

# Micropython

Using Visual Studio Code



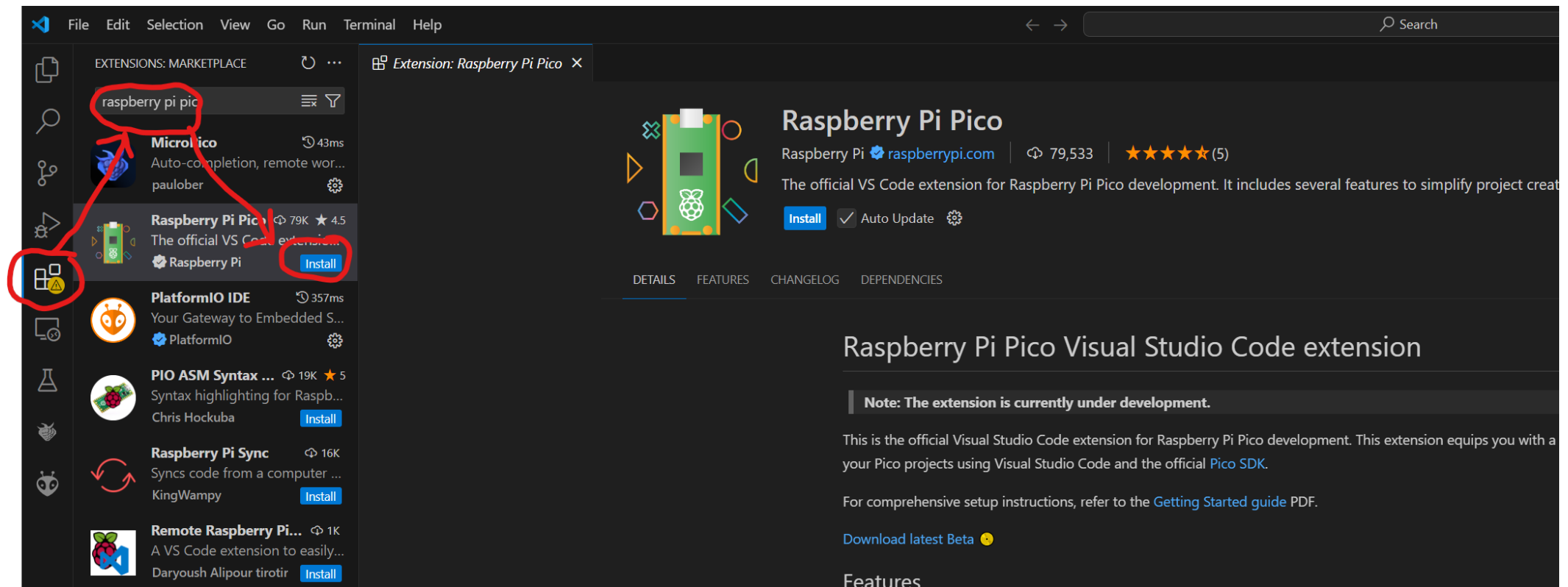
Leo Kuipers – [l.a.Kuipers@saxion.nl](mailto:l.a.Kuipers@saxion.nl)

# Why Visual Studio Code?

- It just works better.
- Thonny IDE is a meh editor.
- Visual Studio Code is a proper IDE.
- VS Code “Extensions” add a lot of functionality. E.g.:
  - Python Intellisense → = Autocomplete!
  - Git directly from within the editor
  - Github Copilot → Use Copilot AI to help you to program code
- Install visual studio code from: <https://code.visualstudio.com>

# RP2040 Micropython support

- Install the Raspberry Pi Pico extension



The screenshot shows the Visual Studio Code interface with the Extensions Marketplace open. The search bar contains "raspberry pi pico". The "Raspberry Pi Pico" extension is highlighted with a red box, and its "Install" button is also circled in red. A red arrow points from the search bar to the extension. The extension details page is visible on the right, showing the extension's name, publisher (Raspberry Pi), version (79,533), and a 5-star rating. A note indicates that the extension is currently under development. The extension description states it is the official VS Code extension for Raspberry Pi Pico development, including features to simplify project creation. A link to the "Getting Started guide PDF" is provided, along with a "Download latest Beta" button.

EXTENSIONS: MARKETPLACE

raspberry pi pico

**Micrónico** 43ms  
Auto-completion, remote wor...  
paulober

**Raspberry Pi Pico** 79K ★ 4.5  
The official VS Code extension...  
Raspberry Pi **Install**

**PlatformIO IDE** 357ms  
Your Gateway to Embedded S...  
PlatformIO

**PIO ASM Syntax ...** 19K ★ 5  
Syntax highlighting for Raspb...  
Chris Hockuba **Install**

**Raspberry Pi Sync** 16K  
Syncs code from a computer ...  
KingWampy **Install**

**Remote Raspberry Pi...** 1K  
A VS Code extension to easily...  
Daryoush Alipour tiroir **Install**

Extension: Raspberry Pi Pico

**Raspberry Pi Pico**  
Raspberry Pi raspberrypi.com | 79,533 | ★★★★★ (5)  
The official VS Code extension for Raspberry Pi Pico development. It includes several features to simplify project creat

**Install**  Auto Update

DETAILS FEATURES CHANGELOG DEPENDENCIES

**Raspberry Pi Pico Visual Studio Code extension**

**Note: The extension is currently under development.**

This is the official Visual Studio Code extension for Raspberry Pi Pico development. This extension equips you with a your Pico projects using Visual Studio Code and the official [Pico SDK](#).

For comprehensive setup instructions, refer to the [Getting Started guide PDF](#).

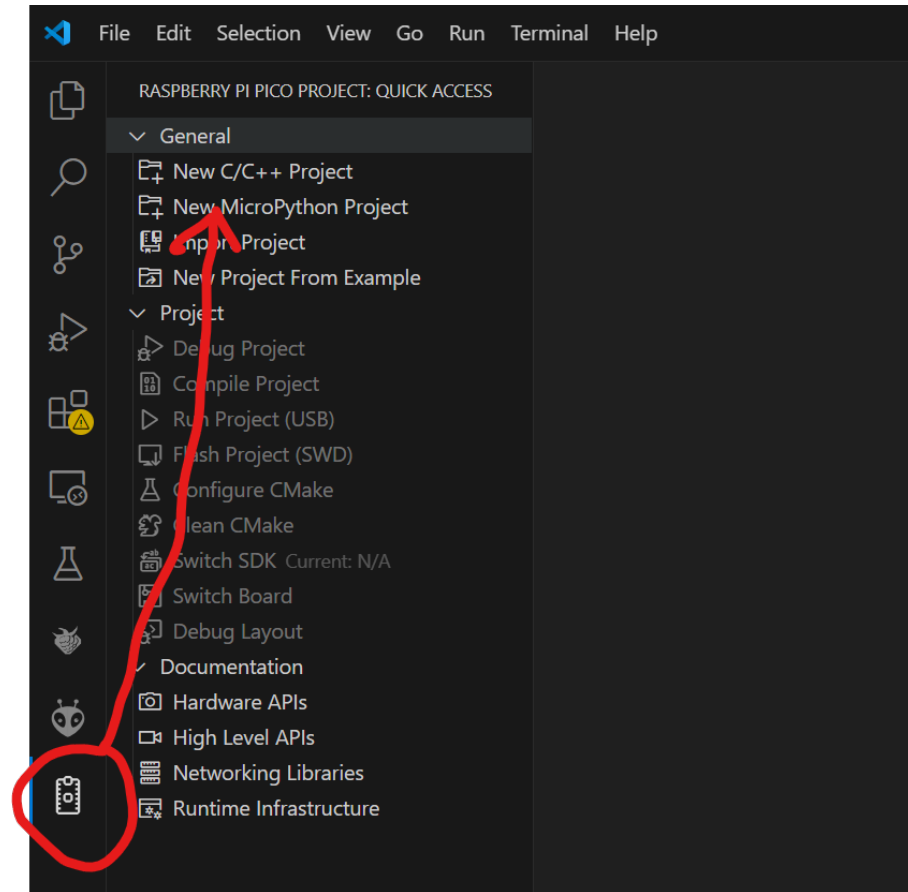
[Download latest Beta](#)

Features

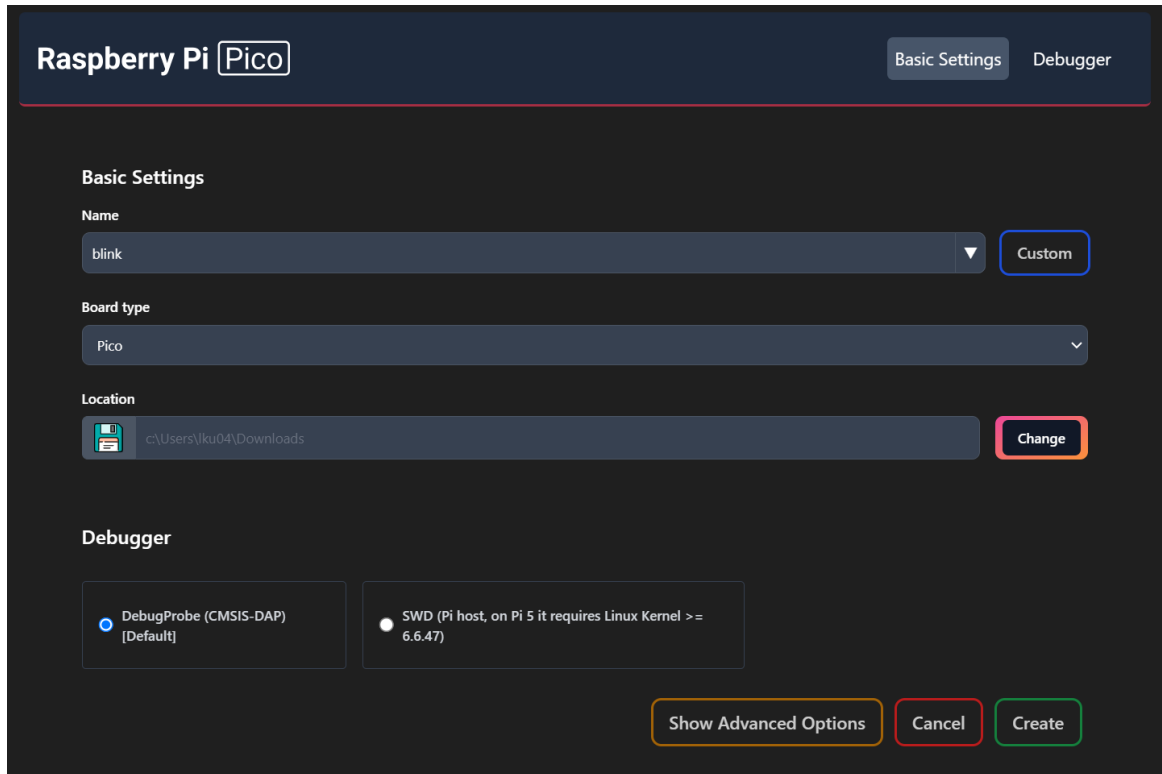
# Raspberry Pi Pico extension

- The Raspberry Pi Pico extension allows you to code in 2 different languages
  1. C/C++ → This uses the Pico SDK as framework. Note that this is NOT Arduino framework! Arduino libraries will not work!
  2. Micropython → this is implemented using the MicroPico extension  
<https://github.com/paulober/MicroPico>

# Howto open the Pico Extension?



# C/C++ project example: Blink



```
File Edit Selection View Go Run Terminal Help
EXPLORER
  BLINK
    .vscode
    build
    .gitignore
    C blink.c
    CMakeLists.txt
    pico_sdk_import.cmake
  C blink.c
    1 /**
    2  * Copyright (c) 2020 Raspberry Pi (Trading) Ltd.
    3  *
    4  * SPDX-License-Identifier: BSD-3-Clause
    5  */
    6
    7 #include "pico/stdlib.h"
    8
    9 // Pico W devices use a GPIO on the WIFI chip for the LED,
   10 // so when building for Pico W, CYW43_WL_GPIO_LED_PIN will be defined
   11 #ifdef CYW43_WL_GPIO_LED_PIN
   12 #include "pico/cyw43_arch.h"
   13 #endif
   14
   15 #ifndef LED_DELAY_MS
   16 #define LED_DELAY_MS 250
   17 #endif
   18
   19 // Perform initialisation
   20 int pico_led_init(void) {
   21 #if defined(PICO_DEFAULT_LED_PIN)
   22     // A device like Pico that uses a GPIO for the LED will define PICO_DEFAULT_LED_PIN
   23     // so we can use normal GPIO functionality to turn the led on and off
   24     gpio_init(PICO_DEFAULT_LED_PIN);
   25     gpio_set_dir(PICO_DEFAULT_LED_PIN, GPIO_OUT);
   26     return PICO_OK;
   27 #elif defined(CYW43_WL_GPIO_LED_PIN)
   28     // For Pico W devices we need to initialise the driver etc
   29     return cyw43_arch_init();
   30 #endif
   31 }
   32
   33 // Turn the led on or off
   34 void pico_set_led(bool led_on) {
   35 #if defined(PICO_DEFAULT_LED_PIN)
   36     // Just set the GPIO on or off
   37     *gpio_put(PICO_DEFAULT_LED_PIN, led_on);
   38 #elif defined(CYW43_WL_GPIO_LED_PIN)
   39     // Ask the wifi "driver" to set the GPIO on or off
   40     cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, led_on);
   41 #endif
   42 }
   43
   44 int main() {
   45     int rc = pico_led_init();
   46     hard_assert(rc == PICO_OK);
   47     while (true) {
   48         pico_set_led(true);
   49         sleep_ms(LED_DELAY_MS);
   50         pico_set_led(false);
   51         sleep_ms(LED_DELAY_MS);
   52     }
   53 }
   54
```

# C/C++ project example: Blink

```
#include "pico/stdlib.h"

// Pico W devices use a GPIO on the WIFI chip for the
// LED,
// so when building for Pico W, CYW43_WL_GPIO_LED_PIN
// will be defined
#ifdef CYW43_WL_GPIO_LED_PIN
#include "pico/cyw43_arch.h"
#endif

#ifndef LED_DELAY_MS
#define LED_DELAY_MS 250
#endif

// Perform initialisation
int pico_led_init(void) {
#ifdef PICO_DEFAULT_LED_PIN
    // A device like Pico that uses a GPIO for the
    // LED will define PICO_DEFAULT_LED_PIN
    // so we can use normal GPIO functionality to
    // turn the led on and off
    gpio_init(PICO_DEFAULT_LED_PIN);
    gpio_set_dir(PICO_DEFAULT_LED_PIN, GPIO_OUT);
    return PICO_OK;
#elif defined(CYW43_WL_GPIO_LED_PIN)
    // For Pico W devices we need to initialise the
    // driver etc
    return cyw43_arch_init();
#endif
}
```

```
// Turn the led on or off
void pico_set_led(bool led_on) {
#ifdef PICO_DEFAULT_LED_PIN
    // Just set the GPIO on or off
    gpio_put(PICO_DEFAULT_LED_PIN, led_on);
#elif defined(CYW43_WL_GPIO_LED_PIN)
    // Ask the wifi "driver" to set the GPIO on
    // or off
    cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN,
led_on);
#endif
}

int main() {
    int rc = pico_led_init();
    hard_assert(rc == PICO_OK);
    while (true) {
        pico_set_led(true);
        sleep_ms(LED_DELAY_MS);
        pico_set_led(false);
        sleep_ms(LED_DELAY_MS);
    }
}
```

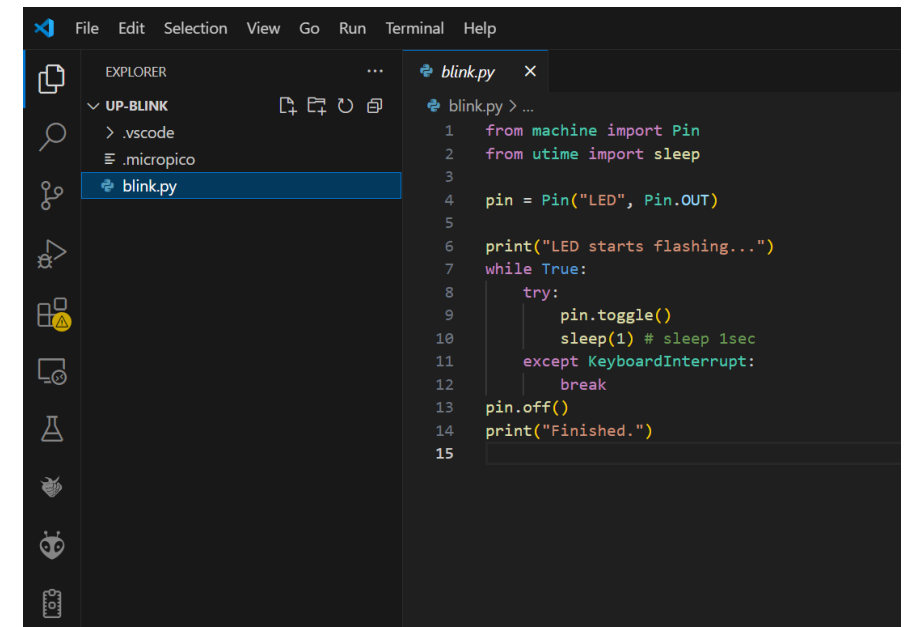
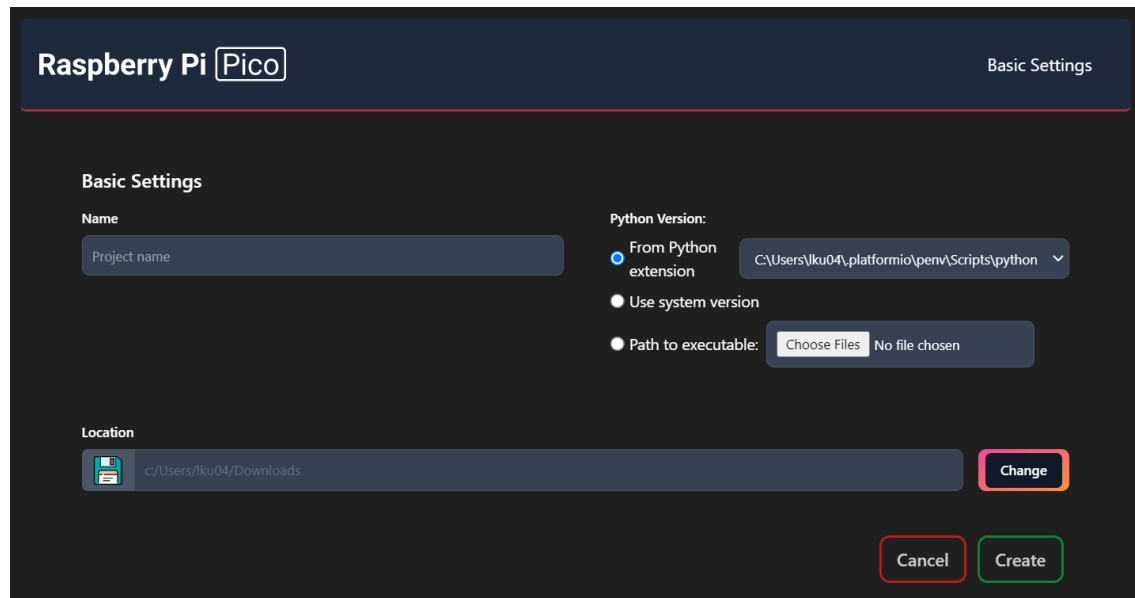
# And then you try to compile the example...

```
.pico-sdk/toolchain/13_3_Rel1/bin/./lib/gcc/arm-none-eabi/13.3.1/././././arm-none-eabi/bin/ld.exe:  
cannot find -lpico_cyw43_arch_none: No such file or directory  
collect2.exe: error: ld returned 1 exit status  
ninja: build stopped: subcommand failed.
```

- Nevermind. Let's continue to micropython 😊



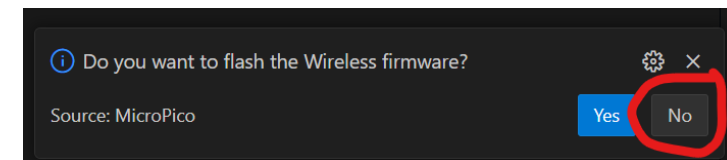
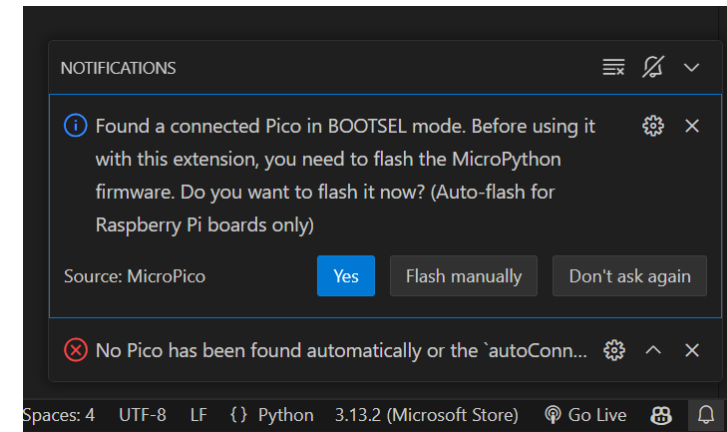
# Micropython example



# Micropython on Xiao RP2040

- Connect the Xiao RP2040
- Put it in bootsel mode
  - hold button B, press & release button R, release button B
- Notice the notification in VSCode
  - Click yes to flash micropython firmware
  - Do NOT flash Wireless firmware
    - Xiao RP2040 is not a Pico-W
    - Press no will flash non-wireless firmware
- You should now see the REPL in the terminal

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR MEMORY XRTOS COMMENTS
MicroPython v1.24.1 on 2024-11-29; Raspberry Pi Pico with RP2040
Type "help()" for more information or .help for custom vREPL commands.
>>> |
```



# Micropython on Xiao RP2040

```
from machine import Pin
from utime import sleep

pinG = Pin(16, Pin.OUT)
pinR = Pin(17, Pin.OUT)
pinB = Pin(25, Pin.OUT)

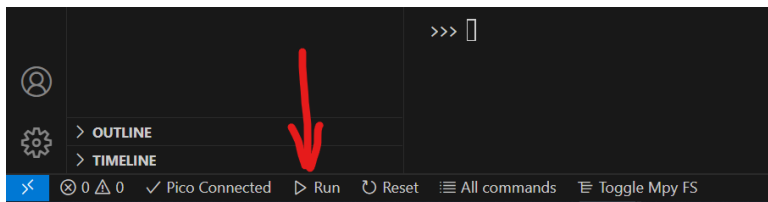
print("LED starts flashing...")
pinR.on()
pinG.on()
pinB.on()
while True:
    try:
        pinG.toggle()
        sleep(1) # sleep 1sec
    except KeyboardInterrupt:
        break
pinG.on()
print("Finished.")
```

```
# XIAO RP2040 pins:

# pin 4 (D4) = I2C SDA
# pin 5 (D5) = I2C SCL
# pin 6 (D6) = UART TX
# pin 7 (D7) = UART RX / SPI Chip select
# pin 8 (D8) = SPI SCK
# pin 9 (D9) = SPI MISO
# pin 10 (D10) = SPI MOSI
# pin 11 = enable pin of RGB LED
#           (high = enabled)
# pin 12 = WS2812 data pin
# pin 16 = onboard green LED (use PIN_LED_G)
# pin 17 = onboard red LED (use PIN_LED_R)
# pin 25 = onboard blue LED (use PIN_LED_B)

# for ONBOARD LEDs: HIGH = OFF, LOW = ON
# so pin.on() = off 😊
```

- Click Run to run



# Further reading

- <https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf>