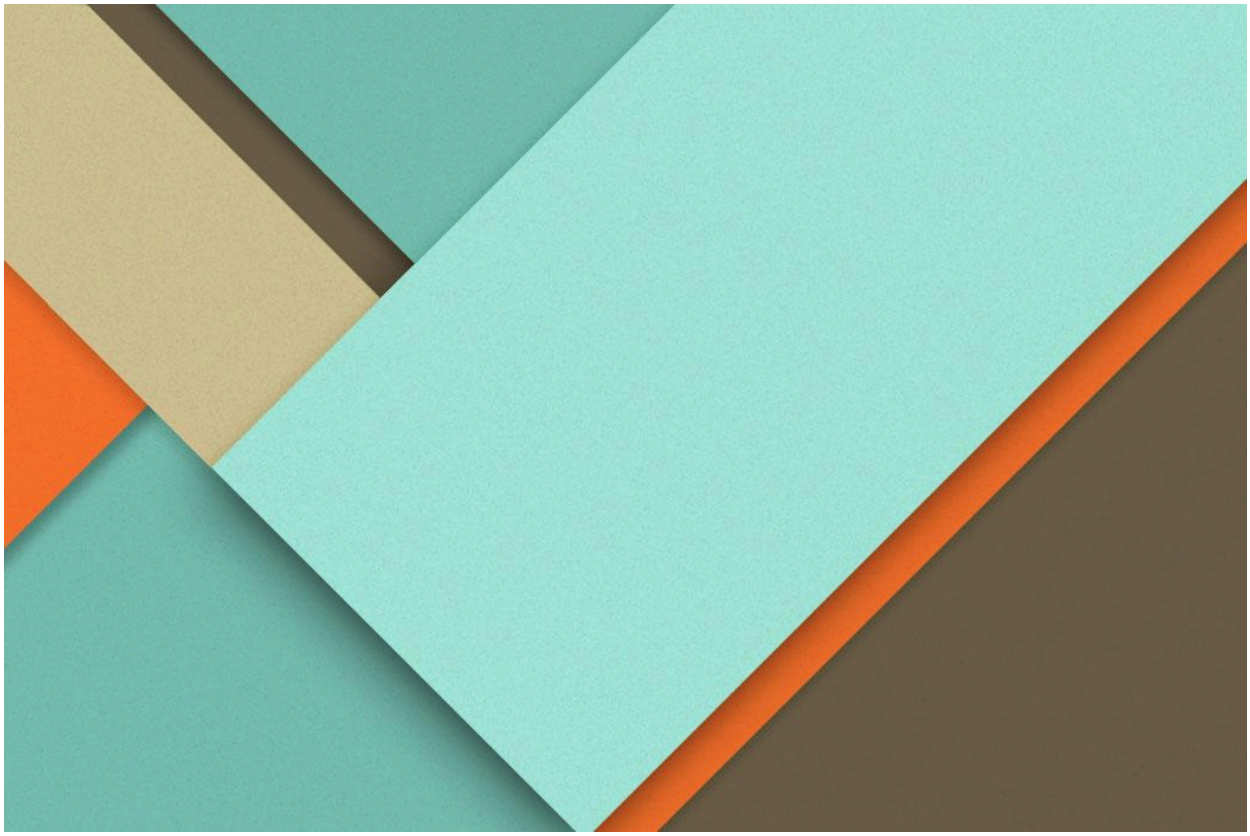


Reporte semanal - Semana 4

23/02/2024

Ismael Montoro Peñasco
Fundación GoodJob

Acciones llevadas a cabo

Realizaré todo aquello que no me dio tiempo a realizar esta semana, aprender un poco de scripting python con la librería Scapy, me dedicaré a mi proyecto personal de Python a lo largo del mes, el proyecto “Espadas”.

Avances realizados

Labor 1: Proyecto Espadas

He reanudado mi proyecto personal de Python, nombre en clave proyecto Espadas, orientado en materia de ciberseguridad.

Labor 1.1: Gnu-Crypto en Python

Quería implementar el algoritmo de hash Haval, un tipo de hash bastante antiguo creado por Yuliang Zheng, Josef Pieprzyk, y Jennifer Seberry, pero habia un obstaculo, este algoritmo solo lo encontre disponible para el lenguaje Java, a través de la librería Gnu-Crypto, por ello tenía que encontrar la manera de ejecutar un ejecutable de Java en Python, para ello he usado la librería subprocess y la API de JAva para ejecutar Java en Python.

Programa Java para realizar el algoritmo Haval.

```java
package Src.pyJava.componentesJava;
// package Src.pyJava;
import java.math.BigInteger;
import gnu.crypto.hash.Haval;

public class cifradosHaval {
    /*
     *
     * Fuente: https://www.bing.com/search?form=NTPCHB&q=Bing+AI&showconv=1
     *
     * */

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        // String mensaje = "hello";
        String mensaje = args[0];

        for (int ronda = 0; ronda < 3; ronda++)
            for (int bits = 0; bits < 5; bits++) {
                String[] cifrado = encriptarHaval(mensaje, bits, ronda);
                System.out.println("Haval" + cifrado[0] + "," + cifrado[1] + " " +
            }
```

```java
public static String[] encriptarHaval(String mensaje, int bits, int ronda) {

    /*
        Este es el enlace de la fuente original, pero no lo utilices, se sospecha que pu
        un malware Chino con la ayuda de virustotal.com, y se ha tenido que realizar cie
        descargar la informe sin el malware.

        https://web.archive.org/web/20050308141821/https://www.calyptix.com/files/haval-

        https://www.virustotal.com/gui/url/5b134512fb654b3194799f2f901a0b508275bbae792a0
    */

    final int size[] = {Haval.HAVAL_128_BIT, Haval.HAVAL_160_BIT, Haval.HAVAL_192_BIT, H
    final int round[] = {Haval.HAVAL_3_ROUND, Haval.HAVAL_4_ROUND, Haval.HAVAL_5_ROUND};
    final String[] tamanos_disponibles = {"128", "160", "192", "224", "256"};
    final String[] rondas_disponibles = {"3", "4", "5"};

    byte[] codificacion = codificacionBytes(mensaje);

    final Haval haval = new Haval(size[bits], round[ronda]);
    haval.update(codificacion, 0, codificacion.length);

    byte[] mensaje_cifrado = haval.digest();
```

Ejecutar archivo Java

```
Src > pyJava > modulopythonjava.py > cifradosHaval
import subprocess

def cifradosHaval(texto: str) -> dict:
    """
    Devolvera todos los hashes de haval
    del texto que pases por parametro
    """
    resultado = subprocess.run(['java', '-cp', \
            'C:\\Users\\ismae\\git\\espadas;C:\\Users\\ismae\\git\\espadas\\Src\\pyJava\\
            'Src.pyJava.componentesJava.cifradosHaval', texto], \
            capture_output=True, text=True)
    salida = resultado.stdout.split("\n")

    procesamiento = dict()
    try:
        for haval in salida:
            hash = haval.split(" ")
            procesamiento[hash[0]] = hash[1]
    except:
        pass

    return procesamiento
```

Usar el archivo

```python
def haval(self, texto: str, version: int = 0):
    """
    Para usar esta funcion tienes que pasar el texto a cifrar y
    el tipo de hash haval a utilizar mediante un numero entero positivo
    aqui dejo un listado de las opciones que tienes disponibles.

    Recomendacion: usa la funcion una vez, y utiliza el diccionario
    devuelto por la funcion para evitar tener que usar la funcion una
    y otra vez, asi tu programa ira mas rápido y será más eficiente.

    Listado:
        "ALL"           0 (Recomendado)\n
        "Haval128,3"    1\n
        "Haval160,3"    2\n
        "Haval192,3"    3\n
        "Haval224,3"    4\n
        "Haval256,3"    5\n
        "Haval128,4"    6\n
        "Haval160,4"    7\n
        "Haval192,4"    8\n
        "Haval224,4"    9\n
        "Haval256,4"    10\n
        "Haval128,5"    11\n
        "Haval160,5"    12\n
        "Haval192,5"    13\n
        "Haval224,5"    14\n
        "Haval256,5"    15
    """
    if version >= 0 and version <= 15:
        versiones = ("ALL", "Haval128,3", "Haval160,3", "Haval192,3", "Haval224,3", "Hava
        eleccion = versiones[version]
        cifrados_haval = traductor.cifradosHaval(texto)

        if eleccion == "ALL":
            return cifrados_haval

        return cifrados_haval[eleccion]

    return "Opcion no valida, introduce un número del 0 al 15, para elegir una opción."

```

Fuente: Bing IA

Labor 1.2: Hashes

He trabajado con varias librerías de hashes durante varios meses, o sea, que tengo conocimiento de librerías de cifrado, hashlib, hashbase, pyaes, rsa, etc. También he trabajado con algoritmos de hash, desde módulos independientes como por ejemplo whirlpool, crc23c o la familia de hashes sha3.

```python
import os, random, zlib, binascii
import hashlib # hashlib es una libreria nativa de python
import crc32c
import Lib.cifrados.whirlpool as whirlpool
import Src.pyJava.modulopythonjava as traductor
import rsa
import pyaes as pyaescrypt
from hashbase import *
```

```
sha512-224    56 fe8509ed1fb7dcefc27e6ac1a80eddbec4cb3d2c6fe565244374061c\n
sha512-256    64 e30d87cfa2a75db545eac4d61baf970366a8357c7f72fa95b52d0accb698f13a\n
ripemd128     32 789d569f08ed7055e94b4289a4195012\n
ripemd160     40 108f07b8382412612c048d07d13f814118445acd\n
ripemd256     64 On PHP: cc1d2594aece0a064b7aed75a57283d9490fd5705ed3d66bf9a\n
                 Python: cc1d2594aece0a064b7aed75a57283d9490fd5705ed3d66bf9adfe3a58b25de5\n
ripemd320     80 eb0cf45114c56a8421fbcb33430fa22e0cd607560a88bbe14ce\n
whirlpool    128 On PHP: 0a25f55d7308eca6b9567a7ed3bd1b46327f0f1ffdc804dd8bb\n
                 Python: 0a25f55d7308eca6b9567a7ed3bd1b46327f0f1ffdc804dd8bb5af40e88d78b88df0d002a89e2fdbd58
```

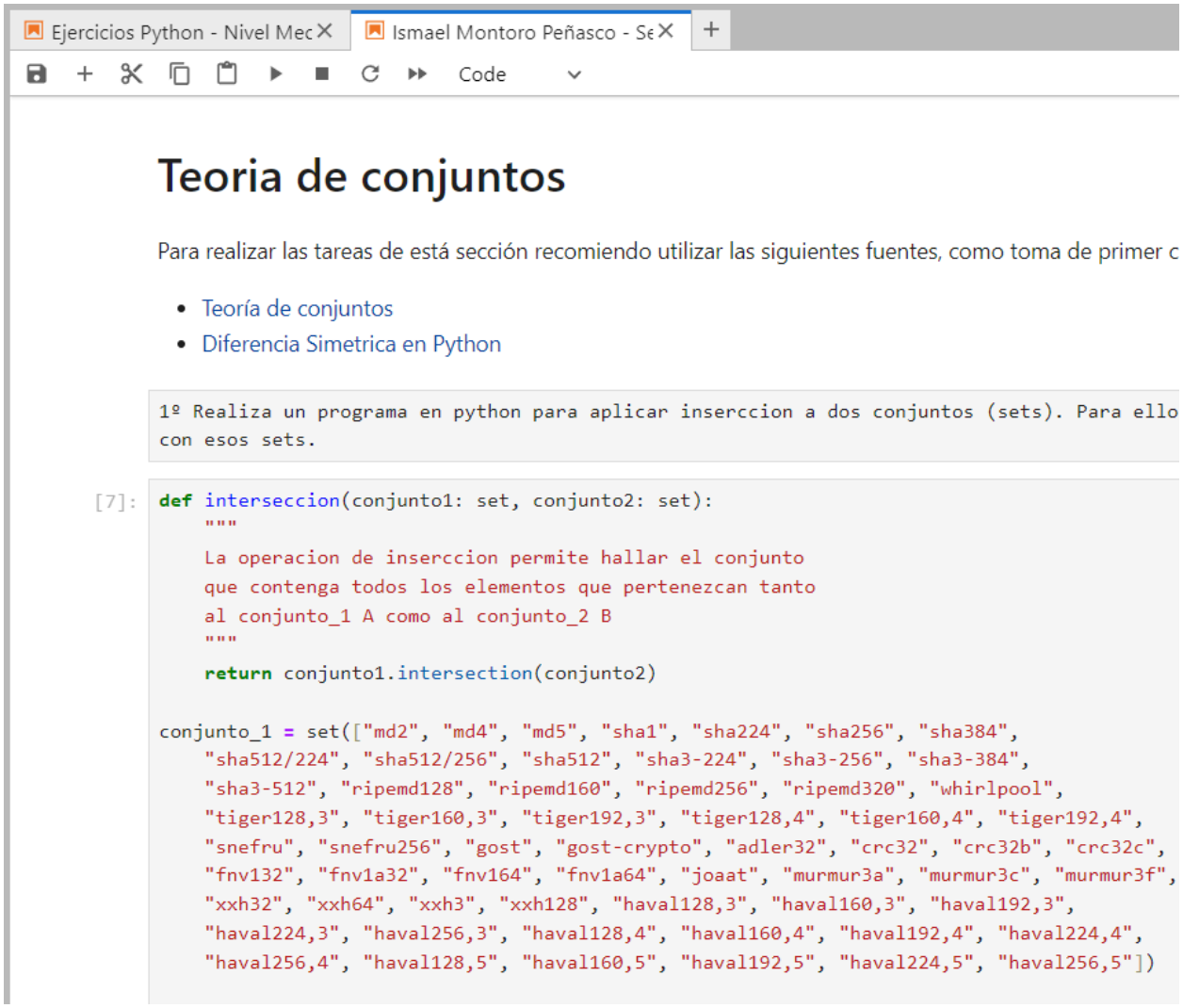
```python
    def ripemd320(self, texto = "hello") -> str:
        """
        Los algoritmos de ripemd sirven para trabajar con aplicaciones
        que prefieren utilizar valores de hash muy grandes y no les es
        necesario tener un nivel de seguridad alto
        """
        return RIPEMD320().generate_hash(texto)

    def whirlpool(self, texto = "hello") -> str:
        lib_whirlpool = whirlpool.new()
        lib_whirlpool.update(texto)
        return lib_whirlpool.hexdigest()

```

Labor 2: Ejercicios de la semana

He realizado los 10 ejercicios de la semana, con los cuales he podido investigar la teoría de conjuntos, los algoritmos y estructuras de datos, funciones de la API de Python.

Labor 2.1: Teoría de conjuntos en Python

He realizado 4 ejercicios para estudiar los conceptos de la teoría de conjuntos aplicada en conjuntos (sets) en Python.



Labor 2.2: Algoritmos y estructuras de datos

He realizado 3 ejercicios de algoritmos y estructuras de datos en Python.

En un algoritmo de árbol binario de búsqueda, podemos encontrarnos con 4 tipos de operaciones,

- Búsqueda: es una operación que buscara primero en el nodo raiz, si coincide se termina la bus
  del nodo raiz, y si el valor a buscar es mayor que el valor del nodo raiz buscara por el subarbol
- Inserccion: permite insertar nuevos elementos al árbol binario de búsqueda, tiene incluido las f
  arbol binario de busqueda no exista, creara un nuevo insertando el valor como el nuevo nodo
- Borrado: permite eliminar elementos que esten dentro del arbol. **Cuidado porque si borrais u**
- Recorridos: los recorridos son varias formas que tiene el algoritmo de buscar un nodo concreta
  - Preorden: recorremos el arbol desde la raiz hasta los nodos inferiores, de izquierda a dere
  - Inorden: recorremos el arbol desde la izquierda del subarbol izquierdo, luego examina la r
  - Posorden: por decirlo del alguna manera, recorremos el arbol primero por el subarbol izqu
    completo a la izquierda, para luego examinar a la izquierda

Fuente: Arbol Binario de Busqueda

]:

```python
from __future__ import print_function

# Esta clase servira para crear objetos que haran de Nodo
class Node:

    """
    Cada Nodo tendra la siguiente informacion
        - El titulo o valor numerico
        - Si pertenece al subarbol izquierdo o derecho
        - Una referencia a su nodo padre
    """
    def __init__(self, label, parent):
        self.label = label
        self.left = None
        self.right = None
        self.parent = parent
```

Labor 2.3: Funciones de Python

He realizado 3 ejercicios para investigar los recursos que nos ofrece la API de Python.

Funciones de Python

Fuente: Funciones en Python

```
8º Investiga la funcion zip incluida en la API de Python.
```

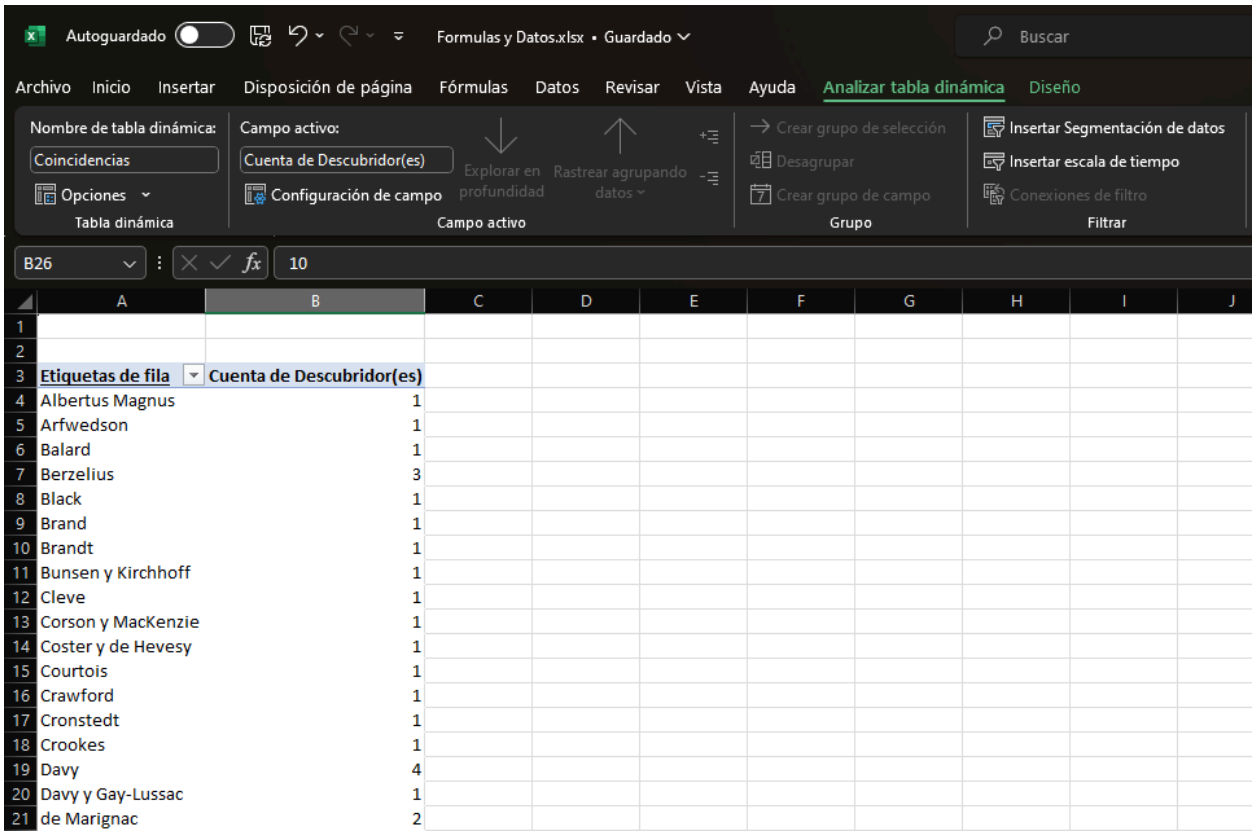
La funcion zip() permite comprimir dos estructuras de datos (listas o tuplas) con contenidos.

[3]:
```
lista = ["Maria", "Jose", "Daniel", "Ester", "Debora"]
identificar = range(len(lista))

print("Compresion completa")
# El parametro strict permite indicar si el numero de elementos de ambas listas tienen que ser iguales
comprimir = list(zip(identificar, lista, strict=True))
print(comprimir)
print("\tIterar archivos compresion")
for paquete in comprimir:
    print("\t", paquete)

print("Compresion incompleta")
comprimir = list(zip(range(2), lista))
print(comprimir)
```

```
Compresion completa
[(0, 'Maria'), (1, 'Jose'), (2, 'Daniel'), (3, 'Ester'), (4, 'Debora')]
        Iterar archivos compresion
         (0, 'Maria')
         (1, 'Jose')
         (2, 'Daniel')
         (3, 'Ester')
         (4, 'Debora')
Compresion incompleta
[(0, 'Maria'), (1, 'Jose')]
```

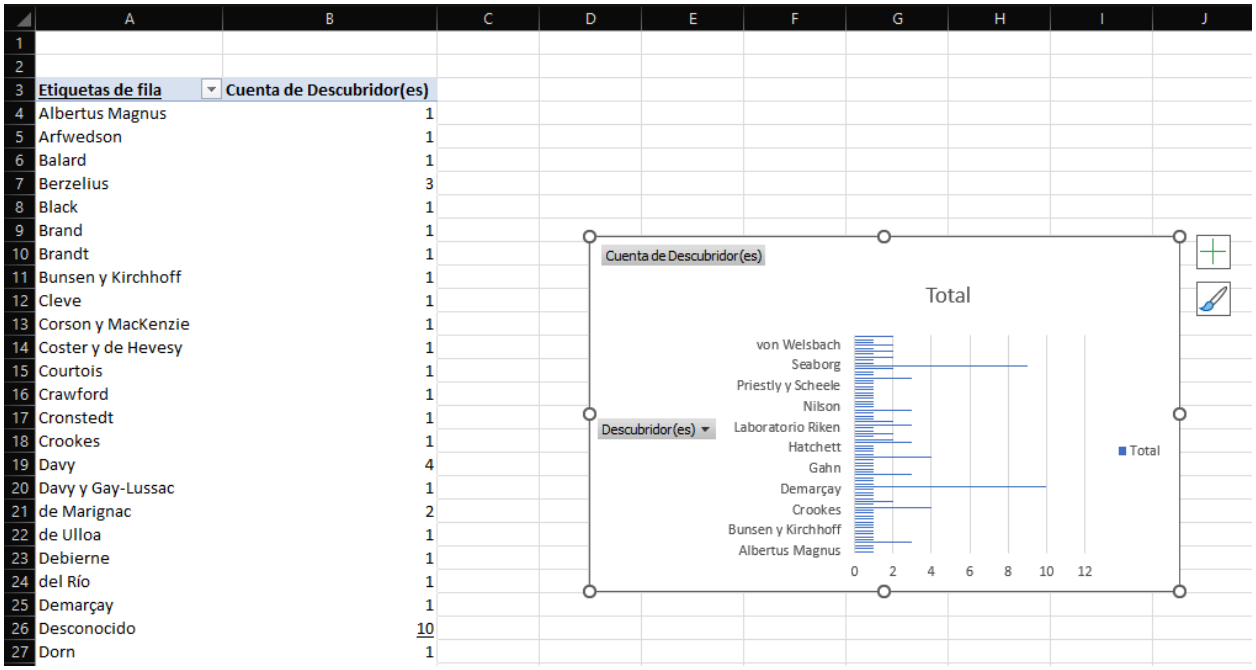
Labor 3: Prácticas con Excel

He creado una tabla dinámica para comprobar cuántos descubrimientos ha hecho cada científico.



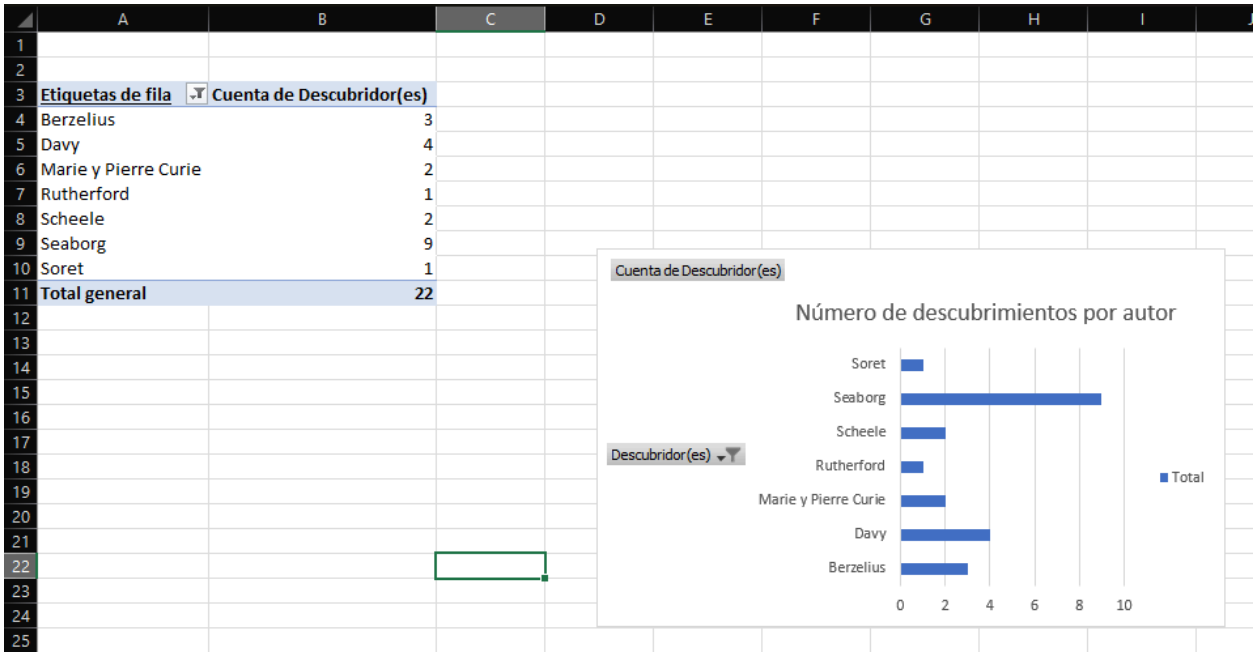
Etiquetas de fila	Cuenta de Descubridor(es)
Albertus Magnus	1
Arfwedson	1
Balard	1
Berzelius	3
Black	1
Brand	1
Brandt	1
Bunsen y Kirchhoff	1
Cleve	1
Corson y MacKenzie	1
Coster y de Hevesy	1
Courtois	1
Crawford	1
Cronstedt	1
Crookes	1
Davy	4
Davy y Gay-Lussac	1
de Marignac	2

Luego de haber creado la tabla dinámica, he creado un gráfico.



Etiquetas de fila	Cuenta de Descubridor(es)
Albertus Magnus	1
Arfwedson	1
Balard	1
Berzelius	3
Black	1
Brand	1
Brandt	1
Bunsen y Kirchhoff	1
Cleve	1
Corson y MacKenzie	1
Coster y de Hevesy	1
Courtois	1
Crawford	1
Cronstedt	1
Crookes	1
Davy	4
Davy y Gay-Lussac	1
de Marignac	2
de Ulloa	1
Debierne	1
del Río	1
Demarçay	1
Desconocido	10
Dorn	1

Y apliqué algunos filtros para no saturar la gráfica de tantos autores y que tuviera un formato más presentable. Además de darle un título a la gráfica.



Etiquetas de fila	Cuenta de Descubridor(es)
Berzelius	3
Davy	4
Marie y Pierre Curie	2
Rutherford	1
Scheele	2
Seaborg	9
Soret	1
Total general	**22**

Labor 4: Reuniones

He asistido a las 2 Masterclass de la semana, “Introducción a Linux” con la que he podido repasar y enterarme de algo nuevo, y “Algoritmia” donde he podido repasar las bases de la programación.

GoodJob

Labor 5: Scapy

He estado realizando un script de Python con funciones para realizar reconocimiento activo en un máquina objetivo, todo gracias a que he consultado a Bing IA ejemplos de uso de la librería Scapy.

He creado las siguientes funciones:

Una función para realizar un ping y que me un diccionario con los datos recabados del paquete:

```python
# ICMP
def ping(ips: list):

    diccionario_respuestas = dict()

    for ip in ips:
        # Creamos el paquete desde el protocolo mas bajo hasta el más alto
        paquete = IP(dst=ip)/ICMP()

        recibidos, perdidos = sr(paquete, timeout=1, verbose=0)
```

```python
if __name__ == "__main__":
    destinos = ["8.8.8.8", "140.126.124.108"]
    print(ping(destinos))
```

```
PS C:\Users\ismae\git\espadas> & c:/Users/ismae/gi
{'8.8.8.8': {'Origen': '192.168.1.34', 'Num_paquet
es_perdidos': 0.0}, '140.126.124.108': {'Origen':
: 0.0, 'Procentaje_paquetes_perdidos': 100.0}}
```

Una función para consultar los puertos de una máquina objetivo.

```
32  def analisis_puertos(ip: str, puertos: list):
33
34      for puerto in puertos:
35          paquete = IP(dst=ip)/TCP(dport=puerto, flags="S")
36          respuesta = sr1(paquete, timeout=1, verbose=0)
37          if respuesta and respuesta.haslayer(TCP):
38              if respuesta[TCP].flags == "SA":
39                  print(f"El puerto {puerto} está abierto")
40              elif respuesta[TCP].flags == "RA":
```

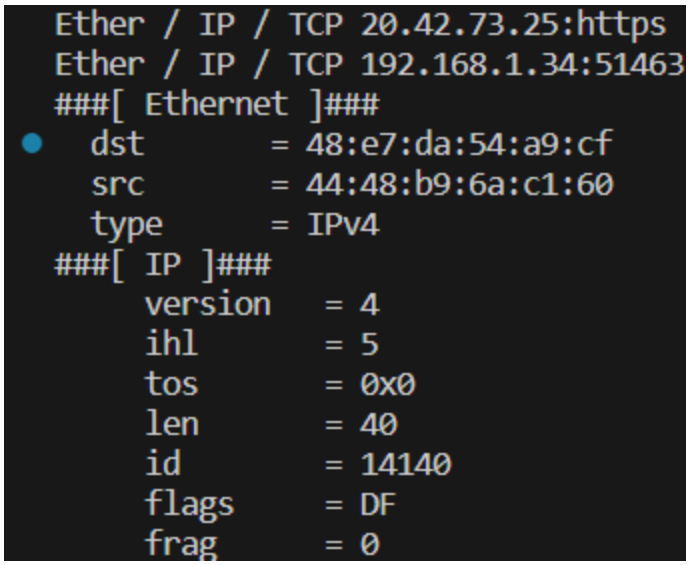
```
46  if __name__ == "__main__":
47      destinos = ["8.8.8.8", "140.126.124.108"]
48      print(ping(destinos))
49      analisis_puertos(destinos[1], [puerto for puerto in range(1, 100 + 1)])
```

```
El puerto 52 no responde
El puerto 53 está abierto
El puerto 54 no responde
El puerto 55 no responde
```

```
El puerto 80 está abierto
El puerto 81 no responde
El puerto 82 no responde
El puerto 83 no responde
```

He realizado una función que hará un sniffing a los paquetes de la red.

```
45  def capturar():
46
47      fichero = "tcp.pcap"
48      # Capturar 10 paquetes y mostrar su resumen
49      sniff(count=10, prn=lambda x: x.summary())
50      # Capturar solo los paquetes TCP y guardarlos en un fichero
51      resultado = sniff(filter="tcp", count=10)
52      wrpcap(fichero, resultado)
53
54      captura = rdpcap(fichero)
55      for paquete in captura:
56          paquete.show()
```



Planes para la próxima semana

Empezaré a investigar librerías de cifrado, codificaciones, redes (Scapy) y hacking a través de mi proyecto “espadas”, además de realizar ejercicios de la semana. Mi prioridad para seguir aprendiendo será mi proyecto espadas, además a partir de ahora iré haciendo los ejercicios de la semana dentro del proyecto de espadas. También seguiré investigando sobre Excel.