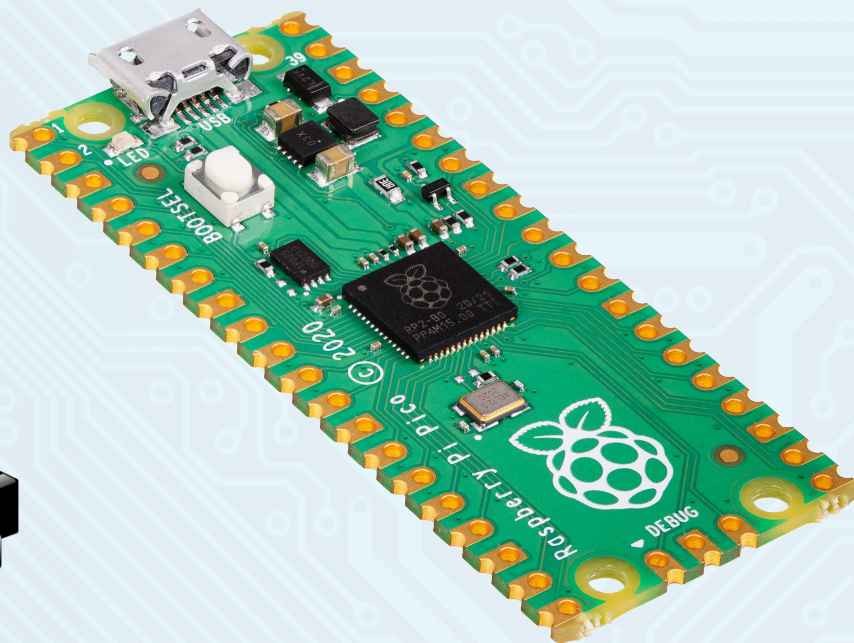
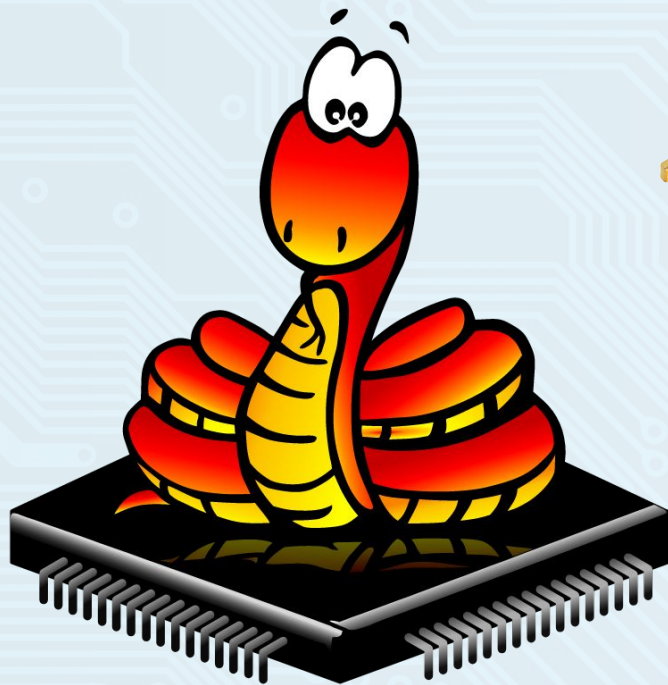


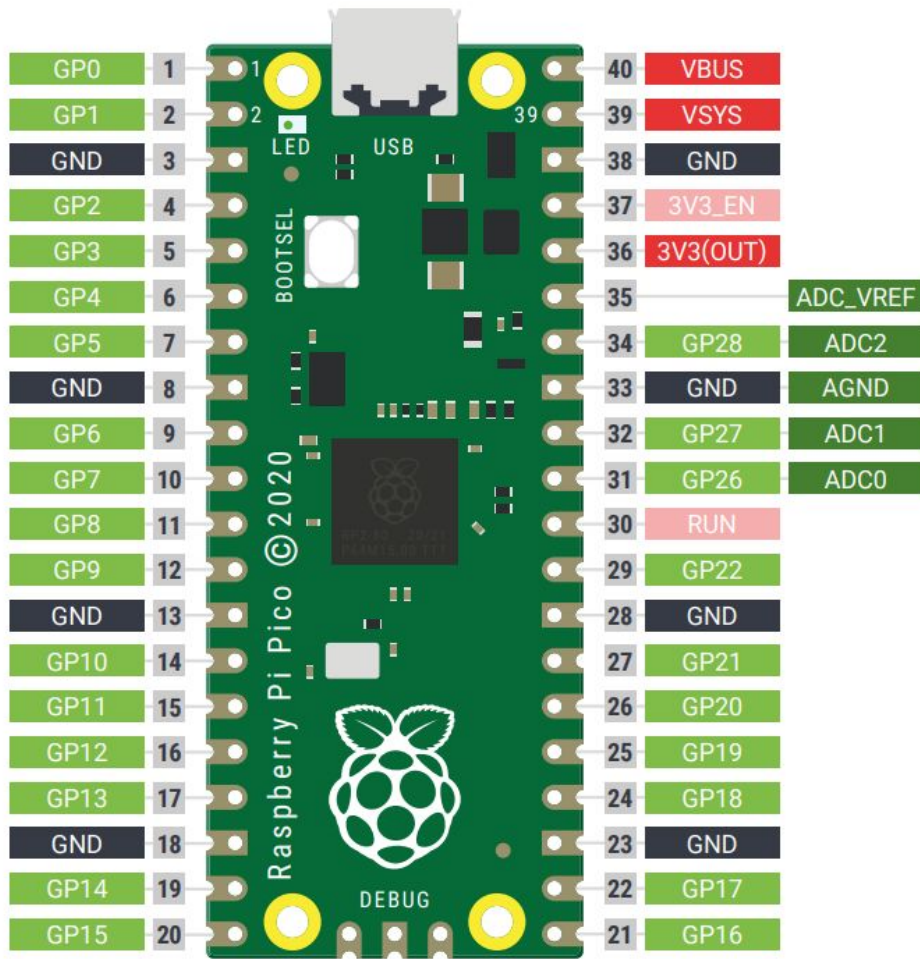
# *Intro to MicroPython Workshop*

*Using the RPi Pico to interface with circuits*



# Raspberry Pi Pico

- The Raspberry Pi Pico is a microcontroller development board
  - NOT the same as other Raspberry Pi boards!!
- Pico is designed for physical computing projects, like controlling LEDs or buttons or even other microcontrollers
- Pico does not come with pin headers attached (these have already been soldered on for you though)
- The bottom of the Pico has labels for each of the pins
  - These pins WILL NOT be visible when the Pico is in a breadboard!



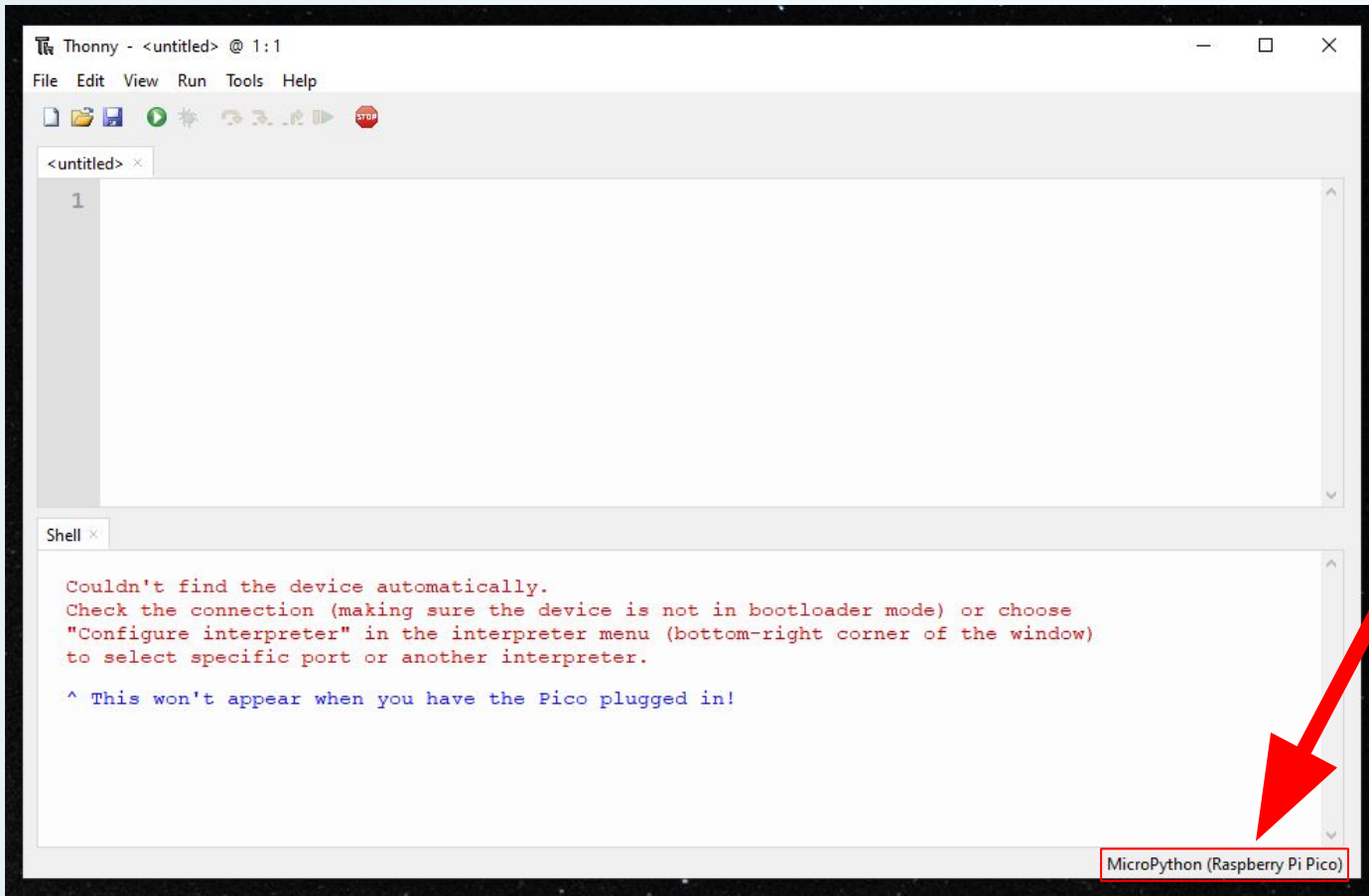
3V3	3.3 volts power	A source of 3.3 V power, the same voltage your Pico runs at internally, generated from the VSYS input. This power supply can be switched on and off using the 3V3_EN pin above it, which also switches your Pico off.
VSYS	~2-5 volts power	A pin directly connected to your Pico's internal power supply, which cannot be switched off without also switching Pico off.
VBUS	5 volts power	A source of 5 V power taken from your Pico's micro USB port, and used to power hardware which needs more than 3.3 V.
GND	0 volts ground	A ground connection, used to complete a circuit connected to a power source. Several of these pins are dotted around your Pico to make wiring easier.
GPxx	General-purpose input/output pin number 'xx'	The GPIO pins available for your program, labelled 'GP0' through to 'GP28'.
GPxx_ADCx	General-purpose input/output pin number 'xx', with analogue input number 'x'	A GPIO pin which ends in 'ADC' and a number can be used as an analogue input as well as a digital input or output – but not both at the same time.
ADC_VREF	Analogue-to-digital converter (ADC) voltage reference	A special input pin which sets a <i>reference voltage</i> for any analogue inputs.
AGND	Analogue-to-digital converter (ADC) 0 volts ground	A special ground connection for use with the ADC_VREF pin.
RUN	Enables or disables your Pico	The RUN header is used to start and stop your Pico from another microcontroller.

# Getting MicroPython on the Pico

- MicroPython is not installed on the boards by default, so we'll be doing the following:
  - While holding the BOOTSEL button, plug the USB cable into the Pico and connect it to your computer
  - After 3 seconds, let it go
  - Your Pico should appear on your computer as a removable drive
  - Open the drive and look for the file "INDEX.HTM"
    - Double click this file and then click on the MicroPython tab and click on the "Download UF2 File" button to download MicroPython to your Pico
  - Navigate to your downloads folder to find the file we just downloaded - should be called "micropython(date/time).uf2"

## Setting up Thonny IDE


- Thonny IDE is an ‘Integrated development environment’ used for Python and MicroPython
- To get set up, go to [thonny.org](https://thonny.org) and download Thonny
- After connecting your Pico to your computer, launch Thonny
- Click on Python Version box in the bottom right of the Thonny window (see next slide)
- Select MicroPython (Raspberry Pi Pico) and click on it
  - If it doesn’t show up, make sure your Pico is connected to your computer properly
- The bottom of Thonny should now say “MicroPython” and you are ready to code!



This is where you  
change 'Python' to  
'MicroPython'



# Coding in MicroPython

- Coding with Thonny in MicroPython follows the same basic syntax rules as Python
  - Code normally runs top to bottom, line by line
  - Indentation is used to control the sequence in which code is read (exactly like Python)
- After writing your code, when you press  to run the program, the code is uploaded and saved to your MicroPython, allowing you to take code on-to-move and use it away from your computer
  - Note: Files can save to the PC or to the Pico itself
  - **main.py** is special! This is the default program the Pico will run on boot.
- Thonny allows users to run code through the shell (which sends information straight to the Pico every time you press enter) or using the script area (this is where we want to be doing our coding)

# Hello, World!

- In the script area in Thonny, type:  

```
print("Hello, World!")
```
- To run your first code, click the Run icon at the top of the Thonny window
- It will ask you to save the file, add a descriptive title and click OK
  - INCLUDE '.py' AT THE END OF YOUR FILE
- Two lines should appear in the shell at the bottom of the Thonny window:
  - %Run -c \$EDITOR\_CONTENT
  - Hello, World!
- The first line is an instruction from Thonny telling the Pico to run the contents of the script area and the second is our output



# Loops

- For and While loops in MicroPython work exactly the same as they do in regular Python3

## Simple *for* loop example

- In a new script ('New' icon above) type the following code:

```
print("Loop starting!")  
for i in range(10):  
    print("Loop number", i)  
print("Loop Finished!")
```

- Note: MicroPython, like Python, is a zero-indexed language, meaning it starts counting from 0 instead of 1.

# Lists

- Lists are a data type that allow you to store multiple values in one variable
- Example:

```
list1 = [1, 2, 3, 'a', 'b', [4,5]]
```

- Individual values can be accessed via [], Example:

```
print("First and last element: ", list1[0], list1[-1])
```

- Multiple values can be accessed using : to specify range, Example:

```
print("Second through 4th element: ", list1[1:4])
```

- Can add elements to the end using append(), Example:

```
list1.append(9.8)
```

# LED and Buttons

- To interface with the GPIO, we need the *Pin* class from *machine*  
from machine import Pin
- Code to turn on onboard LED:  
led = Pin(25,Pin.OUT)  
led.on()
- Buttons work the same way  
button = Pin(15,Pin.IN)  
print(button.value())

## Exercise: Create a binary lock

The design will be as follows:

- The lock will use a binary code as its password, this can be arbitrarily long
  - Identified as a string
- The user will enter a binary code, which can also be arbitrarily long, with the help of three buttons
- The first one will toggle the bit, so pressing it once will change it from 0 to 1, and again will change it from 1 to 0
- The second button will confirm the current state of the bit as an input, so if the button is pressed when the bit is 1, the user input is '1', if it's pressed again when bit is '0', the input is '10'
- The third button will indicate end of input, i.e whatever the input was will be compared with the password to check whether to unlock the lock.

# Exercise: Create a binary lock

