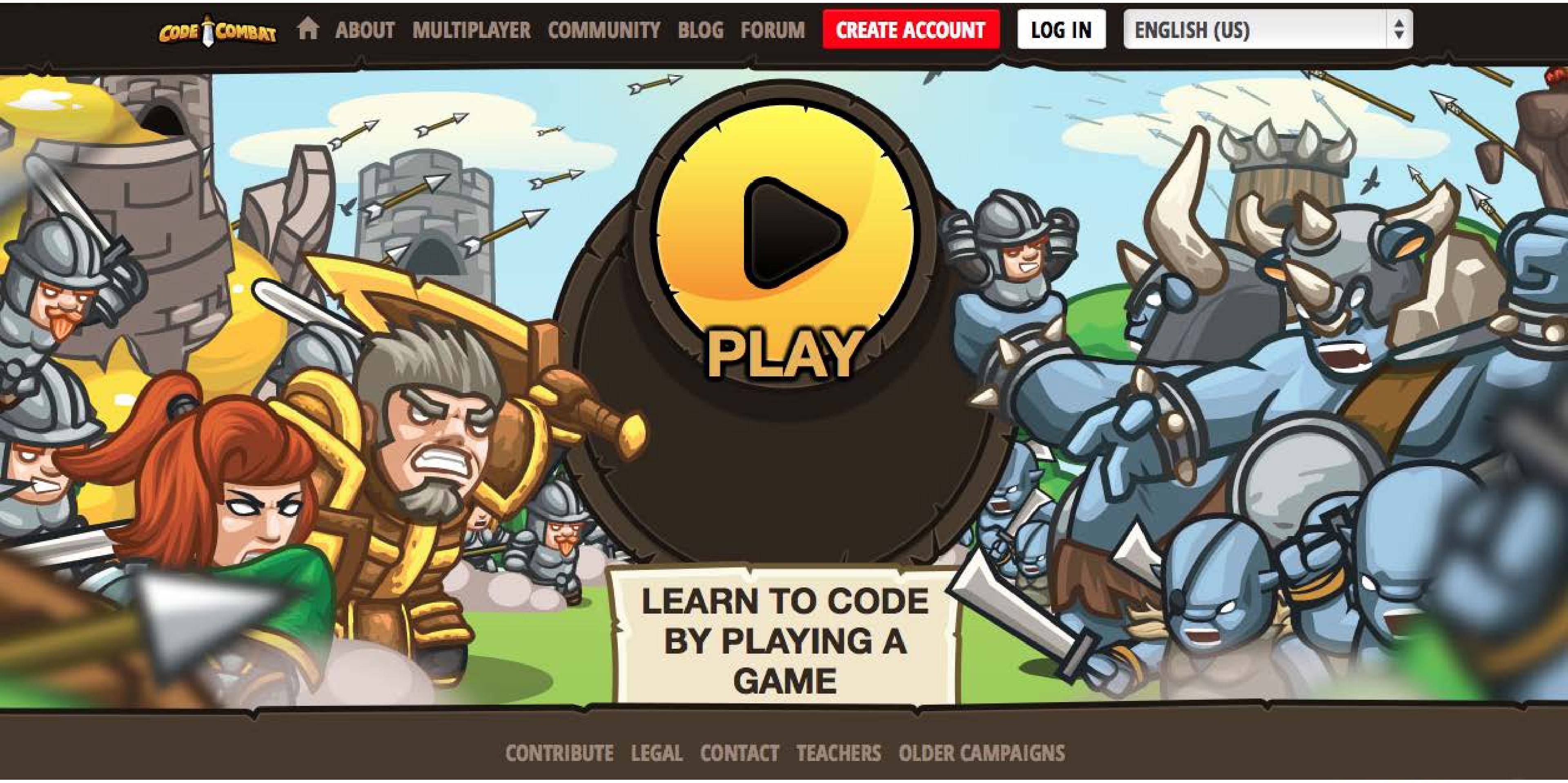
Code Combat


CODE COMBAT
ABOUT
MULTIPLAYER
COMMUNITY
BLOG
FORUM
CREATE ACCOUNT
LOG IN
ENGLISH (US)
PLAY
LEARN TO CODE BY PLAYING A GAME
CONTRIBUTE
LEGAL
CONTACT
TEACHERS
OLDER CAMPAIGNS

Welcome!

So you want to learn to code, eh?

Welcome

I will be your host to the dark arts of … err I mean Programming

CodeCombat is a fun, addictive, and effective way of learning real, line based code.

Introduction

I will teach and test you along the way through challenging puzzle & battle based levels

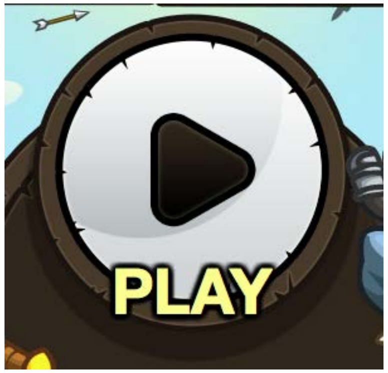
Introduction

The code you are writing uses real **syntax**, meaning the lines you write to solve these puzzles are real lines of code

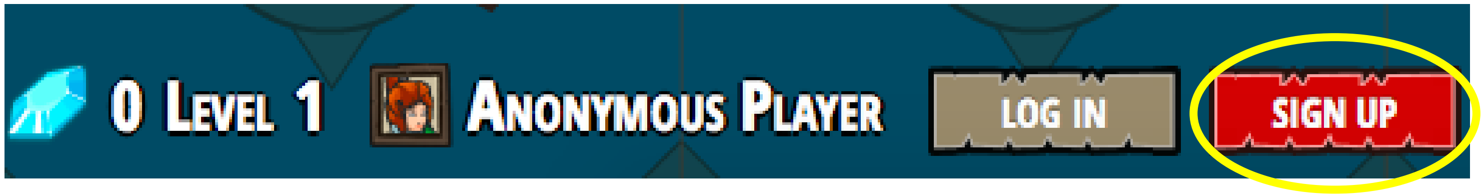
Enough talk - let's start!

Navigate to codecombat.com.

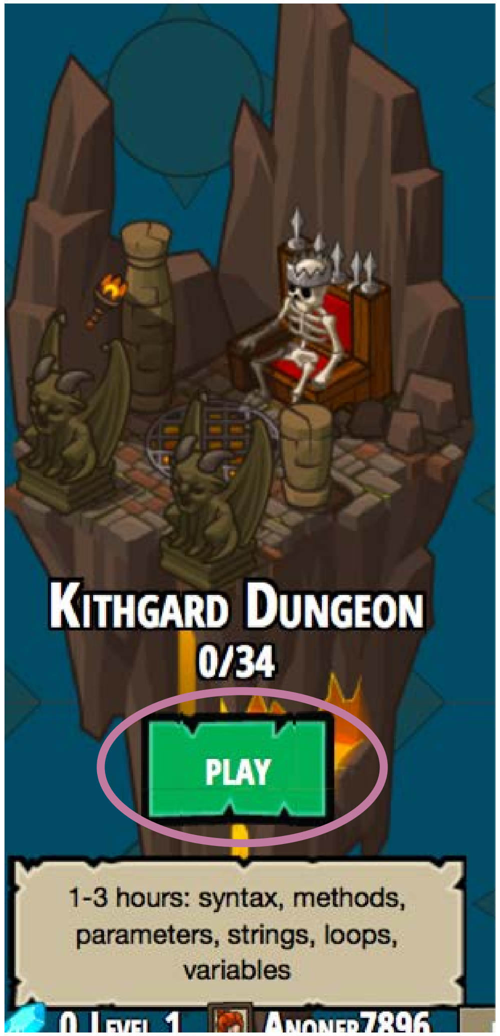
Click the big play button at the center of your screen.



Then down at the bottom of the screen select Sign Up to create your account.



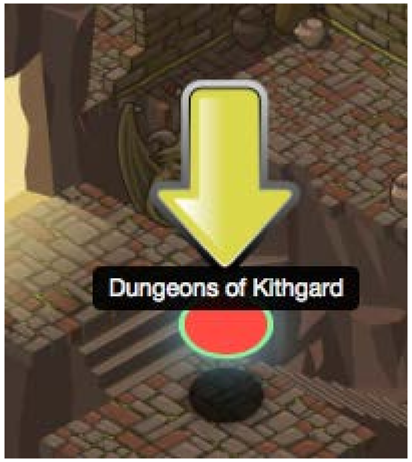
Getting started

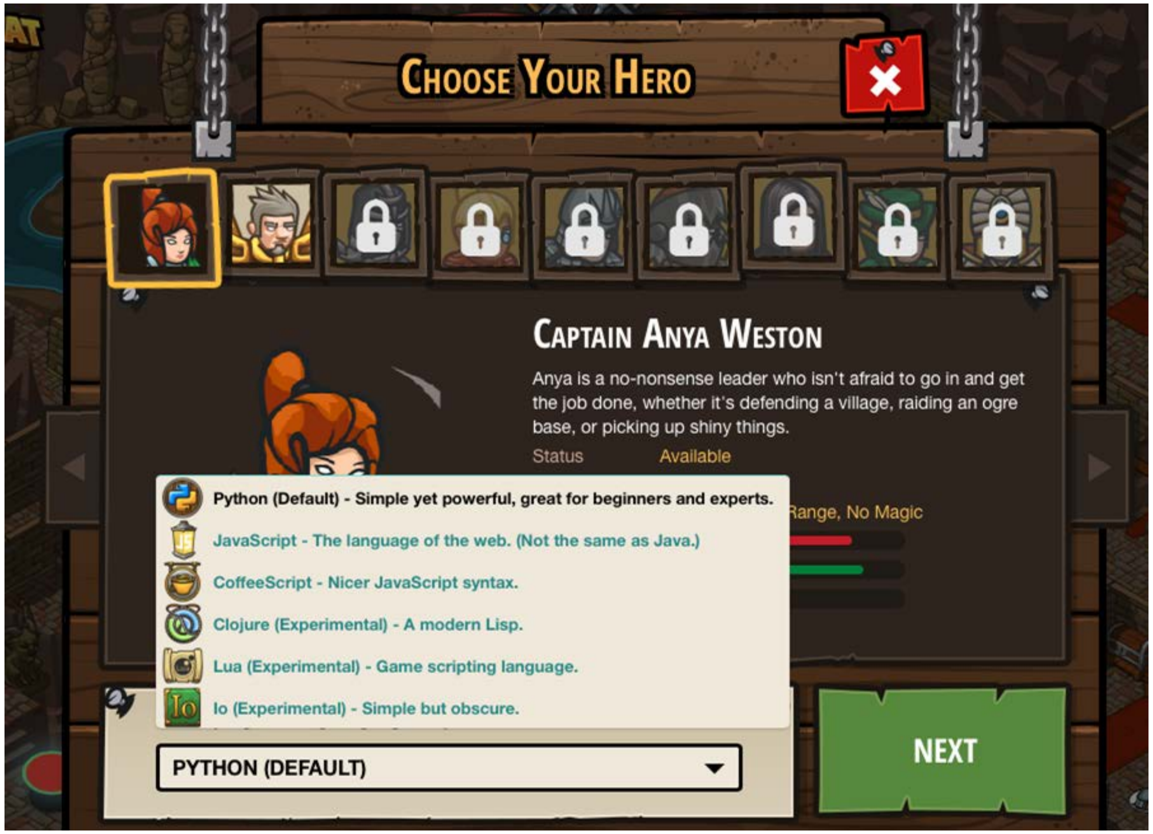


1-3 hours: syntax, methods, parameters, strings, loops, variables

Once you have entered your information, select the first world, Kithgard Dungeon

Getting started





Anya is a no-nonsense leader who isn't afraid to go in and get the job done, whether it's defending a village, raiding an ogre base, or picking up shiny things.

Status Available

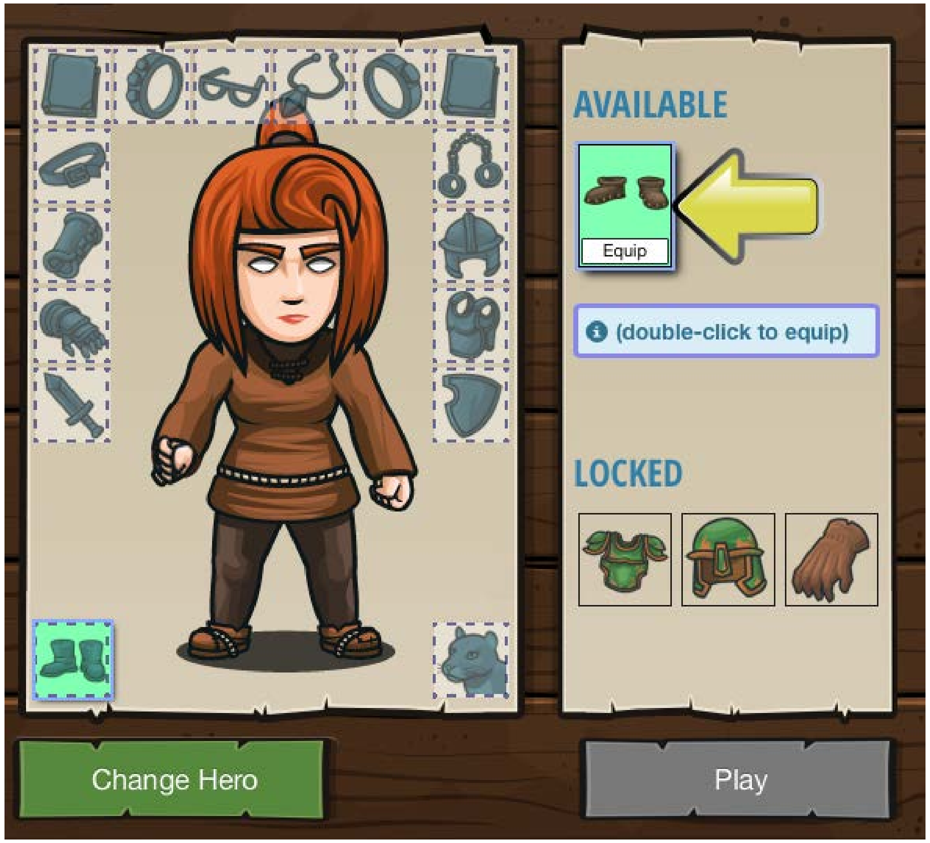
Range, No Magic

Python (Default) - Simple yet powerful, great for beginners and experts.
JavaScript - The language of the web. (Not the same as Java.)
CoffeeScript - Nicer JavaScript syntax.
Clojure (Experimental) - A modern Lisp.
Lua (Experimental) - Game scripting language.
Io (Experimental) - Simple but obscure.

PYTHON (DEFAULT)

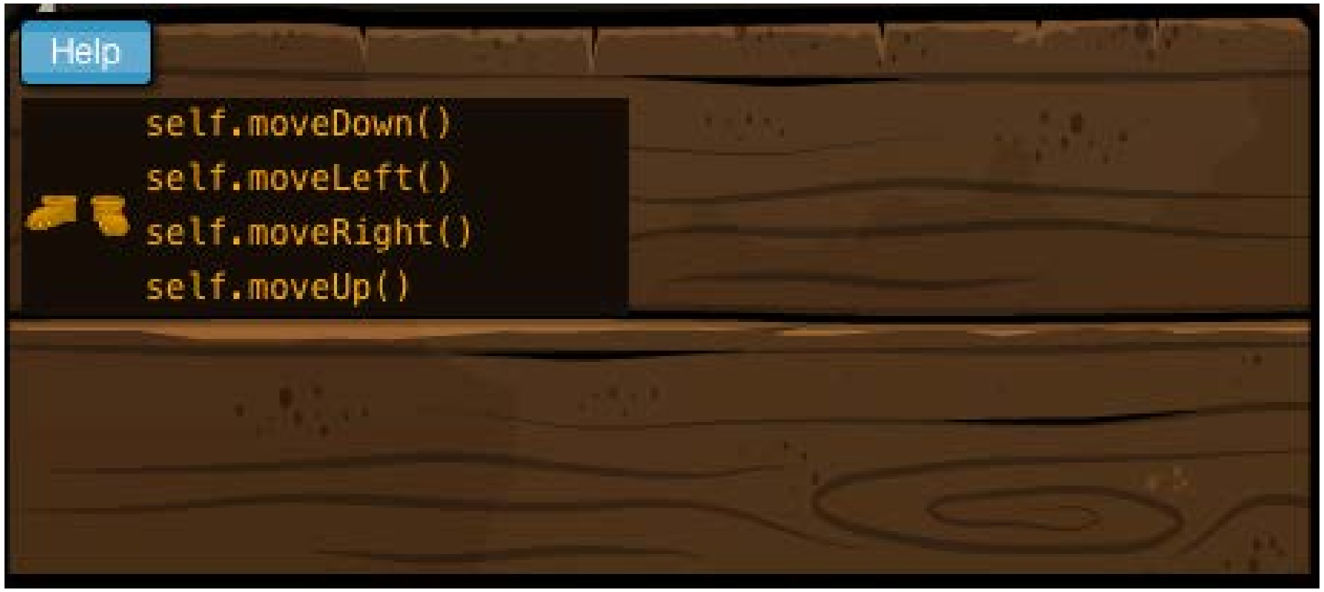
Select the first level, then select your character, being sure to chose Python as your language

Level One



Double click the boots to equip them. Then click play.

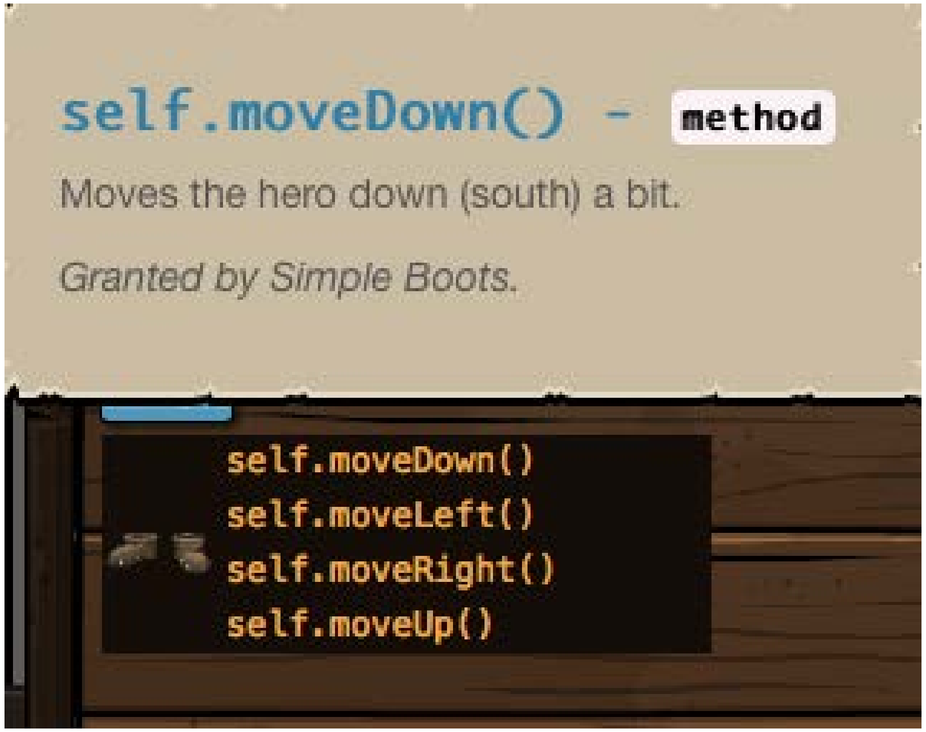
Level One



```
self.moveDown()
self.moveLeft()
self.moveRight()
self.moveUp()
```

In the bottom right there is an inventory. Look there are your boots!

Methods



```
self.moveDown() - method
```

Moves the hero down (south) a bit.

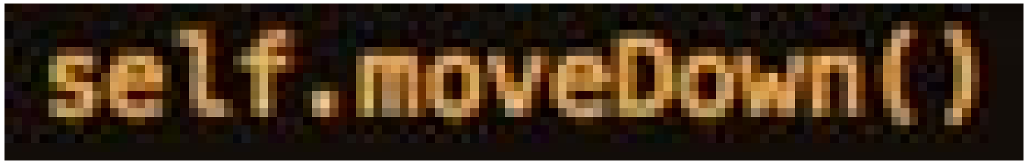
Granted by Simple Boots.

```
self.moveDown()
self.moveLeft()
self.moveRight()
self.moveUp()
```

Hover over one of the orange lines - these are **methods** - to learn a bit about what they do

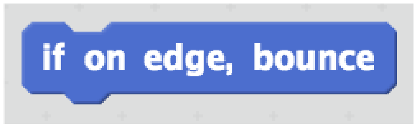
Methods

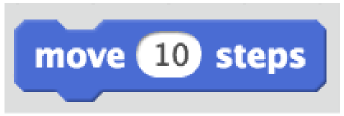
Methods are like functions, but associated with a specific object.



```
self.moveDown()
```

Review: Functions


if on edge, bounce


move 10 steps

Functions are named, reusable sequences of code.

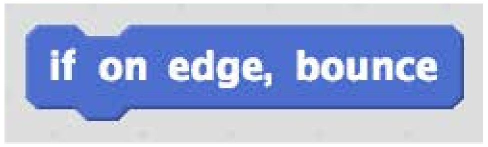
print ( )

input ( )

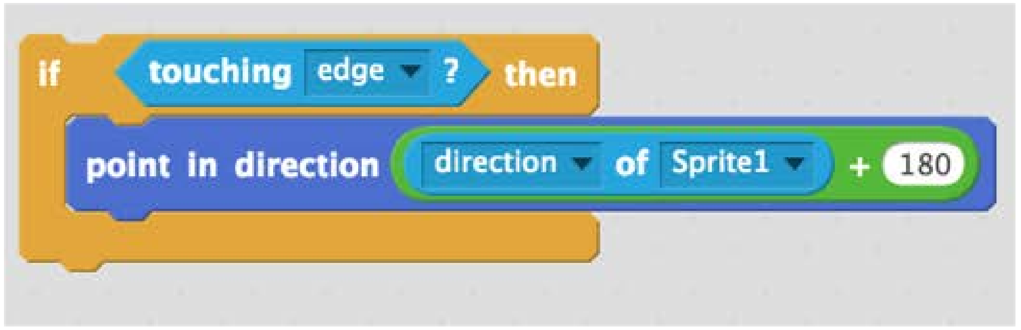
YOUNG engineers OF TODAY

Review: Functions

For example look at built in function block on the left from scratch



=

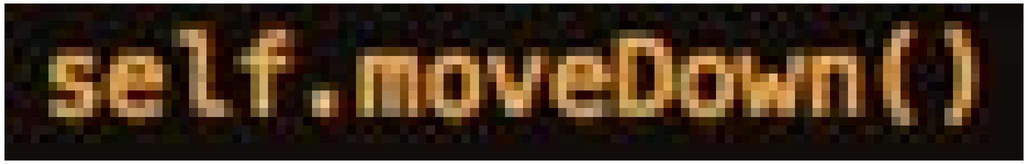


This “If on edge, bounce” block is equivalent to these 5 blocks.

Instead of using the five to the right, you can just use the one block

Methods

So if **methods** are associated with a specific object, what is the object for moveDown?



```
self.moveDown()
```

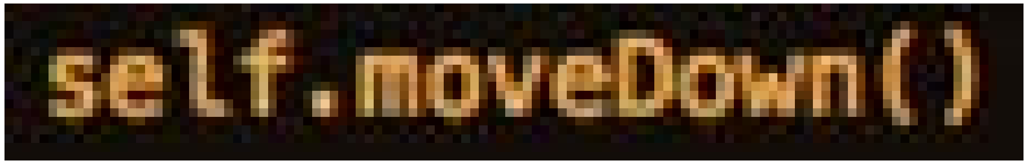
Object

Attached by a period

Method

(note parentheses)

Methods



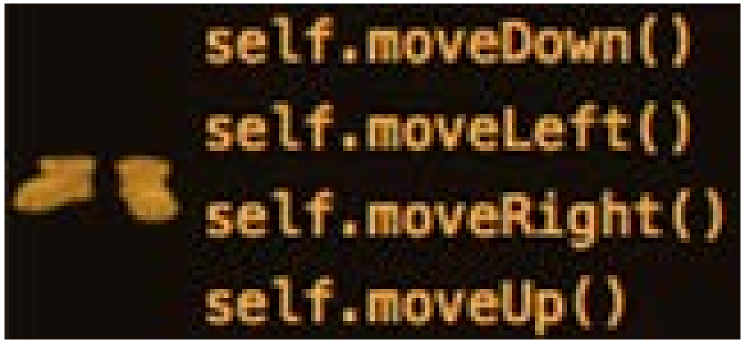
```
self.moveDown()
```

The object here is ‘self’ - which refers to your player.

Self is actually a **variable** that stores information about your player, like their location

Modules

In code combat, it's the boot that allows us to use these move methods.



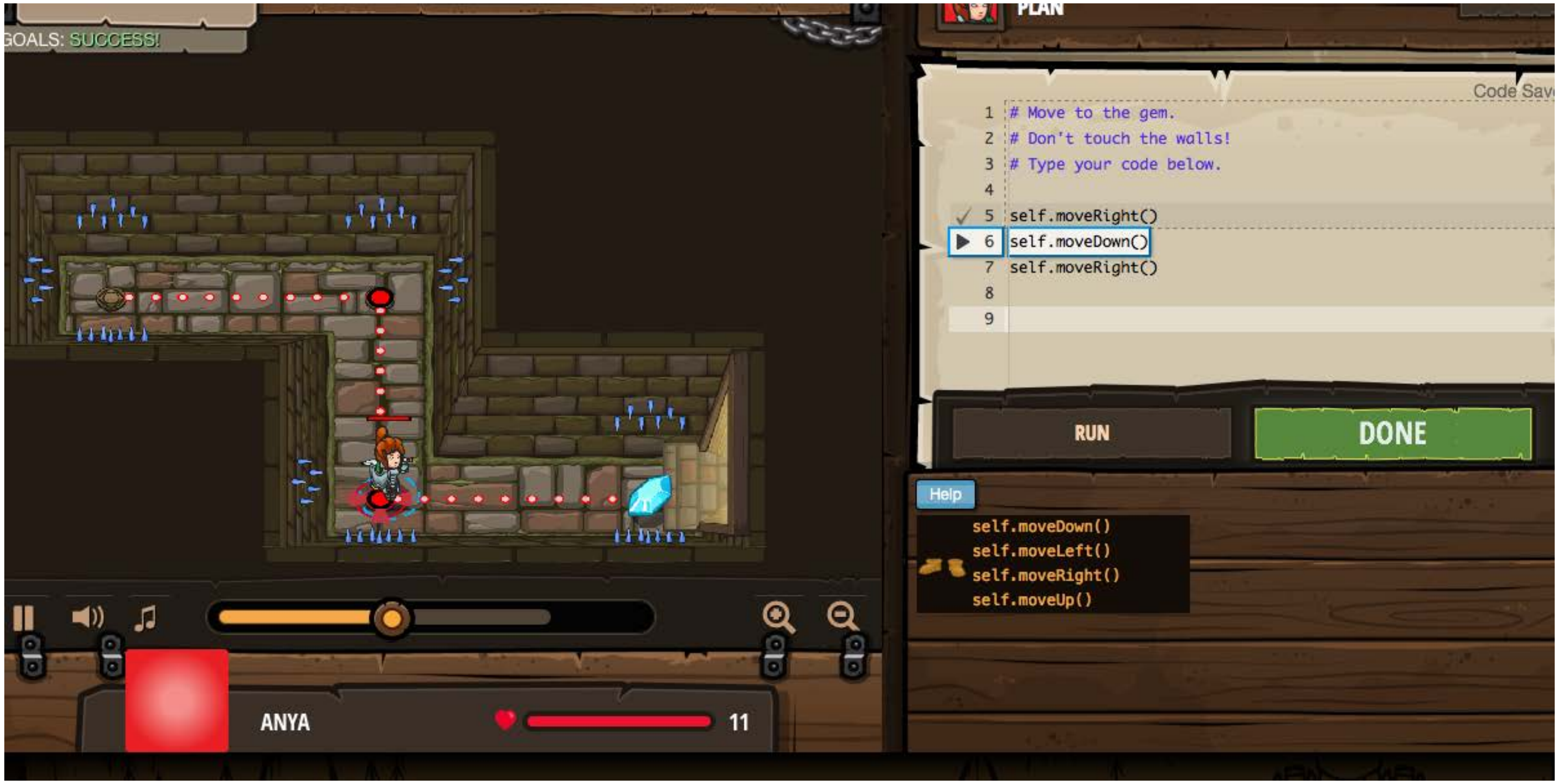
```
self.moveDown()
self.moveLeft()
self.moveRight()
self.moveUp()
```

As the game continues, you will get more armor/weapons etc. that will give you access to additional methods

Modules

In python you use import lines to import libraries or **modules**. These are bundles of functions for you to use.

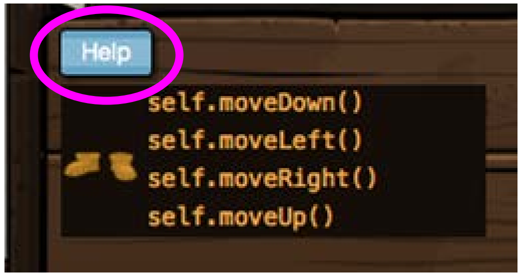
Level One: Dungeons of Kithgard



```
# Move to the gem.
# Don't touch the walls!
# Type your code below.

self.moveRight()
self.moveDown()
self.moveRight()
```

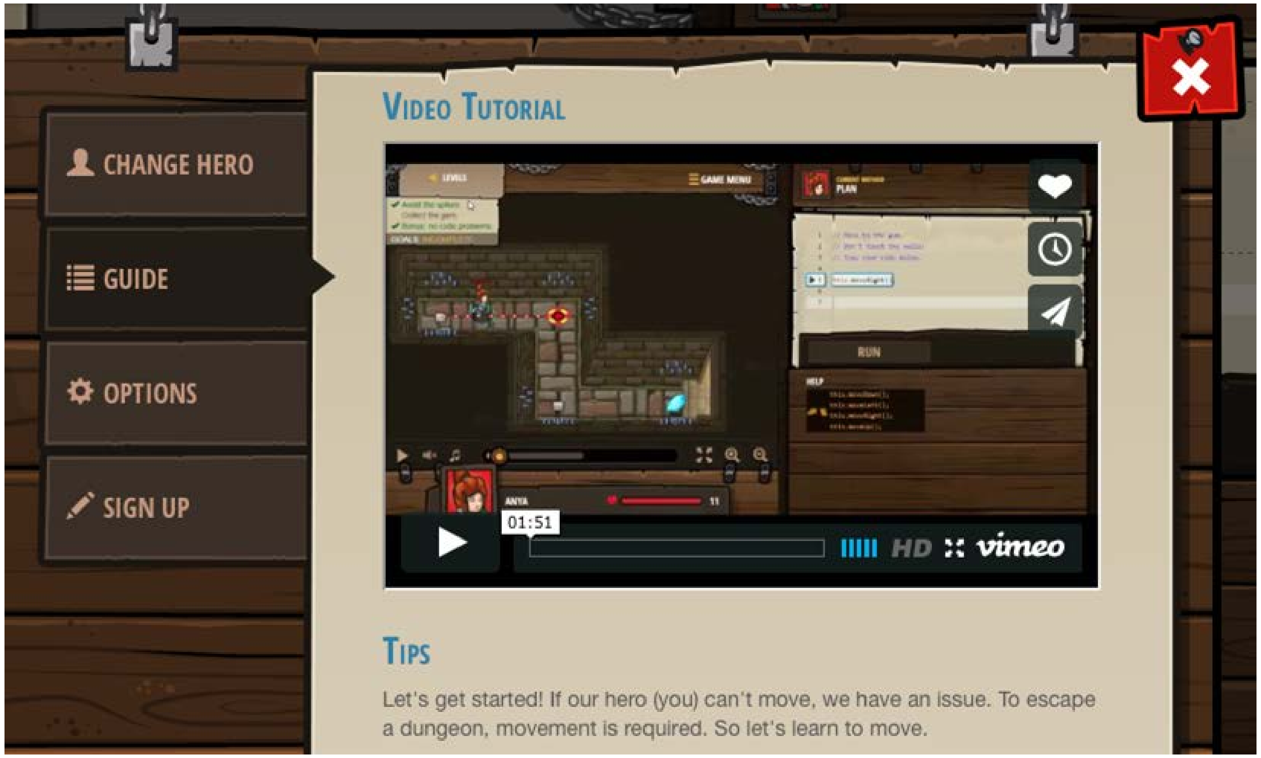
```
self.moveDown()
self.moveLeft()
self.moveRight()
self.moveUp()
```

Help



Help

```
self.moveDown()
self.moveLeft()
self.moveRight()
self.moveUp()
```

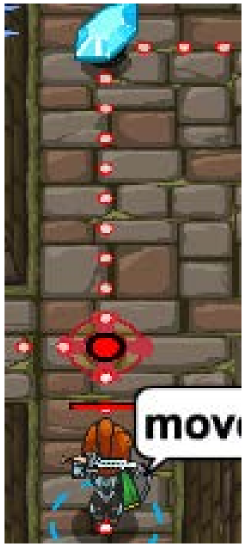


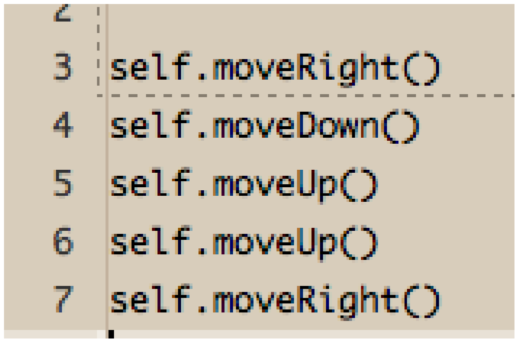
TIPS

Let's get started! If our hero (you) can't move, we have an issue. To escape a dungeon, movement is required. So let's learn to move.

In case you get completely stuck, you can always use the help button

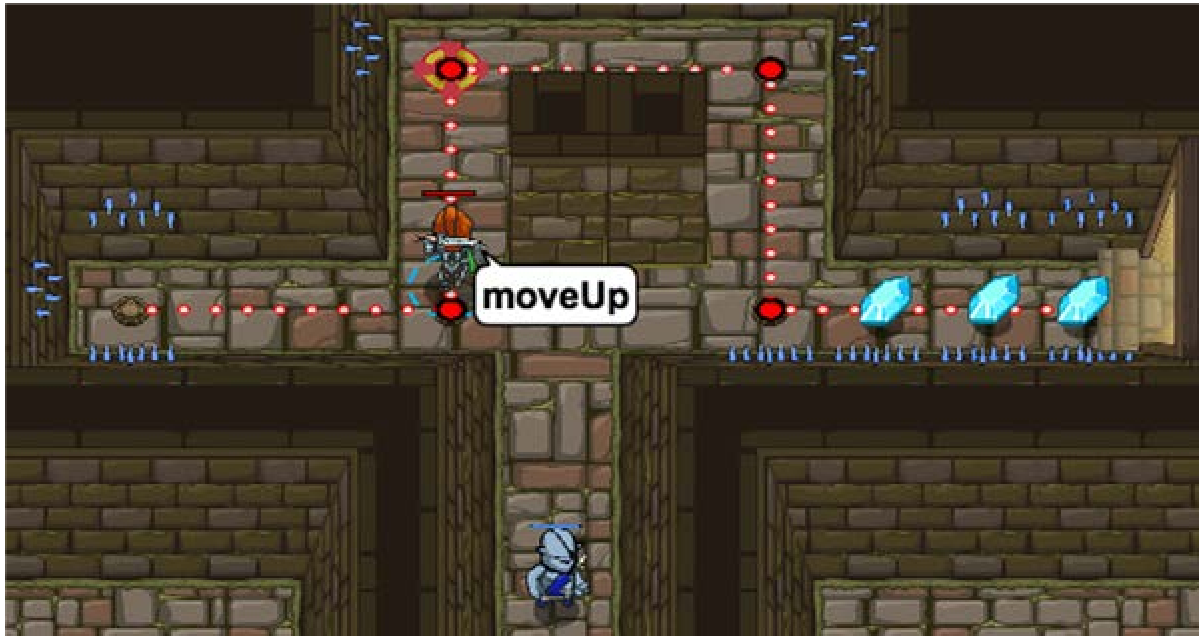
Level Two: Gems in the Deep

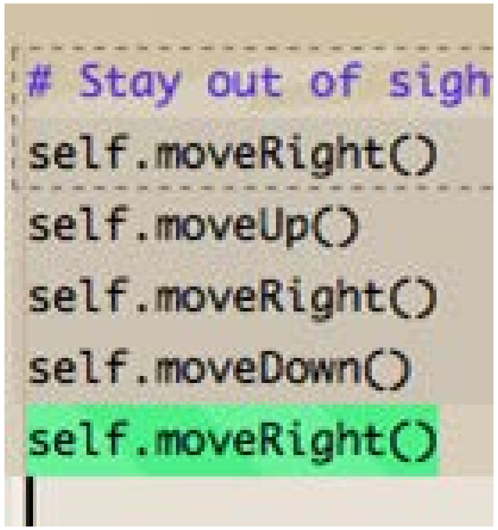



```
self.moveRight()
self.moveDown()
self.moveUp()
self.moveUp()
self.moveRight()
```

Note: you must moveUp twice - first to the red circle, then to the top gem

Level Three: Shadow Guard

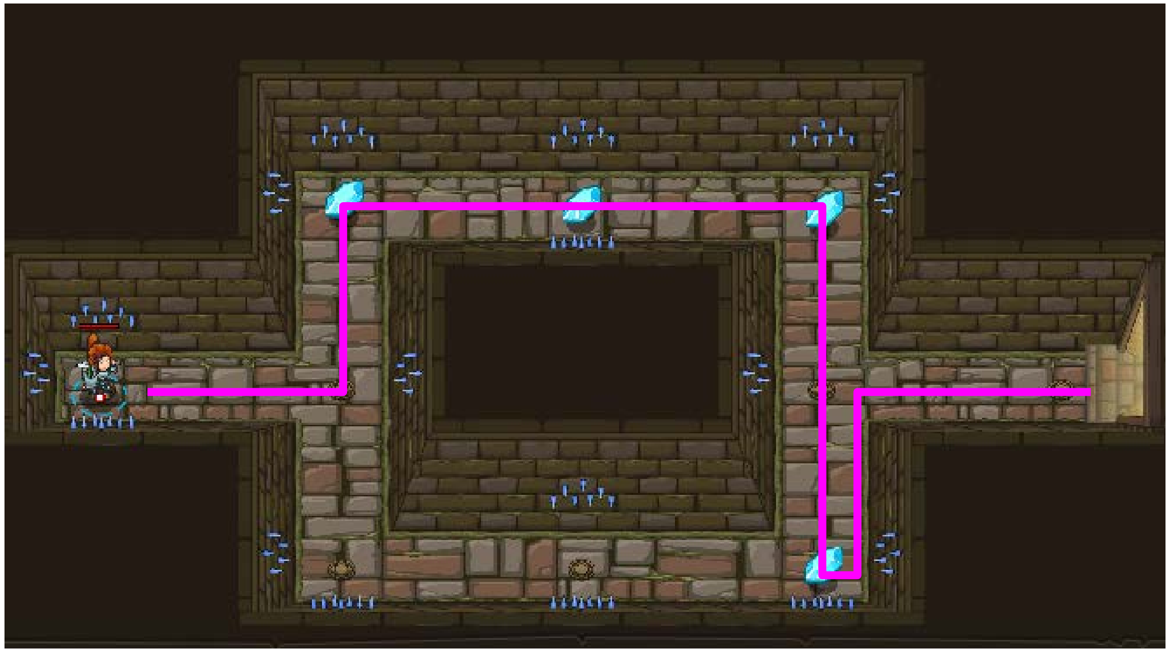


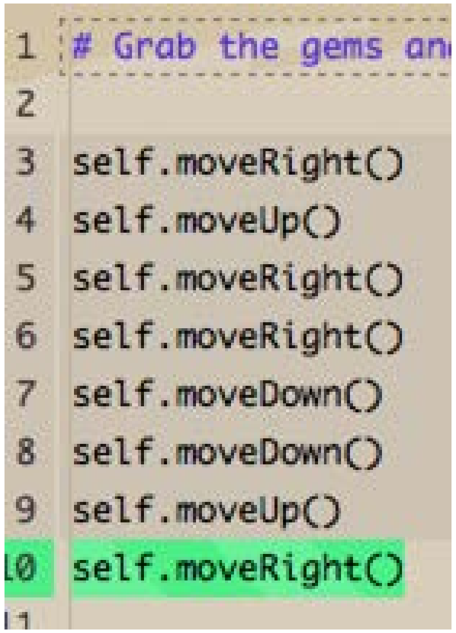


```
# Stay out of sigh
self.moveRight()
self.moveUp()
self.moveRight()
self.moveDown()
self.moveRight()
```

Note: Sometimes it’s not the easiest route that works...

Level Four: Forgetful Gemsmith

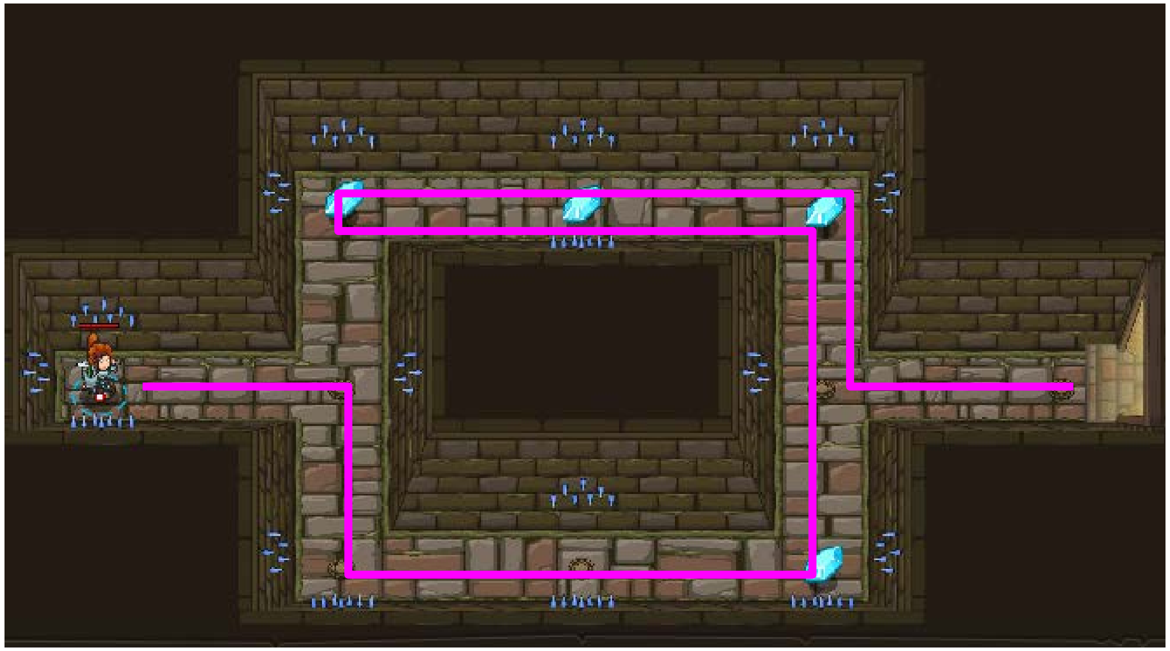




```
# Grab the gems an

self.moveRight()
self.moveUp()
self.moveRight()
self.moveRight()
self.moveDown()
self.moveDown()
self.moveUp()
self.moveRight()
```

Why would taking the lower route first not work?

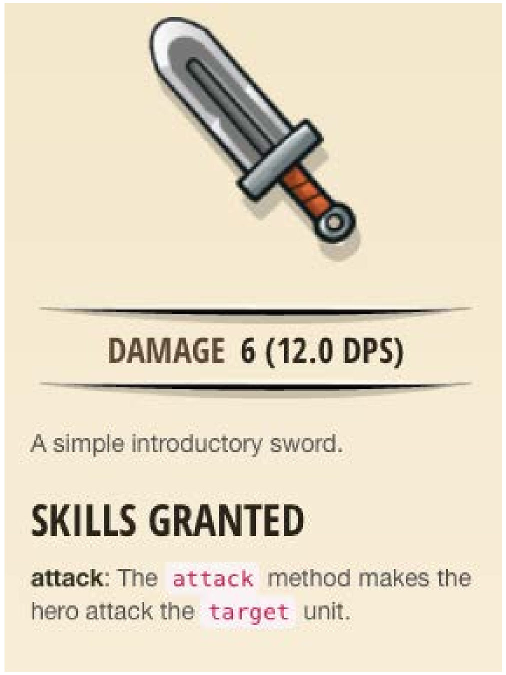
Level Four: Forgetful Gemsmith



Note the difference in length

Level Five: True Names

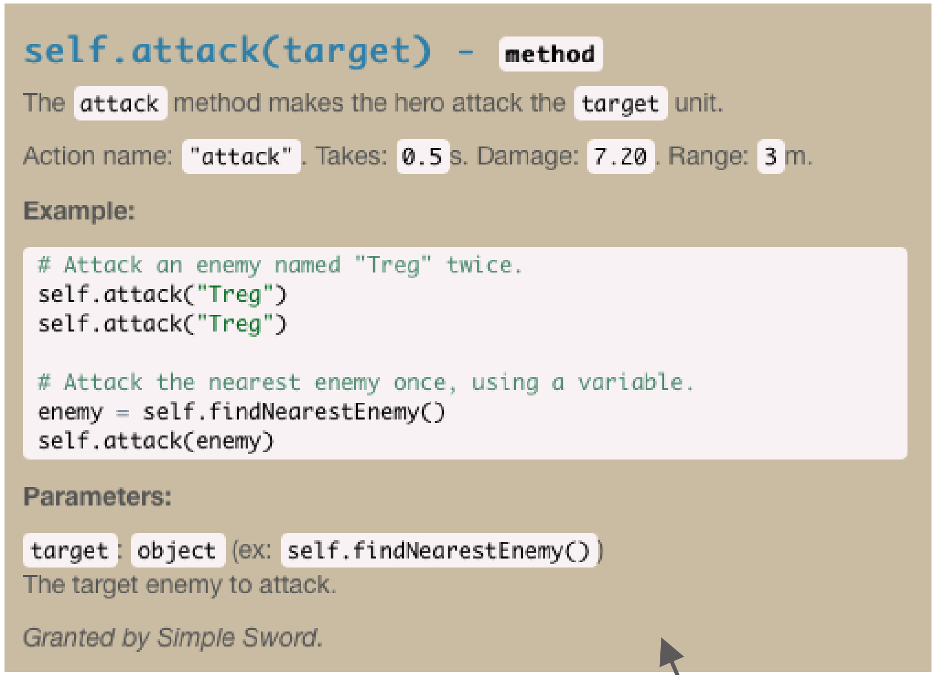
New sword means new method!



A simple introductory sword.

SKILLS GRANTED

attack: The `attack` method makes the hero attack the `target` unit.



`self.attack(target)` - **method**

The `attack` method makes the hero attack the `target` unit.

Action name: `"attack"`. Takes: `0.5` s. Damage: `7.20`. Range: `3` m.

Example:

```
# Attack an enemy named "Treg" twice.
self.attack("Treg")
self.attack("Treg")

# Attack the nearest enemy once, using a variable.
enemy = self.findNearestEnemy()
self.attack(enemy)
```

Parameters:

`target`: `object` (ex: `self.findNearestEnemy()`)
The target enemy to attack.

Granted by Simple Sword.

Hover over sword to reveal information:

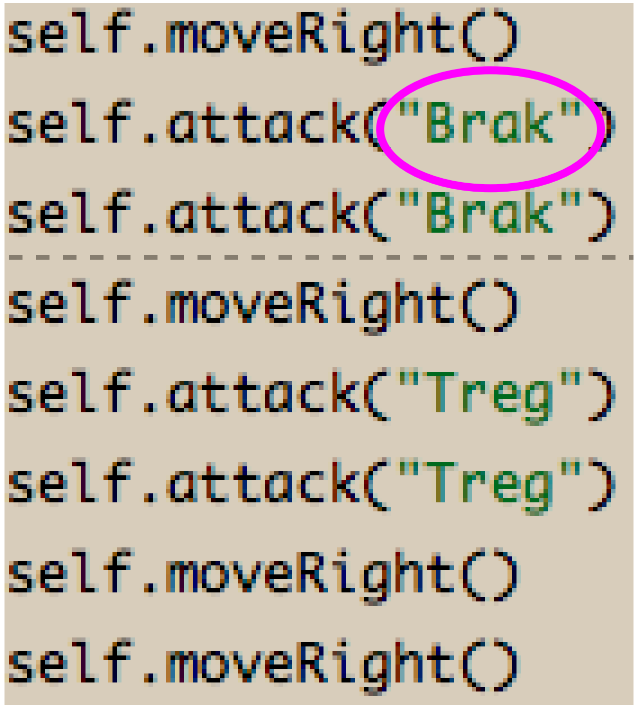
Data Types

Computers understand different kinds of values, for example numbers (integers/floating point numbers) to characters (strings)

Data Types

1. Integers	4
2. Floating Point Numbers	4.0
3. Boolean	True
4. Strings	'four'

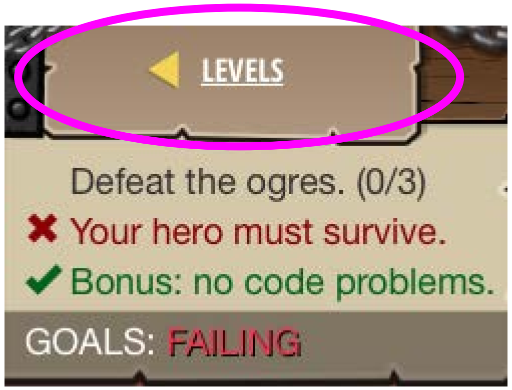
Level Five: True Names



```
self.moveRight()
self.attack("Brak")
self.attack("Brak")
-----------------------------
self.moveRight()
self.attack("Treg")
self.attack("Treg")
self.moveRight()
self.moveRight()
```

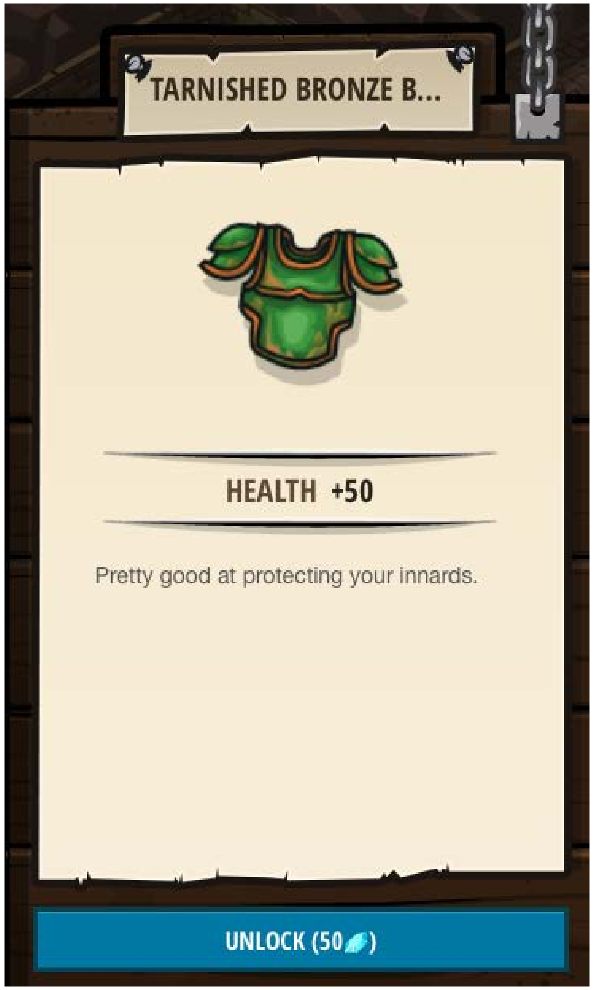
Note: the names are strings

Level Six: The Raised Sword



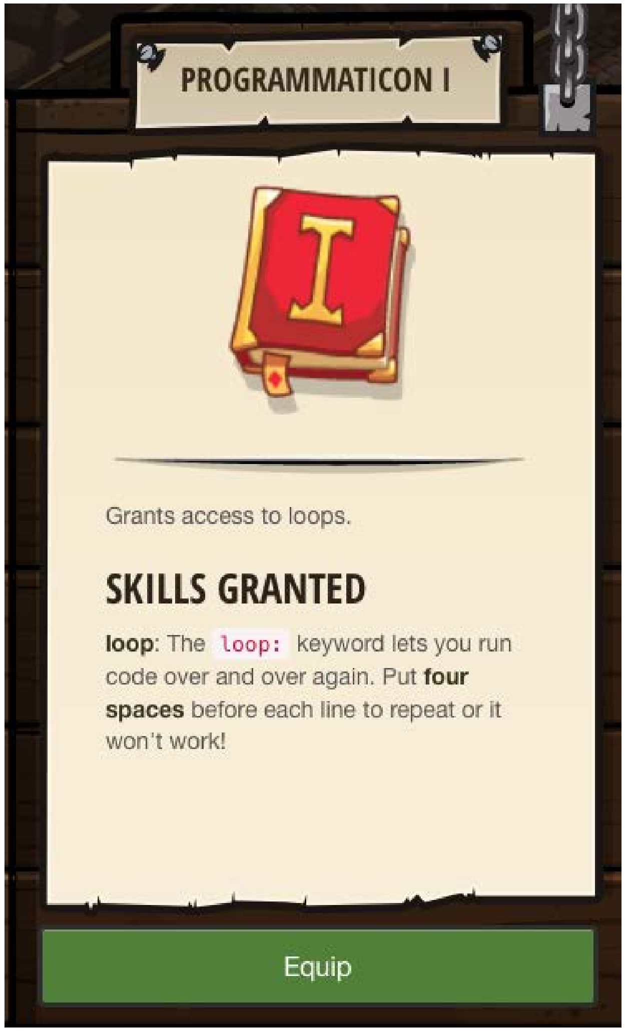
Defeat the ogres. (0/3)
✗ Your hero must survive.
✓ Bonus: no code problems.
GOALS: FAILING

Can’t get past this level, despite correct code? Go back and get some armor!



Level Seven: Haunted Kithmaze

New book! We can now access looping

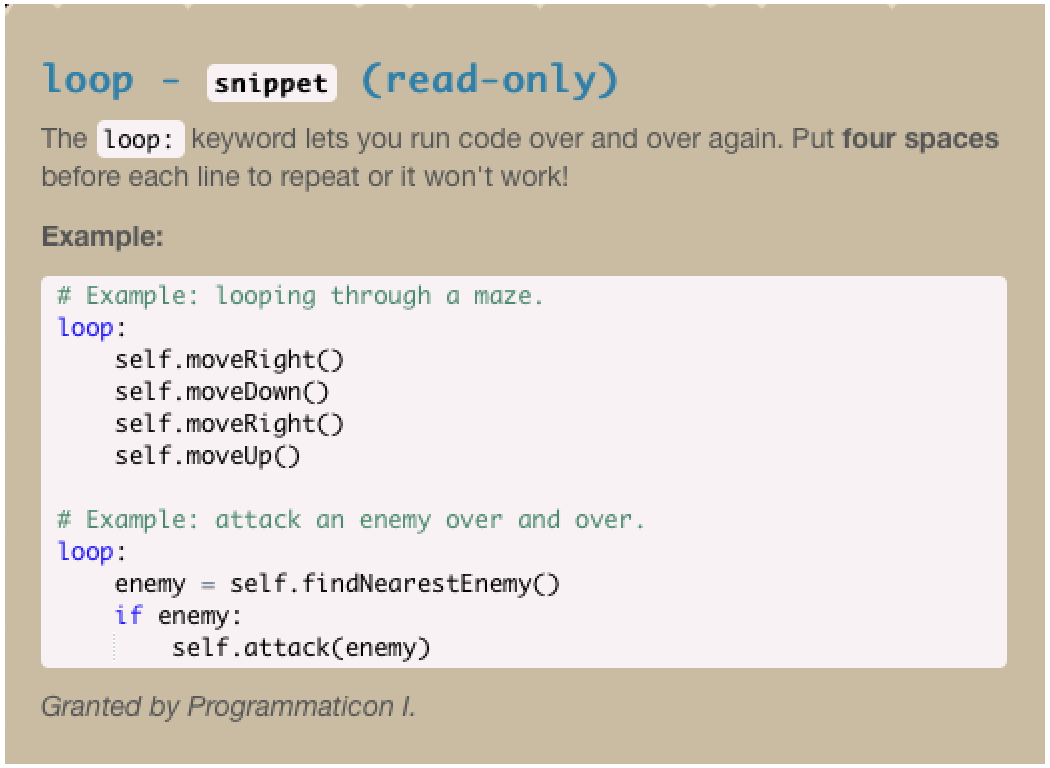


Grants access to loops.

SKILLS GRANTED

loop: The `loop:` keyword lets you run code over and over again. Put **four spaces** before each line to repeat or it won't work!

Level Seven: Haunt Kithmaze


loop - snippet (read-only)

The `loop:` keyword lets you run code over and over again. Put **four spaces** before each line to repeat or it won't work!

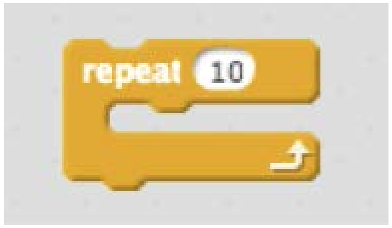
Example:

```
# Example: looping through a maze.
loop:
    self.moveRight()
    self.moveDown()
    self.moveRight()
    self.moveUp()

# Example: attack an enemy over and over.
loop:
    enemy = self.findNearestEnemy()
    if enemy:
        self.attack(enemy)
```

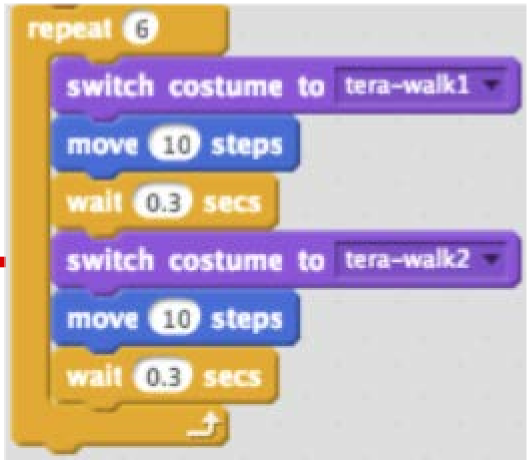
Granted by Programmaticon I.

Hover over book to learn more

Loops



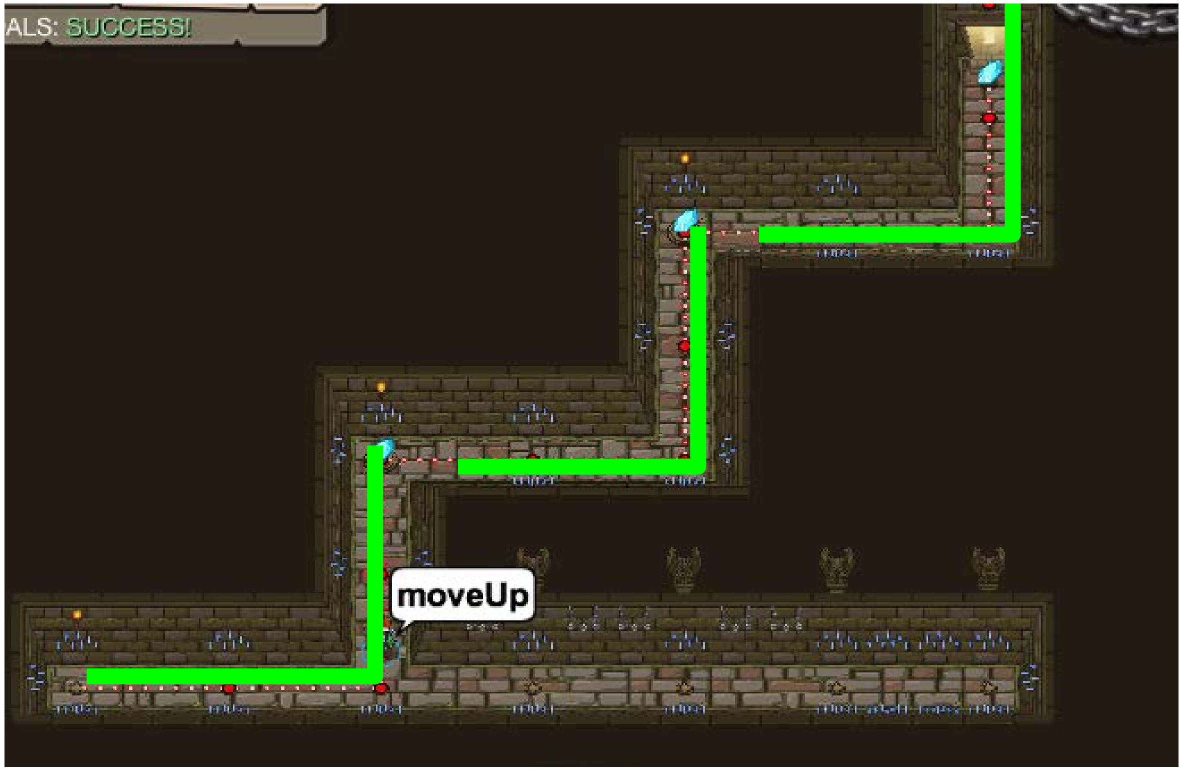
```
repeat 10
```

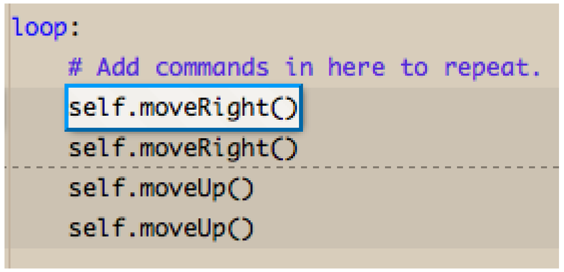


```
repeat 6
    switch costume to tera-walk1
    move 10 steps
    wait 0.3 secs
    switch costume to tera-walk2
    move 10 steps
    wait 0.3 secs
```

Loops are used to repeat a sequence of code, either a set number of times, forever, or until a condition is False

Level Seven: Haunt Kithmaze

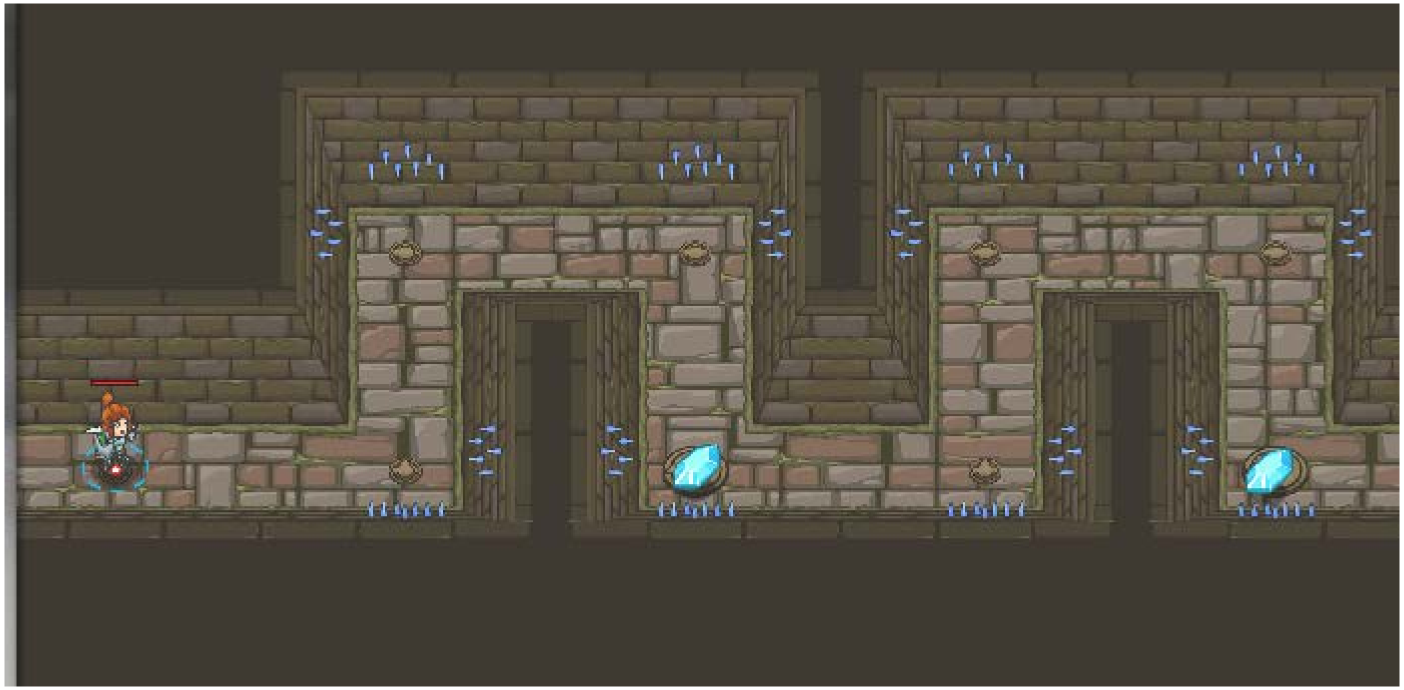




```
loop:
    # Add commands in here to repeat.
    self.moveRight()
    self.moveRight()
    self.moveUp()
    self.moveUp()
```

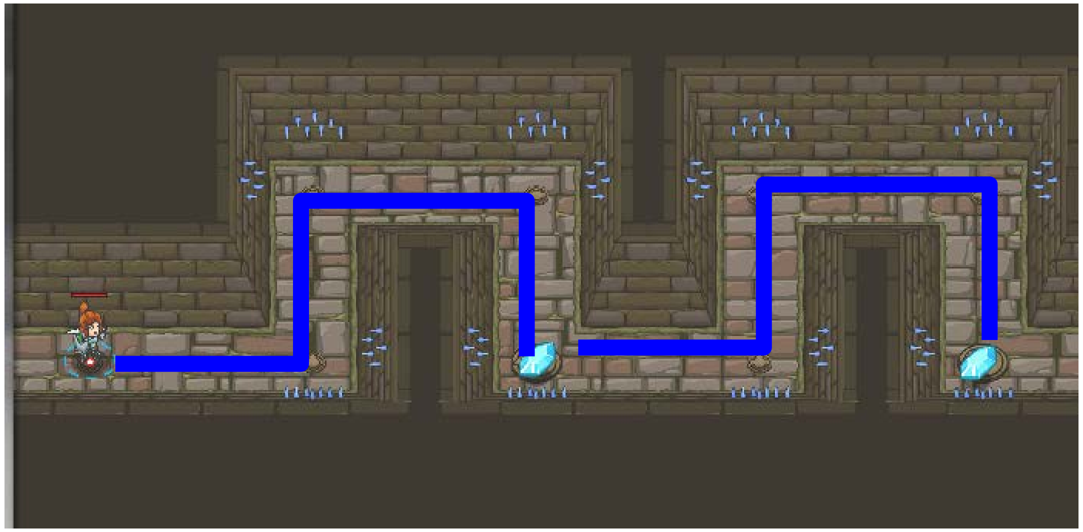
Loop allows the same lines
of code to be repeated

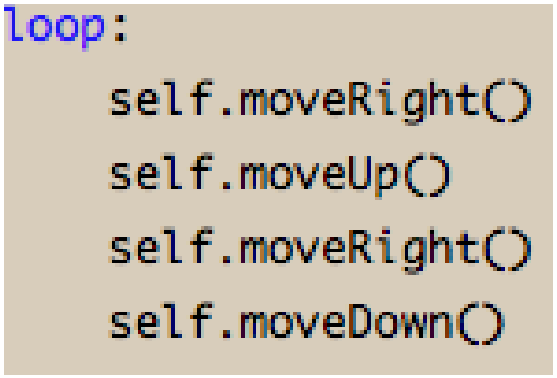
Level Eight: Second Kithmaze



Can you find the pattern?

Level Eight: Second Kithmaze

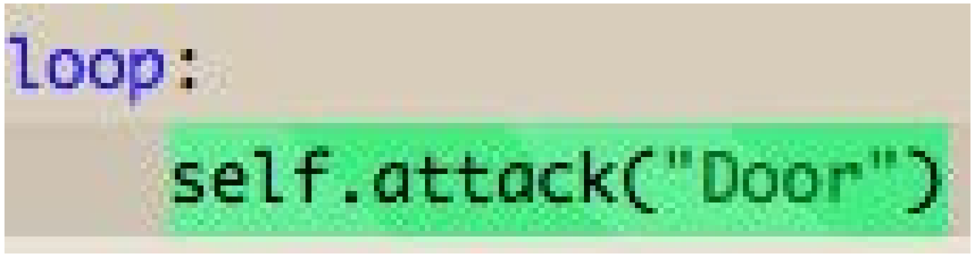




```
loop:
    self.moveRight()
    self.moveUp()
    self.moveRight()
    self.moveDown()
```

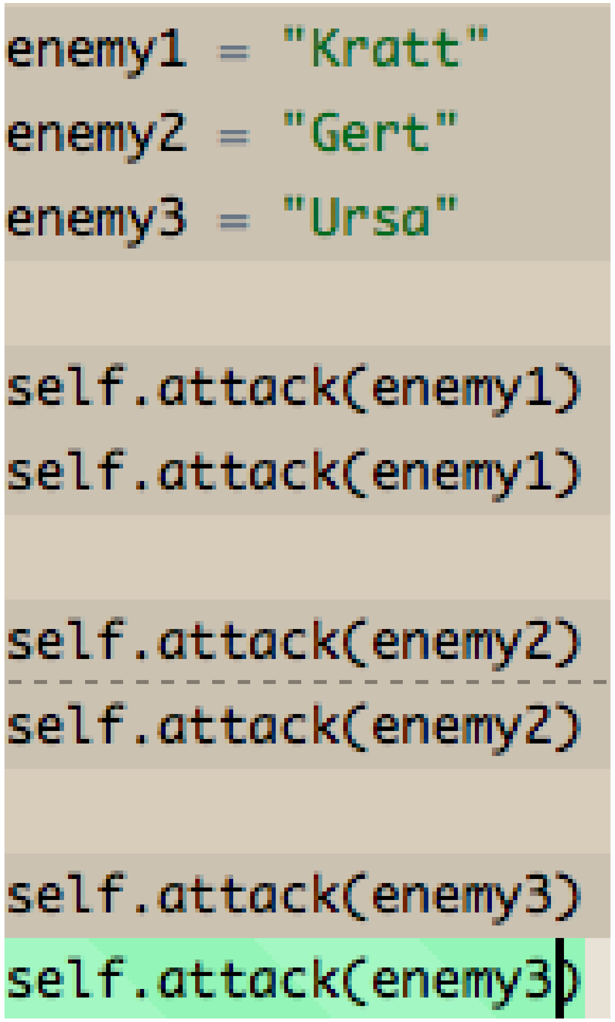
Make sure you indent
the move lines

Level Nine: Dread Door



```
loop:
    self.attack("Door")
```

Level Ten: Known Enemy



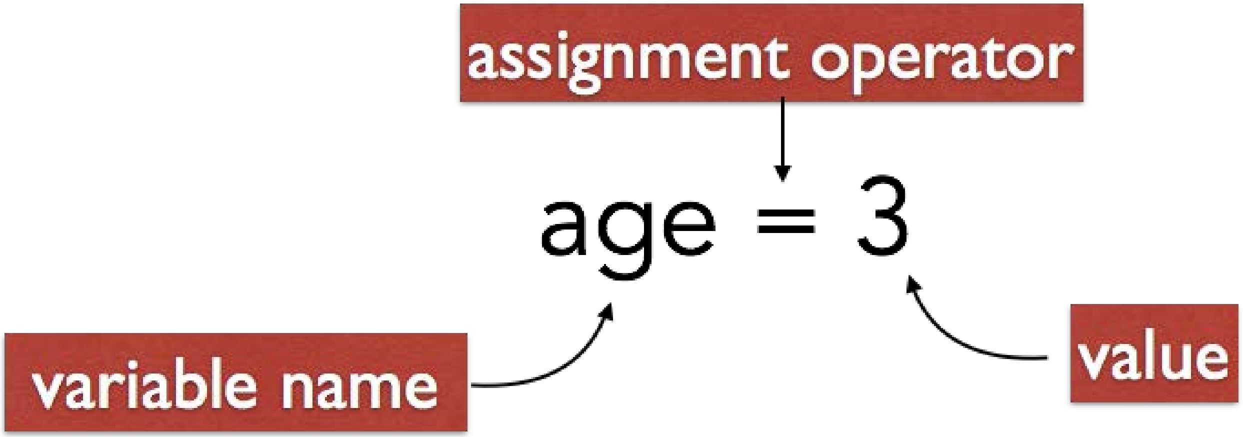
```
enemy1 = "Kratt"
enemy2 = "Gert"
enemy3 = "Ursa"

self.attack(enemy1)
self.attack(enemy1)

self.attack(enemy2)
self.attack(enemy2)

self.attack(enemy3)
self.attack(enemy3)
```

Why is “Kratt” in quotations but enemy1 is not?

Variables

A **variable** is a named container in which an item of data can be stored, much like a real-life object can be stored in a box.

Variable assignment



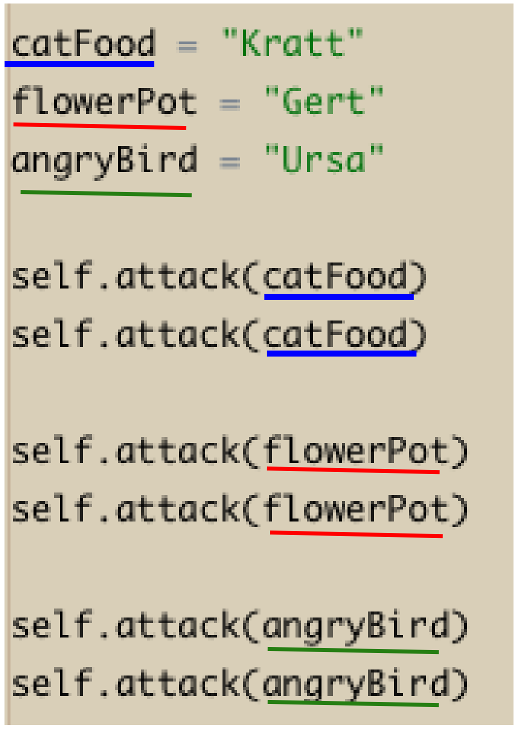
```
age = 3
```

```
name = "Dubick"
```

In line 1 we store the integer value 3 to the variable age.

In line 2 we store the string value “Dubick” to the variable name.

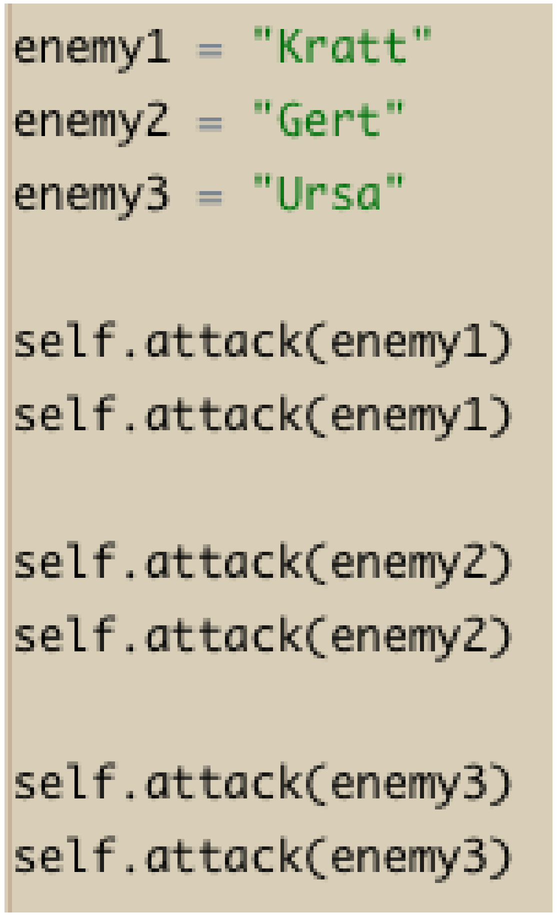
Variable assignment



```
catFood = "Kratt"
flowerPot = "Gert"
angryBird = "Ursa"

self.attack(catFood)
self.attack(catFood)

self.attack(flowerPot)
self.attack(flowerPot)

self.attack(angryBird)
self.attack(angryBird)
```

It is important to note that you *can* name your variable (almost) anything

Variable assignment



```
enemy1 = "Kratt"
enemy2 = "Gert"
enemy3 = "Ursa"

self.attack(enemy1)
self.attack(enemy1)

self.attack(enemy2)
self.attack(enemy2)

self.attack(enemy3)
self.attack(enemy3)
```

But just because you can doesn't mean you should!

Try and name variables so that their purpose is clear. This will save a lot of headaches in the future.

Level Nine: Dread Door

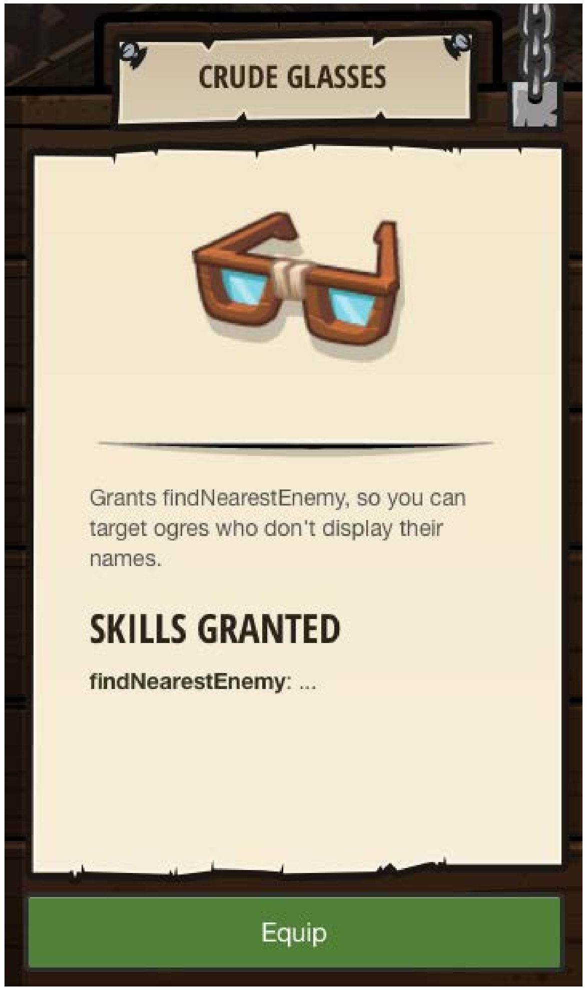
So why is this helpful? Isn’t it more lines of code?

Level Nine: Dread Door

Variables are useful because they are a way to write code for a changing or yet unknown value

We will see why this is useful next level

Level 11: Master of Names


CRUDE GLASSES

Grants findNearestEnemy, so you can target ogres who don't display their names.

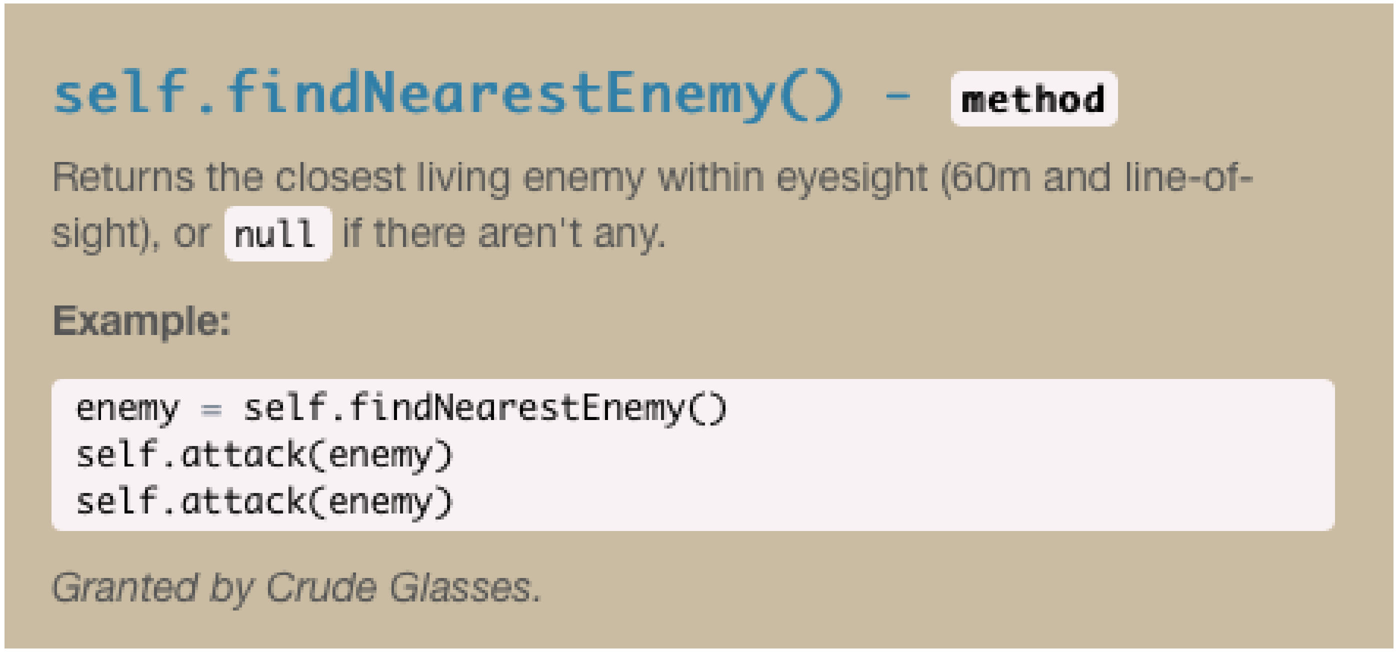
SKILLS GRANTED

findNearestEnemy: ...

New method!

Level 11: Master of Names

self.findNearestEnemy…


self.findNearestEnemy() - method

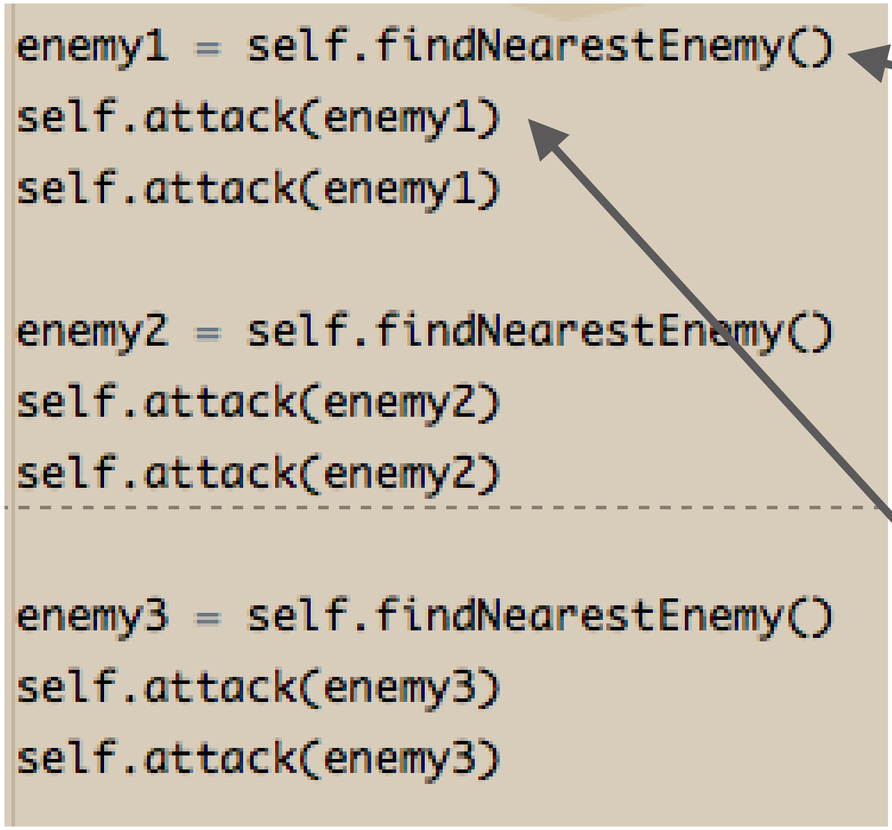
Returns the closest living enemy within eyesight (60m and line-of-sight), or `null` if there aren't any.

Example:

```
enemy = self.findNearestEnemy()
self.attack(enemy)
self.attack(enemy)
```

Granted by Crude Glasses.

Level 11: Master of Names

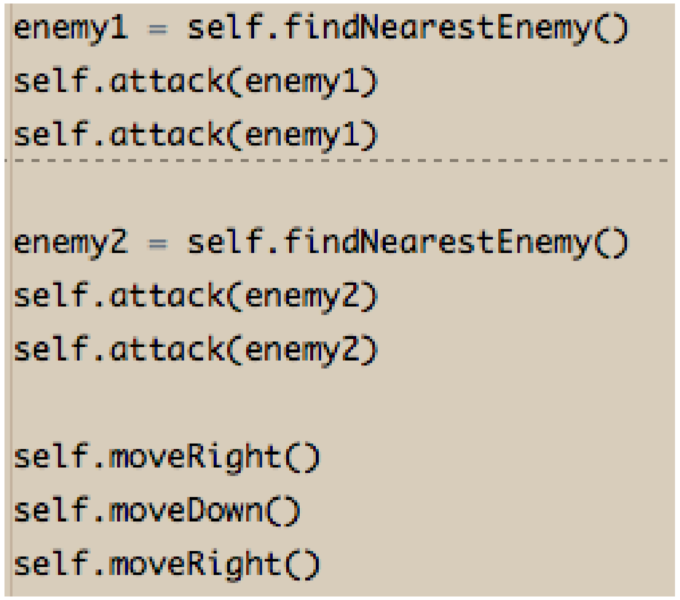


```
enemy1 = self.findNearestEnemy()
self.attack(enemy1)
self.attack(enemy1)

enemy2 = self.findNearestEnemy()
self.attack(enemy2)
self.attack(enemy2)

enemy3 = self.findNearestEnemy()
self.attack(enemy3)
self.attack(enemy3)
```

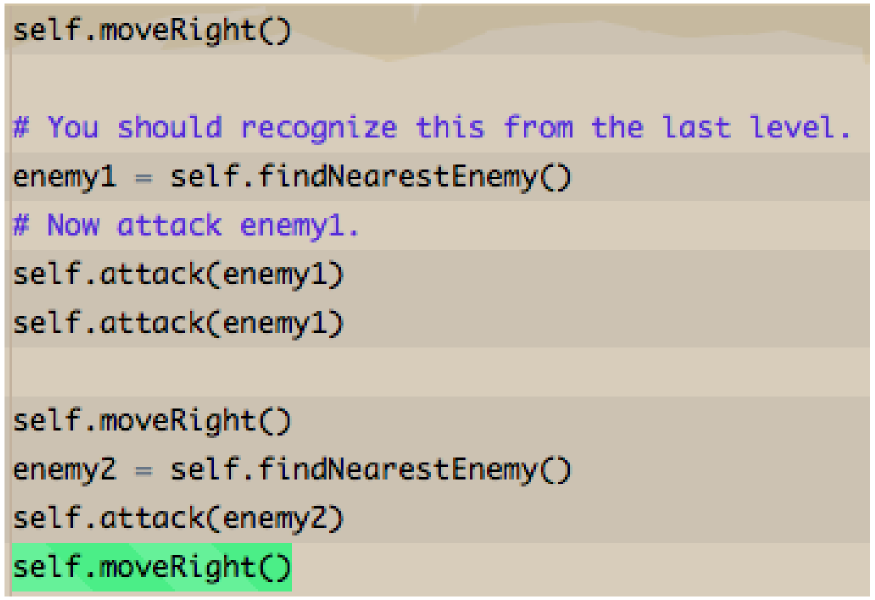
This line locates the enemy nearest to your player and saves their name as a string to the variable enemy 1

This line then attacks that enemy.

12 & 13: Lowly Kithman, Closing the Distance



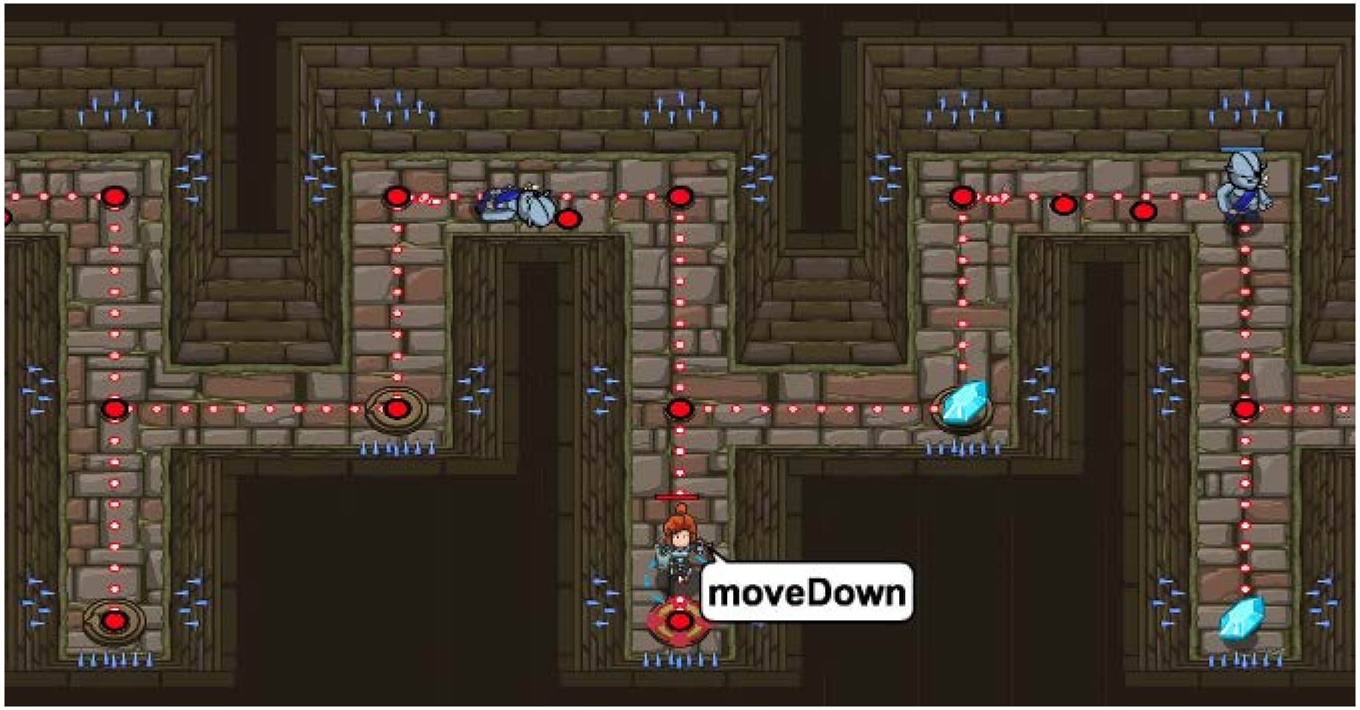
```
enemy1 = self.findNearestEnemy()
self.attack(enemy1)
self.attack(enemy1)

enemy2 = self.findNearestEnemy()
self.attack(enemy2)
self.attack(enemy2)

self.moveRight()
self.moveDown()
self.moveRight()
```



```
self.moveRight()

# You should recognize this from the last level.
enemy1 = self.findNearestEnemy()
# Now attack enemy1.
self.attack(enemy1)
self.attack(enemy1)

self.moveRight()
enemy2 = self.findNearestEnemy()
self.attack(enemy2)
self.moveRight()
```

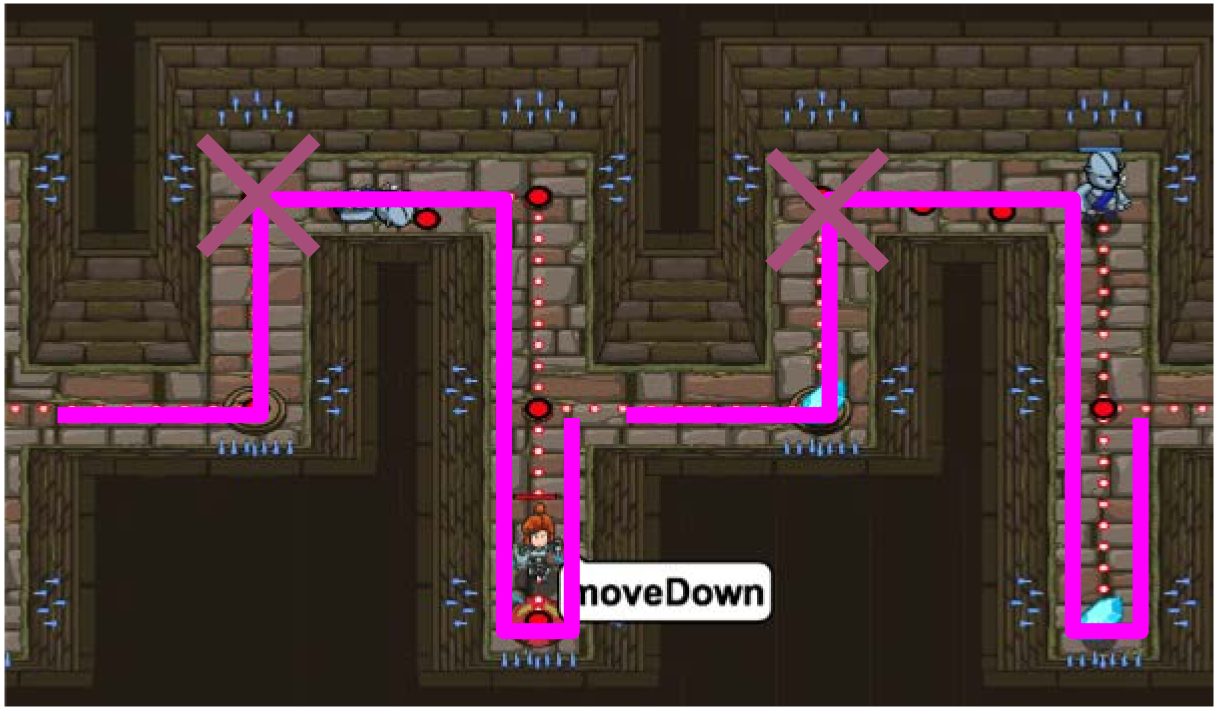
In case you get stuck...

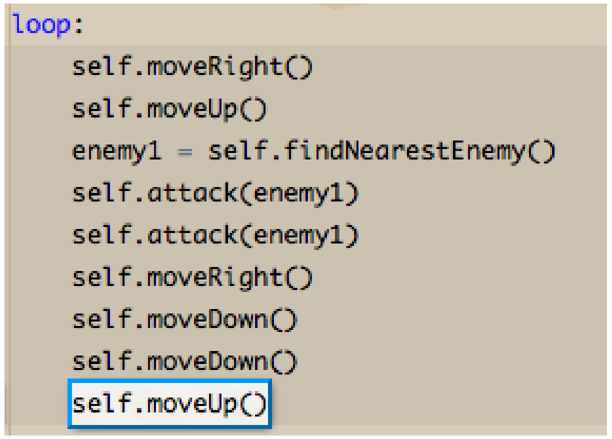
Level 14: The Final Kithmaze



Can you spot the pattern?

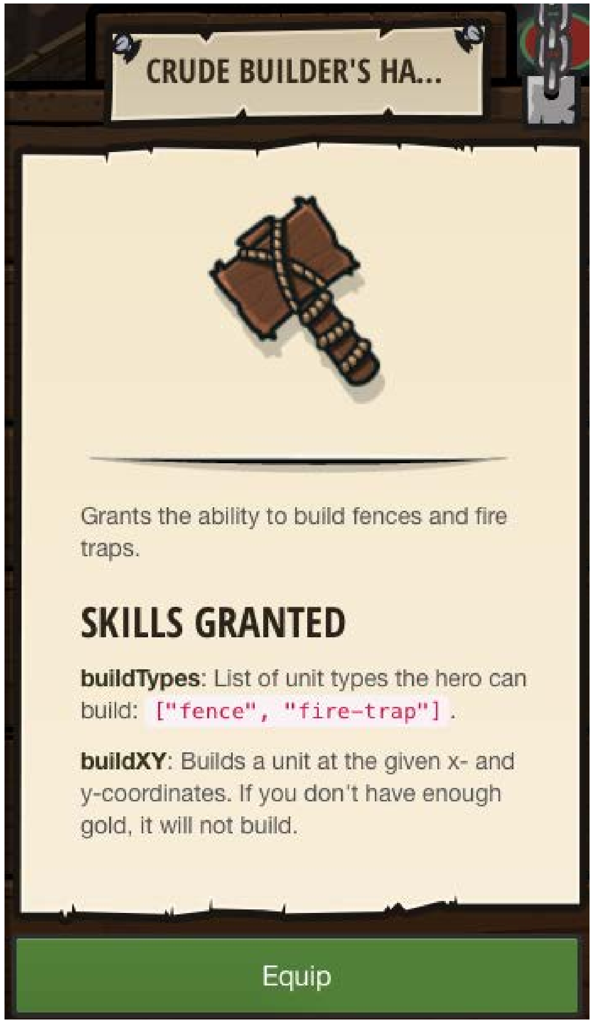
Level 14: The Final Kithmaze





```
loop:
    self.moveRight()
    self.moveUp()
    enemy1 = self.findNearestEnemy()
    self.attack(enemy1)
    self.attack(enemy1)
    self.moveRight()
    self.moveDown()
    self.moveDown()
    self.moveUp()
```

Level 15: The Final Kithmaze



CRUDE BUILDER'S HA...

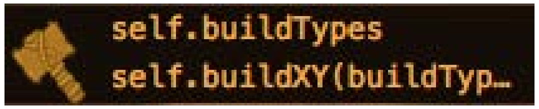
Grants the ability to build fences and fire traps.

SKILLS GRANTED

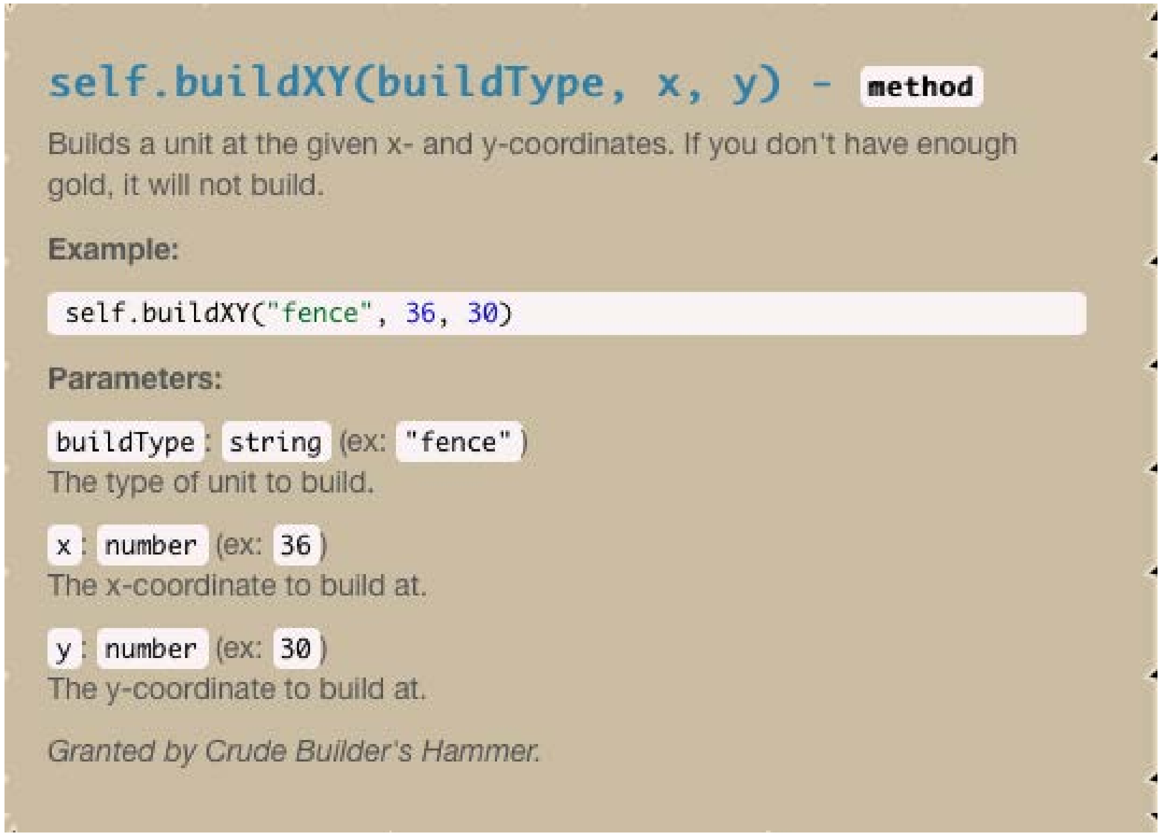
buildTypes: List of unit types the hero can build: `["fence", "fire-trap"]`.

buildXY: Builds a unit at the given x- and y-coordinates. If you don't have enough gold, it will not build.

Level 15: The Final Kithmaze



```
self.buildTypes
self.buildXY(buildTyp…
```



`self.buildXY(buildType, x, y)` - **method**

Builds a unit at the given x- and y-coordinates. If you don't have enough gold, it will not build.

Example:

```
self.buildXY("fence", 36, 30)
```

Parameters:

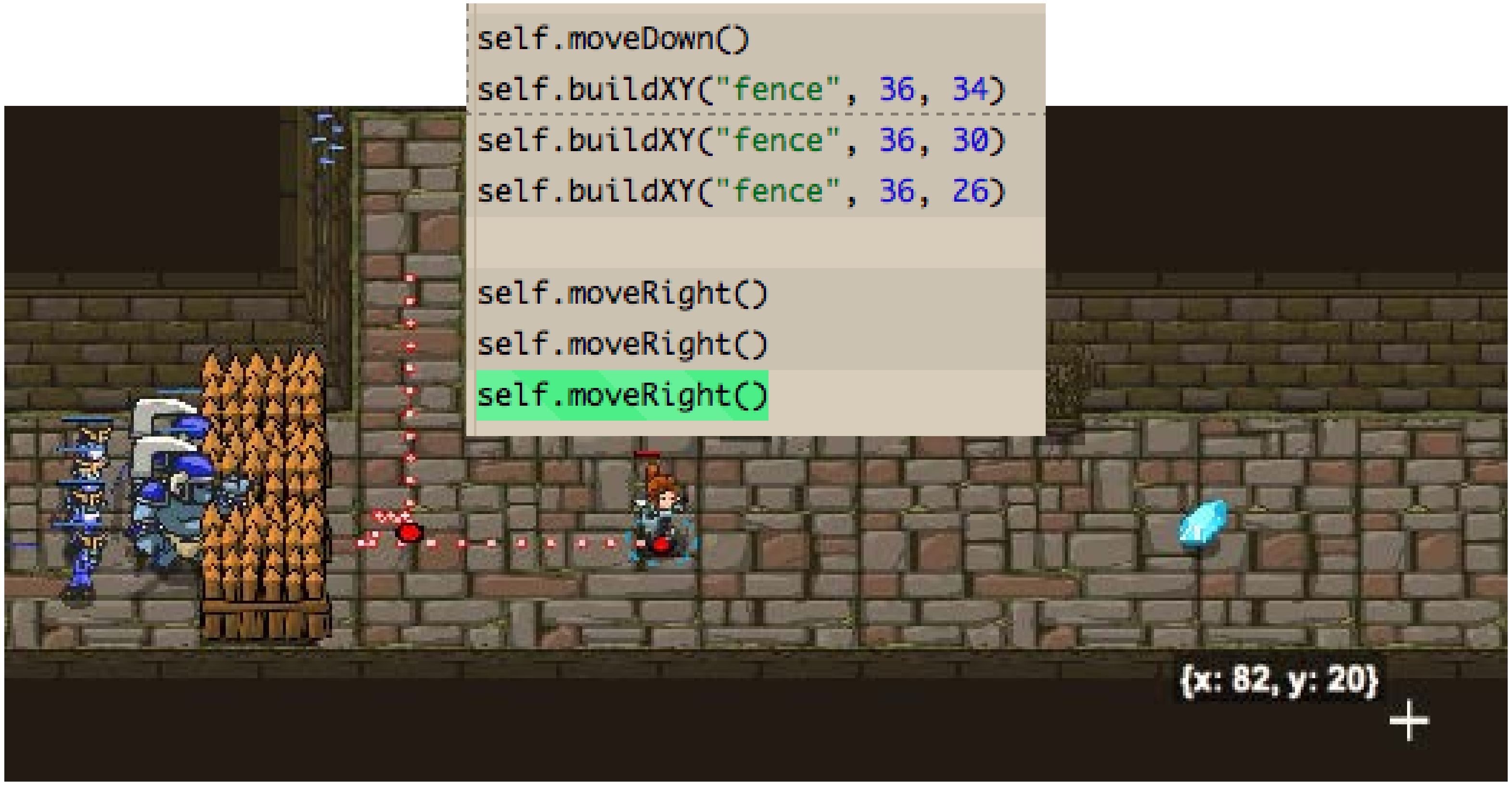
`buildType` : `string` (ex: `"fence"`)
The type of unit to build.

`x` : `number` (ex: `36`)
The x-coordinate to build at.

`y` : `number` (ex: `30`)
The y-coordinate to build at.

Granted by Crude Builder's Hammer.

Level 15: The Final Kithmaze



```
self.moveDown()
self.buildXY("fence", 36, 34)
self.buildXY("fence", 36, 30)
self.buildXY("fence", 36, 26)

self.moveRight()
self.moveRight()
self.moveRight()
```

Onto the next world in Lesson 2!