Projects

Python Quick Reaction Game

Make a quick reaction game to play with your friends

Step 1 Introduction

In this resource, you're going to make a quick reaction game using a few electronic components and a Python script. If you've never before used a breadboard, some buttons, and an LED, you might find it helpful to work through some of the exercises in **Physical Computing with Python** (https://projects.raspberrypi.org/en/projects/physical-computing) first. This will give you a better understanding of how to control components with the Raspberry Pi's GPIO pins.

What you will make

This project gives you the opportunity to use electronics to create a quick reaction game which you will program using Python. If you have little or no experience of creating circuits, don't worry: this guide will walk you through it and by the end you will have a fun game to play with your friends.

What you will learn

By making this quick reaction game, you will learn:

- How to wire a simple circuit that includes a breadboard, LED, resistor, wires, and buttons
- How to write a program to control the circuit
- How to use variables to store information
- How to get user information like a player's name and use it in the game.

This resource covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (https://www.raspberrypi.org/curriculum/):

- **Use basic programming constructs to create simple programs** (https://www.raspberrypi.org/curriculum/programming/creator)
- **Combine inputs and/or outputs to create projects or solve a problem** (https://www.raspberrypi.org/curriculum/physical-computing/builder)

What you will need

Hardware

- 1 x Raspberry Pi
- 1 x Breadboard
- 1 x LED
- 1 x 330 ohm Resistor
- 4 x Male-to-female jumper wires
- 2 x Male-to-male jumper wires
- 2 x Tactile push buttons

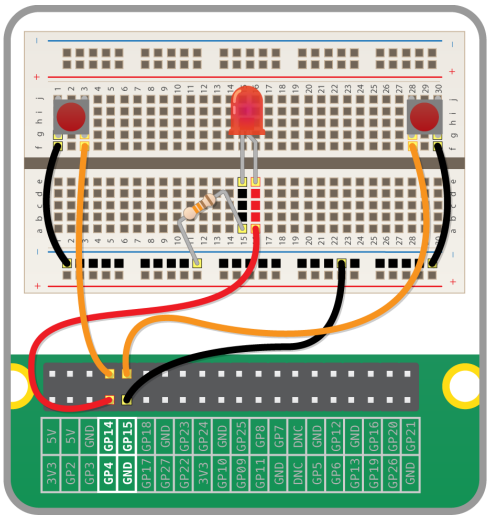
Additional information for educators

If you need to print this project, please use the **printer-friendly version** (https://projects.raspberrypi.org/en/projects/python-quick-reaction-game/print).

Here is a link to the resources for this project (http://rpf.io/python-quick-reaction-game).

Step 2 Building the circuit

This is the circuit you are going to build, consisting of two push-to-make buttons and an LED.



Take one of your tactile buttons and push it into the holes on your breadboard, with one set of legs on row **H** and one set of legs on row **J**.

Repeat the last step with the second button, placing it at the other end of the breadboard on the same row.

Place an LED with the longer leg above the ridge in the breadboard in **D16** and the shorter leg in **D15**. The numbering will depend on your breadboard so make sure that you check the diagram below.

Next push one leg of the resistor into the same column (**15**) as the short leg of the resistor and the other leg into a hole along the blue strip.

Now it's time to add the jumper wires. Start by taking two male-to-male jumper wires and placing one end in a hole next to the outside leg of the left hand button, and the other end in a hole along the blue strip. Repeat this step with the right hand button.

- [ ] Then with a male-to-female jumper wire, connect **GPIO14** to a hole on the breadboard in line with the other leg of the left hand button. Repeat this step for the right hand button, only this time connecting it to **GPIO15**.

- [ ] Using another male-to-female jumper wire, connect **GPIO4** to a hole on the breadboard in line with the long leg of the LED.

- [ ] Finally, connect a **GND** GPIO pin to the blue strip on the breadboard with the remaining male-to-female jumper wire.

Step 3 Controlling the light

When programming, it makes sense to tackle one problem at a time. This makes it easier to test your project at various stages.

Click on the `Menu`>`Programming`>`Mu`

Save the file as **reaction.py** by clicking on `File`>`Save As`

First you will need to import the modules and libraries needed to control the GPIO pins on the Raspberry Pi. Type:

reaction.py

```
from gpiozero import LED, Button
from time import sleep
```

As you are outputting to an LED, you need to set up the pin that the LED connects to on the Raspberry Pi as an output. First use a variable to name the pin and then set the output:

reaction.py

```
from gpiozero import LED, Button
from time import sleep

led = LED(4)
```

Next add a few lines to turn the LED on, wait for 5 seconds and then turn the LED off:

reaction.py

```
from gpiozero import LED, Button
from time import sleep

led = LED(4)

led.on()
sleep(5)
led.off()
```

Finally, test that it works by click on `Run`.

If the LED does not come on for five seconds, go back and see if you can work out what went wrong. This is a very important skill in computing called **debugging**, which means finding and fixing errors or bugs in your code.

Step 4 Adding an element of surprise

The object of the game is to see who can press the button first when the light goes out, so it would be better if the length of time it stayed on were random. You need to add and amend some lines of code in your Python program to make this happen.

Underneath `from time import sleep` add a line to import `uniform`

reaction.py

```
from gpiozero import LED, Button
from time import sleep
from random import uniform

led = LED(4)

led.on()
sleep(5)
led.off()
```

Here, `uniform` allows for the random selection of a decimal (floating point) number from a range of numbers.

Then locate the line `sleep(5)` and amend it so that it reads:

reaction.py

```
from gpiozero import LED, Button
from time import sleep
from random import uniform

led = LED(4)

led.on()
sleep(uniform(5, 10))
led.off()
```

Save your work by clicking on **Save**. Test that everything works by clicking on **Run**

Step 5 Detecting the buttons

The LED is working; now you want to add functionality to your program so that when a button is pressed it is detected. That way you can record the players' scores to see who wins.

As with the last step, some code needs to be added to your current program.

With the file **reaction.py** open add the following variables underneath `led = LED(4)`:

reaction.py

```
from gpiozero import LED, Button
from time import sleep
from random import uniform

led = LED(4)
right_button = Button(15)
left_button = Button(14)

led.on()
sleep(uniform(5, 10))
led.off()
```

Then underneath `led.off()` you can add a function that will be called whenever a button is pressed, which will tell you which **pin** the button was on:

reaction.py

```
from gpiozero import LED, Button
from time import sleep
from random import uniform

led = LED(4)
right_button = Button(15)
left_button = Button(14)

led.on()
sleep(uniform(5, 10))
led.off()


def pressed(button):
    print(str(button.pin.number) + ' won the game')
```

To finish off, when either button is pressed, the function will be called. If the `right_button` is pressed, then you can send the string `'right'` to the `pressed` function. If the `left_button` is pressed, then you can send the string `'left'`.

reaction.py

```
from gpiozero import LED, Button
from time import sleep
from random import uniform

led = LED(4)
right_button = Button(15)
left_button = Button(14)

led.on()
sleep(uniform(5, 10))
led.off()


def pressed(button):
    print(str(button.pin.number) + ' won the game')


right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

Save your program and test it with a friend.

Step 6 Get player names

Wouldn't it be better if the program told you who has won instead of just which button was pressed? For this, you need to find out the players' names. In Python, you can use **input** for this.

To find out the names of the players you can use `input` to ask the players to type in their names. Underneath the imported libraries and modules, and the highlighted code.

reaction.py

```
from gpiozero import LED, Button
from time import sleep
from random import uniform

led = LED(4)
right_button = Button(15)
left_button = Button(14)

left_name = input('left player name is ')
right_name = input('right player name is ')

led.on()
sleep(uniform(5, 10))
led.off()


def pressed(button):
    print(str(button.pin.number) + ' won the game')


right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

Now you can rewrite your pressed function, so that it can print out the name of the player who won.

reaction.py

```
from gpiozero import LED, Button
from time import sleep
from random import uniform

led = LED(4)
right_button = Button(15)
left_button = Button(14)

left_name = input('left player name is ')
right_name = input('right player name is ')

led.on()
sleep(uniform(5, 10))
led.off()


def pressed(button):
    if button.pin.number == 14:
        print(left_name + ' won the game')
    else:
        print(right_name + ' won the game')


right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

Run your program and test your game to see if it works.

You might notice that the game doesn't quit when the button has been pushed. This can be fixed by adding an exit into the `pressed` function. Add the highlighted lines to your code,

reaction.py

```
from gpiozero import LED, Button
from time import sleep
from random import uniform
from sys import exit

led = LED(4)
right_button = Button(15)
left_button = Button(14)

left_name = input('left player name is ')
right_name = input('right player name is ')

led.on()
sleep(uniform(5, 10))
led.off()


def pressed(button):
    if button.pin.number == 14:
        print(left_name + ' won the game')
    else:
        print(right_name + ' won the game')
    exit()


right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

Step 7 What next?

- Can you put the game into a loop (you'll need to remove the `exit()`), so that the LED comes on again?
- Can you add scores for both players that accumulate over a number of rounds, and displays the players' total scores?
- How about adding in a timer, to work out how long it took the players to press the button after the LED turned off?
