



## **Trabajo Práctico - Semana de Integración I**

**Título: “Contador binario de 0 a 15”**

### **Alumnos - Grupo n° 05**

Santiago Octavio Varela, Enderson Eduardo Suarez Porras, Damián Eduardo Tristant, Luis

Fernando Vega, Ximena Maribel Sosa.

**Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.**

**Matemática**

## Breve descripción del proyecto

Este proyecto, basado en la idea N°3 dentro de las ofrecidas en las pautas de este trabajo práctico, consiste en simular un contador binario que cuenta desde 0 hasta 15, mostrando cada número tanto en su forma decimal como en su correspondiente representación binaria.

- Entradas del programa: El programa no requiere entradas por parte del usuario. Todos los valores a procesar están definidos internamente dentro del rango de 0 a 15.
- Resolución del problema: El programa utiliza un bucle `for` para iterar sobre los números del 0 al 15. En cada iteración, convierte el número actual a binario mediante un ciclo anidado `while` que divide el número entre 2 y almacena los residuos usando módulo (`%`), formando así la representación binaria. Utilizamos la función `time.sleep(1)`, como lo sugiere la consigna, para introducir una pausa de un segundo entre cada número, simulando el ritmo de un contador como si el programa “pensara” cada respuesta brevemente.
- Salidas generadas: Por cada número del 0 al 15, el programa imprime en pantalla un mensaje que indica su valor en decimal y su equivalente en binario, sin agregar 0 (ceros) a la izquierda. Por ejemplo: "5 en binario es 101".

## Video explicativo

Link al video en YouTube: [https://youtu.be/2UnyzSSJ6-M?si=rUP7JRJ\\_JiAhVg-](https://youtu.be/2UnyzSSJ6-M?si=rUP7JRJ_JiAhVg-)

## Código Fuente en Python

Link al repositorio en GitHub:

[https://github.com/santiagovOK/UTN-TUPaD-P1/blob/main/mate\\_semanaIntegracion1/tp\\_semana-integracion-01.py](https://github.com/santiagovOK/UTN-TUPaD-P1/blob/main/mate_semanaIntegracion1/tp_semana-integracion-01.py)

Código utilizado (preferentemente ver en GitHub):

---

```
# Contador Binario
```

```
# Escriban un programa que, usando un ciclo, cuente desde 0 hasta 15 y muestre cada número en su representación binaria. Extensión: Utilicen un retardo (por ejemplo, con time.sleep) para simular el conteo de un circuito.
```

```
# ---
```

```
# Comienzo de código

import time # Importamos la biblioteca time para agregar la funcion `time.sleep()`.

for numero in range(0, 16):

    # Usamos `time.sleep()` para pausar 1 seg. la ejecucion del bucle en cada iteración.
    time.sleep(1)

    # Creamos una variable para almacenar el binario en tipo cadena
    binario = ""

    # Hacemos una copia para no modificar 'numero' del `for`
    n = numero

    # Creamos una condicional doble para agregar el 0.
    if numero == 0:

        print("0 en binario es 0")

    else:

        # Dentro de la condicional, agregamos el `while` que incluye el pasaje de cada uno de los
        # números del rango a binario.

        while n > 0:

            # Calculamos el residuo de la división por 2
            residuo = n % 2

            # Concatenamos el residuo al inicio de la cadena binaria.
            binario = str(residuo) + binario

            # Actualizamos n dividiendo por 2.
            n = n // 2

        # Imprimimos los resultados en cada iteración.
        print(f"{numero} en binario es {binario}")

# ---

# Fin de código
```

## Justificación acerca del uso de AI

En lo referente a utilizar inteligencia artificial (AI) como acompañamiento para realizar este pequeño proyecto, debemos arrancar diciendo que en la mayoría de las cuestiones nos fue relativamente sencillo establecer una estructura general del código a fines de solucionar el problema elegido por nosotros para Python sin hacer uso de herramientas de AI.

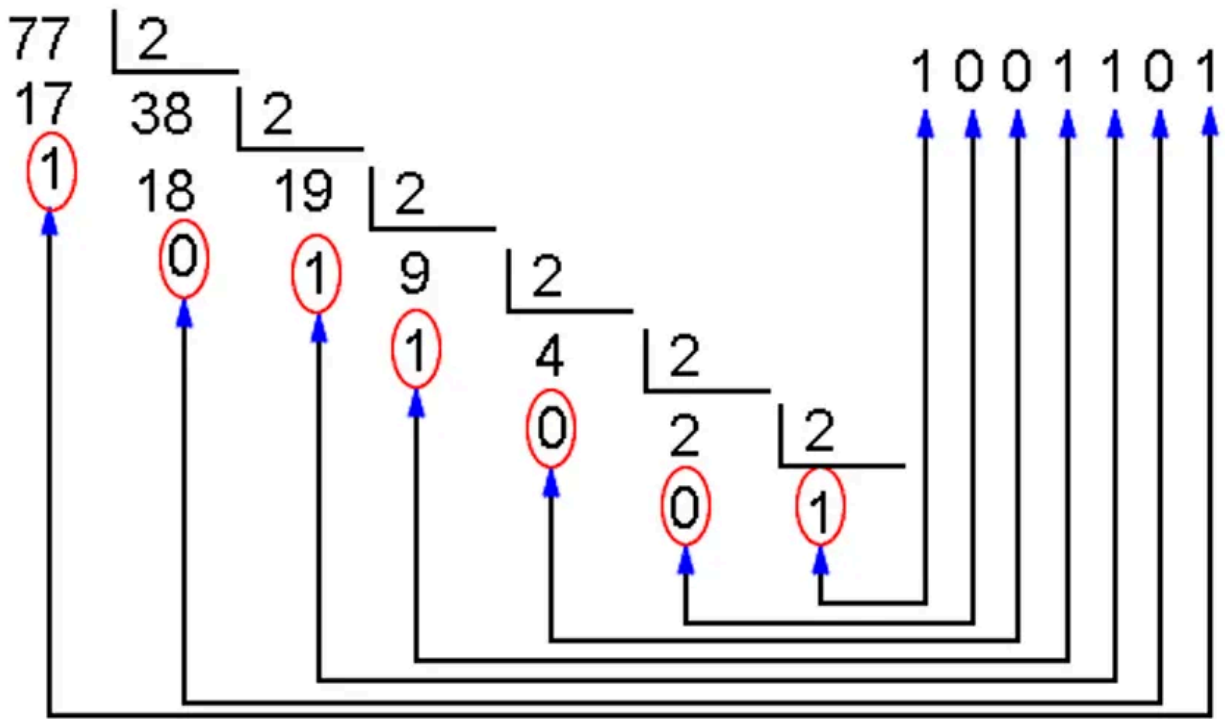
Las ideas básicas sobre cómo establecer un contador `for` para ir pasando número a número entre un rango preestablecido, así como otras cuestiones, ya la teníamos bastante claras porque lo vimos en la Unidad N°4 de Estructuras Repetitivas, en Programación I.

Es por eso que lo más difícil era imaginar la operación de pasaje de número decimal a binario dentro de este bucle, ya que en general, hasta ahora, ha sido una operación que hemos practicado múltiples veces en papel, en el marco de la materia de Matemáticas, pero no habíamos plasmado nunca en código para Python.

Por lo tanto, basándonos en algunos de los videos de la playlist de Programación I, en los cuales el anfitrión desagrega operaciones básicas para pasarlas a Python (por ejemplo, [este video](#)), decidimos pensar paso a paso la operación de pasaje de decimal a binario para poder plasmarla en código.

Es allí donde consideramos que la AI podía ser un buen acompañamiento para nuestro bucle `for` y nos ayudó parcialmente a construir nuestra lógica `while` para seguir usando los restos de la división del número decimal con el divisor 2 (debido a la base 2 de la lógica binaria).

El reto no era saber que nuestro bucle “while” debía usar la función “módulo” para sacar el resto de la división y que esta operación repetirse hasta que el número no se pudiera dividir más, sino saber cómo se iba a realizar el almacenamiento en la variable `binario`. Normalmente para definir el binario se “da vuelta” (se arranca del último resto hasta el primero (de derecha a izquierda), para después escribir el número binario de izquierda a derecha) El verdadero reto era: ¿Cómo hacer esto en Python?



En ese sentido, pasamos nuestro código parcial por AI, la cual, en un primer momento nos arrojó una versión del bucle “while” que usaba funciones que no vimos en profundidad hasta el momento, como `append()`, `reverse()`, `join()`. Debido a lo cual, le exigimos que limite ese mismo código a una lógica sin esas funciones, buscando la concordancia de nuestro código en Python con el recorrido que hemos hecho hasta el momento en Programación I. Para esto le facilitamos a la IA un código de ejemplo de uno de los trabajos prácticos de dicha materia:

Se podría hacer similar a este caso, sin usar `append`, `reverse` o `join`?:

```
py
# Inicializamos una variable como cadena (str) vacía
cadena_invertida = ""

# Iniciamos un bucle for para invertir el número.
# Si por ejemplo tenemos `547`, el rango va a empezar
for i in range(len(numCadena_usuario) - 1, -1, -1):
    cadena_invertida += numCadena_usuario[i]
    # En `cadena_invertida` + `numCadena_usuario[i]`

print(cadena_invertida)
```



👤: ¡Excelente enfoque! Sí, efectivamente puedes reescribir la conversión sin usar `append`, `reverse`, ni `join`, utilizando directamente **concatenación de cadenas**. Aquí tienes una versión que sigue tu estilo:

python

Copy Edit

```
def decimal_a_binario_simple(numero):
    binario = ""
    while numero > 0:
        residuo = numero % 2
        binario = str(residuo) + binario # Agregamos el nuevo dígito al inicio
        numero = numero // 2
    return binario if binario else '0'

# Ejemplo
numero_decimal = int(input("Introduce un número decimal: "))
print(f"El binario es: {decimal_a_binario_simple(numero_decimal)}")
```

🌱 Esta función construye la cadena binaria directamente, anteponiendo cada dígito conforme se calcula, eliminando la necesidad de estructuras adicionales.

El resultado que obtuvimos de la lógica `while`, que modificó ligeramente nuestro código original es el que terminamos usando. Simplemente que en vez de usarla dentro de una función como el output arrojado por la AI, la adaptamos a un bucle `for` para resolver nuestro problema puntual.

Por último, recurrimos a la Inteligencia Artificial, para dividir el guión del video en base al código propuesto, buscando agilizar el proceso, dado que había que grabarlo y luego editarlo (lo que lleva cierto tiempo, aunque el video sea sencillo):

Cómo dividirías una breve explicación del código propuesto para 5 personas, dentro de un video que debe durar entre 3 y 10 minutos

👤: Excelente, amigo mío. Para una presentación colaborativa en video de 3 a 10 minutos, una división efectiva entre cinco personas puede ayudar a mantener el ritmo y claridad. Aquí te propongo una estructura:

### 📺 Distribución de la explicación en 5 partes

#### 👤 **Persona 1 - Introducción y Objetivo (30 seg - 1 min)**

- Presenta el propósito del video.
- Explica brevemente qué es un contador binario.
- Menciona que el programa simula un circuito digital con Python.

#### 👤 **Persona 2 - Entradas y Lógica del Ciclo (1 - 2 min)**

- Comenta que el programa no requiere entradas del usuario.
- Explica el uso del ciclo `for` para contar del 0 al 15.
- Describe brevemente qué significa cada iteración.

#### 👤 **Persona 3 - Conversión a Binario (2 - 3 min)**

- Explica cómo se realiza la conversión de decimal a binario usando `while`.
- Muestra cómo se concatenan los residuos para formar el número binario.
- Menciona la condición especial para el 0.

#### 👤 **Persona 4 - Simulación del Tiempo (30 seg - 1 min)**

- Explica por qué se usa `time.sleep(1)`.
- Relaciona esto con el ritmo de un contador físico en hardware.

#### 👤 **Persona 5 - Salida del Programa y Conclusión (1 - 2 min)**

- Muestra algunos ejemplos de la salida, como "5 en binario es 101".
- Reafirma lo aprendido y cómo puede aplicarse este conocimiento.
- Despede el video o invita a experimentar con el código.