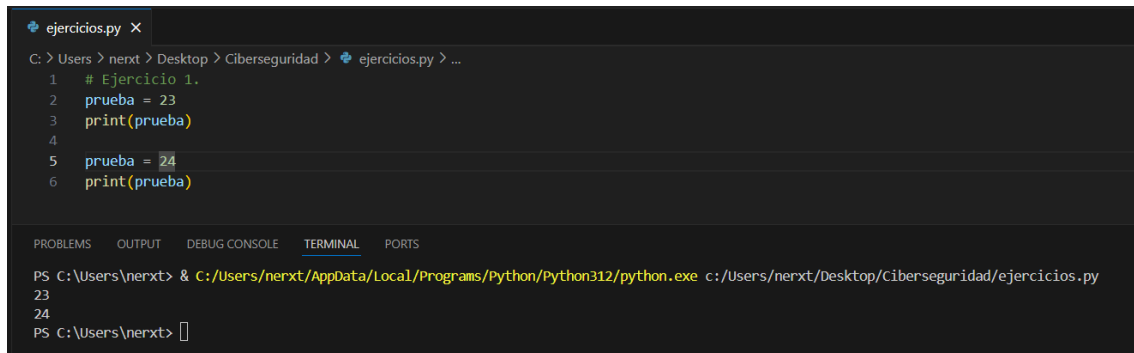


# EJERCICIOS PYTHON

1. Definir una variable que se llama "prueba", asignarle el valor "23" e imprimirla por pantalla. Tras la impresión, asígnale el valor "24" y volver a imprimirlo.

**Explicación:** Una variable (valor que puede variar incluso después de ser asignado) en Python se define al poner el nombre (prueba) seguido de un igual (=) y el valor de la variable (23). Para poder ver el resultado al imprimir, debemos usar la segunda línea, el término "print" seguido de un paréntesis en el que vamos a introducir lo que queremos imprimir, en nuestro caso, la variable llamada prueba. Tras escribir las dos primeras líneas, ejecutamos el código y deberíamos ver un 23. Si tras ejecutarlo cambiamos el valor de la variable a 24 (usando la sintaxis de las líneas siguientes) y lo ejecutamos, podremos comprobar que el valor de la variable ha cambiado.

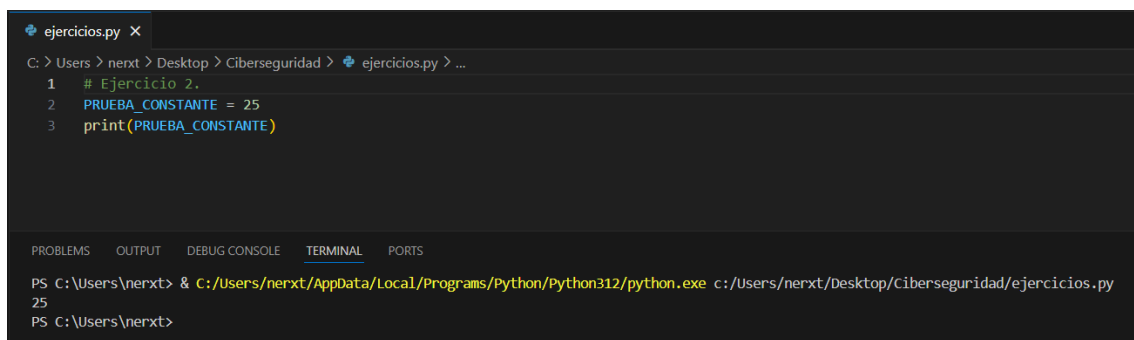


```
ejercicios.py X
C: > Users > nerxt > Desktop > Ciberseguridad > ejercicios.py > ...
1 # Ejercicio 1.
2 prueba = 23
3 print(prueba)
4
5 prueba = 24
6 print(prueba)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nerxt> & C:/Users/nerxt/AppData/Local/Programs/Python/Python312/python.exe c:/Users/nerxt/Desktop/Ciberseguridad/ejercicios.py
23
24
PS C:\Users\nerxt>
```

2. Definir una variable que se llama "PRUEBA\_CONSTANTE", asignarle el valor "25" e imprimirla por pantalla.

**Explicación:** Para este ejercicio, vamos a asignar un nombre de variable con mayúsculas y un guión bajo (\_) para demostrar que una constante puede tener estos caracteres. Luego, lo imprimimos como hemos hecho en el anterior ejercicio.

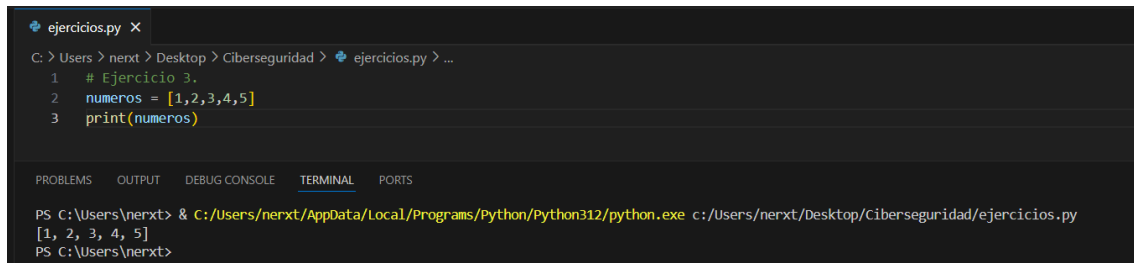


```
ejercicios.py X
C: > Users > nerxt > Desktop > Ciberseguridad > ejercicios.py > ...
1 # Ejercicio 2.
2 PRUEBA_CONSTANTE = 25
3 print(PRUEBA_CONSTANTE)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nerxt> & C:/Users/nerxt/AppData/Local/Programs/Python/Python312/python.exe c:/Users/nerxt/Desktop/Ciberseguridad/ejercicios.py
25
PS C:\Users\nerxt>
```

3. Definir un array de 5 elementos que contengan los números 1, 2, 3, 4 y 5, e imprimirlo por pantalla.

**Explicación:** Un array (estructura de datos que almacena elementos del mismo tipo (como números, cadenas, objetos) en un solo contenedor.) se puede definir de la misma forma que una variable, excepto porque el valor debe ir entre corchetes y separando los elementos con comas. Una vez definido, lo imprimimos para comprobar que sale todo.



```
ejercicios.py X
C: > Users > nerxt > Desktop > Ciberseguridad > ejercicios.py > ...
1 # Ejercicio 3.
2 numeros = [1,2,3,4,5]
3 print(numeros)

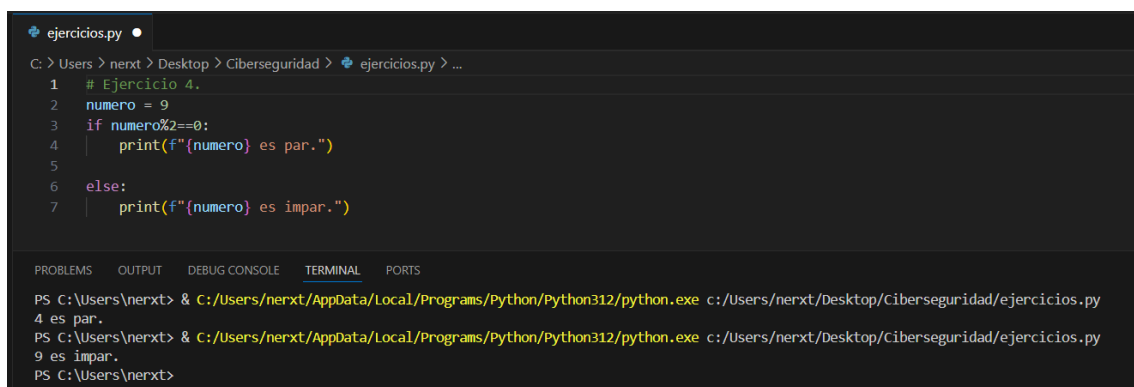
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nerxt> & C:/Users/nerxt/AppData/Local/Programs/Python/Python312/python.exe c:/Users/nerxt/Desktop/Ciberseguridad/ejercicios.py
[1, 2, 3, 4, 5]
PS C:\Users\nerxt>
```

4. Define una variable llamada "numero", y asígnale un valor. A continuación, utilizando un if-else, imprime por pantalla si el número introducido es par o impar.

**Explicación:** En este ejercicio, empezaremos definiendo la variable “número”, asignándole el valor 9 a modo de ejemplo (puede ser cualquier otro número). Después, debemos definir un bucle if-else (se indica una condición que, al evaluarse, devuelve verdadero o falso y en función del resultado ejecuta uno u otro bloque.).

Para definir el bucle, empezamos indicando en el “if” la condición que debe cumplir para imprimir el resultado. En mi caso, la condición era que, si la variable dividida por 2 tantas veces como fuera necesario, daba como resultado 0, se debía imprimir que el número de la variable era par. En caso de que esa condición no se cumpla, pasaría al siguiente bloque y escribiría que el número 9 es impar.

Como se puede apreciar, estamos haciendo uso del conocido como “f-string” (print(f“{numero} es par.”)) para imprimir los valores. Esto se debe a que dentro de la cadena a imprimir, puede haber expresiones que serán evaluadas, y cuyo resultado se incluirá en el texto.



```
ejercicios.py
C: > Users > nerxt > Desktop > Ciberseguridad > ejercicios.py > ...
1 # Ejercicio 4.
2 numero = 9
3 if numero%2==0:
4     print(f"{numero} es par.")
5
6 else:
7     print(f"{numero} es impar.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nerxt> & C:/Users/nerxt/AppData/Local/Programs/Python/Python312/python.exe c:/Users/nerxt/Desktop/Ciberseguridad/ejercicios.py
4 es par.
PS C:\Users\nerxt> & C:/Users/nerxt/AppData/Local/Programs/Python/Python312/python.exe c:/Users/nerxt/Desktop/Ciberseguridad/ejercicios.py
9 es impar.
PS C:\Users\nerxt>
```

En este ejercicio, hay una trampa y esta es el número 0.

Si lo queremos añadir sin considerarlo un número par, cambiaríamos el bloque if para que, en caso de que la variable sea igual a 0, el texto sea el indicado y usaríamos un tercer bloque “elif” entre los dos anteriores con la división entre dos.

Esto se hace en este orden ya que, si ponemos antes la división que el cero, al introducir el valor de la variable como el número 0, siempre se cumplirá que es par y nunca se llegaría al segundo bloque, por lo que la sintaxis será la siguiente:

```
numero = 9

if numero == 0:
    print("Este número es el cero.")
elif numero % 2 == 0:
    print(f"{numero} es par.")
else:
    print(f"{numero} es impar.")
```

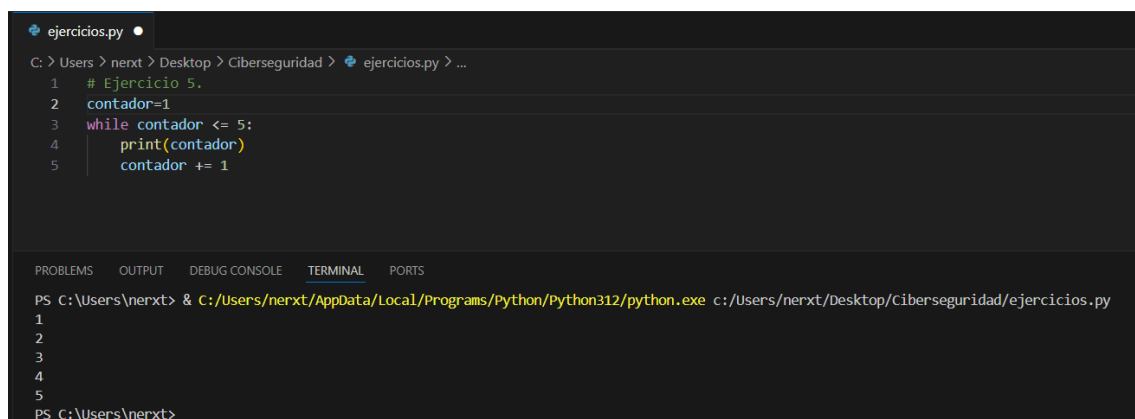
5. Crear un programa que imprima los números del 1 al 5 utilizando un bucle while y una variable llamada “contador”, que se tendrá que incrementar una unidad en cada interacción del bucle.

**Explicación:** De la misma forma que en el anterior ejercicio, usaremos un bucle, pero este caso será un “while” (se ejecuta el bloque contenido en la estructura mientras se cumpla la condición lógica. Se podría dar el caso de que no se ejecutase nunca).

Como siempre, primero se define la variable, ahora debe ser el número 1 ya que va a ser el primer número que se imprima.

Una vez definida la variable, definimos la condición del bucle. Siguiendo las órdenes del enunciado, la condición debe ser que el valor de la variable “contador” sea menor o igual (<=) a 5. Si esto se cumple, se imprime la variable “contador” y se le suma 1 (contador += 1) al valor de la variable.

En el momento en que la variable sea mayor que 5, el bucle para de imprimir.



```
ejercicios.py
C: > Users > nerxt > Desktop > Ciberseguridad > ejercicios.py > ...
1 # Ejercicio 5.
2 contador=1
3 while contador <= 5:
4     print(contador)
5     contador += 1

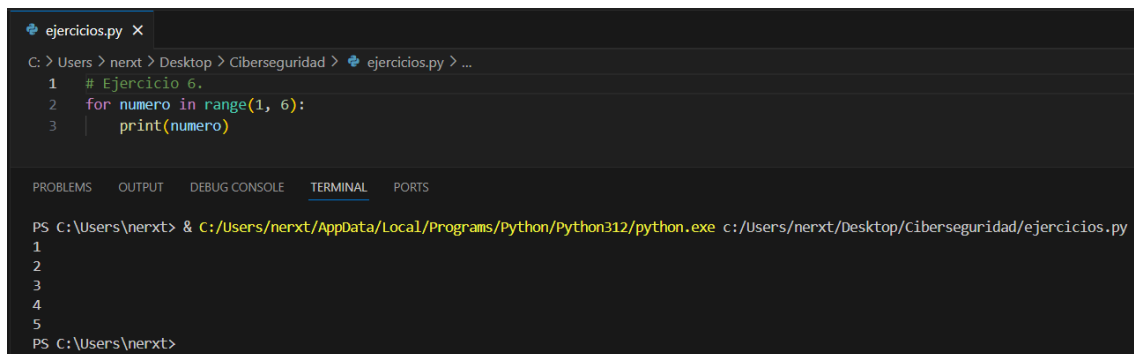
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nerxt> & C:/Users/nerxt/AppData/Local/Programs/Python/Python312/python.exe c:/Users/nerxt/Desktop/Ciberseguridad/ejercicios.py
1
2
3
4
5
PS C:\Users\nerxt>
```

## 6. Crear un programa que imprima los números del 1 al 5 usando un bucle for.

**Explicación:** Para este último ejercicio, usaremos otro bucle o sentencia de control iterativa, el conocido como bucle “for” (además de la condición, posee un bloque de instrucciones que se ejecutan al inicio, el bloque en el que se encuentran la condición a evaluar en cada interacción del bucle. Si bien es posible indicar varias instrucciones separadas por comas en cada elemento del bucle for, no suele ser usual utilizarlo ya que no es intuitivo.) .

Para este ejercicio, no hace falta indicar al principio una variable, pues irá dentro del bucle. Básicamente, lo que se indica en el código es que, para la variable número (for numero) en el rango del 1 al 6 (in range 1, 6), se imprima el número.

Esto se hace de esta manera ya que este bucle nunca muestra el último dígito del rango que se le indique, en este caso llegaría a mostrar hasta el 5.



```
ejercicios.py X
C: > Users > nerxt > Desktop > Ciberseguridad > ejercicios.py > ...
1 # Ejercicio 6.
2 for numero in range(1, 6):
3     print(numero)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nerxt> & C:/Users/nerxt/AppData/Local/Programs/Python/Python312/python.exe c:/Users/nerxt/Desktop/Ciberseguridad/ejercicios.py
1
2
3
4
5
PS C:\Users\nerxt>
```