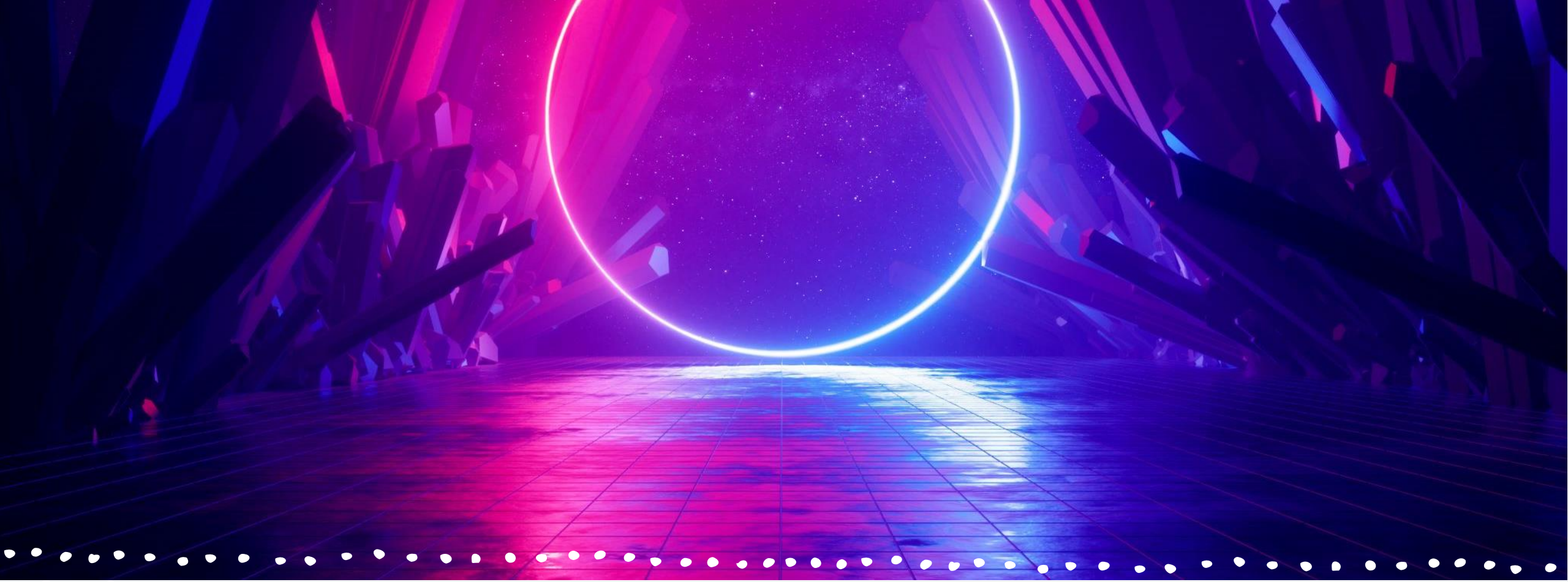


Pseudocódigo e introducción a los bucles (ciclos)

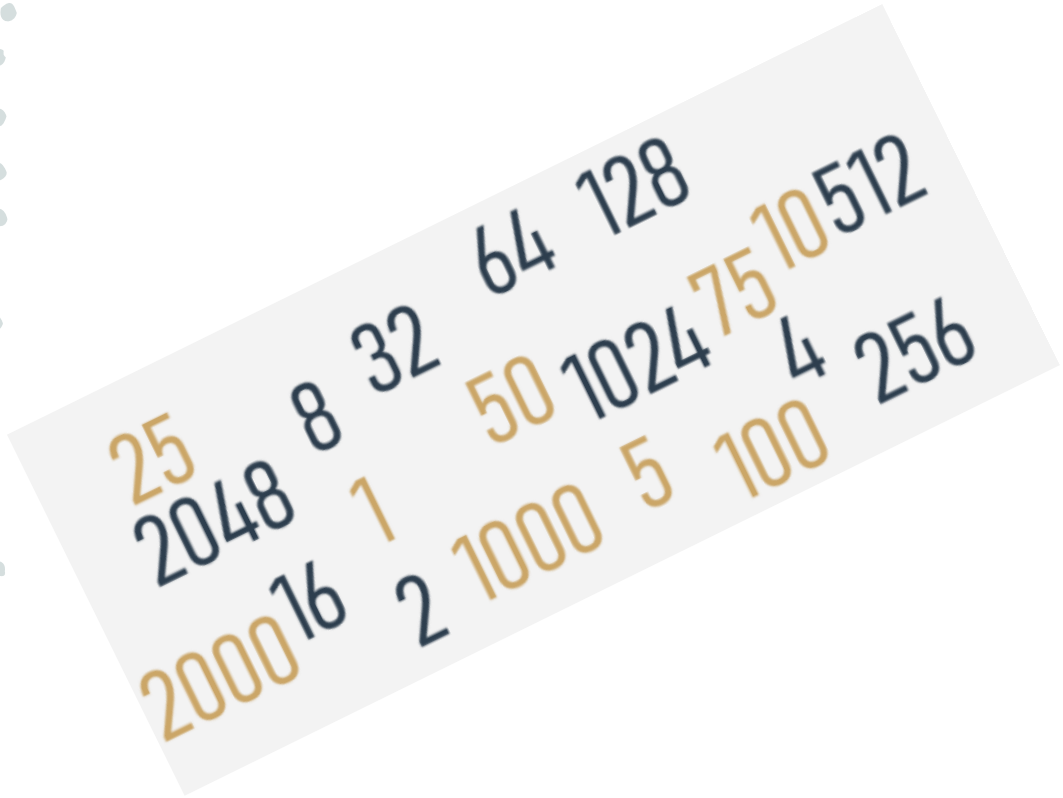
Ahora deberías poder escribir un programa que encuentre el mayor de cuatro, cinco, seis o incluso diez números.

Ya conoces el esquema, por lo que ampliar el tamaño del problema no será particularmente complejo.

¿Pero qué sucede si te pedimos que escribas un programa que encuentre el mayor de doscientos números? ¿Te imaginas el código?

Necesitarás doscientas variables. Si doscientas variables no son lo suficientemente complicadas, intenta imaginar la búsqueda del número más grande de un millón.

Imagina un código que contiene 199 sentencias condicionales y doscientas invocaciones de la función input(). Por suerte, no necesitas lidiar con eso. Hay un enfoque más simple.



- Por ahora ignoraremos los requisitos de la sintaxis de Python e intentaremos analizar el problema sin pensar en la programación real. En otras palabras, intentaremos escribir el **algoritmo**, y cuando estemos contentos con él, lo implementaremos.
- En este caso, utilizaremos un tipo de notación que no es un lenguaje de programación real (no se puede compilar ni ejecutar), pero está formalizado, es conciso y se puede leer. Se llama **pseudocódigo**.
- Veamos nuestro pseudocódigo a continuación:

```
1 largest_number = -999999999
2 number = int(input())
3 if number == -1:
4       print(largest_number)
5       exit()
6 if number > largest_number:
7       largest_number = number
8 # Ir a la línea 02
```

¿Qué está pasando en él?

En primer lugar, podemos simplificar el programa si, al principio del código, le asignamos a la variable largest_number un valor que será más pequeño que cualquiera de los números ingresados. Usaremos -999999999 para ese propósito.

En segundo lugar, asumimos que nuestro algoritmo no sabrá por adelantado cuántos números se entregarán al programa. Esperamos que el usuario ingrese todos los números que desee; el algoritmo funcionará bien con cien y con mil números. ¿Cómo hacemos eso?

Hacemos un trato con el usuario: cuando se ingresa el valor-1, será una señal de que no hay más datos y que el programa debe finalizar su trabajo.

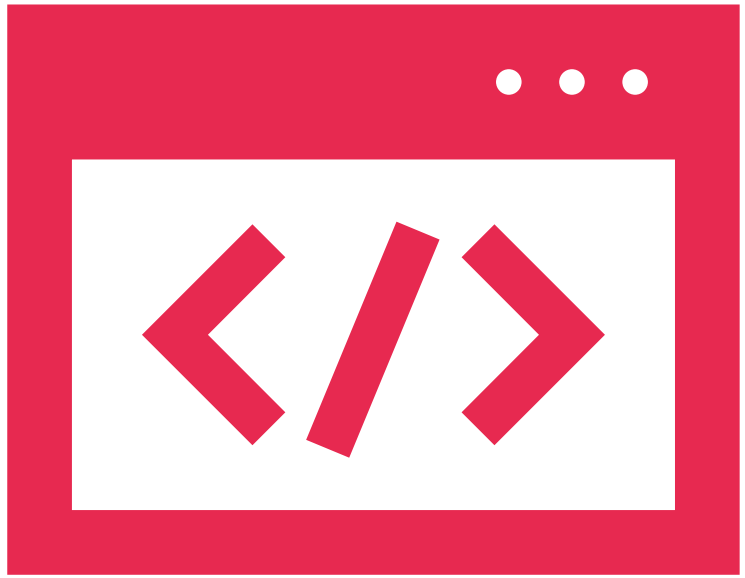
De lo contrario, si el valor ingresado no es igual a -1, el programa leerá otro número, y así sucesivamente.

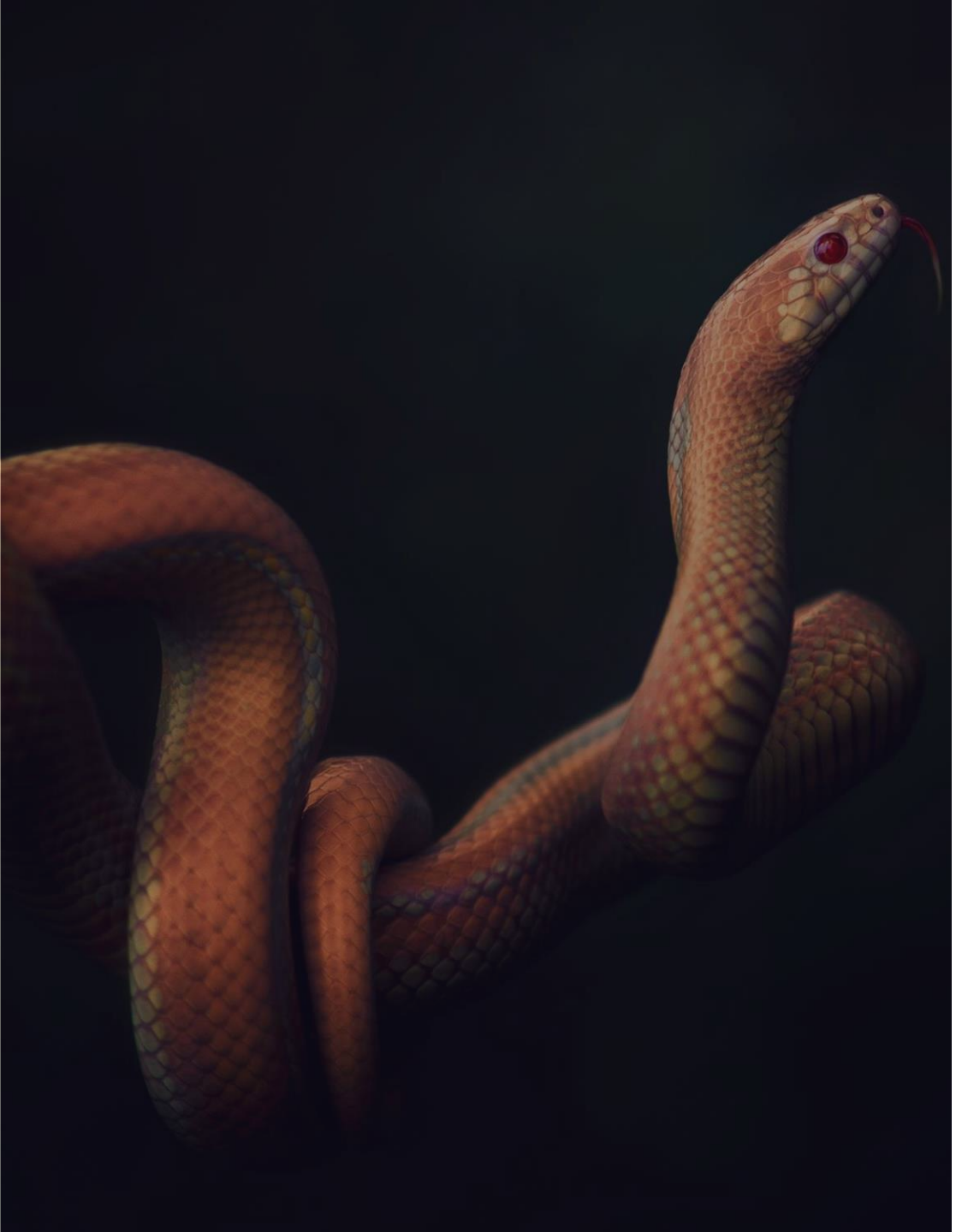
El truco se basa en la suposición de que cualquier parte del código se puede realizar más de una vez, precisamente, tantas veces como sea necesario.

La ejecución de una determinada parte del código más de una vez se denomina bucle. El significado de este término es probablemente obvio para ti.

Las líneas 02 a 08 forman un bucle. Los pasaremos tantas veces como sea necesario para revisar todos los valores ingresados.

¿Puedes usar una estructura similar en un programa escrito en Python?
Si, si puedes.





Python a menudo viene con muchas funciones integradas que harán el trabajo por ti. Por ejemplo, para encontrar el número más grande de todos, puede usar una función incorporada de Python llamada max(). Puedes usarlo con múltiples argumentos.

```python
# Se leen tres números.
number1 = int(input("Ingresa el primer número: "))
number2 = int(input("Ingresa el segundo número: "))
number3 = int(input("Ingresa el tercer número: "))

# Verifica cuál de los números es el mayor
# y pásalo a la variable largest_number

largest_number = max(number1, number2, number3)

# Imprime el resultado.
print("El número más grande es:", largest_number)
```

De la misma manera, puedes usar la función min() para devolver el número más pequeño. Puedes reconstruir el código anterior y experimentar con él en Sandbox.

Vamos a hablar sobre estas (y muchas otras) funciones pronto. Por el momento, nuestro enfoque se centrará en la ejecución condicional y los bucles para permitirte ganar más confianza en la programación y enseñarte las habilidades que te permitirán comprender y aplicar los dos conceptos en tu codigo. Entonces, por ahora, no estamos tomando atajos.