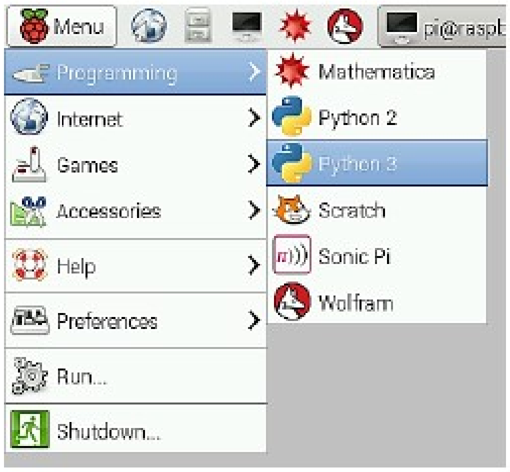
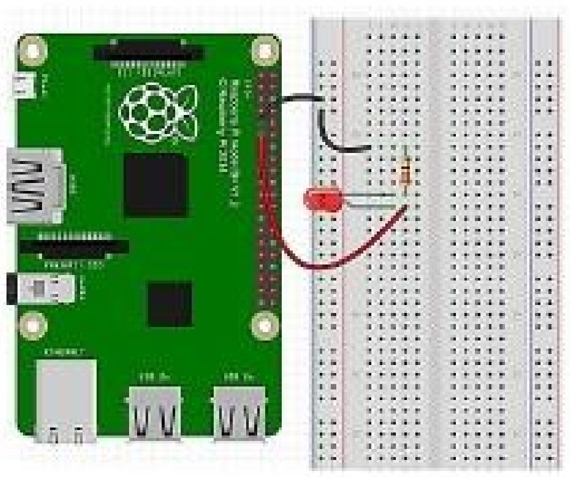
GETTING STARTED WITH GPIO

1. OPEN THE PYTHON IDE



2. BUILD THE FOLLOWING CIRCUIT TO TEST OUTPUT FUNCTIONS



Hints:

- Connect the positive lead of the LED to pin 11.
- Use a 1kΩ resistor
- All resistors in your kit are of the same value

3. Write a Program to Turn on the LED

Create a new Python file called "LED.py" and enter the following lines of code:

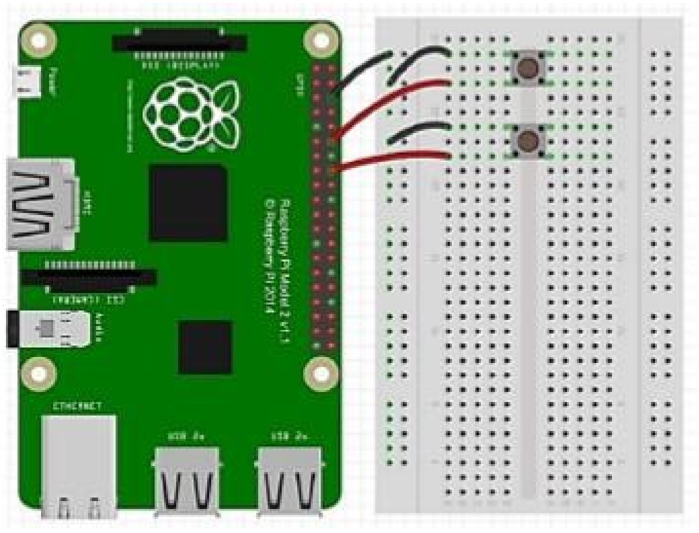
```
#This imports the necessary libraries to access the pins on the Pi
import RPi.GPIO as GPIO

#This tells Python and the Pi that we're using the physical pin numbering
scheme
GPIO.setmode(GPIO.BOARD)

#Alternatively, you can use the BCM numbering scheme with this line:
#GPIO.setmode(GPIO.BCM)
#(but don't…)

led = 11
#According to the physical pin numbering scheme, we want pin 11 to be the
output. If using the GPIO
#numbering scheme, you would use "17" to reference the same pin

#configure pin 11 as an output
GPIO.setup(led, GPIO.OUT)

#This sends a high output to pin 11 and turns the LED on
GPIO.output(led, True)
```

4. Write a Program to Blink the LED

You can modify your code using the **time library** to blink the LED as follows:

```
import RPi.GPIO as GPIO
import time #this allows us to introduce delays in our code

GPIO.setmode(GPIO.BOARD) #physical numbering scheme

led = 11 #connect LED to pin 11

GPIO.setup(led, GPIO.OUT) #set pin 11 as output

#get user input and store it as an integer in a variable
blinkNum = int(input('How many times do you want to blink? '))

for i in range(blinkNum):
      GPIO.output(led, True) #can also use 1 instead of TRUE to turn on
      time.sleep(1) #delay 1 second
      GPIO.output(led, False) #can also use 0 instead of FALSE to turn off
      time.sleep(1) #delay 1 second
GPIO.cleanup()
```

5. Build a circuit to test Input Functions



6. Read Values from Input Pins

When we attach a button to a circuit, we essentially have an open circuit until the button is pressed. When a pin is connected to nothing (like it is when connected to an open button), we call it a floating pin. This causes problems because floating pins are susceptible to interference from nearby devices—they will "float" past the threshold voltage for the pin and alternate between HI and LO. To avoid this, an internal pull up resistor ties the pin to HI (on the device). When ground is connected (button closed), the resistor is shorted out and the pin sees LO.

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)
start = 12 #read from pins 12 and 16
stop = 16

#Declare the pins as inputs and "attach" the pull_up resistors to them
GPIO.setup(start, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(stop, GPIO.IN, pull_up_down=GPIO.PUD_UP)

end = False
while (not end):
    if GPIO.input(start)==0:
        print('Start was pressed')
        sleep(0.5)
    if GPIO.input(stop)==0:
        print('Stop was pressed')
        sleep(0.5)
    if GPIO.input(start)==0 and GPIO.input(stop)==0:
        print('Make up your mind!')
        end = True

GPIO.cleanup() #clear out pin assignments
```