Ejercicios Feyman-Microbit

Ejercicio 1: ¿Para qué sirve el siguiente programa?¿Qué pasa si el valor es 3?

```
import random
valor = random.randint(1, 3)

if valor == 1:
    print("1")
elif valor == 2:
    print("2")
else:
    print("X")
```

El programa está diseñado para **generar un número aleatorio entre 1 y 3** y luego realizar diferentes acciones dependiendo del número generado.

Si `valor` es `1`, el programa imprime `1`.
Si `valor` es `2`, el programa imprime `2`.
Para cualquier otro valor dentro del rango (en este caso, solo puede ser `3`), se ejecuta el bloque `else` e imprime `X`.

Ejercicio 2: ¿Para qué sirve el siguiente programa?¿Qué pasa si el valor es 3?

```
let valor = 0
valor = Math.randomRange(1, 3)

if (valor == 1) {
            basic.showLeds(`
                        . . # . .
                        . . # . .
                        . . # . .
                        . . # . .
```

```
                        . . # . .
                        `)
}
else if (valor == 2) {
                    basic.showLeds(`
                     # . . . #
                     . # . # .
                     . . # . .
                     . # . # .
                     # . . . #
                     `)
}
else {
            basic.showLeds(`
                        # # # # #
                        . . . . #
                        # # # # #
                        # . . . .
                        # # # # #
                        `)
                }
})
```

Genera un número aleatorio entre 1 y 3 y, dependiendo del valor generado, muestra un patrón LED diferente en la pantalla del Micro:bit.

`Math.randomRange(1, 3)` genera un número entero aleatorio entre 1 y 3 (incluyendo ambos límites).
El valor generado se guarda en la variable `valor`

- Si el valor es 1, muestra un patrón LED vertical (columna en el centro).
- Si el valor es 2, muestra un corazón en la pantalla LED.
- Si el valor es 3, muestra un patrón con rectángulos y líneas en la pantalla LED.

El bloque `else`:

- Cubre el caso en el que el valor sea 3 (ya que el rango es limitado entre 1 y 3).

Ejercicio 1: Contador de pasos básico

Un ejercicio inicial perfecto para **Micro:bit** y programación es crear un **contador de pasos** usando el acelerómetro integrado. Este ejercicio utiliza el entorno de bloques de MakeCode, es visual y fácil.

1. Introducir el entorno de programación con bloques (MakeCode).
2. Familiarizarse con conceptos básicos como:
   - Variables.
   - Condicionales.
   - Sensores integrados (acelerómetro).

Instrucciones paso a paso:

1. **Crear el proyecto en MakeCode:**

- Abre MakeCode para Micro:bit.
- Haz clic en "Nuevo proyecto" y nómbralo "Contador de pasos".

2. **Definir la variable:**
   - Ve a la categoría "Variables" y haz clic en "Crear una variable".
   - Llámala “pasos”
   - Arrastra el bloque establecer pasos a 0 dentro del bloque al iniciar.

3. **Detectar movimiento:**
   - Ve a la categoría "Entrada" y selecciona el bloque al agitar.
   - Dentro de este bloque, arrastra el bloque cambiar pasos por 1 desde la categoría "Variables".

4. **Mostrar los pasos en la pantalla:**
   - Desde la categoría "Básico", arrastra el bloque mostrar número y ponlo en el evento al presionar botón A.
   - Configúralo para mostrar la variable pasos.

5. **Opcional: Resetear el contador (botón B):**
   - Agrega otro bloque al presionar botón B desde la categoría "**Entrada**".
   - Dentro, arrastra el bloque establecer pasos a 0 desde "**Variables**".

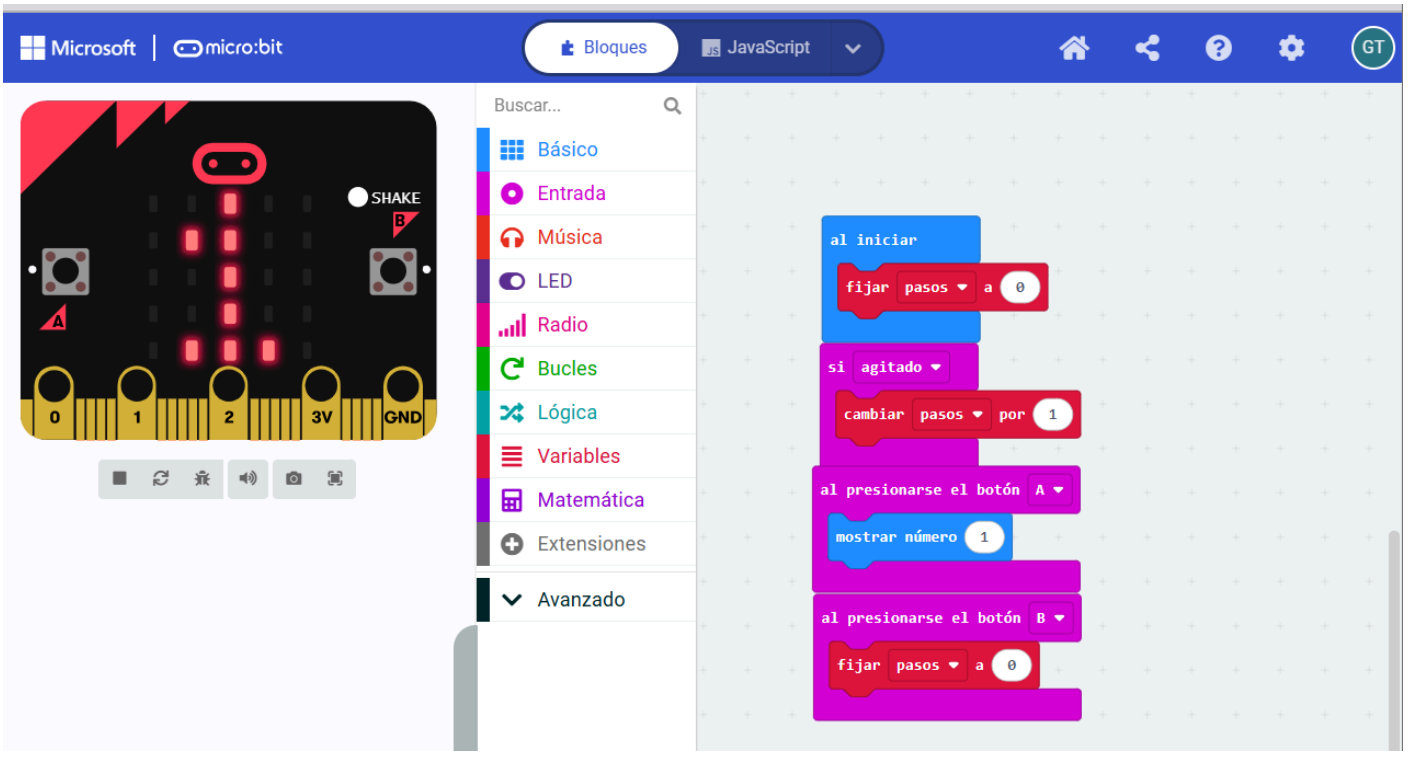
Resumen Código final (en bloques):

- **Al iniciar:** Establecer pasos en 0.
- **Al agitar:** Incrementar pasos en 1.
- **Al presionar botón A:** Mostrar el valor de pasos en la pantalla LED.
- **Al presionar botón B:** Restablecer pasos a 0.

Explicación del ejercicio:

- El Micro:bit tiene un **acelerómetro** que detecta el movimiento. Este sensor se utiliza para aumentar el contador cada vez que el dispositivo se agita.

- Al presionar los botones A o B, los alumnos interactúan directamente con el dispositivo para consultar o reiniciar el contador.



Ejercicio 1b: Contador de pasos básico con Python

Un ejercicio similar al contador de pasos, pero en **Python** (usando Micro:bit y su editor), puede ser ideal para introducir conceptos básicos de programación como variables, condicionales y entrada/salida.

Instrucciones paso a paso:

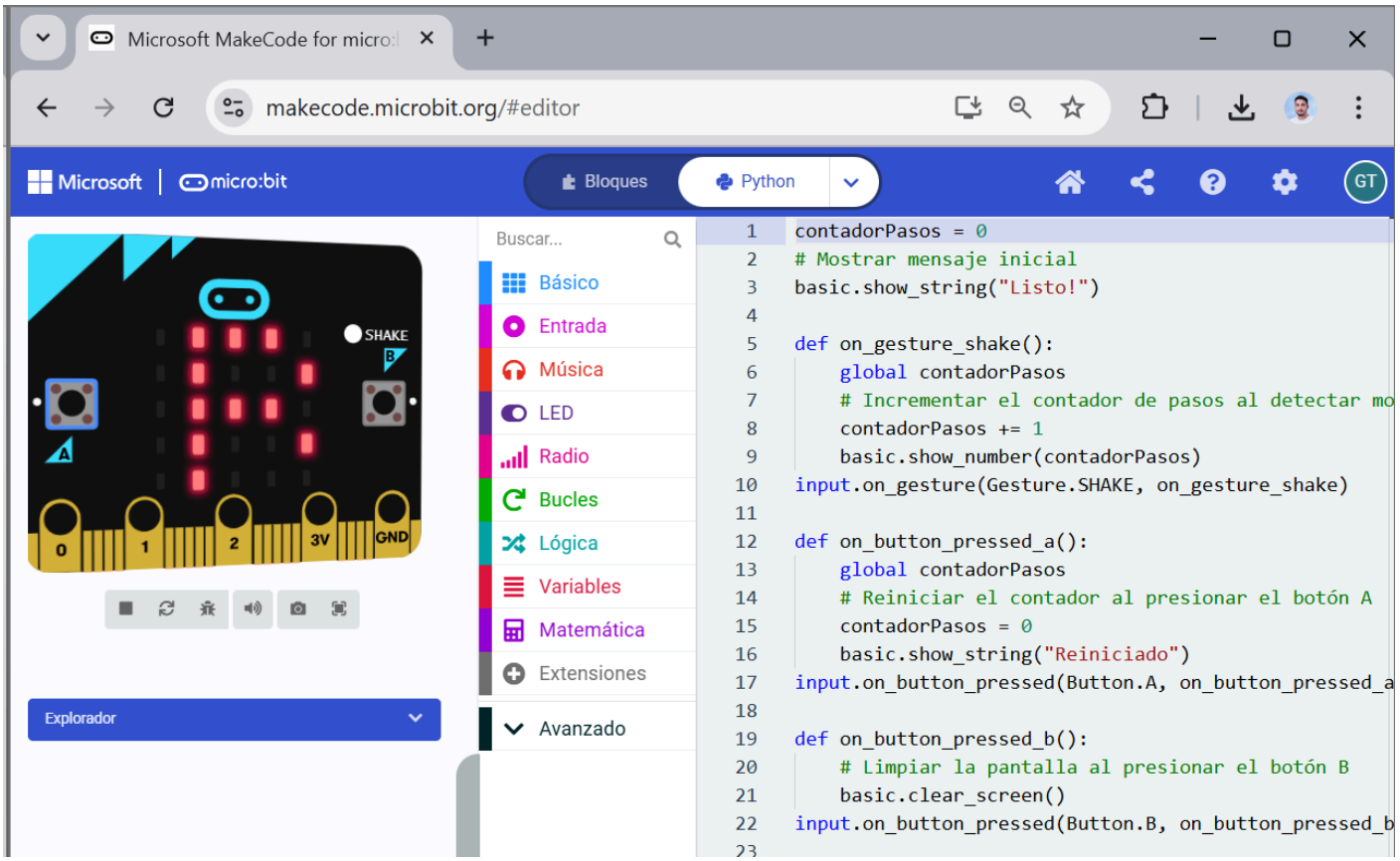
1. **Preparar el entorno:**
   - Abre el editor Python para Micro:bit (MicroPython en MakeCode).
   - Crea un nuevo proyecto y guárdalo como contador_pasos.py.

código básico:

```
from microbit import *

# Inicializar el contador de pasos
pasos = 0

while True:
    # Detectar si el Micro:bit está siendo agitado
    if accelerometer.was_gesture("shake"):
        pasos += 1  # Incrementar el contador

    # Mostrar pasos al presionar botón A
    if button_a.is_pressed():
        display.scroll(str(pasos))

    # Reiniciar el contador al presionar botón B
    if button_b.is_pressed():
        pasos = 0
        display.show("R")  # Mostrar una "R" como confirmación
```

Explicación del código:

1. **Importar la librería:**
   - from microbit import * incluye todas las funciones necesarias para interactuar con el Micro:bit.
2. **Inicializar la variable:**
   - pasos = 0 define y establece el contador en 0.
3. **Bucle principal:**
   - while True: asegura que el programa se ejecute continuamente.
4. **Detectar movimiento:**
   - accelerometer.was_gesture("shake") detecta si el Micro:bit ha sido agitado.
   - Si se detecta el movimiento, se incrementa el contador.
5. **Mostrar los pasos:**
   - button_a.is_pressed() muestra el número de pasos en la pantalla LED al presionar el botón A.
6. **Reiniciar el contador:**
   - button_b.is_pressed() reinicia el contador y muestra una confirmación (R) en la pantalla.


Microsoft MakeCode for micro:
makecode.microbit.org/#editor
Microsoft
micro:bit
Bloques
Python
GT
SHAKE
A
B
0
1
2
3V
GND
Explorador
Buscar...
Básico
Entrada
Música
LED
Radio
Bucles
Lógica
Variables
Matemática
Extensiones
Avanzado
1 contadorPasos = 0
2 # Mostrar mensaje inicial
3 basic.show_string("Listo!")
4
5 def on_gesture_shake():
6 global contadorPasos
7 # Incrementar el contador de pasos al detectar mo
8 contadorPasos += 1
9 basic.show_number(contadorPasos)
10 input.on_gesture(Gesture.SHAKE, on_gesture_shake)
11
12 def on_button_pressed_a():
13 global contadorPasos
14 # Reiniciar el contador al presionar el botón A
15 contadorPasos = 0
16 basic.show_string("Reiniciado")
17 input.on_button_pressed(Button.A, on_button_pressed_a
18
19 def on_button_pressed_b():
20 # Limpiar la pantalla al presionar el botón B
21 basic.clear_screen()
22 input.on_button_pressed(Button.B, on_button_pressed_b
23

Ejercicio 2: : Generador de resultados para una quiniela

Objetivo:

1. Familiarizar a los alumnos con:
    - Uso de botones (A y B).
    - Mostrar caracteres (1, X, 2) en la pantalla LED.
    - Generación de resultados aleatorios.

2. Introducir el concepto de **aleatoriedad** en programación

Enunciado:
Programa el Micro:bit para que sea un **generador de quinielas**. Al presionar el botón **A**, mostrará un resultado aleatorio entre 1, X y 2. Si presionas el botón **B**, limpiará la pantalla.

Instrucciones:
1. Usa la función de números aleatorios para seleccionar entre tres opciones: 1, X, 2.
2. Muestra el resultado en la pantalla LED al presionar el botón A.
3. Al presionar el botón B, limpia la pantalla para preparar el próximo resultado.

Ejemplo de código en bloques (MakeCode):

1. Ve a la categoría **Entrada** y selecciona al presionar botón A.

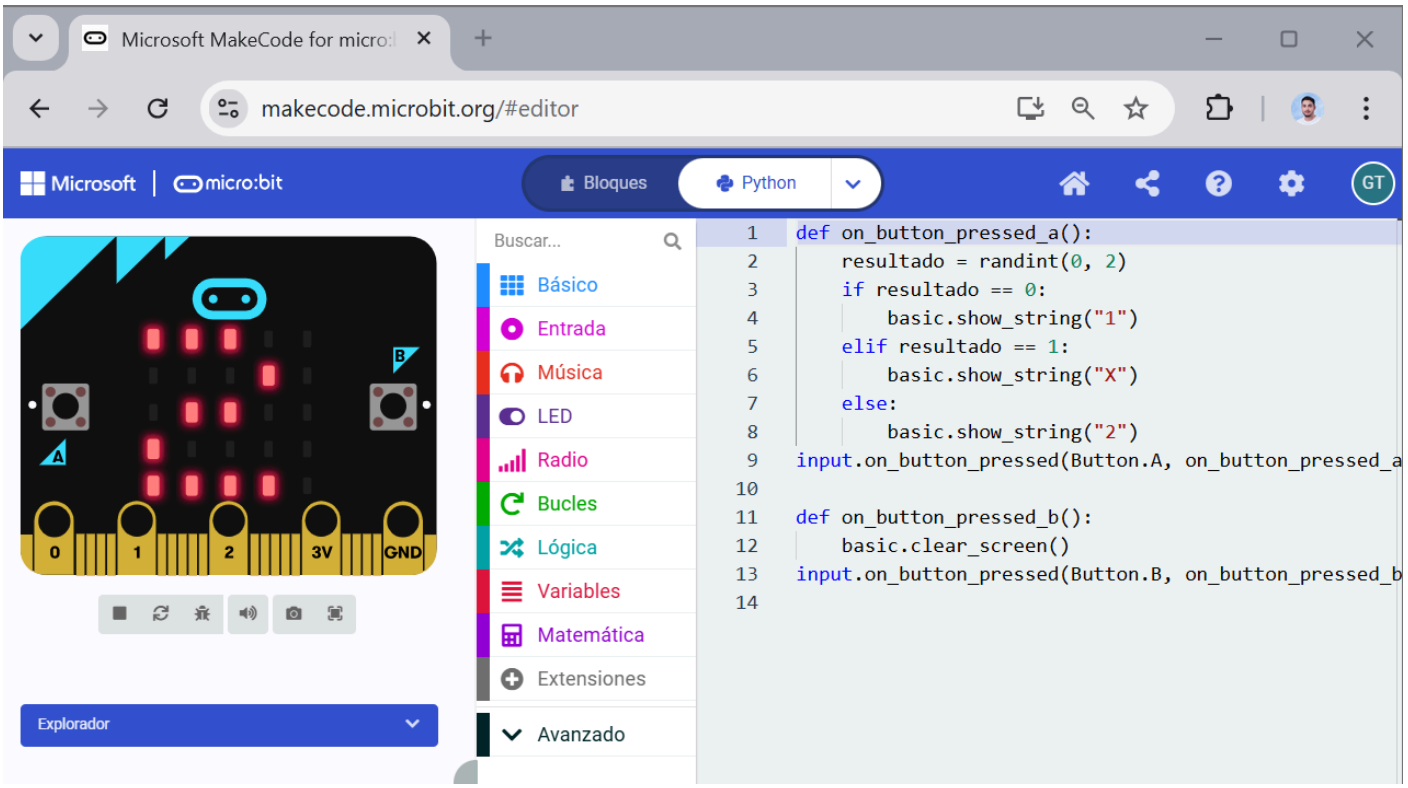
2. Usa la categoría **Matemáticas** para agregar elegir aleatoriamente.
    - Configúralo para elegir entre los valores 1, "X" y 2.

3. Usa la categoría **Básico** para mostrar el número o carácter en pantalla.

4. Agrega un bloque al presionar botón B y dentro pon borrar pantalla.

El programa debería verse así:
- **Cuando se presiona A:** Mostrar un valor aleatorio 1, X o 2.
- **Cuando se presiona B:** Borrar la pantalla.

```
from microbit import *
import random

while True:
    # Mostrar un resultado aleatorio al presionar A
    if button_a.is_pressed():
        resultado = random.choice(["1", "X", "2"])
        display.show(resultado)

    # Limpiar la pantalla al presionar B
    if button_b.is_pressed():
        display.clear()
```


Microsoft MakeCode for micro:
makecode.microbit.org/#editor
Microsoft
micro:bit
Bloques
Python
GT
0
1
2
3V
GND
A
B
Explorador
Buscar...
Básico
Entrada
Música
LED
Radio
Bucles
Lógica
Variables
Matemática
Extensiones
Avanzado
1 def on_button_pressed_a():
2 resultado = randint(0, 2)
3 if resultado == 0:
4 basic.show_string("1")
5 elif resultado == 1:
6 basic.show_string("X")
7 else:
8 basic.show_string("2")
9 input.on_button_pressed(Button.A, on_button_pressed_a
10
11 def on_button_pressed_b():
12 basic.clear_screen()
13 input.on_button_pressed(Button.B, on_button_pressed_b
14

Ejercicio 3: Máquina del amor - Medidor de compatibilidad amorosa

Objetivo:

1. Usar el sensor de temperatura del Micro:bit.
2. Introducir conceptos básicos:
   - Lectura de sensores.
   - Generación de puntuaciones a partir de datos.
   - Mostrar resultados en pantalla.

Enunciado:
Construye una máquina del amor con tu Micro:bit. La máquina medirá tu compatibilidad amorosa basándose en la temperatura de tu mano o en la estabilidad del dispositivo. Al presionar el botón **A**, mostrará un número entre 0 y 100 como tu "puntuación de amor". Usa la función de temperatura como entrada para hacer el resultado más interesante.

Instrucciones para los alumnos:

1. Usa el sensor de temperatura del Micro:bit.
2. Calcula una puntuación "amorosa" basándote en la temperatura medida.
3. Muestra el resultado al presionar el botón **A**.
4. Opcionalmente, agrega un mensaje romántico si la puntuación es alta (por ejemplo, mayor a 75).

Ejemplo de código en bloques (MakeCode):

1. Usa la categoría **Entrada** para leer la temperatura (temperatura ambiente).
2. Genera una puntuación "amorosa" basada en la temperatura:
   - Multiplica la temperatura por un valor, por ejemplo, 4, y ajusta con una fórmula para que el rango esté entre 0 y 100.
3. Al presionar el botón A:
   - Muestra la puntuación en la pantalla.
   - Si la puntuación es alta, muestra un icono de corazón.
4. Opcionalmente, al presionar el botón B, limpia la pantalla.

Código en Python:

```python
from microbit import *

while True:
    # Al presionar el botón A, calcular puntuación
    if button_a.is_pressed():
        temperatura = temperature()  # Leer temperatura ambiente
        puntuacion = (temperatura * 4) % 101  # Generar puntuación entre 0 y 100

        # Mostrar puntuación en pantalla
        display.scroll(str(puntuacion))

        # Mostrar un corazón si la puntuación es alta
        if puntuacion > 75:
            display.show(Image.HEART)
        else:
            display.clear()

    # Limpiar pantalla con el botón B
    if button_b.is_pressed():
        display.clear()
```

Explicación del código:

1. **Lectura de la temperatura:**

temperature() lee la temperatura en grados Celsius del sensor del Micro:bit.

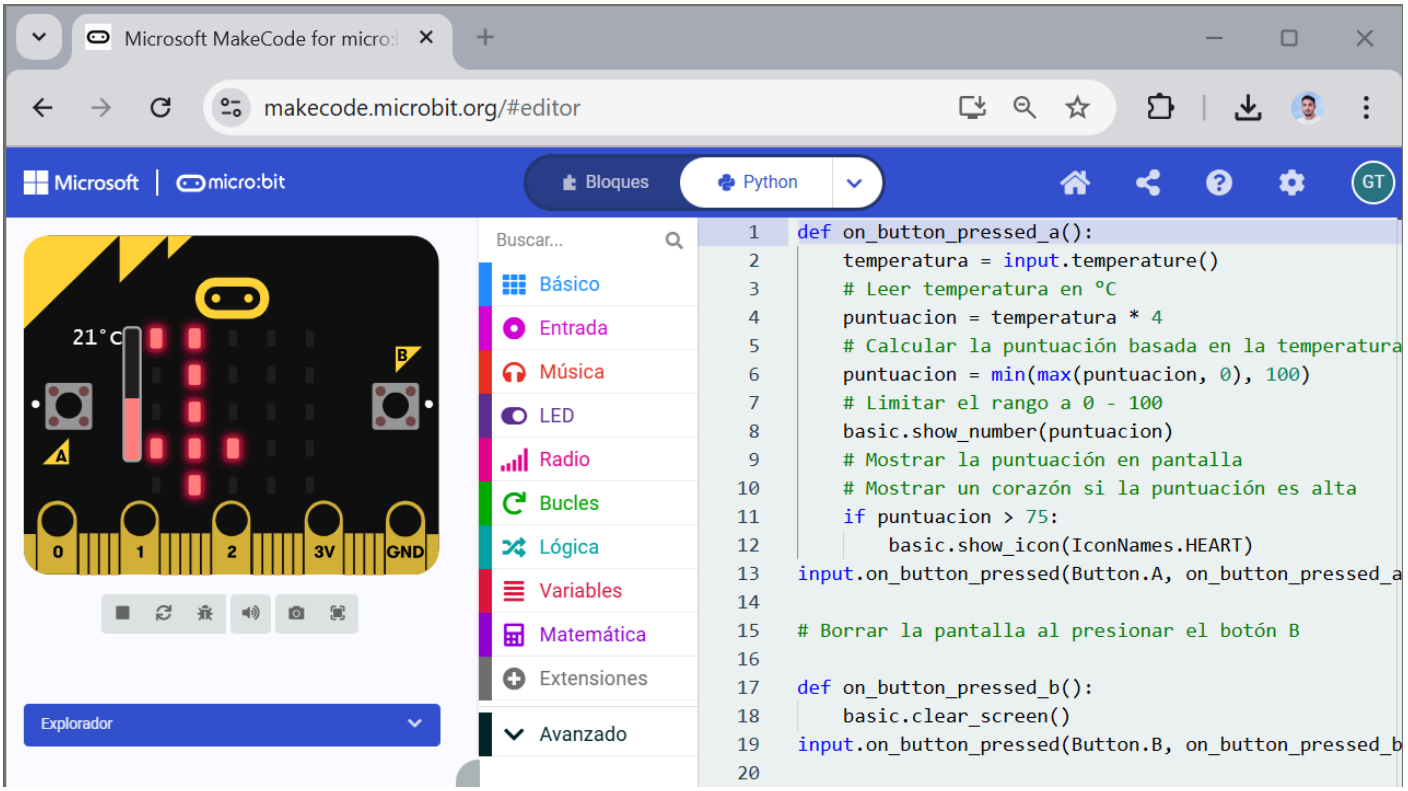
2. **Cálculo de puntuación:**
   - (temperatura * 4) % 101 convierte el rango de temperatura en una puntuación "amorosa" entre 0 y 100

3. **Mostrar resultados:**
   - Usa display.scroll para mostrar la puntuación.
   - Si la puntuación supera 75, muestra un corazón (Image.HEART).

4. **Limpieza opcional:**
   - Limpia la pantalla con el botón **B** para iniciar una nueva medición.

Extensiones para hacerlo más interesante:

1. **Añadir mensajes románticos**
   - Mostrar "HOT" si la puntuación es mayor a 90 o "MEH" si está por debajo de 30.
2. **Sensores adicionales:**
   - Usar el acelerómetro para añadir "estabilidad" como otro factor en la puntuación.
3. **Efectos visuales:**
   - Añadir una animación mientras se calcula la puntuación.



https://chatgpt.com/share/674e476a-85b4-8009-a402-ccf808f7fe1a