# Hack Super Mario Bros. with Python!

**Get Mario-Level-1 from the Raspberry Pi**

Launch a terminal/command prompt. Google how to do this if you can't remember how.

Log into the Pi via ssh:

```
$ ssh pi@dex.local
```

Look at what's in the pi user's home directory:

```
$ ls
```

You should see a file called "kiavo-pythongames-1.zip".

Open up a new terminal/command prompt window that you will use to access your own computer instead of the pi. If you don't alread have a directory called "pythongames" in a place that is easy to find, use "cd" to navigate to the place you would like to create it and use mkdir to make a new directory:

```
$ mkdir pythongames
```

Making sure that your command prompt is inside the pythongames folder, copy the kiavo-pythongames-1.zip file from the Pi to your computer via scp:

```
$ scp pi@dex.local:~/kiavo-pythongames-1.zip .
```

The single dot at the end tells scp to place the downloaded file in your current directory.

Now look at your directory and you should see the file kiavo-pythongames-1.zip:

Mac/Linux: `$ ls`
Windows: `> dir`

**Unzip the kiavo-pythongames-1.zip file:**

Mac/Linux: `$ unzip -a kiavo-pythongames-1.zip`
if this doesn't work, just find the zip file in Finder and double click on it. This will extract the files.

Windows: Download http://stahlworks.com/dev/unzip.exe using your web browser and double click on the file that you downloaded.
Then go back to the command prompt and type:
`> unzip kiavo-pythongames-1.zip`

If this still doesn't work, find the zip file in your File Explorer and double click on it to extract it.

**Play Mario-Level-1**

You should have already installed PyGame Zero, which comes packaged with regular PyGame.

Check to see that it is there with:

```
$ pgzrun
```

If you have it installed, you should get something like:

```
pygame 1.9.4
Hello from the pygame community. https://www.pygame.org/
contribute.html
Usage: pgzrun [options]

pgzrun: error: You must specify which module to run.
```

If not, install Pygame Zero by typing:

```
$ pip install pgzero
```

Go into the kiavo-pythongames-1 folder if it is there, or otherwise go into the pythonsupermario directory if that is what you see.

Launch the game with Python:

```
$ python mario_level_1.py
```

Play the game a couple of times!

The arrow keys move Mario around, 'a' makes Mario jump and 's' makes him run fast.

**Improving Mario's behavior**

Is there anyhing you don't like about Mario's movement on the screen? Is he too slow? Does he jump high enough?

Go back to the pythonsupermario directory on the command line. Use `ls` (Mac/Linux) or `dir` (Windows) to list the directory contents. You should see a directory called markmillr-Mario-Level-1. Rename this directory with your own name with the move command:

Mac/Linux: `mv markmillr-Mario-Level-1 loganmarieella-Mario-Level-1`
Windows: `move markmillr-Mario-Level-1 loganmarieella-Mario-Level-1`

(replace loganmarieella with your own name or screenname).

Go into that directory with `cd` and play the game with `$ python mario_level_1.py` . What's different?

It turns out that the young man who developed this Python version of Super Mario, Justin Armstrong in Victoria, British Columbia, Canada (github name justinmeister) used a list of constants to specify the behavior of the Mario actor in his game. The file that specifies Mario's attributes is in the data folder: $ cd data and in a file called `constants.py`.

We can use IDLE to modify the file, so at the command line type `$ idle constants.py`

Here you'll see a bunch of variable definitions like screen width and height, the RGB (red, green, blue) definition of various colors, and a lot of game-specific constants that Justin defined to make his game work.

Find the comment `# MARIO FORCES`

The numbers here are used by Justin's code to change how Mario is affected by physics within the game. VEL stands for velocity (speed). Pick a variable that you would like to change. Comment out the original code by putting a hashtag in front of it, and copy the text with control-c (Windows/Linux) or command-c (Mac), and paste the text below with ctrl-v (Windows/Linux) or cmd - v (Mac).

For example, to change JUMP_VEL, you could change the code so that it looks like the following:

```
 #MARIO FORCES
WALK_ACCEL = .15
RUN_ACCEL = 20
SMALL_TURNAROUND = .35

GRAVITY = 1.01
JUMP_GRAVITY = .31
#JUMP_VEL = -10
JUMP_VEL = -15.5
FAST_JUMP_VEL = -12.5
MAX_Y_VEL = 11

MAX_RUN_SPEED = 800
MAX_WALK_SPEED = 6
```

Notice how one JUMP_VEL is commented out, and there is a copy of it below with a different value. This is a way of keeping the original value there for reference. Eventually when you are finished tuning your parameters/constants, you can just delete all the comment lines so that your code is easy to read.

Try changing only one variable at a time to see what it does. You'll have to make sure to save the file each time. On the command line, since you're in the "data" directory and the main game executable file is in the directory above, you will have to tell Python where to look for the game file.

$ python ../mario-level-1.py

Two dots, "..", means "look in the folder up from where I am." One dot (".") means "look in this folder".

Now you are a scientist and an engineer, tweaking and testing little changes in your code to see what happens. You'll end up playing Mario dozens of times as you test your code!

Rather than retyping "python ../mario-level-1.py" dozens of times every time you want to run it, you can simply hit the up arrow and the Terminal/Command Prompt will tell you what your last commands were. This saves a lot of typing and time.

**Make Mario invincible**

At the command line, go into the "components" directory by typing:

```
$ cd components
```

Open up mario.py in IDLE:

```
$ idle mario.py
```

Here you will see a definition of the Mario "class". Once a class is defined, you can make as many of them as you like, and they will all have the same functions and attributes. The Mario class definition is the mould for creating new Mario objects. In turn, in this code, the Mario class inherits all of the features from PyGame's "Sprite" class, and then here in this file the author of the code added a bunch of custom features specific to this game.

Find the line that says `def setup_state_booleans(self):`

Here you can see where I made some custom tweaks to the Mario class. Remember how Mario is invincible right from the start in this custom version? This is because according to the definition I added here, he is created with the attribute `self.invincible = True`.

Switch the comment hashtags and make `self.invincible = False` so that your code looks like this:

```
    def setup_state_booleans(self):
        """Sets up booleans that affect Mario's behavior"""
        self.facing_right = True
        self.allow_jump = True
        self.dead = False
        self.invincible = False
        #self.invincible = True
```

Play the game again. Is Mario the way you expected?

Tweak the different variables in setup_state_booleans() and play the game repeatedly to test your hypotheses. Have fun hacking Mario!