Welcome to GPN

Tech
Inclusion

Thank you to our Sponsors!

Platinum Sponsor:

Tech
Inclusion

Who are the tutors?

Tech
Inclusion

Who are you?

Tech
Inclusion

Log on

Log on and jump on the GPN website

girlsprogramming.network/workshop

You can see:

- These **slides** (to take a look back or go on ahead).
- A digital copy of your **workbook**.
- Help bits of text you can **copy and paste**!

There's also links to places where you can do more programming!

Tell us you're here!

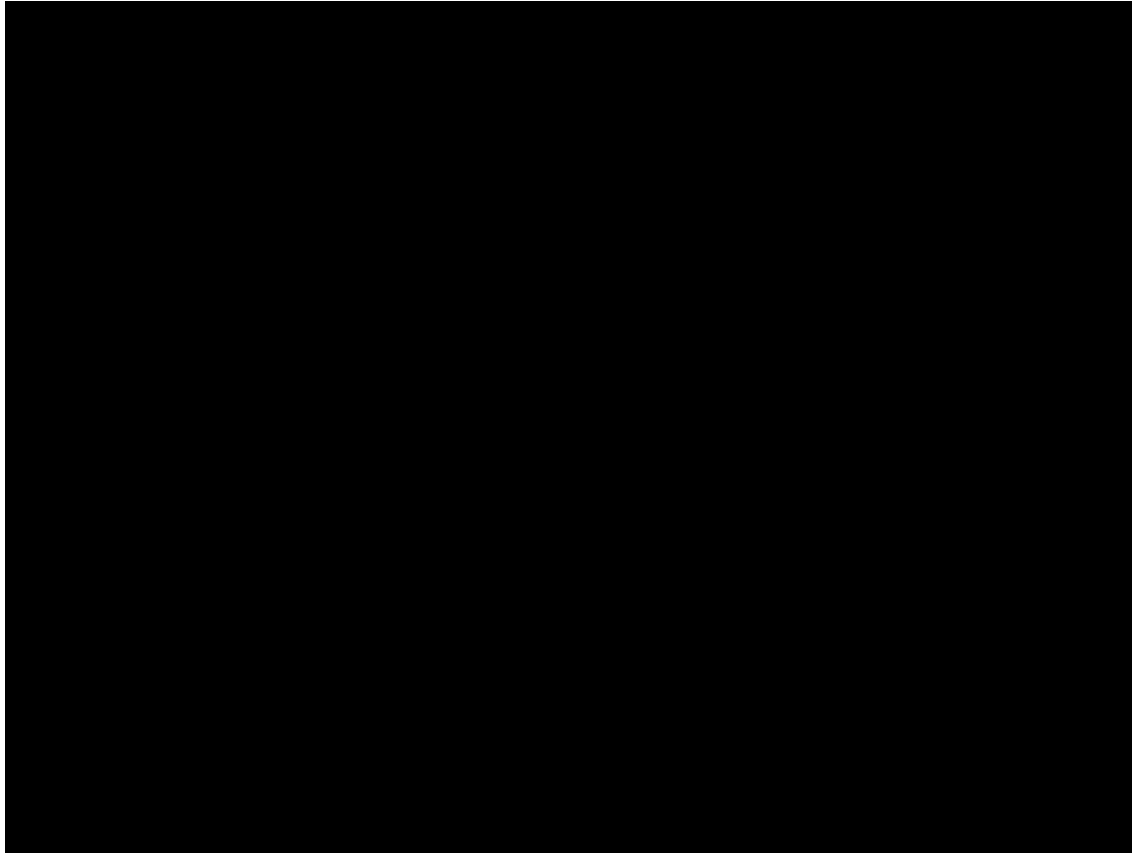Click on the
Start of Day Survey
and fill it in now!

Tech
Inclusion

Today's Project!

Flappy Bird!

Tech
Inclusion

What will the game look like?

Using the workbook!

The workbooks will help you put your project together!

Each **Part** of the workbook is made of tasks!

Tasks - The parts of your project

Follow the tasks **in order** to make the project!

Hints - Helpers for your tasks!

Stuck on a task, we might have given you a hint to help you **figure it out**!

The hints have **unrelated** examples, or tips. **Don't copy and paste** in the code, you'll end up with something **CRAZY**!

Task 6.2: Add a blah to your code!

This has instructions on how to do a part of the project

1. **Start by doing this part**
2. **Then you can do this part**

Task 6.1: Make the thing do blah!

Make your project do blah ….

Hint

A clue, an example or some extra information to help you **figure out** the answer.

Tech Inclusion

Using the workbook!

The workbooks will help you put your project together!

Check off before you move on from a **Part**! Do some bonuses while you wait!

Checklist - Am I done yet?

Make sure you can tick off every box in this section before you go to the next Part.

☑ **CHECKPOINT** ☑

If you can tick all of these off you're ready to move the next part!
☐ Your program does blah
☐ Your program does blob

Lecture Markers
This tells you you'll find out how to do things for this section during the names lecture.

For Loops →

Bonus Activities
Stuck waiting at a lecture marker?
Try a purple bonus. They add extra functionality to your project along the way.

★ **BONUS 4.3: Do some extra!**

Something to try if you have spare time before the next lecture!

Tech Inclusion

Intro to Python

Let's get coding!

Tech
Inclusion

Getting set up

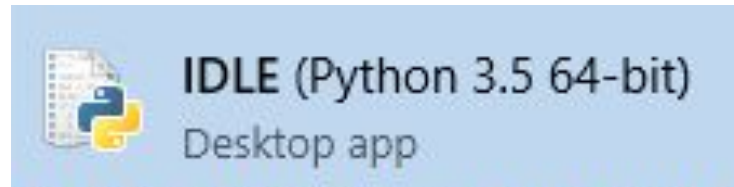Go to your desktop and open the Flappy bird python

Flappy Bird Python

Double click the IDLE(Python GUI).exe file.

(This will download IDLE onto your desktop)
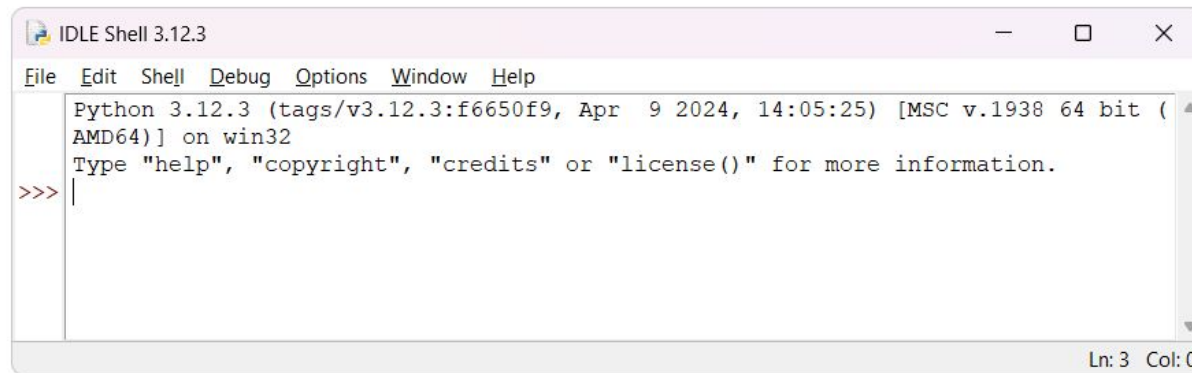
It should look like this

IDLE (Python GUI)

Where do we program? In IDLE

Once it's downloaded open IDLE.

You should get a screen that looks like this!

Tech Inclusion

Type by **button mashing** the keyboard!

Then press enter!

`asdf asdjlkj;pa j;k4uroei`

Did you get a big red error message?

Mistakes are great!

Good work you made an error!

- Programmers make A LOT of errors!
- Errors give us hints to find mistakes
- Run your code often to get the hints!!
- Mistakes won't break computers!

KeyError: 'Hairy Potter'

AttributeError: 'NoneType' object has no attribute 'foo'

TypeError: Can't convert 'int' object to str implicitly

Tech Inclusion

Adding a comment!

Sometimes we want to write things in code that the computer doesn't look at! We use **comments** for that!

Use comments to write a note or explanation of our code

Comments make code easier for humans to understand

```
# This code was written by Sheree
```

We can make code into a comment if we don't want it to run (but don't want to delete it!)

```
# print("Goodbye world!")
```

Tech Inclusion

Write some code!!

This is the first bit of code we will do. What do you think it does?

```
print('hello world')
```

Write some code!!

This is the first bit of code we will do. What do you think it does?

```python
print('hello world')
```

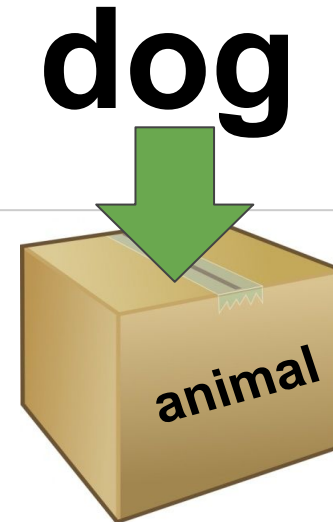It prints the words "hello world" onto the screen!

Storing information

We can store information in variables:

```python
animal = "dog"
print("My favourite animal is a " + animal)

>>> My favourite animal is a dog
```

dog

Variables are like putting things into a **labeled cardboard box**.

animal

They can help us remember things for later... especially when those things end up changing

Reusing variables

We can replace values in variables:

```python
animal = "dog"
print("My favourite animal is a " + animal)
animal = "cat"
print("My favourite animal is a " + animal)
animal = animal + "dog"
print("My favourite animal is a " + animal)
```

What will this output?

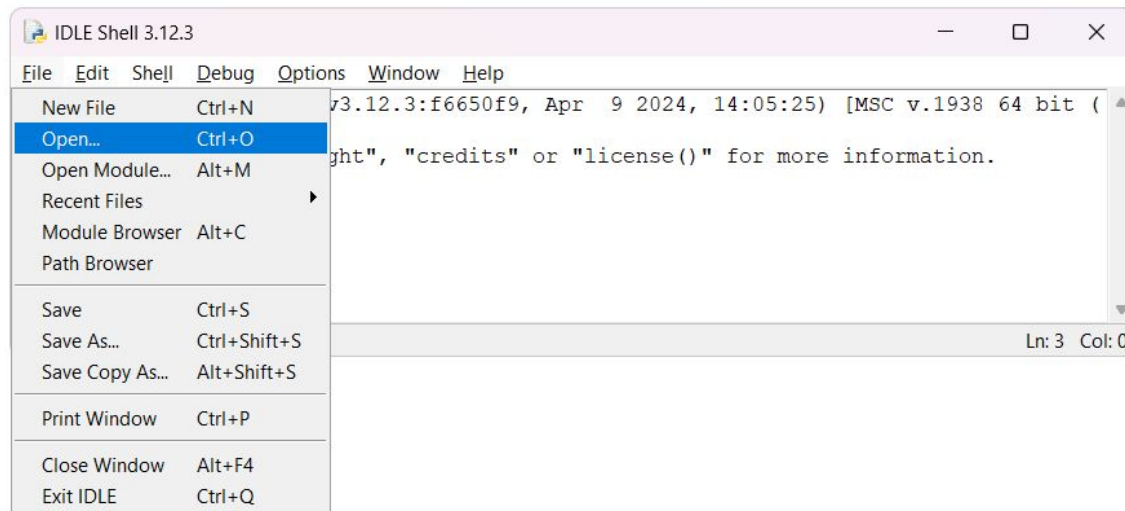# Reusing variables

We can replace values in variables:

```python
animal = "dog"
print("My favourite animal is a " + animal)
animal = "cat"
print("My favourite animal is a " + animal)
animal = animal + "dog"
print("My favourite animal is a " + animal)
```

What will this output?
```
My favourite animal is a dog
My favourite animal is a cat
My favourite animal is a catdog
```

Coding in a file!

Code in a file is code we can run multiple times! Make a reusable "hello world"!

1. Open a file called "flappy_bird.py" (it's in your folder)
2. Put your `print('hello world')` code in it
3. Run your file using the F5 key

Tech Inclusion

Project time!

You now know all about printing and variables and input!

Let's put what we learnt into our project

Try to do Part 0
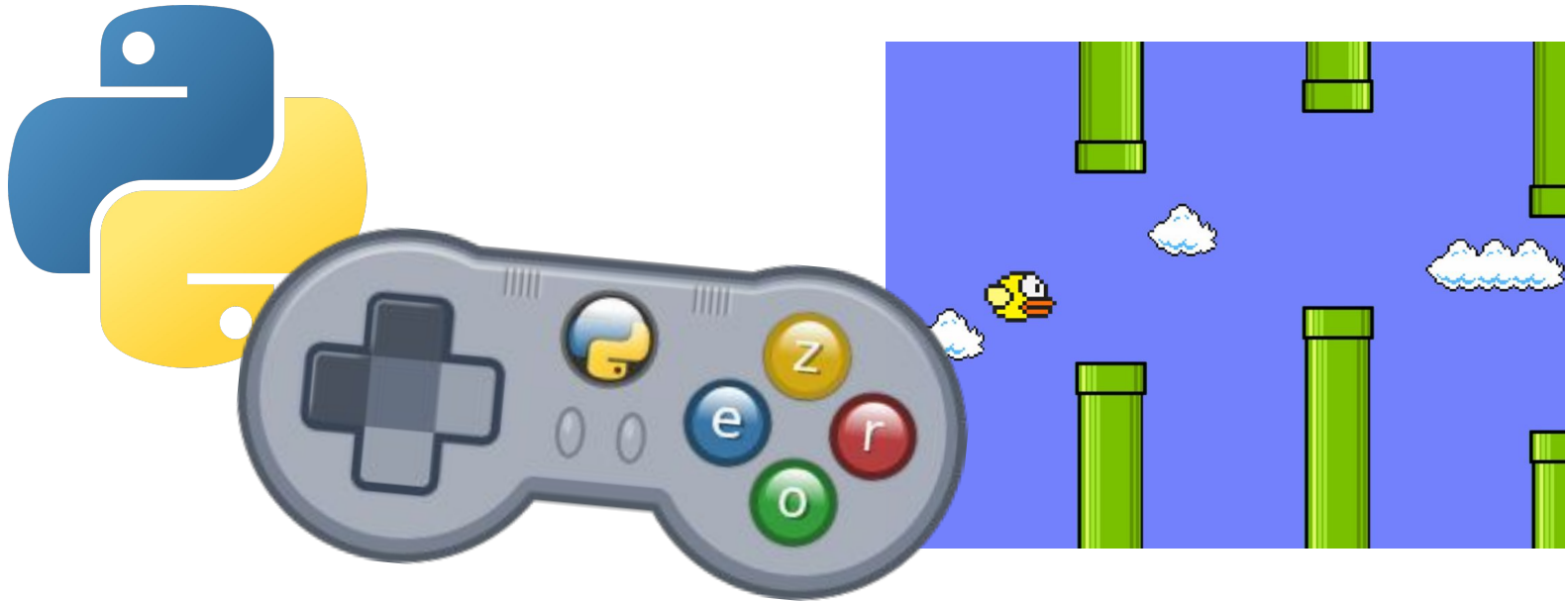
The tutors will be around to help!

Intro to PyGame Zero

Making it into a game!

What is Pygame Zero?

Pygame zero is a bit of extra code we add on to our regular python to "teach" it how to do some new things like drawing images other things to make a game like playing sounds.

Pygame Zero Setup

The first thing we need to do is to "import" pygame zero. This tells idle that it should be working with pygame zero to run your code.

To do that we need to write something like this in your file.

```
>>> import pgzrun
```

Pygame Zero Setup

The first thing we need to do is to "import" pygame zero. This tells idle that it should be working with pygame zero to run your code.

To do that we need to write something like this in your file.

```
>>> import pgzrun
```

Now to make sure PyGame Zero runs our code we also need another line at the end of our code

```
>>> pgzrun.go()
```

Some Pygame Zero basics

Here's some of the basics of Pygame Zero that you'll need for your game.

Screen:

Your main screen for the game will be a screen that pops up whenever you run your game. You can create a screen by setting its size using the keywords WIDTH and HEIGHT

1. Try making a 100 x 100 screen and running your file!

The screen should be blank for now

Some Pygame Zero basics

Here's some of the basics of Pygame Zero that you'll need for your game.

Screen:

Your main screen for the game will be a screen that pops up whenever you run your game. You can create a screen by setting its size using the keywords WIDTH and HEIGHT

1. Try making a 100 x 100 screen and running your file!

   ```
   >>> WIDTH = 100
   >>> HEIGHT = 100
   ```

The screen should be blank for now

Tech
Inclusion

Project time!

You now know all about the basics of Pygame Zero!

Let's put what we learnt into our project

**Try to do Part 1**

The tutors will be around to help!

Tech Inclusion

PyGame Zero images

Adding things to our screen!

Tech
Inclusion

Images in Pygame zero

Images in Pygame zero are called **Actors**

This is because you can make them move around and do things like actors in a play. Pygame zero stores some information about each of the actors in our game like their position on the screen and what image the actor is.
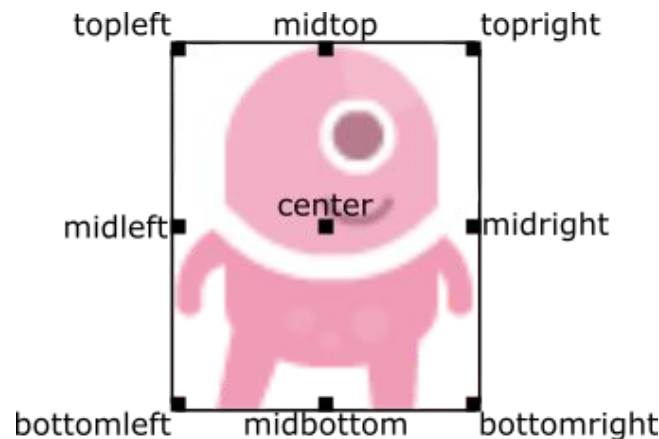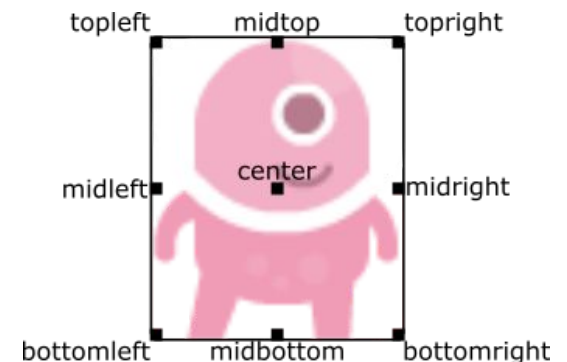
How to make an actor

To make a new actor and tell Pygame zero what image it is you need to write the code:

```
>>> myActor = Actor("myImage")
```

Here the name of our actor is **myActor** and if we need to change anything about it we have to use it's name

Tech
Inclusion

How to make an actor

To make a new actor and tell Pygame zero what image it is you need to write the code:
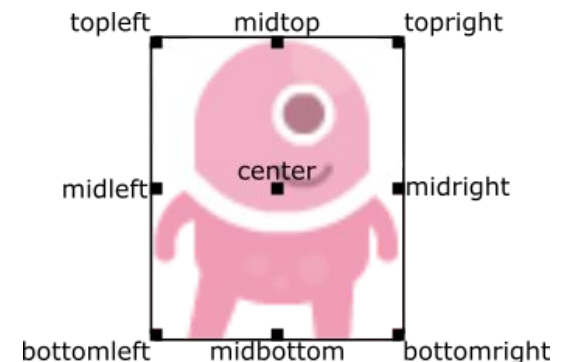
```
>>> myActor = Actor("myImage")
```

Here the name of our actor is **myActor** and if we need to change anything about it we have to use it's name

To set our actor's x and y position you use the code:

```
>>> myActor.x = 50
>>> myActor.y = 50
```


topleft   midtop   topright

midleft   center   midright

bottomleft   midbottom   bottomright

Tech
Inclusion

TODO

Add a screenshot of function blocks in block coding

Getting an actor on screen!

Pygame zero needs some pretty specific things in order to make our game work. To do these there are three main functions:

```python
def draw():
    # This function is to add things to the screen every frame

def update():
    # This function is to change things every frame

def on_mouse_down():
    # This function's code runs every time the player clicks their mouse
```
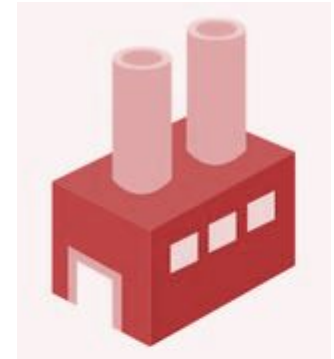
What are functions?

Functions are like factories!
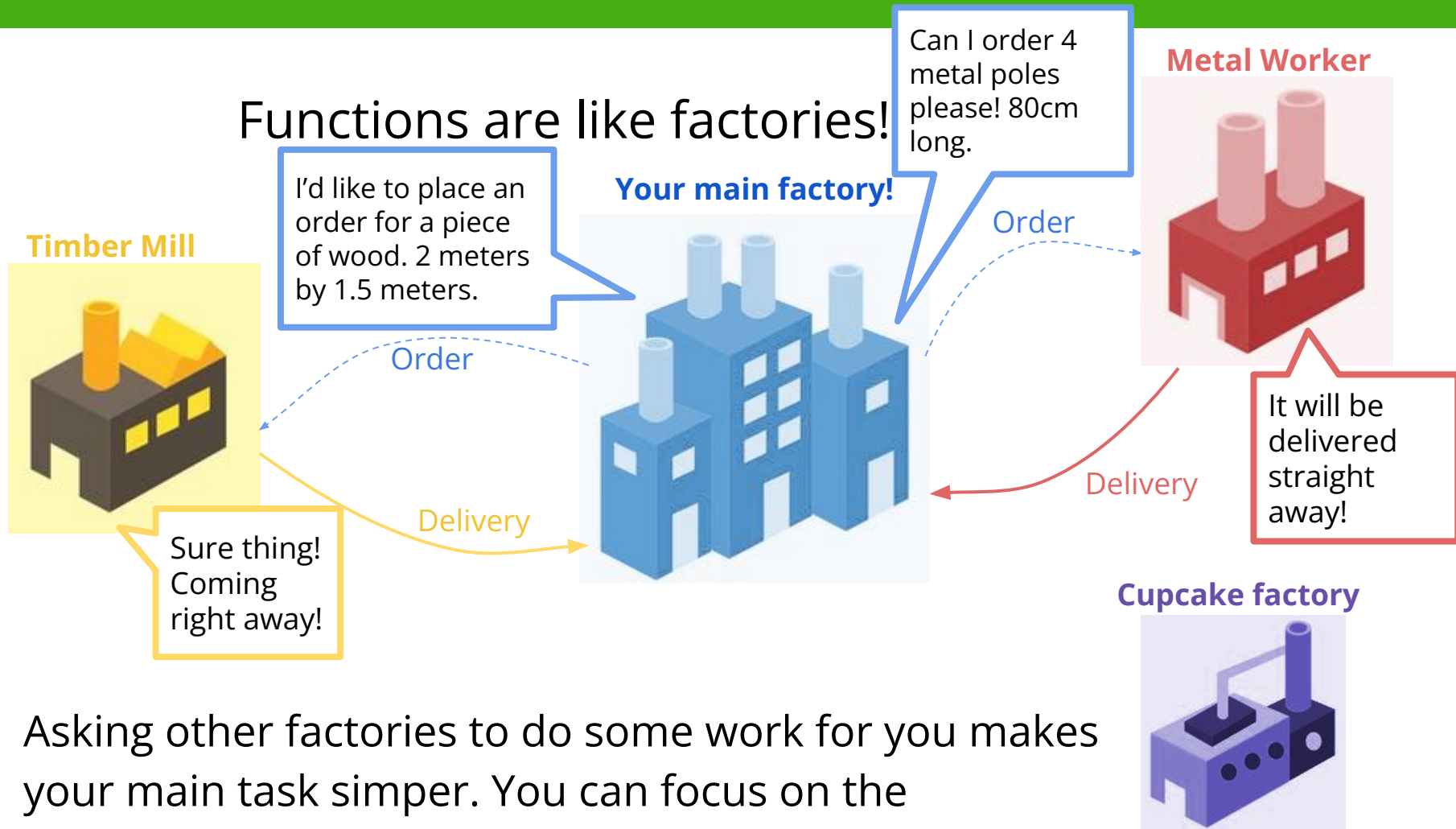
Metal Worker

Your main factory!

Timber Mill

Cupcake factory

Running a factory doesn't mean doing all the work yourself, you can get other factories to help you out!

Tech
Inclusion

What are functions?

Functions are like factories!
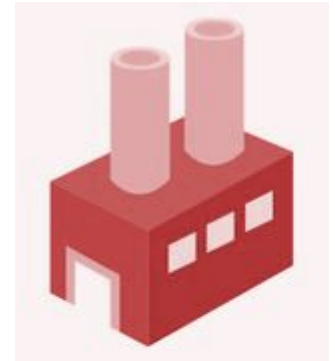
Metal Worker

Your main factory!

Timber Mill

Look at this beautiful table I made!

Outsourcing made it simple!

Cupcake factory

Some important code

Pygame zero needs some pretty specific things in order to make our game work. To do these there are three main functions:

```python
def draw():
    # This function is to add things to the screen every frame

def update():
    # This function is to change things every frame

def on_mouse_down():
    # This function's code runs every time the player clicks their mouse
```

What is a function?

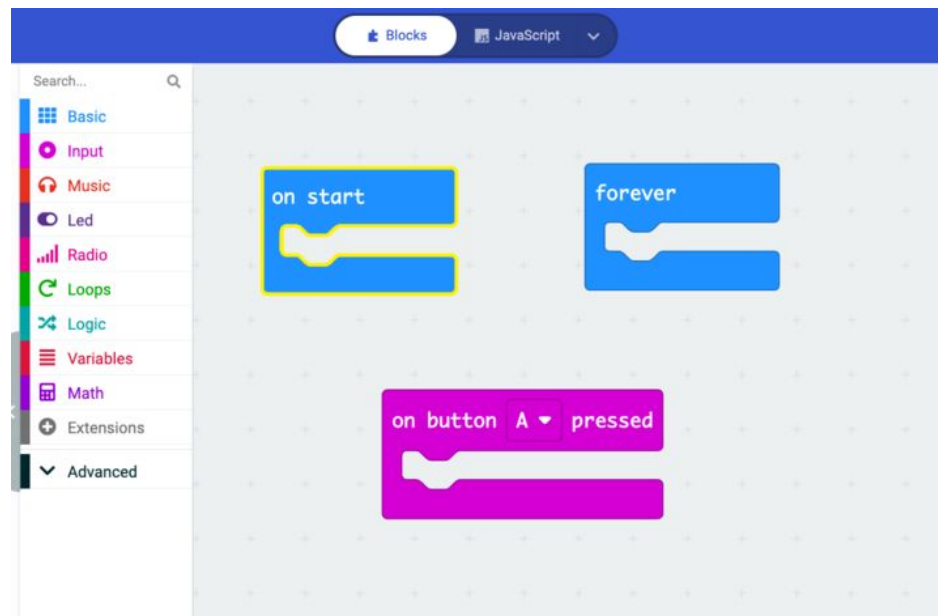**What you need to know about functions:**

They are a piece of code that gets run a lot! These functions get run everytime you say their name.

Beetlejuice, Beetlejuice, Beetlejuice!

Tech Inclusion

Functions in blockly
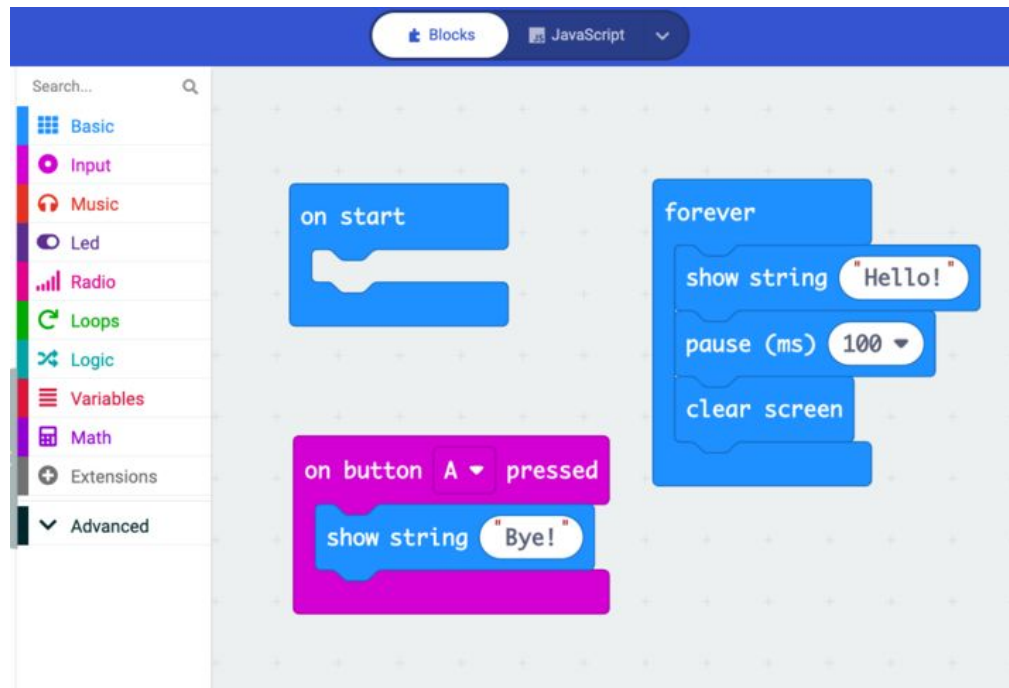
Here are some functions in blockly - maybe they seem familiar from earlier in school:

Blockly always runs those chunks of code. Pygame Zero is similar!

Functions in blockly

With the blocks you stick other blocks inside so they get run, at the start of the program, each loop, or when a button is pressed.

Pygame Zero continuously looks for 3 functions to make the game run

3 Key functions

```python
def draw():
    # This function is to add things to the screen every frame

def update():
    # This function is to change things every frame

def on_mouse_down():
    # This function's code runs every time the player clicks their mouse
```

We'll put our code inside and Pygame Zero will run them to make the game work!

Getting an actor on screen!

The first function we need in Pygame Zero is the draw() function.The draw() function tells Pygame Zero what things need to appear on screen.

You can use it to "draw" an actor on the screen by using these lines of code:

```
>>> def draw():
...     myActor.draw()
```
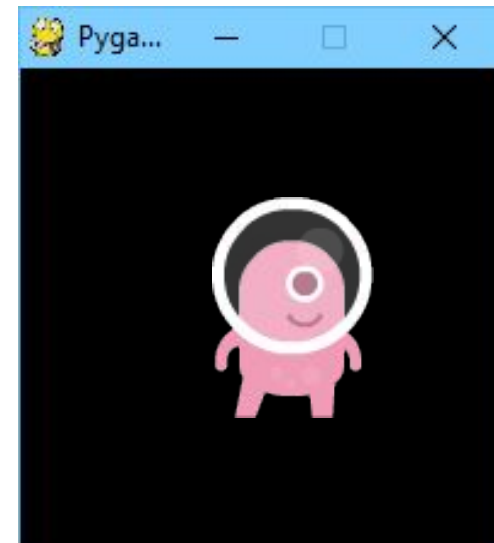
Tech
Inclusion

Changing the actor

The update() function tells Pygame Zero what things need to change so that it can "animate" the game frame by frame

You can use it to do things like update an actor's image or x or y coordinates:

```
>>> def update():
...     myActor.x = myActor.x + 5
```

Tech Inclusion

When the mouse clicks

The on_mouse_down() function only runs when the player has clicked. This means that you can make changes to your character when the player clicks their mouse.

You can use it to do things like change an actor's image or x or y coordinates when the player clicks the mouse:

```
>>> def on_mouse_down():
...     myActor.image("image2")
```

Tech Inclusion

Project time!

You now know all about how to put a character on the screen and how to animate it!

Let's put what we learnt into our project

Try to do Part 2

The tutors will be around to help!

If Statements and Lists

Some quick revision

Conditions!

Conditions let us make decision.

First we test if the condition is met!

Then maybe we'll do the thing

<u>**If it's raining**</u> take an umbrella

Yep it's raining

...... take an umbrella

Tech Inclusion

Conditions

So to know whether to do something, they find out if it's **True**!

```python
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```

Conditions

So to know whether to do something, they find out if it's **True**!

```python
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```

What do you think happens?
>>>

Conditions

So to know whether to do something, they find out if it's **True**!

```python
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```

<u>What do you think happens?</u>

```
>>> that's a small number
```

Tech Inclusion

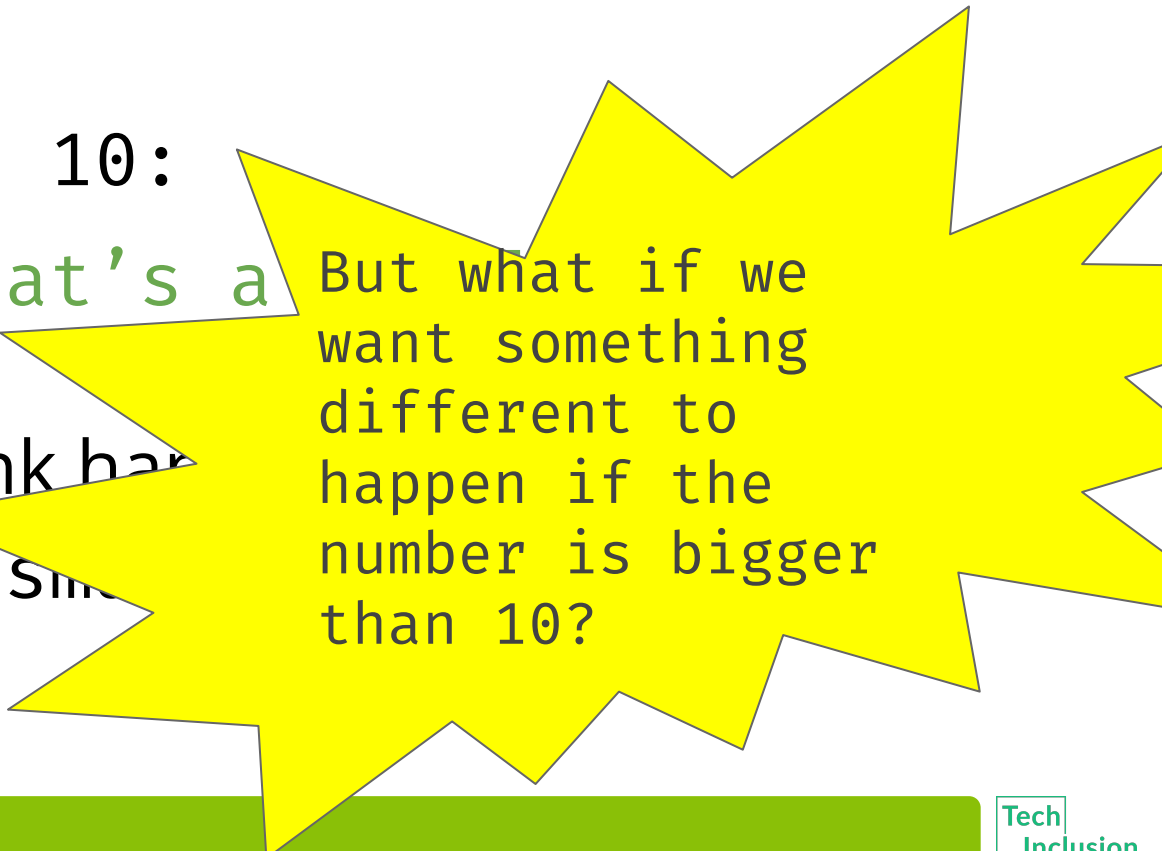# Conditions

So to know whether to do something, they find out if it's True!

```
fave_num = 5
if fave_num < 10:
    print("that's a
```

<u>What do you think ha</u>

>>> that's a sm

But what if we want something different to happen if the number is bigger than 10?

Tech Inclusion

Elif statements

elif
Means we can give specific instructions for other scenarios

else
statements means something still happens if the **if** statement was **False**

```python
fav_number = 10
if fav_number < 10:
    print("That is a small number")
elif fav_number > 10:
    print("That is a big number!")
else:
    print("That number is just right!")
```

What happens?

Tech Inclusion

Elif statements

elif
Means we can give specific instructions for other scenarios

else
statements means something still happens if the **if** statement was **False**

```python
fav_number = 10
if fav_number < 10:
    print("That is a small number")
elif fav_number > 10:
    print("That is a big number!")
else:
    print("That number is just right!")
```
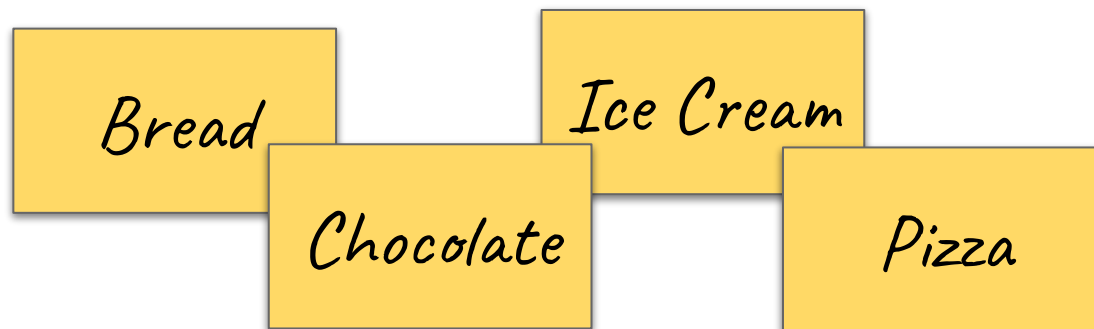
What happens?
>>> That number is just right!

Tech Inclusion

Lists

When we go shopping, we write down what we want to buy!

But we don't store it on lots of little pieces of paper!

Bread

Chocolate

Ice Cream

Pizza

We put it in one big shopping list!

- Bread
- Chocolate
- Ice Cream
- Pizza

Lists

It would be annoying to store it separately when we code too

```
>>> shopping_item1 = "Bread"
>>> shopping_item2 = "Chocolate"
>>> shopping_item3 = "Ice Cream"
>>> shopping_item4 = "Pizza"
```

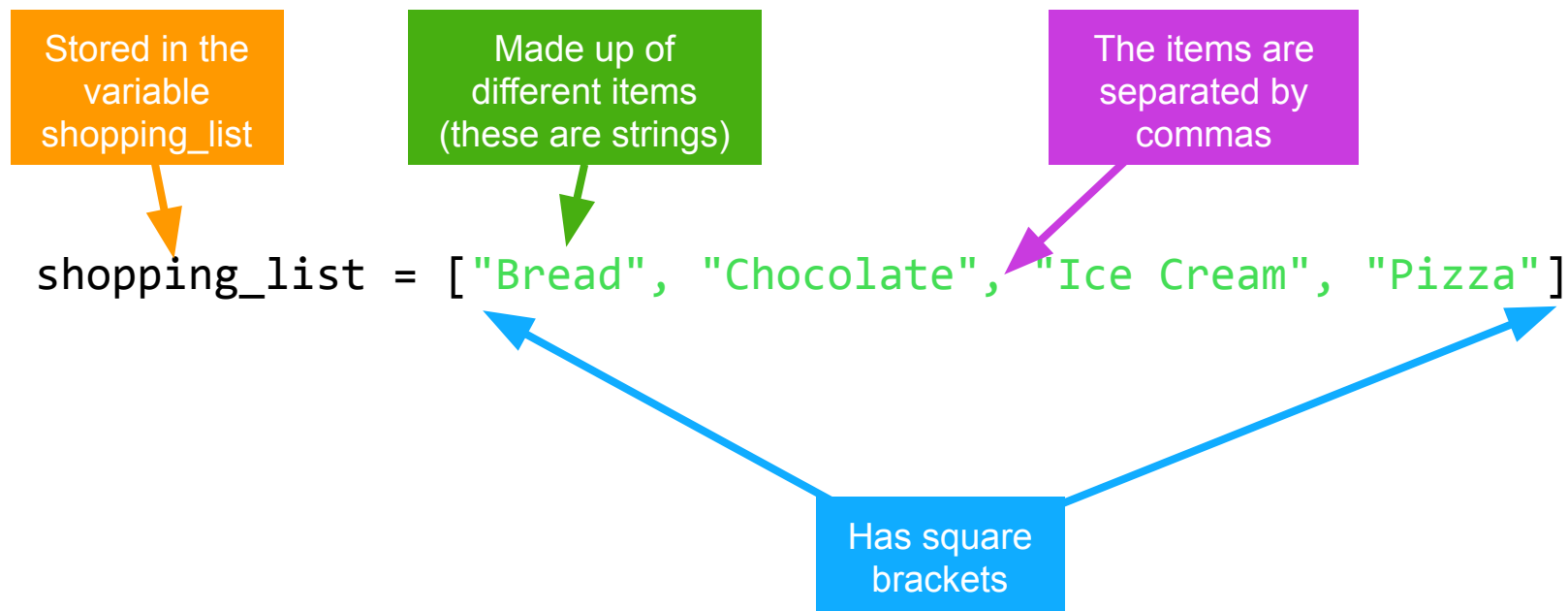So much repetition!

Instead we use a python list!

```
>>> shopping_list = ["Bread", "Chocolate", "Ice Cream", "Pizza"]
```

List anatomy

Stored in the variable shopping_list

Made up of different items (these are strings)

The items are separated by commas

```
shopping_list = ["Bread", "Chocolate", "Ice Cream", "Pizza"]
```

Has square brackets

Tech Inclusion

Project Time!

You now know all about **if** and lists!

See if you can do Part 3

The tutors will be around to help!

Classes

Classes and Objects

What is a 'class'?

A **class** is a group of things that have similar characteristics and behaviour. It's used in lots of different games to allow characters to behave the same while still being different and having different attributes. One "instance" or one occurrence of a class is called an **object**

For example let's think of Pokemon

```
class Pokemon:
```

Name of the class or group of things

Tech Inclusion

Our example...

There are many different types of pokemon but they all have similar behaviours and general statistics. For example each pokemon has a **type**, **attacks**, and **weaknesses**. The specifics of these vary with each pokemon but on a general level this is what makes them part of a class as they have similar characteristics and behaviours.

Why do we use them?

For our example let's focus on **Bulbasaur!**

What are some things we know about bulbasaur?

Name: Bulbasaur
Type: Grass/Poison
Attacks: Tackle, Vine whip, Growth, etc.
Weaknesses: Fire, Ice, Flying, Psychic

Making a class

Let's make an example class using this information

```python
class Pokemon:

    def __init__(self, name, type, attacks, weaknesses):
        self.name = name
        self.type = type
        self.attacks = attacks
        self.weaknesses = weaknesses
        print('A new pokemon has been created')
```

Declaring its properties

This is printed whenever a new pokemon object is made.

Making an object

Now let's make a bulbasaur object...

```python
class Pokemon:

    def __init__(self, name, type, attacks, weaknesses):
        self.name = name
        self.type = type
        self.attacks = attacks
        self.weaknesses = weaknesses
        print('A new pokemon has been created')

name = "Bulbasaur"
type = "Grass/Poison"
attacks = ["Tackle","Vine Whip","Growth",...]
weaknesses = ["Fire","Ice","Flying","Psychic"]
Bulbasaur = Pokemon(name, type, attacks, weaknesses)
```
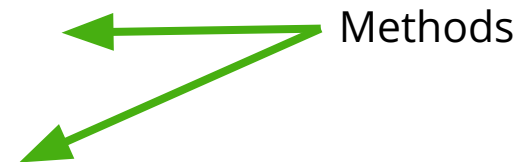
Methods

Right now our pokemon don't do anything so let's fix that. A class can have specific behaviours or "methods" attached to it. These are functions that can only be performed by the class. However, this means that every object in a class has these same "behaviours". So let's make some behaviors for our Pokemon

```python
def evolve(self, new_name):
        print(f"Congratulations, your {self.name}
has now evolved to a {new_name}")
        self.name = new_name


def attack(self):
    attack = attacks[0]
    print(f"{self.name} performs move {attack}")
```

Methods

Methods

Let's test out these new behaviours with our bulbasaur

```python
def evolve(self, new_name):
    print(f"Congratulations, your {self.name} has now
evolved to a {new_name}")
    self.name = new_name

def attack(self):
    attack = attacks[0]
    print(f"{self.name} performs move {attack}")

Bulbasaur.attack()
```

What happens?
>>>

Methods

Let's test out these new behaviours with our bulbasaur

```python
def evolve(self, new_name):
    print(f"Congratulations, your {self.name} has now
evolved to a {new_name}")
    self.name = new_name

def attack(self):
    attack = attacks[0]
    print(f"{self.name} performs move {attack}")

Bulbasaur.attack()
```

What happens?
>>> Bulbasaur performs move Tackle

Methods

Let's test out these new behaviours with our bulbasaur

```python
def evolve(self, new_name):
    print(f"Congratulations, your {self.name} has now
evolved to a {new_name}")
    self.name = new_name

def attack(self):
    attack = attacks[0]
    print(f"{self.name} performs move {attack}")


Bulbasaur.evolve("Ivysaur")
```

What happens?
>>>

Methods

Let's test out these new behaviours with our bulbasaur

```python
def evolve(self, new_name):
    print(f"Congratulations, your {self.name} has now
evolved to a {new_name}")
    self.name = new_name

def attack(self):
    attack = attacks[0]
    print(f"{self.name} performs move {attack}")

Bulbasaur.evolve("Ivysaur")
```

What happens?
>>> Congratulations, your Bulbasaur has now
evolved to a Ivysaur

Attribute values

Now that we know the basics of what a class is and what it can do we have one more thing to cover... Attributes

Remember when we created the class it had some weird things at the beginning like self.name self.type etc? Well we can actually find out the value of these variables from outside the class by using the object's name instead of self. Let's look at some examples:

1. `print(Bulbasaur.name)`
   `>>>`

2. `print(Bulbasaur.type)`
   `>>>`

Attribute values

Now that we know the basics of what a class is and what it can do we have one more thing to cover... Attributes

Remember when we created the class it had some weird things at the beginning like self.name self.type etc? Well we can actually find out the value of these variables from outside the class by using the object's name instead of self. Let's look at some examples:

1. ```
   print(Bulbasaur.name)
   >>> Bulbasaur
   ```

2. ```
   print(Bulbasaur.type)
   >>>
   ```

Attribute values

Now that we know the basics of what a class is and what it can do we have one more thing to cover… Attributes

Remember when we created the class it had some weird things at the beginning like self.name self.type etc? Well we can actually find out the value of these variables from outside the class by using the object's name instead of self. Let's look at some examples:

1. `print(Bulbasaur.name)`
   `>>> Bulbasaur`

2. `print(Bulbasaur.type)`
   `>>> Grass/Poison`

Tech
Inclusion

Project time!

You now know all about classes

Let's put what we learnt into our project

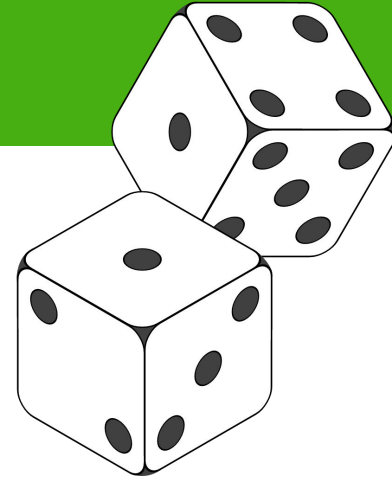**Try to do Part 4**

The tutors will be around to help!

Random!

That's so random!

There's lots of things in life that are up to chance or random!

We want the computer to be random sometimes!

Python lets us **import** common bits of code people use! We're going to use the **random** module!

Tech Inclusion

Using the random module

Let's choose something randomly from a list!

This is like drawing something out of a hat in a raffle!

Try this!

1. **Import the random module!**
   ```
   >>> import random
   ```

2. **Copy the shopping list into IDLE**
   ```
   >>> shopping_list = ["eggs", "bread", "apples", "milk"]
   ```

3. **Choose randomly! Try it a few times!**
   ```
   >>> random.choice(shopping_list)
   ```

Tech
Inclusion

Using the random module

You can also assign your random choice to a variable

```
>>> import random
>>> shopping_list = ["eggs", "bread", "apples", "milk"]
>>> random_food = random.choice(shopping_list)
>>> print(random_food)
```

Raaaaaaaaaandom! Can you handle that?

Let's try use it in our project!
Try to do Part 5

The tutors will be around to

Tech
Inclusion