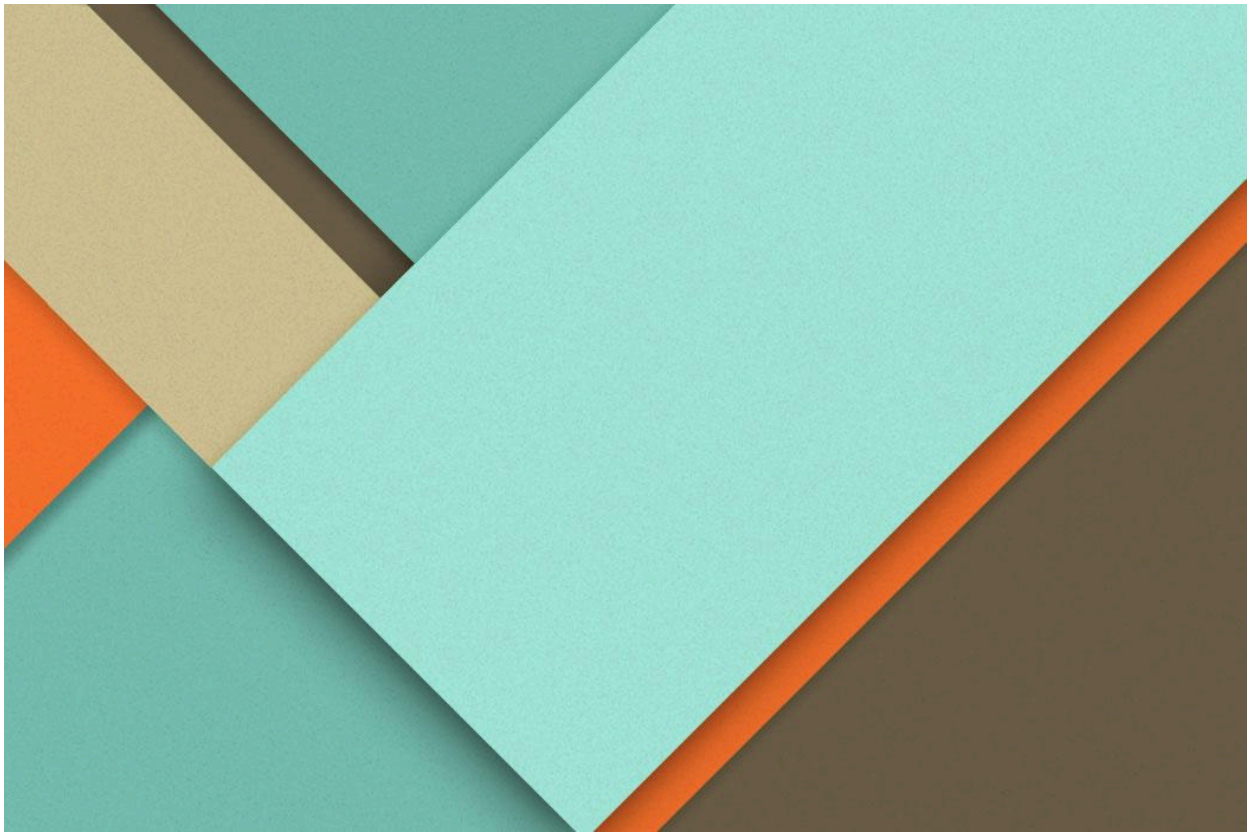


Reporte semanal - Semana 1

02/02/2024

Ismael Montoro Peñasco
Fundación GoodJob

Acciones llevadas a cabo

Decidí llevar a cabo un plan de estudio para subsanar la debilidad que tengo con Python, realizando ejercicios propuestos por GoodJob a modo de repaso del curso de Python impartido por Nanfor, y realizar pequeños experimentos para aumentar mis conocimiento de la API de Python, además de realizar un poco de Javascript.

Avances realizados

Labor 1: Generador de diccionarios

He avanzado con un proyecto personal con Python 3 para generar diccionarios para ataques de fuerza bruta mucho más eficientes a la hora de leerlos, además este programa en python 3 a pesar de ser un lenguaje interpretado, a demostrado ser eficiente gracias al uso del paradigma de la programación en paralelo y la programación multiproceso gracias a los procesos.

Además he podido buscar los cálculos matemáticos necesarios para hacer más eficiente el programa, para por ejemplo encontrar para repartir de manera eficiente el número de cadenas que le tocaría procesar cada proceso, gracias a Bing IA.

Tecnologías investigadas

- La librería itertools.product de Python 3.
- La librería multiprocessing de Python 3.
- Búsquedas en Bing IA.

Muestras

Aquí dejo dos trozos del código fuente del generador del diccionario.

```python
import itertools
import multiprocessing

def generador_diccionario(listado_palabras: list, nombre_diccionario: str) -> None:
    if nombre_diccionario:
        with open(nombre_diccionario, "w", encoding="utf-8") as archivo:
            archivo.writelines(listado_palabras)
    else:
        print("Introduce un nombre")

def generador_cadena(decimales_ascii: tuple) -> str:
    """
    Aqui dejo un chueta del numero de cadenas a generar
        Mayusculas (May) = 25 (o 0 en caso de que no se usen) [ascii: 65 - 90]
        minusculas (min) = 25 (o 0 en caso de que no se usen) [ascii: 97 - 122]
        Digitos (D) = 10 (o 0 en caso de que no se usen) [ascii: 48 - 57]
        Metacaracteres (Mc) = 15; 6 [ascii: 32 - 47; ascii: 58 - 64)]

        (May + min + D + Mc) ** Longitud_Cadena = Num_cadenas
        (0 + 26 + 0 + 0) ** 3 = 26 ** 3 = 676
        (26 + 26 + 0 + 0) ** 3 = 52 ** 3 = 2704

        Cada proceso no debe de tener una carga de trabajo de
        500 cadenas de texto a generar cada uno, el proceso padre
        solo puede tener 7 procesos hijo activos
    """
    dec_ascii = list(decimales_ascii)

    map_chr_ascii = map(chr, dec_ascii)
    chr_ascii = list(map_chr_ascii)

    return "".join(chr_ascii) + "\n"

def calc_num_cadenas_generar(mayus: int, minus: int, digit: int,
                             metachar: int, longitud_cadena: int) -> int:
    # calculo = (26 + 26 + 0 + 0) ** 2
    num_cadenas = (mayus + minus + digit + metachar) ** longitud_cadena
    return num_cadenas

def finalizar(salida: int = 0) -> None:
    exit(salida)
```

Aquí hago las funciones básicas del programa.

- generador_diccionario: permite crear un fichero txt con el diccionario.
- generador_cadena: permite crear mediante un objeto de tipo producto el conjunto de cadenas a generar cadenas, si se procesa adecuadamente.
- calc_num_cadenas_generar: permite determinar el número exacto de combinaciones posibles si se le pasan ciertos tipos de caracteres.
- finalizar: finaliza el programa, el valor de salida por defecto es 0 (SIN ERRORES)

```
# Generaremos un objeto de tipo chain, que es un objeto para empaquetar varios objetos
# dentro del parametro estamos desempaquetando todos los range de la lista para despues
# empaquetarlos dentro del objeto chain.
caracteres_seleccionados = itertools.chain(*caracteres_listados)

# Generaremos un nuevo objeto de tipo product, en el cual iteramos todos los objetos range
# del chain para que conviertan a tuplas dentro del objeto producto, repeat es para indicar
# cuantas veces se va a replicar el chain (el rango de varios rangos) dentro del producto,
# y para indicar cuantas veces se tiene que replicar se tiene que especificar en el parametro
# repeat. Tu piensa que "caracteres_seleccionados" es una columna de caracteres, y repeat el
# numero de veces que se repite dicha columna, para luego realizar todas las combinaciones
# de cadenas posibles, generando cadenas unicas
producto = itertools.product(caracteres_seleccionados, repeat=longitud)
```

Con este código generó todas las posibles combinaciones que se pueden realizar con los caracteres que selecciones el usuario.

```
526 if longitud < 3:
527     cont = 0
528     for sublist in producto:
529         cadena_generada = generador_cadena(sublist)
530         # cont = cont + 1
531 
532     # print(cont, resultado)
533     finalizar() # Finalizo el programa aqui dado que hemos logrado el objetivo
534 
535 # Aplicar programacion multiproceso cuando la longitud de la cadena (repeat) sea más de dos
536 if longitud > 2 and __name__ == "__main__":
537 
538     """
539     Cadenas de longitud 3 necesitan 7 procesos
540     Cadenas de longitud 4 necesitan  procesos
541     """
542     pool = multiprocessing.Pool(7)
543     diccionario = pool.map(generador_cadena, producto)
544 
545     generador_diccionario(diccionario, "Ideas\\diccionario.txt")
546     pool.close()
547     pool.join()
```

En caso de que el conjunto de cadenas tengan una longitud de tres caracteres cada una se procesarán mediante un bucle for debido a que la cantidad de cadenas a procesar es mínima, sin embargo en caso de que la longitud de cada cadena sea mayor a dos caracteres, se tendrá que procesar las cadenas con programación paralela, mediante multiprocessing,

Labor 2: Primer módulo del curso de Javascript

He completado el primer módulo del Curso de Javascript de manera autodidacta, con ello he podido repasar algunos conceptos básicos de Javascript. Además he podido refrescar la memoria de algunas cosas que he investigado en Visual Studio Code.

Tecnologías investigadas

- Hojas internas y externas de Javascript.
- Algunas utilidades del Editor de Visual Studio Code
    - Abreviación Emmet que se usa escribiendo !, que sirve para crear el código básico de HTML
    - El autocompletado que ya viene configurado por defecto.
    - La anotación HTML:5 que tiene la misma función que la Abreviación Emmet, pero para HTML5, no está en el Curso de JavaScript, pero cuanto se me mostró la Abreviación Emmet en el curso lo recordé.
- Algunas instrucciones básicas.
- Las reglas básicas de cómo utilizar Javascript.

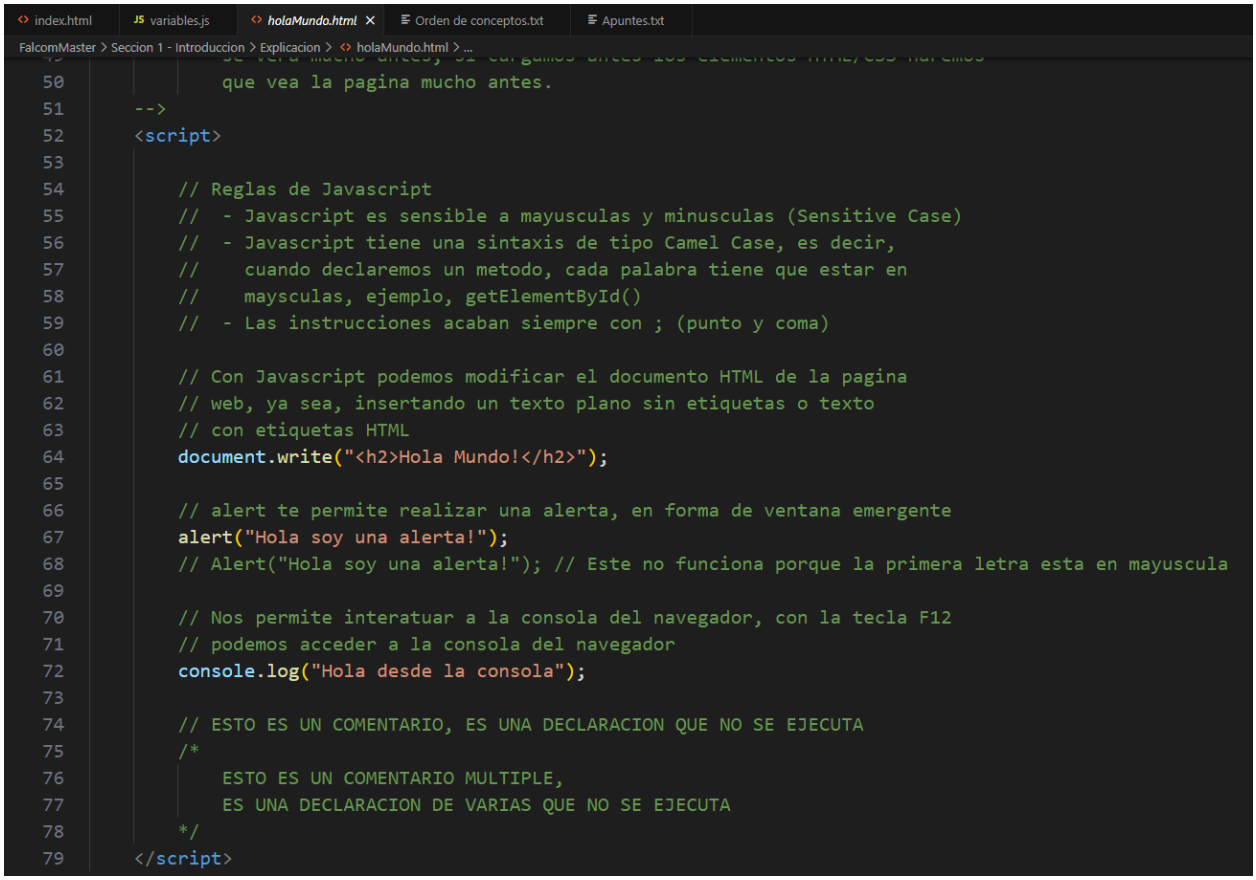
Muestras

```
<!--
    PENIULTIMA COSA: podemos poner la hoja interna de Javascript dentro del head

    IMPORTANTE: como estamos ejecutando codigo desde el head por defecto, el
    codigo de javascript se ejecutara antes que carguen los elementos HTML/CSS
    lo que disminuira el rendimiento, para evitar esto, hay un atributo para
    la etiqueta <script> que se llama defer que lo puedes poner en cualquier
    etiqueta script del documento y que sirve para indicar al navegador que
    que primero tiene que cargar el contenido del <body> primero
-->
<script defer>
    alert("Se ejecuta desde el head");
</script>
<!--
    ULTIMA COSA: puedes agregar codigo Javascript con hoja externa y
    la etiqueta script y el atributo src

    IMPORTANTE: esta es la mejor forma de trabajar con Javascript
-->
<script defer src="./holaMundo.js"></script>
```

Uso de hojas internas y hojas externas.



Instrucciones y nomenclatura de JavaScript.

Labor 3: Ejecutar Java en Python

Hoy he podido utilizar los recursos del lenguaje de Java dentro de un script de Python, gracias a las búsquedas de Bing IA.

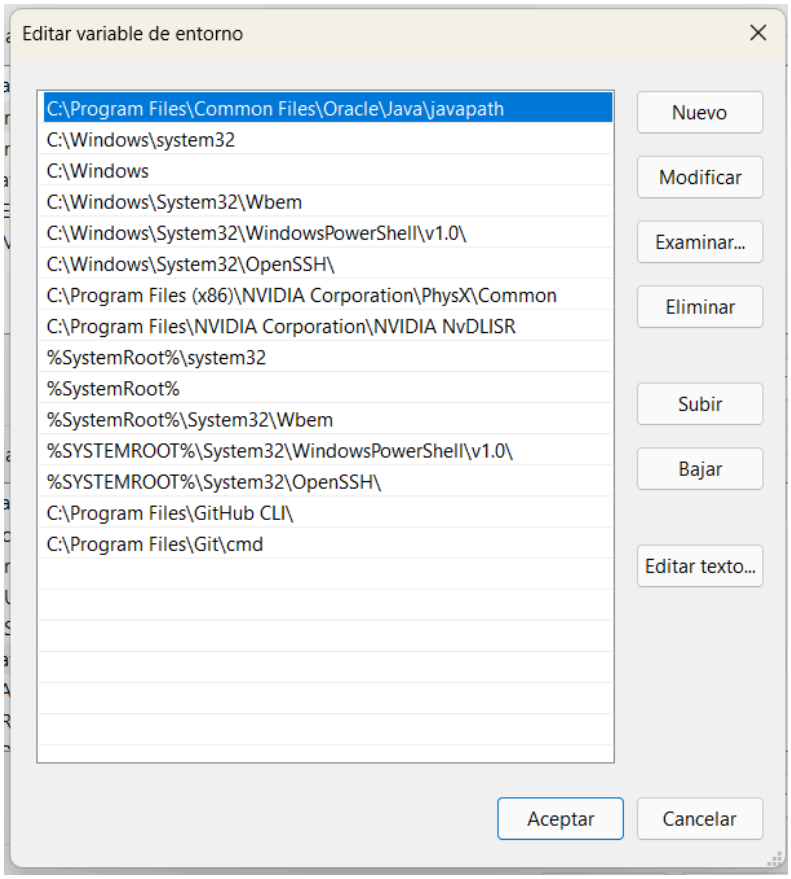
Tecnologías utilizadas

- La librería subprocess de Python 3.
- La API de Java JDK 21.
- La librería de gnu-crypto.jar.
- Buscador de Bing IA.

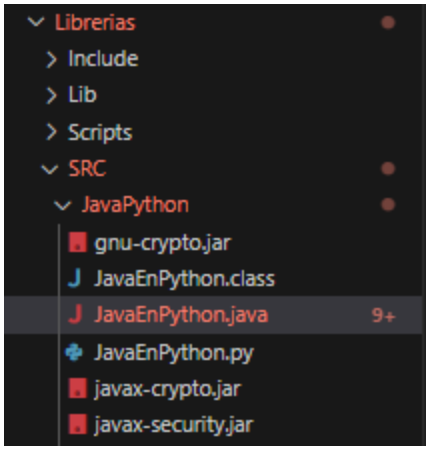
Muestras

1. Instalar la API de JDK 21.

2. Reiniciar el ordenador para aplicar los cambios necesarios, y después de reiniciar, comprobar si la API de Java está referenciada en las variables de entorno del sistema.



3. Colocamos el código Java, la librería de GNU-Crypto con sus dependencias en la misma carpeta. Importante esto para no tener tantos problemas con las rutas a la hora de programar.



4. Crear un código de Java que imprima un resultado por pantalla, en este caso he creado un código para utilizar el algoritmo de hash Haval.

NOTA: No hagas caso de los errores, son falsos positivos porque el Visual Studio Code no detecta la librería, pero no necesitamos que la detecte.

```java
package Librerias.SRC.JavaPython;
import java.math.BigInteger;
import gnu.crypto.hash.Haval;

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

    Run | Debug
    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] + " " + cifrado[2]);
            }

        // System.out.println("\n" + encriptarTiger(mensaje));
    }

    public static byte[] codificacionBytes(String mensaje) {
        return mensaje.getBytes();
    }

    public static String codificacionHexadecimal(byte[] mensaje) {
        return new BigInteger(signum:1, mensaje).toString(radix:16);
    }
```

```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 pueda tener implementado
            un malware Chino con la ayuda de virustotal.com, y se ha tenido que realizar ciertas acciones para
            descargar la informe sin el malware.

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

            https://www.virustotal.com/gui/url/5b134512fb654b3194799f2f901a0b508275bbae792a0029bc2ff9d93211ec5f/detection
        */

        final int size[] = {Haval.HAVAL_128_BIT, Haval.HAVAL_160_BIT, Haval.HAVAL_192_BIT, Haval.HAVAL_224_BIT, Haval.HAVAL_256_BIT};
        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();

        return new String[] {tamanos_disponibles[bits], rondas_disponibles[ronda], codificacionHexadecimal(mensaje_cifrado)};

    }

    /*public static String encriptarTiger(String mensaje) {

        Tiger tiger = new Tiger();
        byte[] codificacion = codificacionBytes(mensaje);
        tiger.update(codificacion, 0, codificacion.length);

        return codificacionHexadecimal(tiger.digest());
    }*/

}
```

5. Crear un Script en Python 3 que importe la librería subprocess.

```python
Librerias > SRC > JavaPython > JavaEnPython.py > ...
# Recuerda primero instalar el entorno de Java JDK,
# puedes utilizar el instalador que hay dentro del
# directorio
    # https://www.java.com/es/download/
    # https://www.oracle.com/java/technologies/downloads/#java21

import subprocess
```

6. Invocamos los subprocess que necesitamos para primero compilar el código de Java, y después ejecutarlo con un parámetro en el comando, en este caso le pasamos el texto “hola”.

```python

# Compila tu archivo Java con la gnu-crypto jar
subprocess.run(["javac", "-cp", \
                "C:\\Users\\ismae\\Downloads\\EstudiosPython\\Librerias\\SRC\\JavaPython\\gnu-crypto.jar", \
                    "C:\\Users\\ismae\\Downloads\\EstudiosPython\\Librerias\\SRC\\JavaPython\\JavaEnPython.java"])

# Ejecuta tu archivo Java con la gnu-crypto jar
resultado = subprocess.run(["java", "-cp", \
                "C:\\Users\\ismae\\Downloads\\EstudiosPython;C:\\Users\\ismae\\Downloads\\EstudiosPython\\Librerias\\SRC\
                    "Librerias.SRC.JavaPython.JavaEnPython", "hola"], capture_output=True, text=True)

resultado = resultado.stdout.split("\n")

if __name__ == "__main__":
    for haval in resultado:
        print(haval)
```

```
PROBLEMAS 12  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS                                                    Python

PS C:\Users\ismae\Downloads\EstudiosPython> & c:/Users/ismae/Downloads/EstudiosPython/Librerias/Scripts/python.exe c:/Users/ismae/Downloads/EstudiosPython/Librerias/SRC/JavaPython/JavaEnPython.py
Haval128,3 323ea1968f683774abcf19027772bf2
Haval160,3 ee2a176edd33d9fc430c3ceca145ad5a30247c7b
Haval192,3 298bdff86ade3446f7c7a76c7e7cff34f9414a9303574d76
Haval224,3 184a3fe8669a3a282f919587aeb380027a8e9e1aa3ea232fb683f108
Haval256,3 55761bcc66f74f1ddade2b4d4141f967eac41868d99e5d2c6d8c00f9ab803e45
Haval128,4 b2448d944dde3dc692e8b76af049d740
Haval160,4 a79c5af62e5bec73a65f5ab0da7f4cee1e2c67f2
Haval192,4 1b975ec75f7c105069c6bcbeaff1873d23f18403a644e292
Haval224,4 d15f355e8f81c67ee257dd0110d0ba42610ef22cf7ceddb6ebab37cb
Haval256,4 ebbb9d37b148b3d363112b4a8f2a78677864a98610b7fee8e1a9d0d9f6447bf6
Haval128,5 f802ec3a576b5734665bb856234f7686
Haval160,5 3bec88a42eadd264b7ec8a48ec655c26cac9d673
Haval192,5 f64fbe4b2a633f4600f48541b11f090475b29df8f1b0c3dc
Haval224,5 b18f142dae51a841f246978a9abf2fb3877a7381fb7a14928317cfe4
Haval256,5 6717de82842973fa1b3ccfbf06f98d8ee6e1a186b470959a6fc60dcbcc0ae5ca

PS C:\Users\ismae\Downloads\EstudiosPython>
```

Como puedes comprobar el programa devuelve todos los hashes que devuelve el algoritmo de Haval a partir del parámetro “hola”.

Labor 4: Los 10 ejercicios de Python

Como Román Ramírez Giménez de RootedCon nos dijo que cada semana nos irá pidiendo 10 ejercicios de Python hace ya algunas semanas, el día 18 de Enero pedi 10 ejercicios de Python a Bing IA, y los hice, cuando termine dichos ejercicios que eran fáciles, pase a ejercicios más complicados, y pude realizar algunos de ellos. Aquí dejo una captura del archivo de Júpiter Notebook que es una mezcla de los ejercicios fáciles y ejercicios difíciles.

Ejercicios de Python.ipynb

Code Notebook Python (Pyodide)

1º Crea una función que tome una lista y devuelva una nueva lista con los elementos únicos de la lista original.

[30]:
```python
lista = [5,8,9,2,6,4,10,5,9,2,1,5,9,10,9,6,1,10]

# Creamos una funcion que utilice un set para eliminar la redundancias
def eliminar_redundancias(lista: list):
    return list(set(lista))

print("La lista", lista, "ha sido resumida a", eliminar_redundancias(lista))
```

```
La lista [5, 8, 9, 2, 6, 4, 10, 5, 9, 2, 1, 5, 9, 10, 9, 6, 1, 10] ha sido resumida a [1, 2, 4, 5, 6, 8, 9, 10]
```

2º Escribe un programa que imprima los primeros 10 números de la serie de Fibonacci.

[29]:
```python
fibonacci = [0,1] # Creamos una lista inicial con los primeros valores de la secuencia de Fibonacci.

for _ in range(8): # Elegimos cuantas veces vamos a realizar la operacion de Fibonacci con un for y un range
    fibonacci.append(fibonacci[-1] + fibonacci[-2]) # Sumamos el penultimo y el ultimo valor de la lista

print(fibonacci)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

3º Crea una función que implemente el algoritmo de ordenación por mezcla (merge sort)

Planes para la próxima semana

Aumentar el tiempo que le dedicaré, a medida que pasen los días le dedicaré un poco más de tiempo a descubrir nuevas librerías externas de Python y funciones nativas de la API de Python, y si es posible le dedicaré más tiempo a Javascript.

Empezaré los tutoriales de la suite Office empezando por Excel, me concentraré concretamente en la pestaña de Datos, Fórmulas y si da tiempo investigaré Python para Excel.

Empezaré a usar herramientas de inteligencia artificial en local para ir probando distintos modelos de IA.

Mínimo

1. Investigar herramientas de IA.
2. Investigar Fórmulas y Datos de Excel.
3. Librerias Externas de Python.
4. API de Python.