CamJam EduKit Worksheet Four

Project Interact with the user and turn on LEDs appropriately.

Description In this project, you will control the red, yellow, or green LEDs depending on your choice.

Equipment Required

The circuit built in Worksheet Two.

Code

You are going to use the same circuit again, but this time you are going to control the LEDs with user input. This worksheet will also show you how to use variables to store information that will be used in later code.

Explanations have been placed within the code. These are called 'comments' and in Python they are the text following the '#' symbol. Nothing after the # will be run, and can be left out if you want. However, best practice is to use comments to remind you what you intended your code to do.

Plug the small end of the micro-USB cable in to the Pico and connect the other end to your computer. Start Thonny and create a new file. Type in the following:

```
# CamJam EduKit 1 - Basic
# Worksheet 4 - User Input

# Import Libraries
import time  # A collection of time related commands
from picozero import LED  # The LED function from picozero

# Set pins 15, 14 and 13 to be LEDs
red = LED(15)
yellow = LED(14)
green = LED(13)

# Ask the user which colour LED to blink
print("Which LED would you like to blink?")
print("1: Red?")
print("2: Yellow?")
print("3: Green?")
led_choice = input("Choose your option: ")
# Click on the space after : in the Shell area
# to move the cursor to correct position

# Ensure that the led_choice variable is a whole number (integer)
led_choice = int(led_choice)

# Ask the user how many times they want the LED to blink
count = input("How many times would you like it to blink? ")

# Ensure that the count variable is a whole number (integer)
count = int(count)

# Sets the variable 'led_chosen' to be the LED chosen
if led_choice == 1:
    print("You picked the Red LED")
    led_chosen = red
elif led_choice == 2:
```

```
    print("You picked the Yellow LED")
    led_chosen = yellow
elif led_choice == 3:
    print("You picked the Green LED")
    led_chosen = green

# If we have chosen a valid choice, flash the LED
if led_choice > 0:
    # While the count variable is greater than zero
    while count > 0:
        led_chosen.on()  # Turn the chosen LED on
        time.sleep(1)  # Sleep for 1 second
        led_chosen.off()  # Turn the chosen LED off
        time.sleep(2)  # Sleep for 2 seconds
        count = count - 1  # Decrease the count by one
```

Once complete, save the file onto your Pico as `4-user-input.py`.

Explanation of the Code

```
led_choice = input("Choose your option: ")
led_choice = int(led_choice)
```

Your choice of 1, 2 or 3 will be stored in a variable called 'led_choice'

'=' assigns the value of the right-side item to the left-side item.

'==' compares the two values and returns True if they are equal, of False if they are not.

int() ensures that Python understands that the value is an integer (a whole number).

```
if led_choice == 1:
    print("You picked the Red LED")
    led_chosen = red
```

The 'if' statement will test to see whether led_choice is equal to 1. The value is either:

- 'True', meaning led_choice is equal to 1, or
- 'False', meaning led_choice is not equal to 1.

If the test condition returns a 'True' value, the indented block of code after the '`if`' line will be executed, and execution will continue at the next line after the whole '`if .... elif ....`' section.

```
    print("You picked the Red LED")
    led_chosen = red
```

If the test condition returns a 'False' value, the block of code will be skipped and execution continues at the next line after the body.

```
elif led_choice == 2:
```

This line will execute if the previous line, '`led_choice == 1`' is False. This is the '`else`' part of the line '`if led_choice == 1:`'

The '`elif`' statement will test to see whether led_choice is equal to 2. This is the '`if`' part of '`elif`'.

If the test condition returns 'True', the body of the '`elif`' line will be executed.

```
        print("You picked the Yellow LED")
        led_chosen = yellow
```

If the test condition returns 'False', the body will be skipped. Execution continues at the next line **after** the body of '`elif led_choice == 2:`'. In this example, the line '`elif led_choice == 3:`' will be executed.

```
if led_choice > 0:
```

This is a test condition which determines whether led_choice is greater than 0.

If 'True', the indented block will be executed.

If 'False', the indented block will be skipped. Since there are no more lines to execute, the program will end.

```
while count > 0:
```

The 'while' statement tests whether the variable called 'count' is greater than 0.

If 'True', the next indented block will be executed. Notice that the last line of the body modifies this 'loop-control variable'.

```
        led_chosen.on()  # Turn the chosen LED on
        time.sleep(1)  # Sleep for 1 second
        led_chosen.off()  # Turn the chosen LED off
        time.sleep(2)  # Sleep for 2 seconds
        count = count - 1  # Decrease the count by one
```

After the block has been executed, the program will 'go back' to the '`while`' line and '`count > 0`' will be re-tested.

If 'True', the block will be executed one more time. This repetition is known as 'looping'.

If 'False', execution continues at the next line after the block. Since there are no more lines following, the program will end.

Running the Code

Click the green Run icon on the top menu bar. The screen will clear, and you will be prompted for which LED you want to turn on or off.

In the Thonny Shell area, click one space after the ':' to move the cursor to the correct position. Enter 1, 2, or 3. You will then be prompted for how many times you want the LEDs to flash. The LED you chose will then flash the number of times you requested.

Note: Do not disassemble this circuit as it will be used in the following worksheets.

The concepts that you have learned in this worksheet

Selection

Simple 'if'

'if' select path based on a **single** condition.

```
if condition_0:
```

If True, execute this block.

If False, skip this block.
Continues at the next line that is at the same indentation of the '`if`'.

if ... else ...

```
if (condition_0):
```

If True, execute the indented block and continue at the next line after the block of the **whole** '`if … else …`' structure.

```
else:
```

If False, execute this block and then continue at the next line after the **whole** '`if … else …`' structure

if ... elif ...

The `if … elif …` construct selects "execution paths" based on **multiple** conditions. You can have as many conditions as you like.

```
if condition_0:
```

if True, execute this block and then continue at the next line after the block of the **whole** '`if … elif …`' structure.

If False, skip this block and continue at next line after this block.

```
elif condition_1:
```

If True, execute this block and then continue at the next line after the block of the **whole** '`if… elif …`' structure.

If False, skip this block and then continue at the next line after this block.

Repetition

A 'while' statement can control the number of repetitions of the next code block.

```
while count > 0:
```

If True, execute this block and then, go back to the '`while`' line. The 'loop-control variable' can be changed within the block. For clarity, it is preferable to place the change of the loop-control variable at the beginning or end of the block, but it can be anywhere within the block.

For example, in the code in this worksheet, the 'loop-control variable' is decremented by 1 each time the while block is executed (`count = count - 1`). This is the last line of the `while` block.

Scope and nesting

Scope is marked by the level of indentation. There can be multiple levels of indentation; one deeper than the other. This is called 'nesting'.

In this illustration, the second 'deeper' or 'inner' indentation level is shown in bold. Within the scope of a given indentation level, you can have one or as many lines as you want.

You can have as many levels of indentation as you like. For the ease of reading, the level of nesting should be kept relatively small.

```
if led_choice > 0:
    # First line for scope of 'if'
    ......
    while count > 0:
        # First line for scope of 'while'
        led_chosen.on()  # Turn the chosen LED on
        ......
        # last line for scope of 'while'
    # line(s) at same indentation of 'while'
    # and also within scope of 'if'
    ......
    # last line for scope of 'if'
# next line at same indentation of 'if'
```