Cohort Session 3, Week 3
Blinky

Objectives
1. Implement conditionals and loops in Python.
2. Learn to interface with hardware (LED, switchs) using Raspberry Pi.
3. Write a Python program to control an LED with a switch.

Please work on this lab in a group of **two or three**. Be sure to email your partner all the modified code, printouts and data. You may have to use them during your exams.

You should have completed the pre-activity handout before starting work on this project.

1 Equipment & Software

Download the software for this project from **1D Project website**.

Each group should have:

1. A Raspberry Pi.
2. A cobbler.
3. An LCD touch screen.
4. A wireless keyboard.
5. A wireless mouse.
6. 2 LEDs.
7. A switch.
8. 2 Resistors (330Ω).
9. A few jumper wires.
10. A breadboard. Refer to **GPIO Pins and Breadboard Layout** for its layout information.
11. `wk3_template.py`, which you would use as template to write your code.

2 LED Blinking

Task
Write a program to control two LEDs with a switch.

Modify the `wk3_template.py` file in your Raspberry Pi to perform the following:

1. Check whether the switch is closed or opened.
2. If the switch is closed, turn off the LED on the left and blink the LED on the right.
3. If the switch is opened, turn off the LED on the right and blink the LED on the left.
4. The blink interval is 1 second.

You should have noticed that the code in the template file is mostly similar to the `raspberrypi_sample.py` file that was used in the pre-activity (If you have not attempted the pre-activity, you should do it before working on this task).

However, some of the differences worth noting are:

```
7  # Use GPIO23 for LED 1, GPIO24 for LED 2 and GPIO18 for switch
8  led = [23, 24]
9  switch = 18
10
11 # Set the GPIO23 and GPIO24 as output.
12 GPIO.setup(led, GPIO.OUT)
```

1. Line 8 of the template uses a list to store the GPIO numbers that will be used for the LEDs. In line 12, the list is passed in as an argument into the `GPIO.setup()` function. The first argument of the `GPIO.setup()` can take in either an int or a sequence of GPIO numbers. By grouping the GPIO numbers into a list, it makes the code neater than having multiple lines of GPIO number assignment and `GPIO.setup()` to set them as input/output.

```
17 def blink(gpio_number, duration):
18     '''This function takes in two input: gpio_number and duration. The
19     gpio_number specifies the GPIO number which the LED (to be blinked) is
20     connected to. The duration is the blink interval in seconds.'''
21
22     # Write your code here
23     pass
```

2. A function blink has been defined. You should make use of this function to blink the LED. The blink function takes in two input: `gpio_number` and `duration`. The `gpio_number` specifies the GPIO number which the LED (to be blinked) is connected to. The `duration` is the blink interval in seconds.

```
25 while True:
26     # Check whether the switch is closed or opened. When the switch is closed,
```

```
27      # turn off the LED at GPIO24 and blink the LED at GPIO23. When the switch
28      # is opened, turn off the LED at GPIO23 and blink the LED at GPIO24. The
29      # blink interval should be 1 second.
30
31      # Write your code here
32      pass
```

3. Similar to the `raspberrypi_sample.py`, there is a while loop where you should check the status of the switch and control the LEDs accordingly. You should make use of the `led` list when you are trying to set an output to be HIGH or LOW. Recall that you can access a list element using an index.

You may refer to the following figure to wire up the the LEDs and switch.

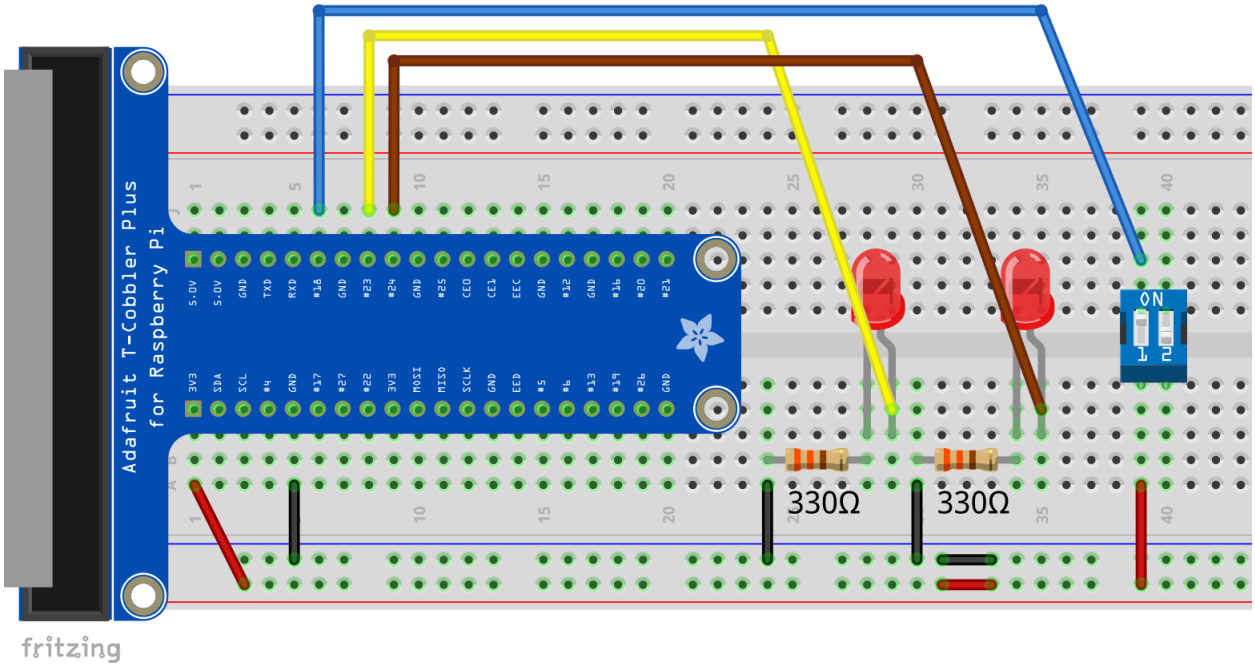

Figure 1: Note that the longer leg of the LED is the positive end. Hence it should be connected to the GPIO pin.

WARNING!
The state of the GPIOs will be retained after the program terminates. That is, if any GPIO output was HIGH at the moment the program terminates, it would remain as HIGH even after the program terminates.

Consequently, it is possible to accidentally damage the Raspberry Pi by connecting GPIOs that was HIGH directly to ground. In order to 'reset' the GPIOs, you can use gpiocleanup.py (found at **Courseware website**) and run it in your Raspberry Pi. The program sets every GPIO as input. As such, they would not get damaged even if they are connected directly to either HIGH or LOW.

Checkoff 1
Explain and demonstrate the working program to a staff member. The program should make use of the blink function and access the GPIO numbers using the `led` list in the while loop.