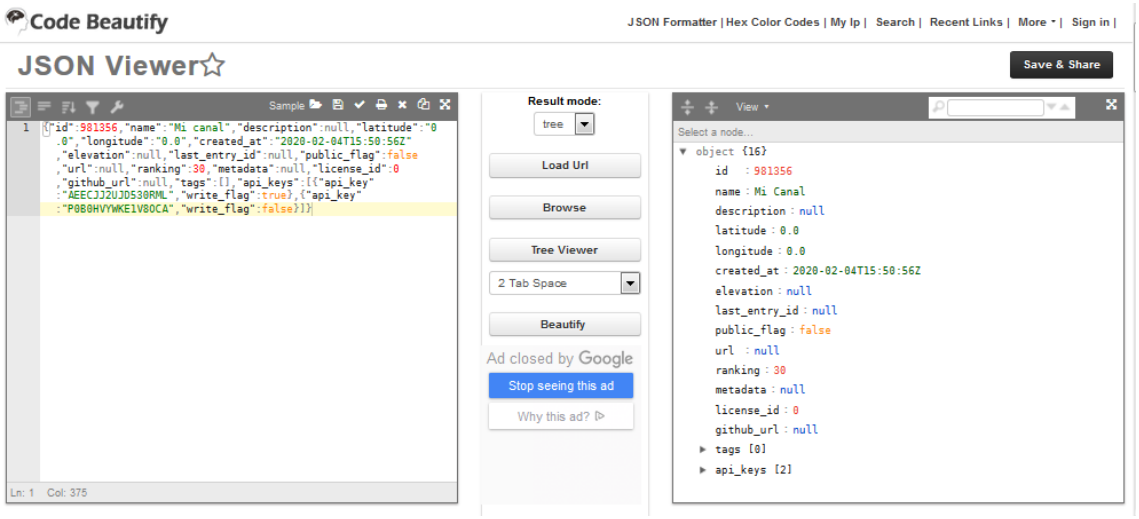
```
import requests
import urllib


metodo = 'POST'
uri = "https://api.thingspeak.com/channels.json"
cabeceras =  {'Host': 'api.thingspeak.com',
              'Content-Type': 'application/x-www-form-urlencoded'}
contenido = {'api_key': '52PRF2TFLS98MXW7',
             'name': Mi Canal',
             'field1': "%CPU",
             'field2': "%RAM"}
contenido_encoded = urllib.urlencode(contenido)
cabeceras['Content-Length'] = str(len(contenido_encoded))
respuesta = requests.request(metodo, uri, data=contenido encoded,
                             headers=cabeceras, allow redirects=False)


codigo = respuesta.status_code
descripcion = respuesta.reason
print(str(codigo) + " " + descripcion)
contenido = respuesta.content
print(contenido)
```

12. Parsear la respuesta HTTP con los datos del canal recién creado en formato JSON

La respuesta HTTP del apartado anterior devuelve una estructura de datos en formato JSON.

{"id":981356,"name":"Mi canal","description":null,"latitude":"0.0","longitude":"0.0","created_at":"2020-02-04T15:50:56Z","elevation":null,"last_entry_id":null,"public_flag":false,"url":null,"ranking":30,"metadata":null,"license_id":0,"github_url":null,"tags":[],"api_keys":[{"api_key":"AEECJJ2UJD530RML","write_flag":true},{"api_key":"P0B0HVYWKE1V8OCA","write_flag":false}]}

Para analizar la estructura de los datos JSON, utilizar la siguiente herramienta que permite visualizar los datos de forma ordenada: https://codebeautify.org/jsonviewer



De esta estructura de datos, debéis extraer por programa el ID del canal recién creado y la clave de permiso de escritura (WRITE_API_KEY) para que sean utilizados en la siguiente solicitud (subida de datos). Estructura de datos JSON para el manejo de Python:

https://www.w3schools.com/python/python_json.asp

13. Desarrollar un código que suba cada 15 segundos a ThinkSpeak dos datos

https://es.mathworks.com/help/thingspeak/rest-api.html

REST API reference → Write Data → Write Data

https://es.mathworks.com/help/thingspeak/writedata.html

OBSERVACIONES:

- En este punto no utilizar la clave de autorización extraída en el apartado anterior, crear un canal manualmente e introducir la clave de autorización en el código manualmente.
- En este punto no subáis %CPU ni %RAM actual. Subir dos números cualesquiera introducido manualmente en el código.

14. Juntar los códigos 7, 12 y 13

Es decir, crear un canal en ThingSpeak, subir dos datos a ese canal cada 15 segundos y desarrollar un programa que finaliza la ejecución pulsando Ctrl+C.

Nota: en este punto no subir %CPU y %RAM actuales. Subir dos números cualesquiera introducidos manualmente en el código.

15. Desarrollar un código que elimine datos de un canal de ThinkSpeak

Buscar en la API de ThingSpeak las características de la petición HTTP correspondientes a esta función.

16. Juntar los códigos de los apartados 14 y 15

Se crea un canal de ThingSpeak, se desarrolla un programa que sube los datos %CPU y %RAM cada 15 segundos, pulsando Ctrl+C se vacía el canal y finaliza la ejecución del programa