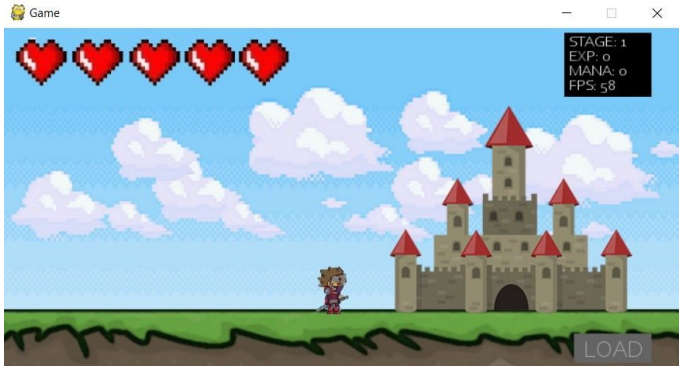
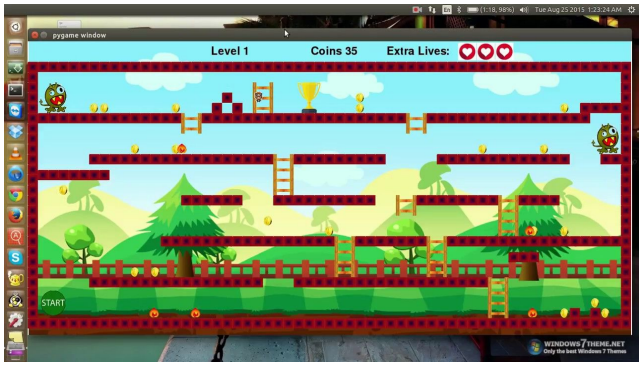
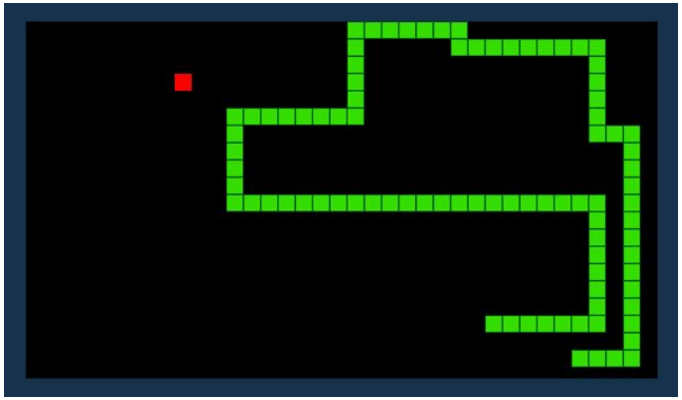
Intro to Pygame

What is Pygame?

Pygame is a tool we can use to make cool games using Python like these:







How do we use it?

First we need to tell Python that we want to use Pygame, and tell it to start the game

```
from pygame import *

init()
```

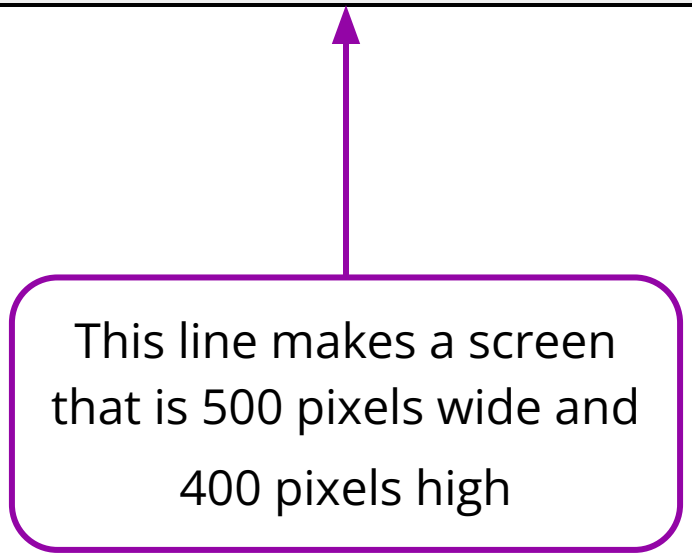
This line tells Python that we want to use pygame

This line starts Pygame

Make a scene!

We can make a screen to show our game

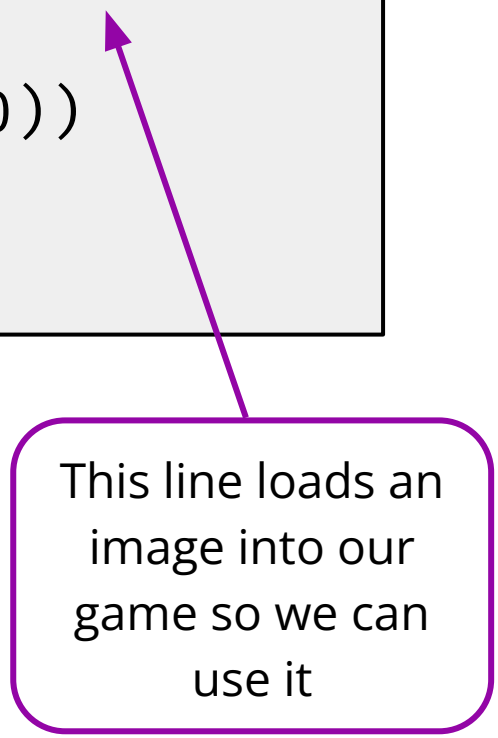
```
screen = display.set_mode(500, 400)
```



Make a scene!

Once we have a screen we can put images on it!

```
cat_image = image.load("cat.png")

screen.blit(cat_image, (30, 40))

display.update()
```



Make a scene!

Once we have a screen we can put images on it!

```
cat_image = image.load("cat.png")

screen.blit(cat_image, (30, 40))

display.update()
```

This line puts the image on the screen at the coordinates 30, 40

This line loads an image into our game so we can use it

Make a scene!

Once we have a screen we can put images on it!

```
cat_image = image.load("cat.png")

screen.blit(cat_image, (30, 40))

display.update()
```

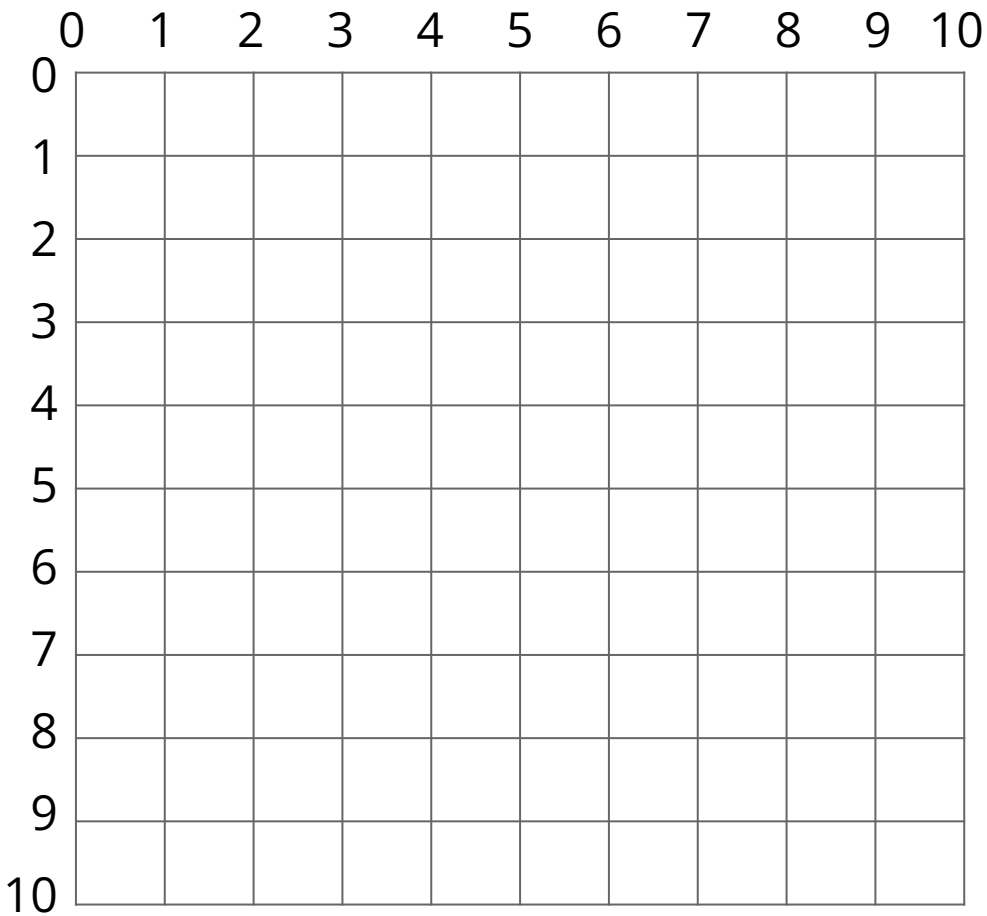
This line tells Pygame to update our display, which will show our new image

This line puts the image on the screen at the coordinates 30, 40

This line loads an image into our game so we can use it

Pygame Coordinates

Coordinates in Pygame are a little strange...



Because they start at the top left instead of the bottom left

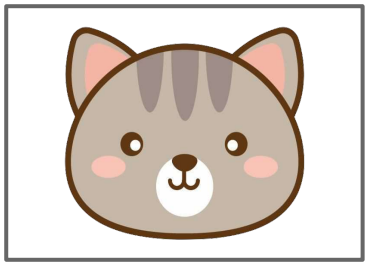
Pygame Coordinates

When you have an image in pygame like this:

Pygame Coordinates

When you have an image in pygame like this:

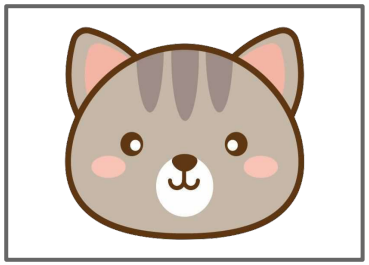
Pygame thinks of it like a rectangle like this:



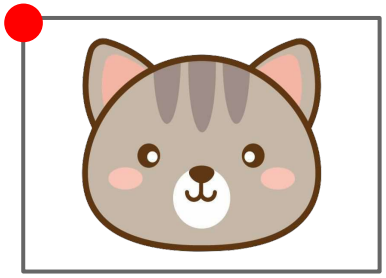
Pygame Coordinates

When you have an image in pygame like this:

Pygame thinks of it like a rectangle like this:

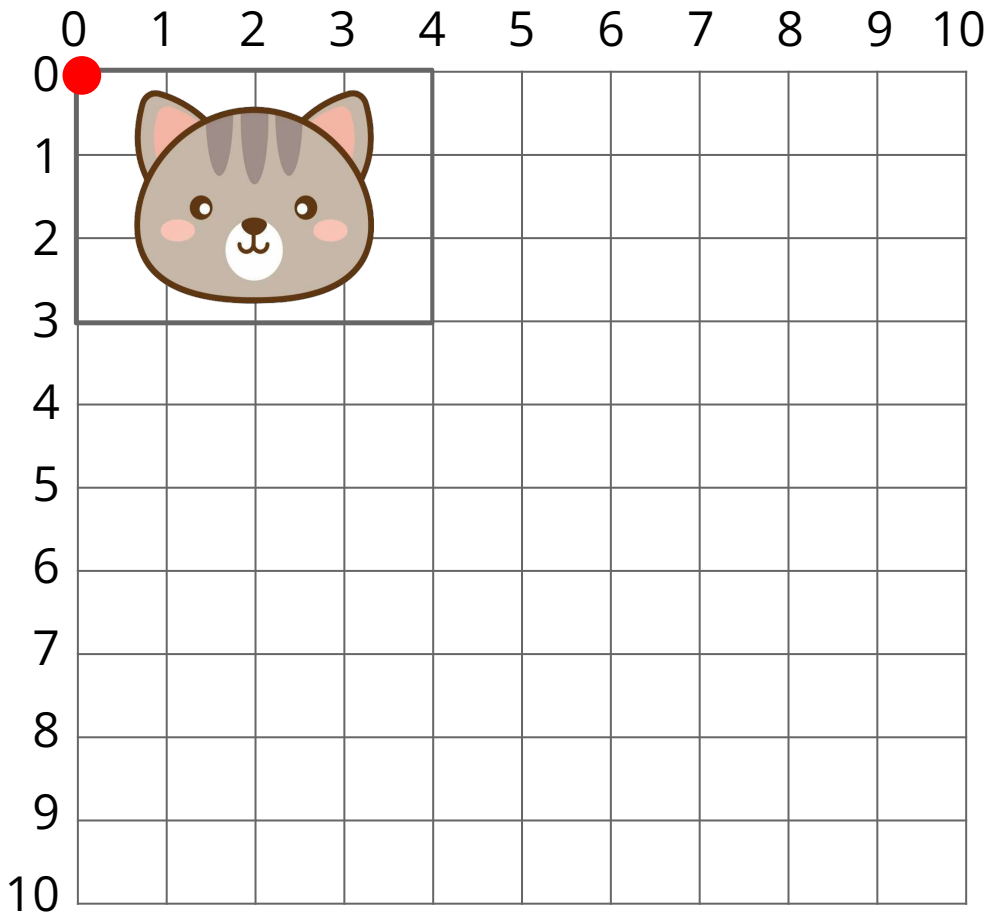


And the coordinates for the image are the top left corner like this:



Pygame Coordinates

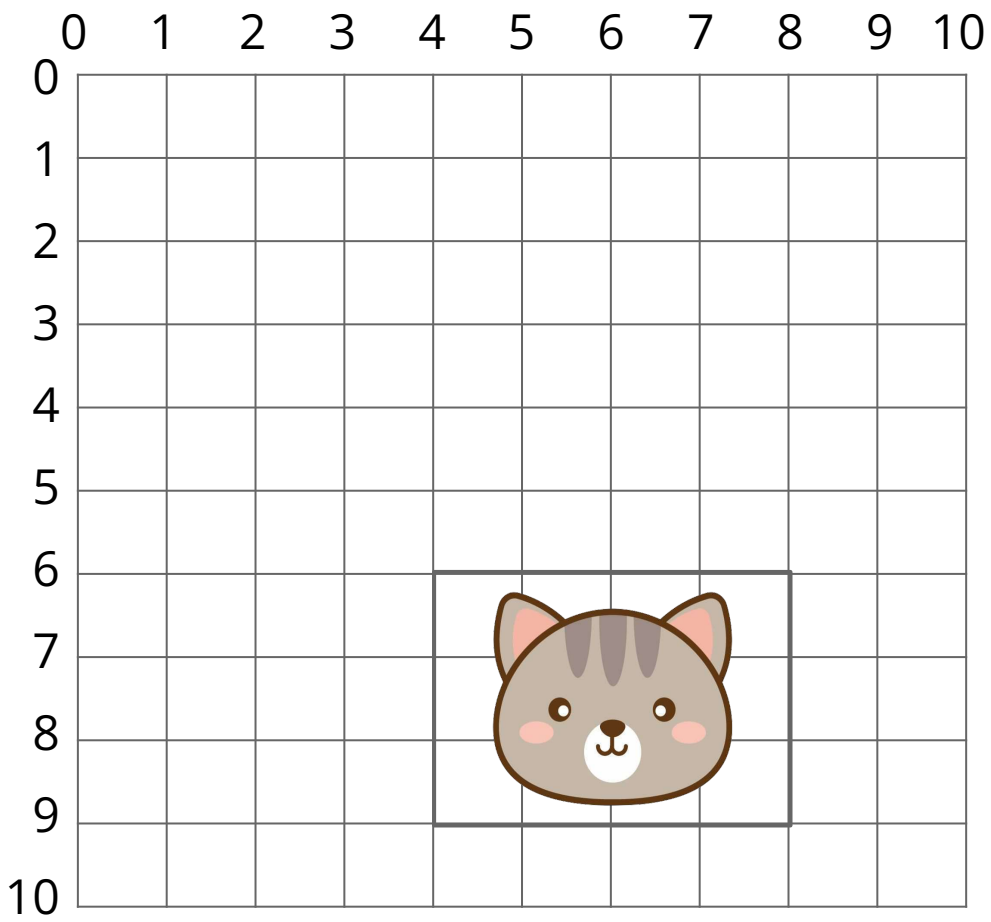
So to if we wanted our cat image to be on the top left we would use the coordinates (0, 0)



That means that the top left corner of the image will be in the top left corner of the screen

Pygame Coordinates

What are the coordinates of our image now?



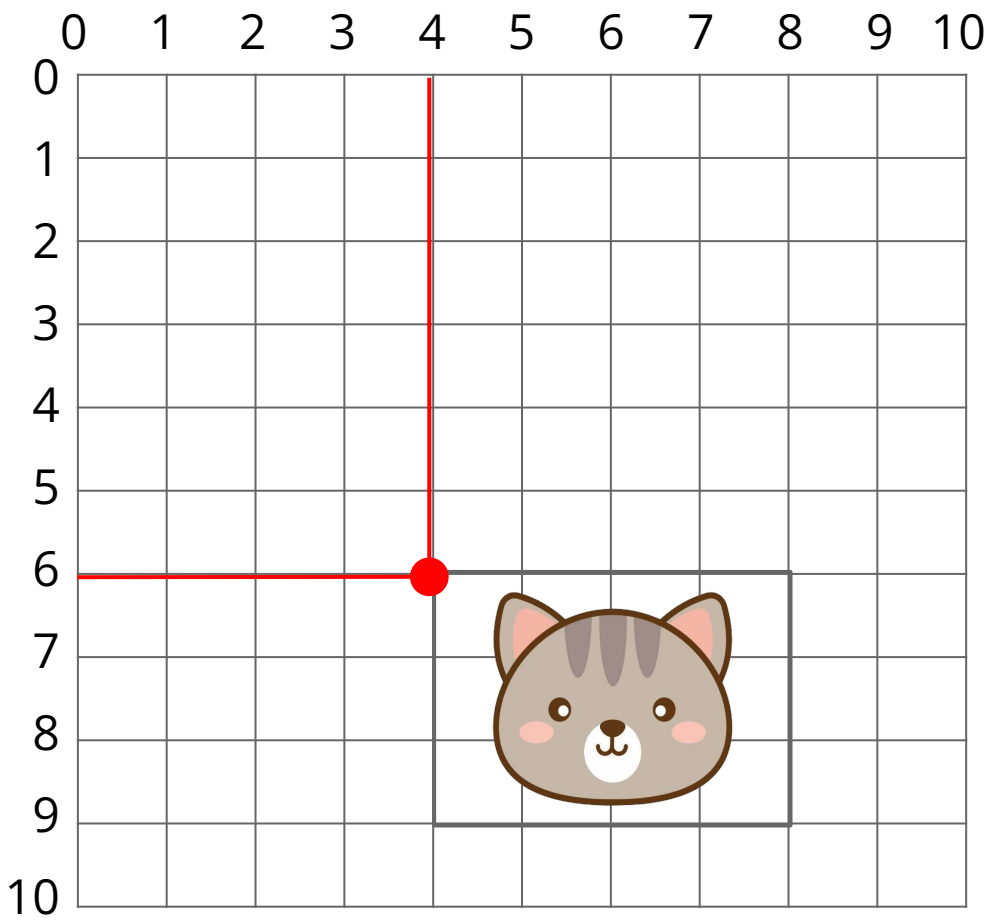
Remember that when we write coordinates we say the x (horizontal) number first and the y (vertical) number second

Pygame Coordinates

What are the coordinates of our image now?

They are:

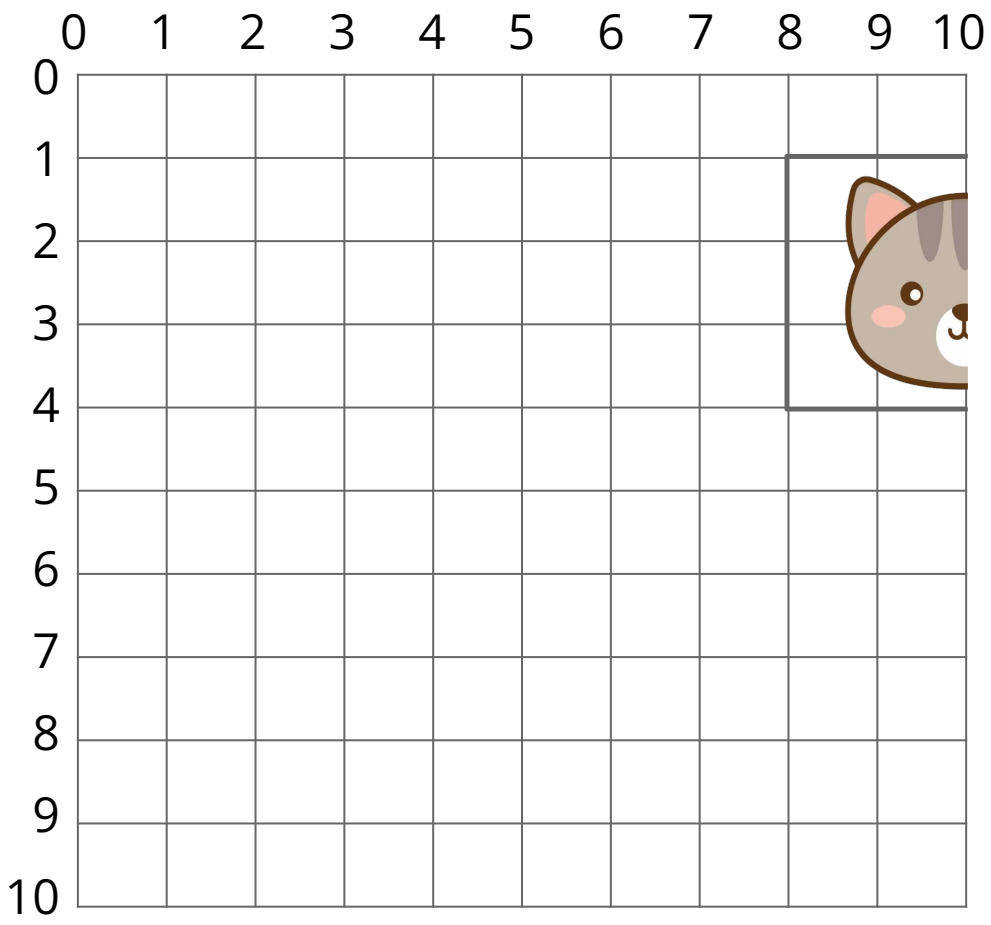
(4, 6)



Remember that when we write coordinates we say the x (horizontal) number first and the y (vertical) number second

Pygame Coordinates

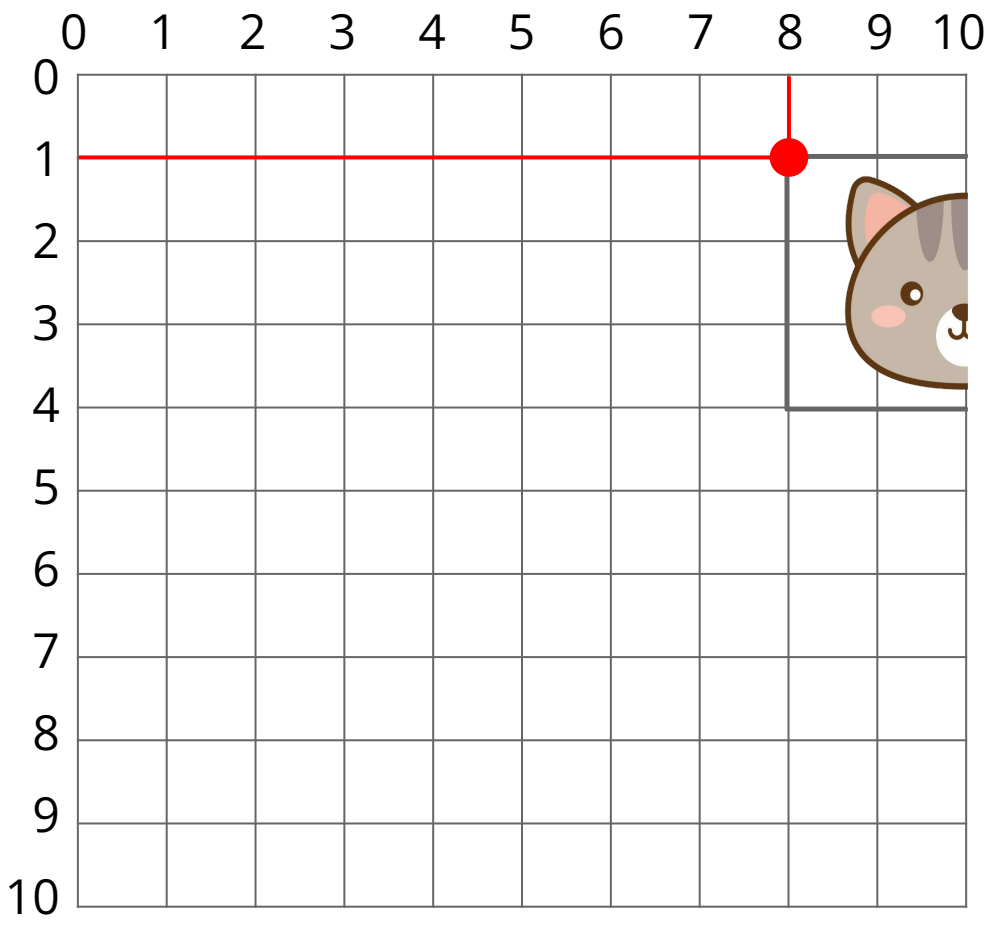
What are the coordinates of our image now?



Sometimes we can put the image off the screen by giving coordinates that are far to the bottom or the right

Pygame Coordinates

What are the coordinates of our image now?



They are:

(8, 1)

Sometimes we can put the image off the screen by giving coordinates that are far to the bottom or the right

Project time!

Now that you've **update**d your knowledge...

Try to do the next Part!

The tutors will be around to help!

Pygame Events!

Pygame Events

Pygame can do more than just show images on a screen!

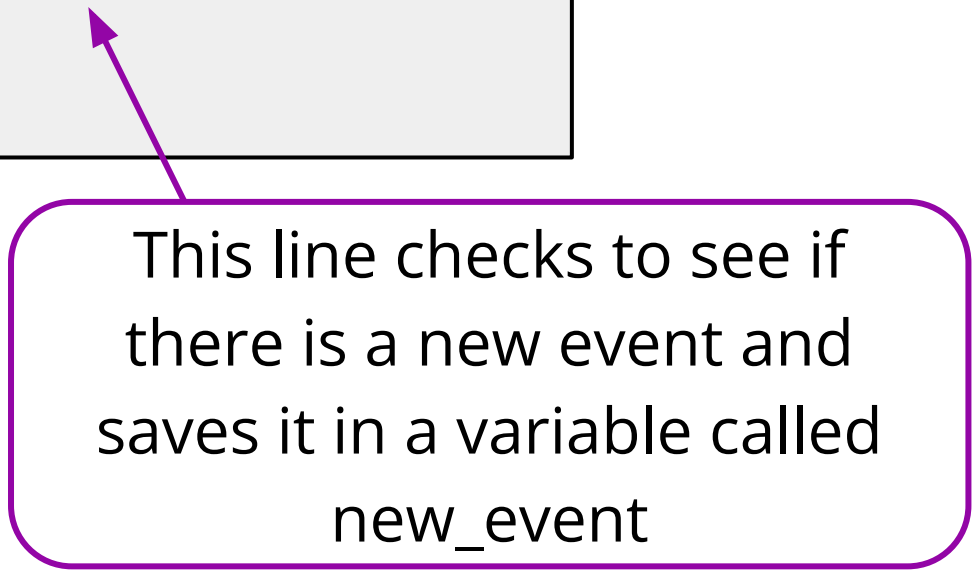
We can use it to figure out if the mouse moved or if a keyboard button was pressed!

These actions are called “events”! Let’s learn how to use them

How do we use it?

First we need to ask Pygame to tell us what has happened recently

```
while True:
  new_event = event.poll()
```



How do we use it?

First we need to ask Pygame to tell us what has happened recently

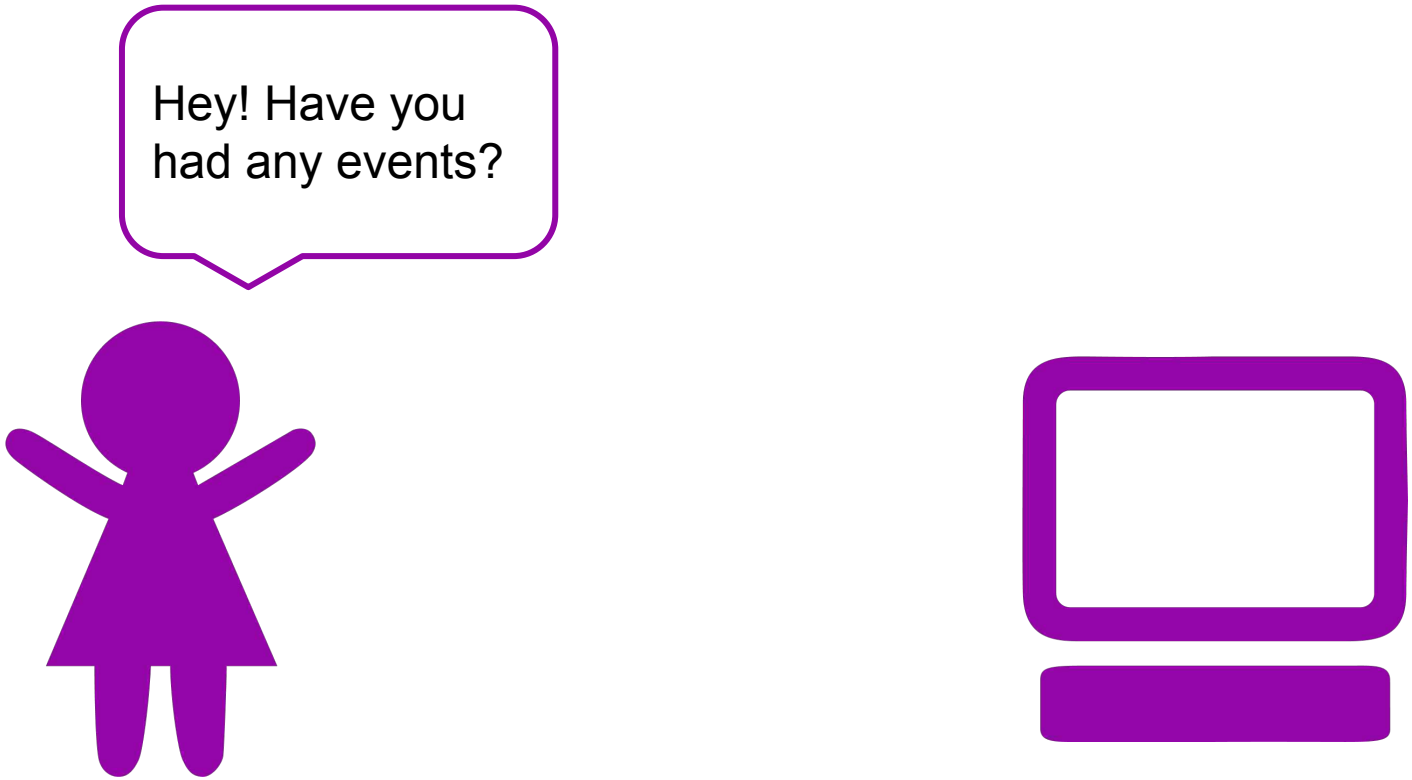
```
while True:
    new_event = event.poll()
```

But what does this line do??

This line checks to see if there is a new event and saves it in a variable called new_event

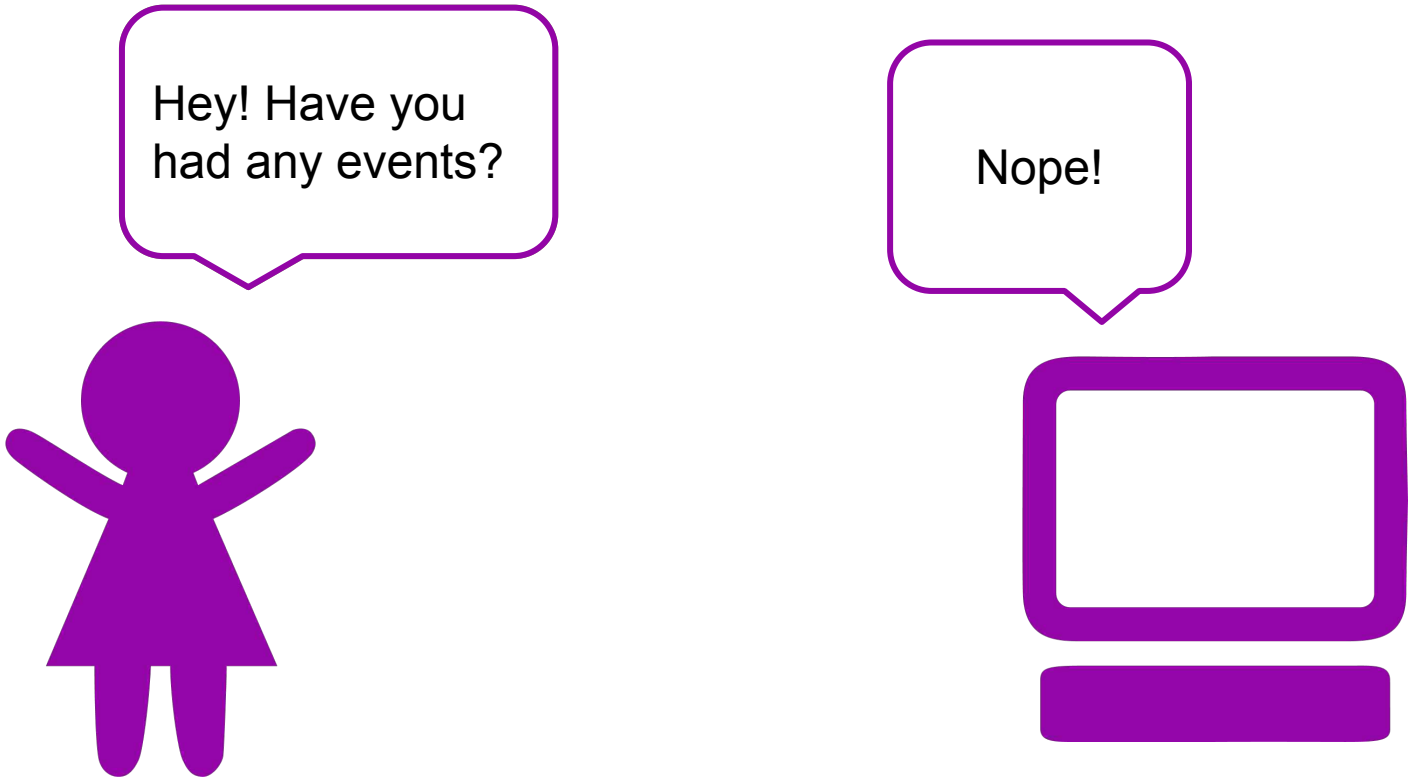
Looking for Events

Let's think of how Pygame checks for events like this:



Looking for Events

Let's think of how Pygame checks for events like this:

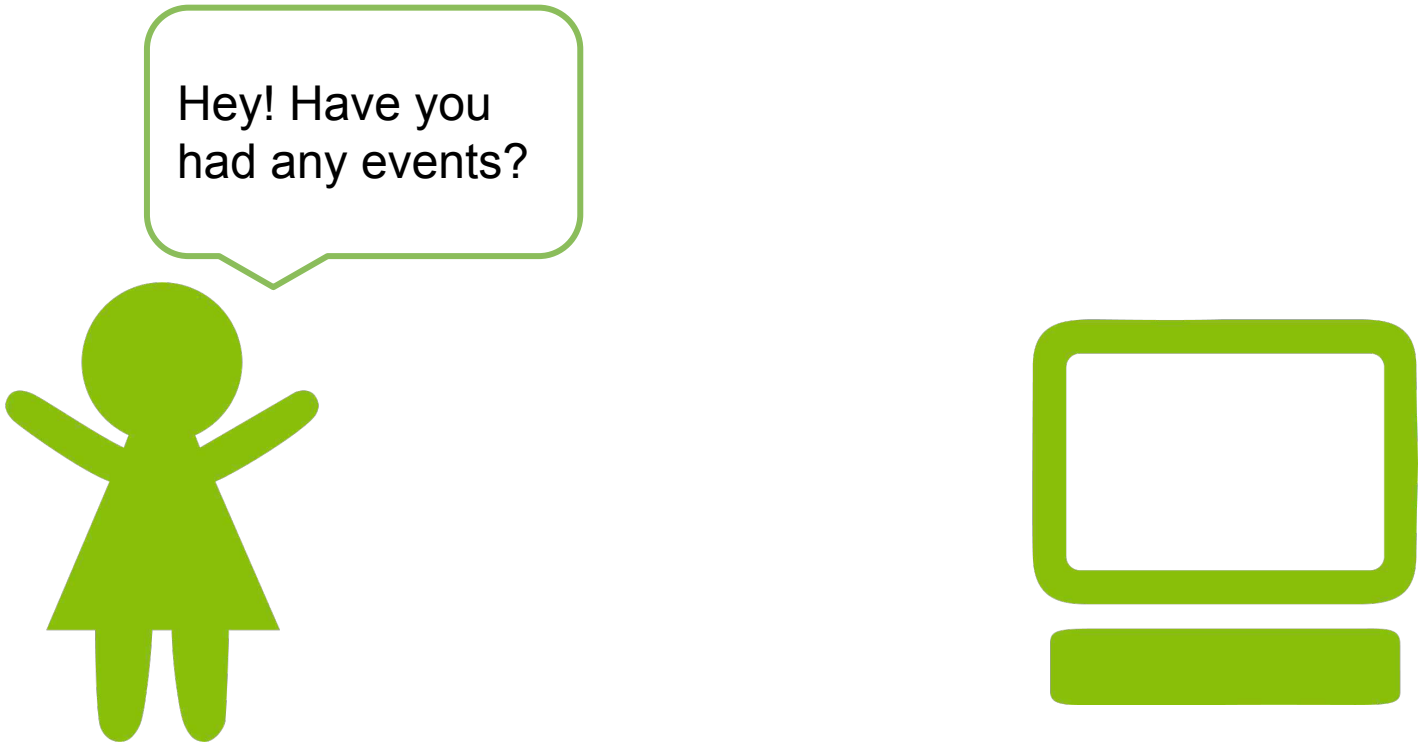


Looking for Events

If we only ask once then we won't know if an event happens later

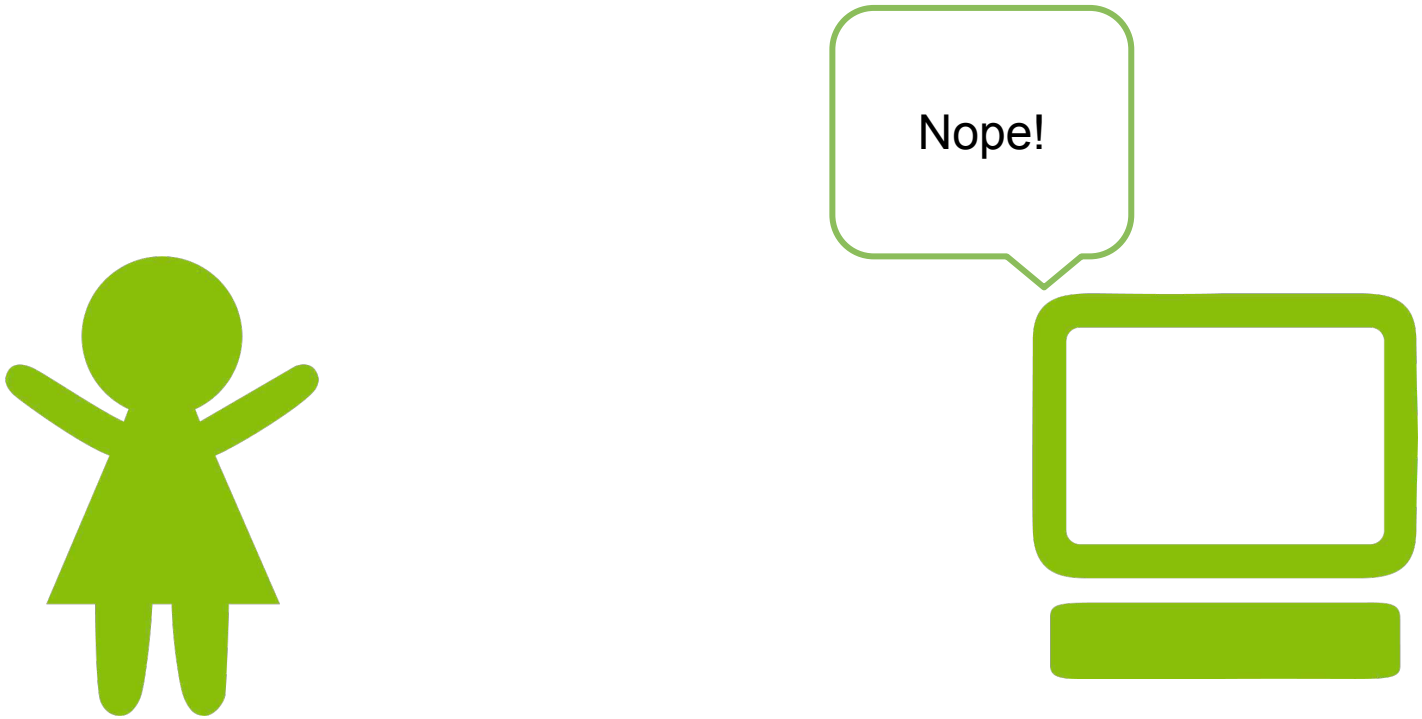
Looking for Events

We need to keep asking over and over again!



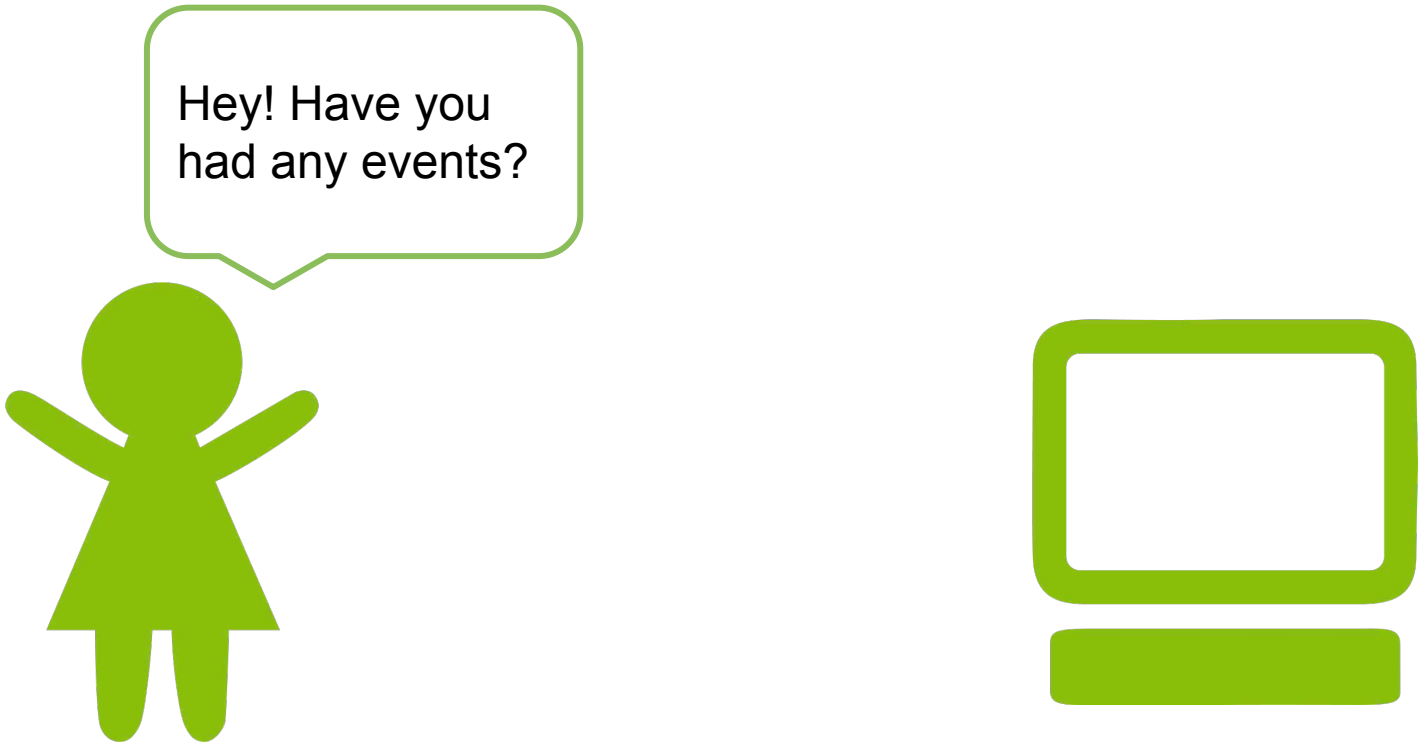
Looking for Events

We need to keep asking over and over again!



Looking for Events

We need to keep asking over and over again!

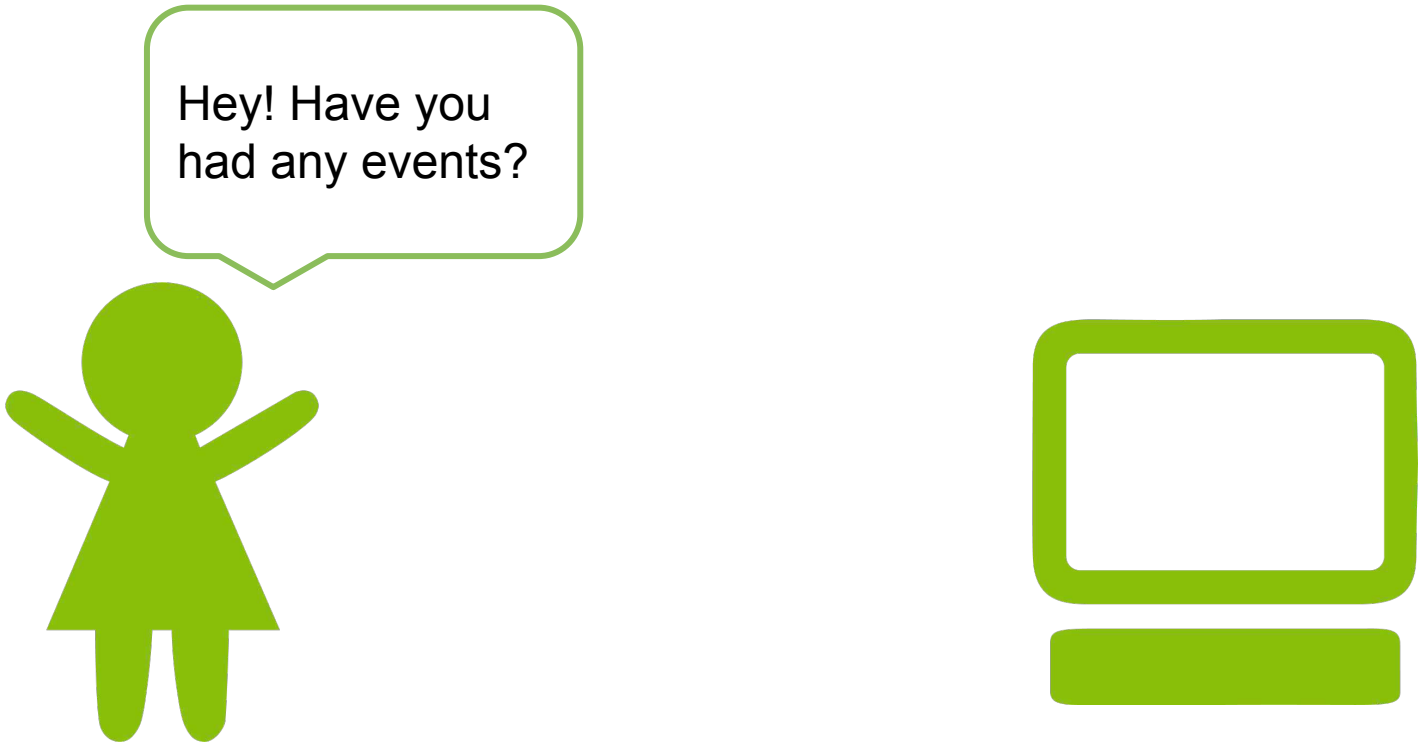


Looking for Events

We need to keep asking over and over again!

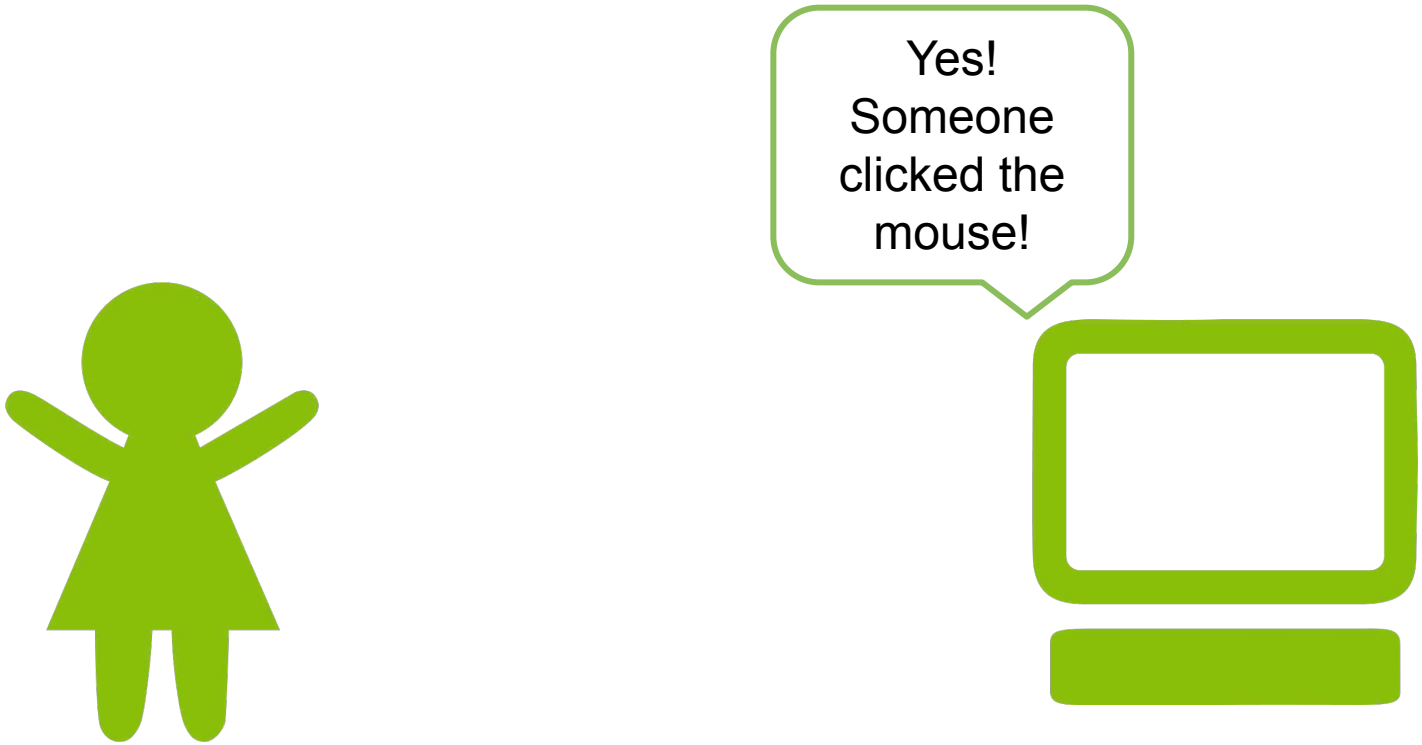
Looking for Events

We need to keep asking over and over again!



Looking for Events

We need to keep asking over and over again!



Loops

We can do something over and over again in our code using Loops! Like this:

```
while True:
  print(“Hello”)
```

What do you think this code does?

Loops

We can do something over and over again in our code using Loops! Like this:

```
while True:
  print(“Hello”)
```

What do you think this code does?

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

Loops

We can do something over and over again in our code using Loops! Like this:

```
while True:
  print(“Hello”)
```

It prints “Hello” forever! Let’s have a look at what it’s doing

Loops

```
while True:
  print("Hello")
```

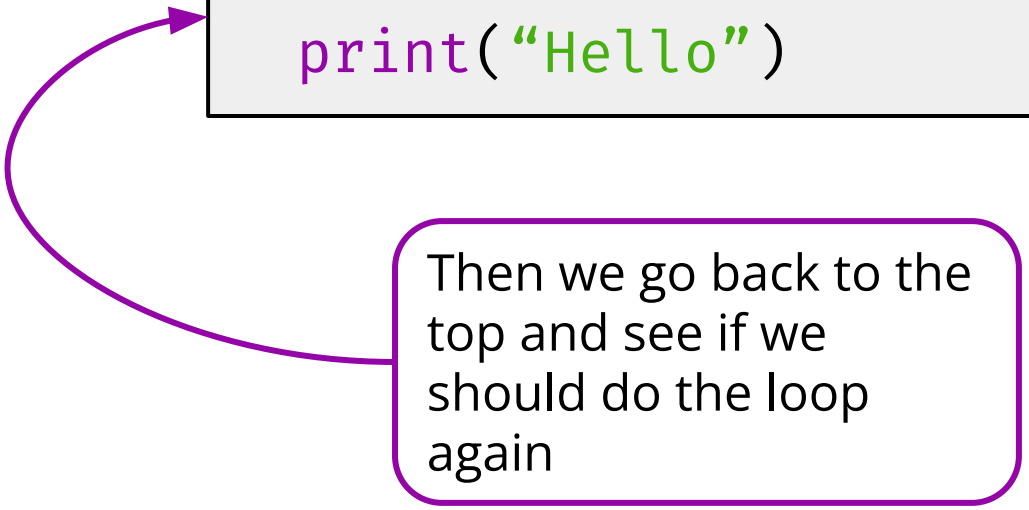
First, it checks to see if it should go into the loop. Because we wrote True here it will **always** go into the loop

Loops

```
while True:
   print(“Hello”)
```

Loops

```
while True:
  print("Hello")
```



Loops

```
while True:
  print("Hello")
```

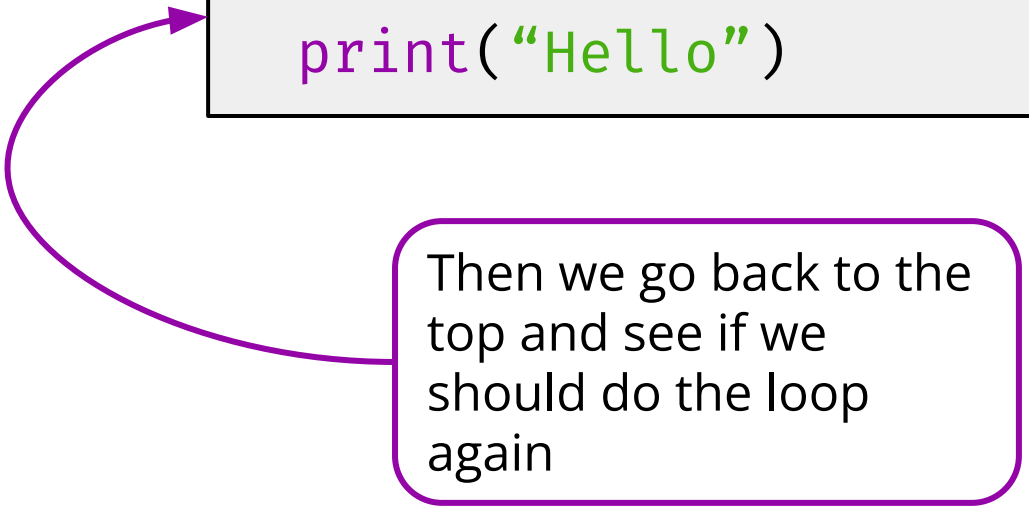
Because we wrote True here it will **always** go into the loop

Loops

```
while True:
   print(“Hello”)
```

Loops

```
while True:
  print("Hello")
```



Loops

This pattern keeps going on and on forever! (or until you quit the program)

```
while True:
  print(“Hello”)
```

Looking for Events

Now that we understand that we need to keep asking over and over, let's have another look at that code!

Looking for Events

Now that we understand that we need to keep asking over and over, let's have another look at that code!

```
while True:
  new_event = event.poll()
```

This line checks to see if there is a new event and saves it in a variable called new_event

Looking for Events

Now that we understand that we need to keep asking over and over, let's have another look at that code!

```
while True:
    new_event = event.poll()
```

This line tells python to do it over and over. It's called a loop!

This line checks to see if there is a new event and saves it in a variable called new_event

Looking for Events

```
while True:
    new_event = event.poll()
```

First we enter the loop here

Looking for Events

```
while True:
  new_event = event.poll()
```

Hey! Have you had any events?

Then we ask if there is a new event

Looking for Events

```
while True:
  new_event = event.poll()
```

Then we ask if there is a new event

Nope!

Looking for Events

```
while True:
  new_event = event.poll()
```

Then we go back to the top and do it again

Looking for Events

```
while True:
  new_event = event.poll()
```

Hey! Have you had any events?

Now we're doing this line again!

Looking for Events

```
while True:
  new_event = event.poll()
```

Nope!

Now we're doing this line again!

Finding Events

Okay so now we know how to ask for new Events. But what do we do when we find one?

Finding Events

Okay so now we know how to ask for new Events. But what do we do when we find one?

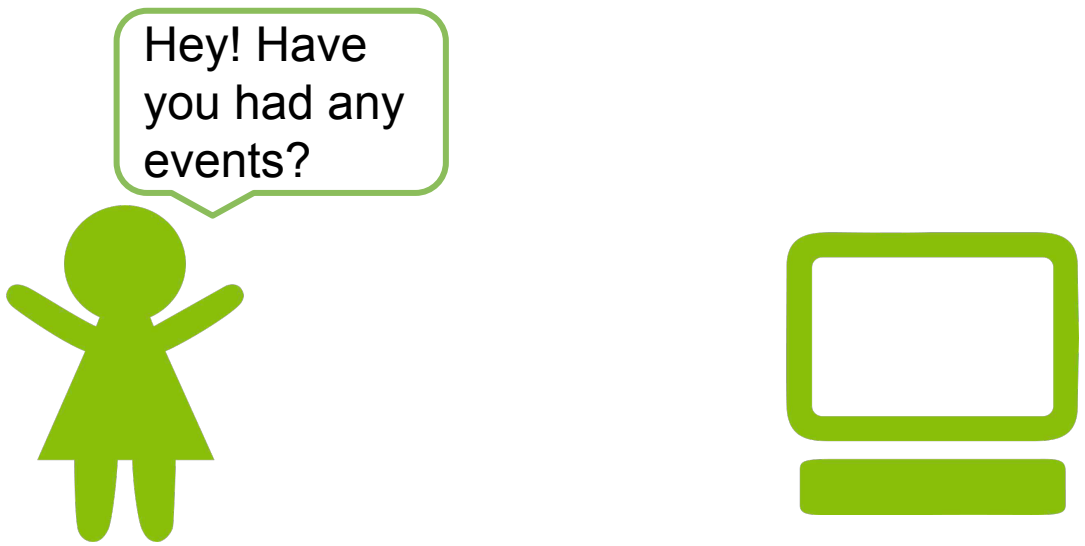
```
while True:
  new_event = event.poll()
  if new_event.type == KEYDOWN:
    print("You pressed a key!")
```

This if statement checks if the type of event was a KEY on the keyboard being pressed DOWN

Finding Events

```
while True:
    new_event = event.poll()
    if new_event.type == KEYDOWN:
        print("You pressed a key!")
```

First we check if there are any events



Finding Events

```
while True:
    new_event = event.poll()
    if new_event.type == KEYDOWN:
        print("You pressed a key!")
```

Then we check what type of event it was

Yep! It was a KEYDOWN event

Finding Events

```
while True:
    new_event = event.poll()
    if new_event.type == KEYDOWN:
        print("You pressed a key!")
```

If it's the event we want then we print this line

Pressing Keys

But we want to know *which* key they pressed! Not just if they pressed any key on the keyboard!

Pressing Keys

But we want to know *which* key they pressed! Not just if they pressed any key on the keyboard!

```
while True:
  new_event = event.poll()
  if new_event.type == KEYDOWN and new_event.key == K_SPACE:
    print("You pressed the space key!")
```

This now also checks if they key was the SPACE key

Finding Events

```
while True:
  new_event = event.poll()
  if new_event.type == KEYDOWN and new_event.key == K_SPACE:
    print("You pressed the space key!")
```

First we check if there are any events

Finding Events

```
while True:
  new_event = event.poll()
  if new_event.type == KEYDOWN and new_event.key == K_SPACE:
    print("You pressed the space key!")
```

Then we check what type of event it was

And we check what key was pressed

Yep! It was a KEYDOWN event using the SPACE key!

Finding Events

```
while True:
  new_event = event.poll()
  if new_event.type == KEYDOWN and new_event.key == K_SPACE:
    print("You pressed the space key!")
```

Now we know what the event is we can print

Project time!

The **key** to doing the next part was all in these slides

Try to do the next Part

In the **event** of confusion, the tutors will be around to help!

Pygame Collisions!

Pygame Collisions

Our game is looking great so far! But it would be even better if the things in our game could react when something collides with them!

Pygame already knows how to work this out, let's learn how to do it!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

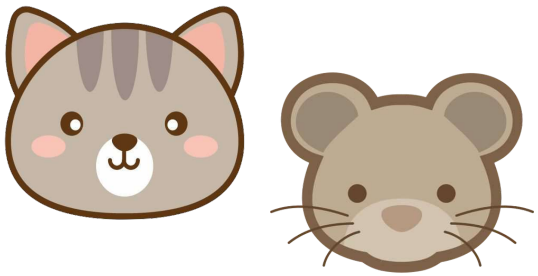
Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!



Cat & Mouse

Let’s have a look at this example where we have a game where the cat has to catch the mouse!

Cat & Mouse

Let's have a look at a little bit of the code from this game. We've left out some of it to keep it short

```
while True:
  cat = screen.blit(cat_image, (cat_x, cat_y))
  mouse = screen.blit(mouse_image, (mouse_x, mouse_y))
  display.update()
```

Cat & Mouse

Let's have a look at a little bit of the code from this game. We've left out some of it to keep it short

```
while True:
  cat = screen.blit(cat_image, (cat_x, cat_y))
  mouse = screen.blit(mouse_image, (mouse_x, mouse_y))
  display.update()
```

What does this code do?

Cat & Mouse

Let's have a look at a little bit of the code from this game. We've left out some of it to keep it short

```
while True:
  cat = screen.blit(cat_image, (cat_x, cat_y))
  mouse = screen.blit(mouse_image, (mouse_x, mouse_y))
  display.update()
```

What does this code do?

It blits the cat image and the mouse image to the screen and then updates the display!

Colliderect

We need to be able to tell when our cat collides with our mouse! Let's have a look at the code to do that

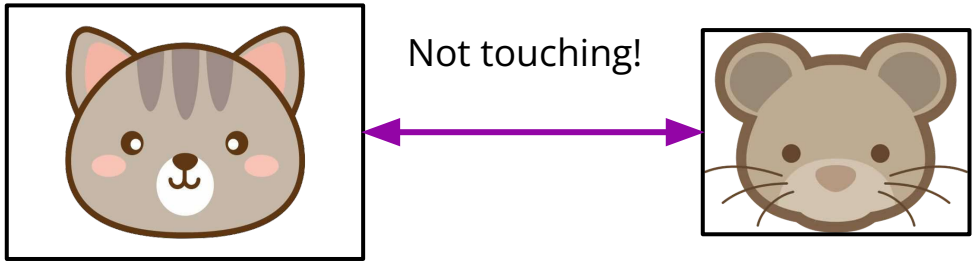
```
while True:
  cat = screen.blit(cat_image, (cat_x, cat_y))
  mouse = screen.blit(mouse_image, (mouse_x, mouse_y))
  display.update()

  if cat.colliderect(mouse):
    print("I win!")
```

Colliderect

Let's take a closer look!

```
while True:
  cat = screen.blit(cat_image, (cat_x, cat_y))
  mouse = screen.blit(mouse_image, (mouse_x, mouse_y))
  display.update()

  if cat.colliderect(mouse):
    print("I win!")
```

This if statement checks if the cat's rectangle collides (touches) the mouse's rectangle



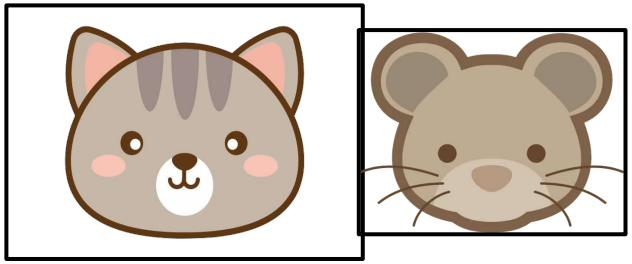
Colliderect

Let's take a closer look!

```
while True:
  cat = screen.blit(cat_image, (cat_x, cat_y))
  mouse = screen.blit(mouse_image, (mouse_x, mouse_y))
  display.update()

  if cat.colliderect(mouse):
    print("I win!")
```

This if statement checks if the cat's rectangle collides (touches) the mouse's rectangle

Touching!



Project time!

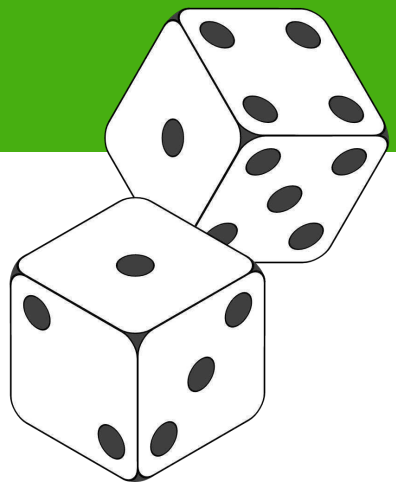
Now we can **collide** our knowledge with the workbook and do the next part!

Try to do the next Part

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!

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

3. Choose randomly! Try it a few times!

```
>>> random.choice(shopping_list)
```

Using the random module

You can also assign your random choice to a variable

```
>>> import random
>>> shopping_list = ["Bread", "Chocolate", "Ice Cream",
    "Pizza"]
>>> random_food = random.choice(shopping_list)
>>> print(random_food)
```

Project Time!

Raaaaaaaaaandom! Can you handle that?

Let's try use it in our project!

Try to do the next Part

The tutors will be around to help!

Classes

What is an object?

What do you think an object is?

What is an object?

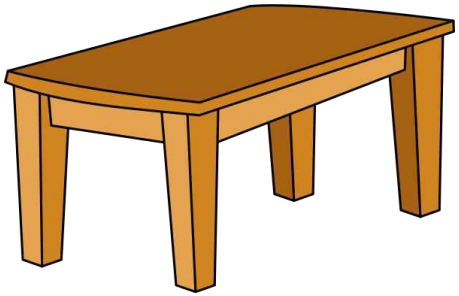
What do you think an object is?

What is an object?

What do you think an object is?

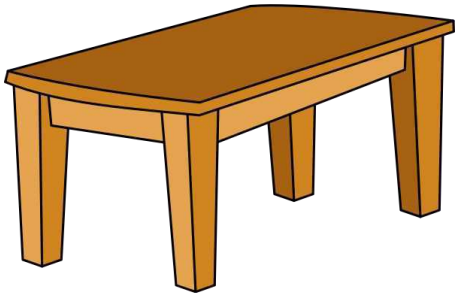
What is an object?

What do you think an object is?



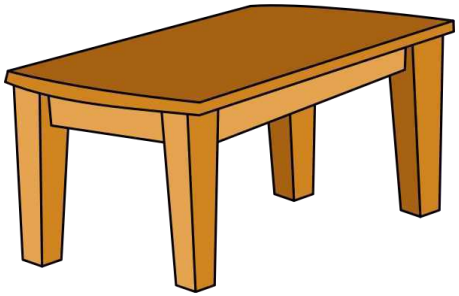
What is an object?

What do you think an object is?



What is an object?

What do you think an object is?



What is an object in code?

An object is something that we know information about and that can do things

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What information might we know about a cat?

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What information might we know about a cat?

Name

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What information might we know about a cat?

Name

Age

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What information might we know about a cat?

Name

Age

Colour

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What information might we know about a cat?

Name **Owner**

Age

Colour

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What information might we know about a cat?

Name

Owner

Age

Weight

Colour

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What information might we know about a cat?

Name

Owner

Age

Weight

Colour

Microchip #

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What things might a cat do?

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What things might a cat do?

Meow

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What things might a cat do?

Meow

Eat

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What things might a cat do?

Meow

Eat

Scratch

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What things might a cat do?

Meow

Eat

Scratch

Sleep

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What things might a cat do?

Meow

Eat

Scratch

Sleep

Purr

What is an object in code?

An object is something that we know information about and that can sometimes do things

Like a cat!

What things might a cat do?

Meow

Sleep

Eat

Purr

Scratch

Jump

What does that look like in Python?

Let’s have a look at how we might make a Cat object in Python code!

What does that look like in Python?

Let's have a look at how we might make a Cat object in Python code!

```
class Cat():
    def __init__(self, name, age, colour):
        self.name = name
        self.age = age
        self.colour = colour
```

Here we tell python that we are making a new type (or class) of object called Cat

What does that look like in Python?

Let's have a look at how we might make a Cat object in Python code!

__init__ is how we tell Python how to make a new Cat

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour
```

What does that look like in Python?

Let's have a look at how we might make a Cat object in Python code!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour
```

Here we tell Python what information we need to know about the Cat

Note: self is special and we always need it

What does that look like in Python?

Let’s have a look at how we might make a Cat object in Python code!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour
```

Here we save the information we got so we can use it again

What does that look like in Python?

How do we make a new Cat?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

emmy = Cat(“Emmy”, 3, “Dark brown”)
```

What does that look like in Python?

What does this print out?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

emmy = Cat(“Emmy”, 3, “Dark brown”)
print(emmy.name)
print(emmy.age)
print(emmy.colour)
```

What does that look like in Python?

What does this print out?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

emmy = Cat(“Emmy”, 3, “Dark brown”)
print(emmy.name)
print(emmy.age)
print(emmy.colour)
```

```
Emmy
3
Dark Brown
```

What about doing things?

We said an object was something with information that could sometimes do things. Our Cat object doesn't do anything right now - let's add a way for it to meow!

What about doing things?

We said an object was something with information that could sometimes do things. Our Cat object doesn't do anything right now - let's add a way for it to meow!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def meow(self):
    print("Meow")
```

What about doing things?

What does this code do?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def meow(self):
    print(“Meow”)

emmy = Cat(“Emmy”, 3, “Dark brown”)
emmy.meow()
```

What about doing things?

What does this code do?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def meow(self):
    print(“Meow”)

emmy = Cat(“Emmy”, 3, “Dark brown”)
emmy.meow()
```

```
Meow
```

What else can it do?

Let’s have our cat have a Birthday that makes it get older by 1 year!

What else can it do?

Let’s have our cat have a Birthday that makes it get older by 1 year!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def meow(self):
    print(“Meow”)

  def birthday(self):
    self.age = self.age + 1
```

What else can it do?

What does this code do?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def meow(self):
    print(“Meow”)

  def birthday(self):
    self.age = self.age + 1

emmy = Cat(“Emmy”, 3, “Dark brown”)
emmy.birthday()
print(emmy.age)
```

What else can it do?

What does this code do?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def meow(self):
    print(“Meow”)

  def birthday(self):
    self.age = self.age + 1

emmy = Cat(“Emmy”, 3, “Dark brown”)
emmy.birthday()
print(emmy.age)
```

4

I have more than 1 cat!

Emmy has a little sister, Saphira! Let’s add her to our code too!

```
cat1 = Cat(“Emmy”, 3, “Dark brown”)
cat2 = Cat(“Saphira”, 1, “Grey”)
```

Cat Crime!

There has been a cat crime!

One of the cats has gotten on the kitchen counter and eaten some of my lunch!

They both look innocent but they left a hair behind at the scene of the crime! Let’s write some code to work out who did it

Cat Crime

Who did it??

```
cat1 = Cat(“Emmy”, 3, “Dark brown”)
cat2 = Cat(“Saphira”, 1, “Grey”)

hair_colour = “Grey”

if hair_colour == cat1.colour:
  print(“That hair belongs to”, cat1.name)
elif hair_colour == cat2.colour:
  print(“That hair belongs to”, cat2.name)
```

Cat Crime

Who did it??

```
cat1 = Cat(“Emmy”, 3, “Dark brown”)
cat2 = Cat(“Saphira”, 1, “Grey”)

hair_colour = “Grey”

if hair_colour == cat1.colour:
  print(“That hair belongs to”, cat1.name)
elif hair_colour == cat2.colour:
  print(“That hair belongs to”, cat2.name)
```

```
That hair belongs to Saphira
```

Classes

Arguments and Returns

Remember our Cat class?

Let’s have another look at our Cat class!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def meow(self):
    print(“Meow”)

  def birthday(self):
    self.age = self.age + 1
```

Rename that Cat

Sometimes we rename our cat - like when we first bring them home from the shelter (Emmy’s original name was Tawny)

Rename that Cat

Let’s add some code that lets us rename our cat!

Rename that Cat

Let's add some code that lets us rename our cat!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def rename(self, new_name)
    self.name = new_name
```

Rename that Cat

Let's add some code that lets us rename our cat!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def rename(self, new_name)
    self.name = new_name
```

This is called an argument - it's some extra information that we can give our object to help it do its job!

Rename that Cat

What does this code do?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def rename(self, new_name)
    self.name = new_name

cat1 = Cat("Tawny", 3, "Dark brown")
print(cat1.name)
cat1.rename("Emmy")
print(cat1.name)
```

Rename that Cat

What does this code do?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def rename(self, new_name)
    self.name = new_name

cat1 = Cat("Tawny", 3, "Dark brown")
print(cat1.name)
cat1.rename("Emmy")
print(cat1.name)
```

Tawny

Emmy

Give it back!

Arguments are how we give information to the class, but how do we get information back from them? We can use a return

Give it back!

Let’s have a look at an example!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def matches_hair(self, hair_colour):
    return self.colour == hair_colour
```

Give it back!

Let’s have a look at an example!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def matches_hair(self, hair_colour):
    return self.colour == hair_colour
```

Here we use an argument to tell the class what hair colour we are trying to match with

Give it back!

Let’s have a look at an example!

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def matches_hair(self, hair_colour):
    return self.colour == hair_colour
```

Here we return whether or not the hair_colour is equal to our cats colour

Give it back!

What does matches_hair return here?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def matches_hair(self, hair_colour):
    return self.colour == hair_colour

cat1 = Cat(“Emmy”, 3, “Dark brown”)
cat1.matches_hair(“Grey”)
```

Give it back!

What does matches_hair return here?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def matches_hair(self, hair_colour):
    return self.colour == hair_colour

cat1 = Cat("Emmy", 3, "Dark brown")
cat1.matches_hair("Grey")
```

False

Give it back!

What about now?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def matches_hair(self, hair_colour):
    return self.colour == hair_colour

cat1 = Cat(“Emmy”, 3, “Dark brown”)
cat1.matches_hair(“Dark brown”)
```

Give it back!

What about now?

```
class Cat():
  def __init__(self, name, age, colour):
    self.name = name
    self.age = age
    self.colour = colour

  def matches_hair(self, hair_colour):
    return self.colour == hair_colour

cat1 = Cat(“Emmy”, 3, “Dark brown”)
cat1.matches_hair(“Dark brown”)
```

True

Cat Crimes!

Let's look at our cat crimes code again and see if we can make it better with our new code!

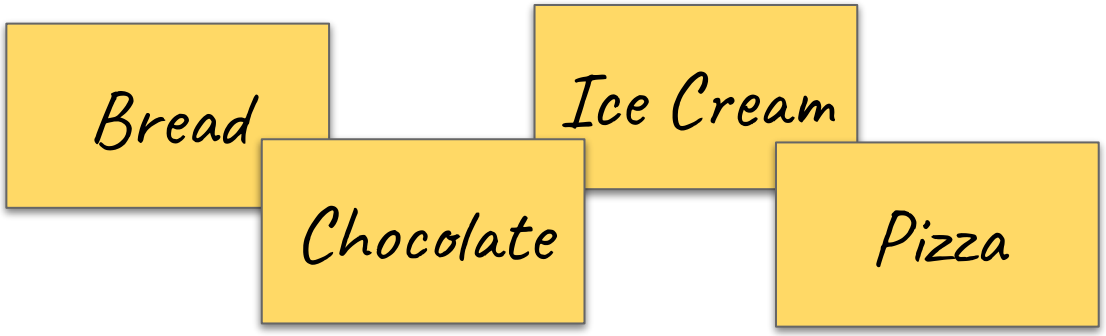
```
cat1 = Cat("Emmy", 3, "Dark brown")
cat2 = Cat("Saphira", 1, "Grey")

hair_colour = "Grey"

if cat1.matches_colour(hair_colour):
  print("That hair belongs to", cat1.name)
elif cat2.matches_colour(hair_colour):
  print("That hair belongs to", cat2.name)
```

Lists

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!



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"]
```

You can put (almost) anything into a list

- You can have a list of integers

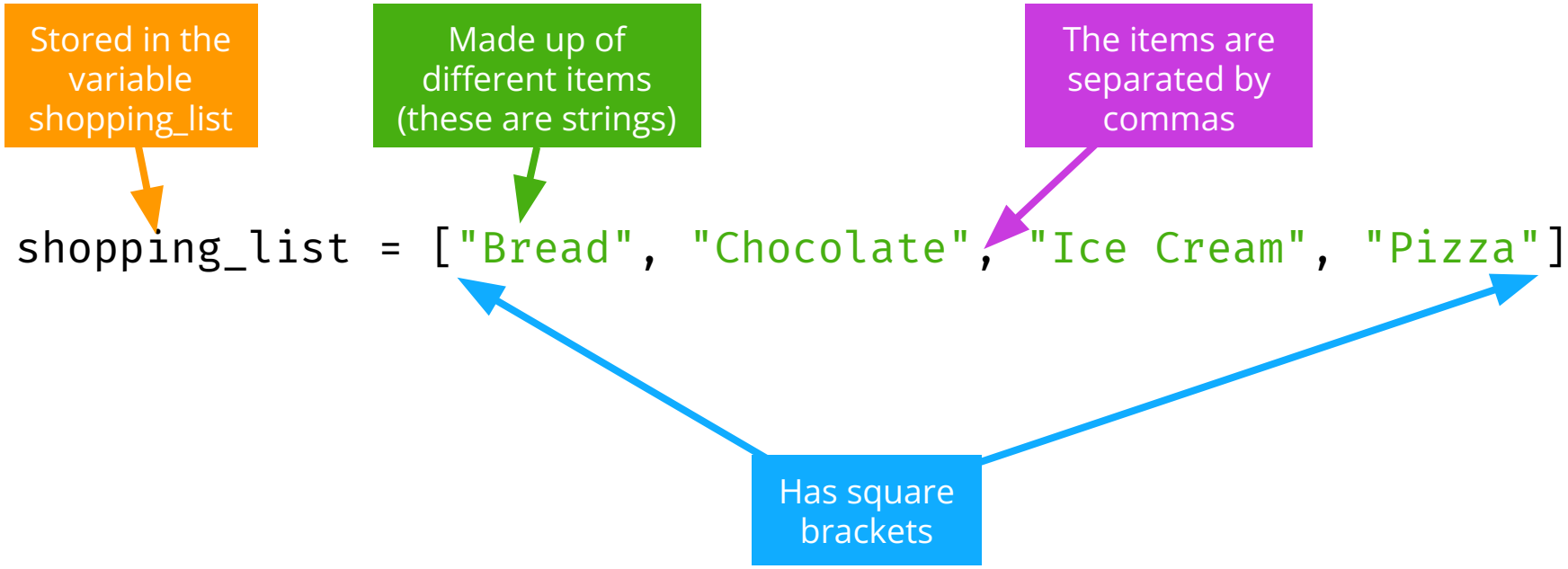
```
>>> primes = [1, 2, 3, 5, 11]
```

- You can have lists with mixed integers and strings

```
>>> mixture = [1, 'two', 3, 4, 'five']
```

- But this is almost never a good idea! You should be able to treat every element of the list the same way.

List anatomy



Try this!

1. Make a list of your favourite things

```
>>> faves = ['books', 'butterfly', 'chocolate',
             'skateboard']
```

2. Use `print` to print out your favourite things list

3. Can you make it print on one line?

```
These are a few of my favourite things ['books',
'butterfly', 'chocolate', 'skateboard']
```

>> **Hint: use print with a comma!**

Accessing Lists!

The favourites `list` holds four strings in order.

We can count out the items using index numbers!

0	1	2	3
			

Remember: Indices start from zero!

Accessing Lists

We access the items in a `list` with an index such as `[0]`:

```
>>> faves[0]
'books'
```

What code do you need to access the second item in the list?

Accessing Lists

We access the items in a `list` with an index such as `[0]`:

```
>>> faves[0]
'books'
```

What code do you need to access the second item in the list?

```
>>> faves[1]
'butterfly'
```

0	**[1]**	2	3
			

Going Negative

Negative indices count backwards from the end of the `list`:

```
>>> faves[-1]
'skateboard'
```

What would `faves[-2]` return?

Going Negative

Negative indices count backwards from the end of the `list`:

```
>>> faves[-1]
'skateboard'
```

What would `faves[-2]` return?

```
>>> faves[-2]
'chocolate'
```

-4 | -3 | **[-2]** | -1




Falling off the edge

Python complains if you try to go past the end of a `list`

```
>>> faves = ['books', 'butterfly', 'chocolate',
             'skateboard']
>>> faves[4]


Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Updating items!

We can also update things in a list:

```
>>> faves = ['books', 'butterfly',
                 'chocolate', 'skateboard']
>>> faves[2]
'chocolate'
>>> faves[2] = 'lollipops'
>>> faves
```

Updating items!

We can also update things in a list:

```
>>> faves = ['books', 'butterfly',
                 'chocolate', 'skateboard']
>>> faves[2]
'chocolate'
>>> faves[2] = 'lollipops'
>>> faves
['books', 'butterfly', 'lollipops', 'skateboard']
```




Removing items!

We can remove items from the list if they're no longer needed!

What if we decided that we didn't like butterflies anymore?

```
>>> faves
['books', 'butterfly', 'lollipops', 'skateboard']
>>> faves.remove('butterfly')
```

What does this list look like now?

Removing items!

We can remove items from the list if they're no longer needed!

What if we decided that we didn't like butterflies anymore?

```
>>> faves
['books', 'butterfly', 'lollipops', 'skateboard']
>>> faves.remove('butterfly')
```

What does this list look like now?

```
['books', 'lollipops', 'skateboard']
```

Adding items!

We can also add new items to the list!

What if we decided that we also liked programming?

```
>>> faves
['books', 'lollipops', 'skateboard']
>>> faves.append('programming')
```

What does this list look like now?

Adding items!

We can also add new items to the list!

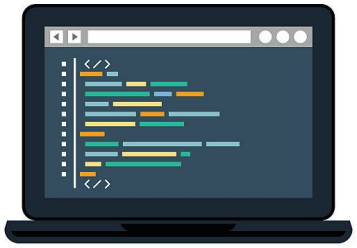
What if we decided that we also liked programming?

```
>>> faves
['books', 'lollipops', 'skateboard']
>>> faves.append('programming')
```

What does this list look like now?

```
['books', 'lollipops', 'skateboard', 'programming']
```



List of lists!

You really can put anything in a list, even more lists!

We could use a list of lists to store different sports teams!

```
tennis_pairs = [
    ["Alex", "Emily"], ["Kass", "Annie"], ["Amara", "Viv"]
]
```

Get the first pair in the list

```
>>> first_pair = tennis_pairs[0]
>>> ["Alex", "Emily"]
```

Now we have the first pair handy, we can get the first the first player of the first pair

```
>>> fist_player = first_pair[0]
>>> "Alex"
```

Project time!

You now know all about lists!

Let’s put what we learnt into our project

Try to do the next Part

The tutors will be around to help!

Files

Filing it away!

What happens if we want to use different data in our program? What if that data is too big to write in with the keyboard?

We'd have to change our code!!

It would be better if we could keep all our data in a file and just be able to pick and choose what file we wanted to play today!

people.txt

```
Aleisha,brown,black,hat
Brittany,blue,red,glasses
Charlie,green,brown,glasses
Dave,blue,red,glasses
Eve,green,brown,glasses
Frankie,hazel,black,hat
George,brown,black,glasses
Hannah,brown,black,glasses
Isla,brown,brown,none
Jackie,hazel,blonde,hat
Kevin,brown,black,hat
Luka,blue,brown,none
```

Opening files!

To get access to the stuff inside a file in python we need to **open** it!

That doesn't mean clicking on the little icon!

```
f = open("test.txt", "r")
```

You'll now be able to read the things in `f`

A missing file causes an error

Here we try to open a file that doesn't exist:

```
f = open("missing.txt", "r")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IOError: [Errno 2] No such file or
directory: 'missing.txt'
```

You can read a whole file into a string

```
>>> f = open("haiku.txt", "r")
>>> my_string = f.read()
>>> my_string
'Wanna go outside.\nOh NO!
Help! I got outside!\nLet me
back inside!

>>> print(my_string)
Wanna go outside.
Oh NO! Help! I got outside!
Let me back inside!
```

haiku.txt

```
Wanna go outside.
Oh NO! Help! I got outside!
Let me back inside!
```

You can also read in one line at a time

You can use a for loop to only get 1 line at a time!

```
f = open("haiku.txt", "r")
for line in f:
    print(line)
```

```
Wanna go outside.

Oh NO! Help! I got outside!

Let me back inside!
```

Why is there an extra blank line each time?

Chomping off the newline

The newline character is represented by '\n':

```
print('Hello\nWorld')
Hello
World
```

We can remove it from the lines we read with .strip()

```
x = 'abc\n'
x.strip()
'abc'
```

x.strip() is safe as lines without newlines will be unaffected

Reading and stripping!

```
for line in open("haiku.txt", "r"):
    line = line.strip()
    print(line)
```

```
Wanna go outside.
Oh NO! Help! I got outside!
Let me back inside!
```

No extra lines!

Write to files!

You can also write to files!

```
f = open("newfile.txt", "a")
f.write("This is my new line!")
```

Notice we used `"a"` instead of `"r"`? We opened it in append mode!

This will create a new file if it doesn't exist, and add the new line to the bottom of the file. This is called "appending"!

Write to files!

You can also write over files!

```
f = open("newfile.txt", "a")
f.write("This is my new file!")
```

Notice we used `"w"` instead of `"a"`? We opened it in write mode!

This will create a new file if it doesn't exist, and **delete** everything in the file and replace it with what we write.

Closing Time

Always remember to close your file when you're finished with it:

```
f.close()
```

This will close your file and save it.

Using with!

This is a special trick for opening files!

```
with open("words.txt", "r") as f:
  for line in f:
    print(line.strip())
```

It automatically closes your file for you!

It's good when you are writing files in python!

Project time!

I hope you `file`d that knowledge away

Use it in the next section of the project!

Try to do the next Part

The tutors will be around to help!

Alternative slides using `with`

Filing it away!

What happens if we want to use different data in our program? What if that data is too big to write in with the keyboard?

We'd have to change our code!!

It would be better if we could keep all our data in a file and just be able to pick and choose what file we wanted to play today!

people.txt

```
Aleisha,brown,black,hat
Brittany,blue,red,glasses
Charlie,green,brown,glasses
Dave,blue,red,glasses
Eve,green,brown,glasses
Frankie,hazel,black,hat
George,brown,black,glasses
Hannah,brown,black,glasses
Isla,brown,brown,none
Jackie,hazel,blonde,hat
Kevin,brown,black,hat
Luka,blue,brown,none
```

Opening files!

To get access to the stuff inside a file in python we need to **open** it!

That doesn't mean clicking on the little icon!

```
with open("test.txt", "r") as f:
```

You'll now be able to read the things in f

If your file is in the same location as your code you can just use the name!

A missing file causes an error

Here we try to open a file that doesn't exist:

```
with open("missing.txt", "r") as f:
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IOError: [Errno 2] No such file or
directory: 'missing.txt'
```

You can read in one line at a time

You can use a for loop to read 1 line at a time!

```
with open("haiku.txt", "r") as f:
    for line in f:
        print(line)
```

```
Wanna go outside.

Oh NO! Help! I got outside!

Let me back inside!
```

Why is there an extra blank line each time?

Chomping off the newline

The newline character is represented by '\n':

```
print('Hello\nWorld')
Hello
World
```

We can remove it from the lines we read with .strip()

```
x = 'abc\n'
x.strip()
'abc'
```

x.strip() is safe as lines without newlines will be unaffected

Reading and stripping!

```
with open("haiku.txt", "r") as f:
    for line in f:
        line = line.strip()
        print(line)
```

```
Wanna go outside.
Oh NO! Help! I got outside!
Let me back inside!
```

No extra lines!

Write to files!

You can also write to files!

```
with open("newfile.txt", "a") as f:
    f.write("This is my new line!\n")
```

Notice we used `"a"` instead of `"r"`? We opened it in write mode!

This will create a new file if it doesn't exist, and add the new line to the bottom of the file. This is called "appending"!