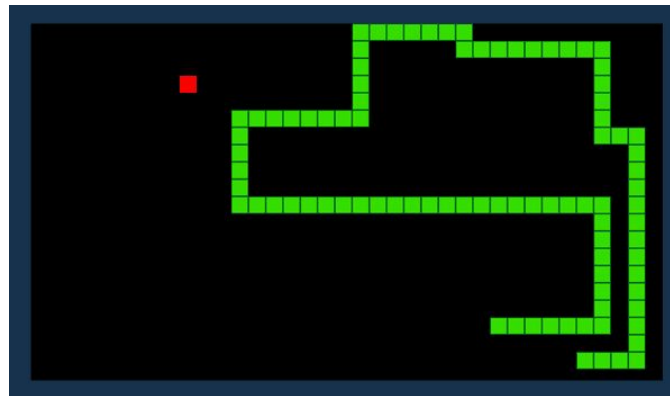
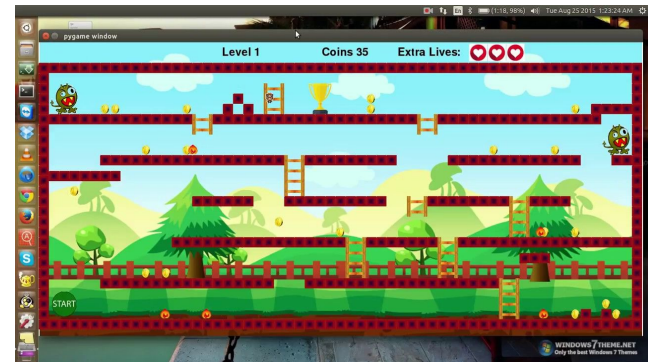
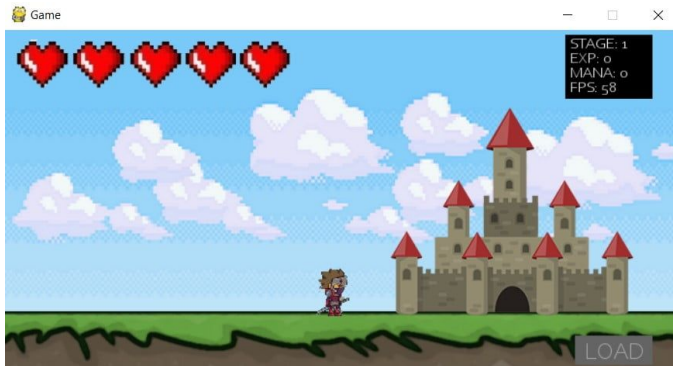


# Intro to Pygame



# 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)
```




This line makes a screen  
that is 500 pixels wide and  
400 pixels high

# 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 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 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()
```

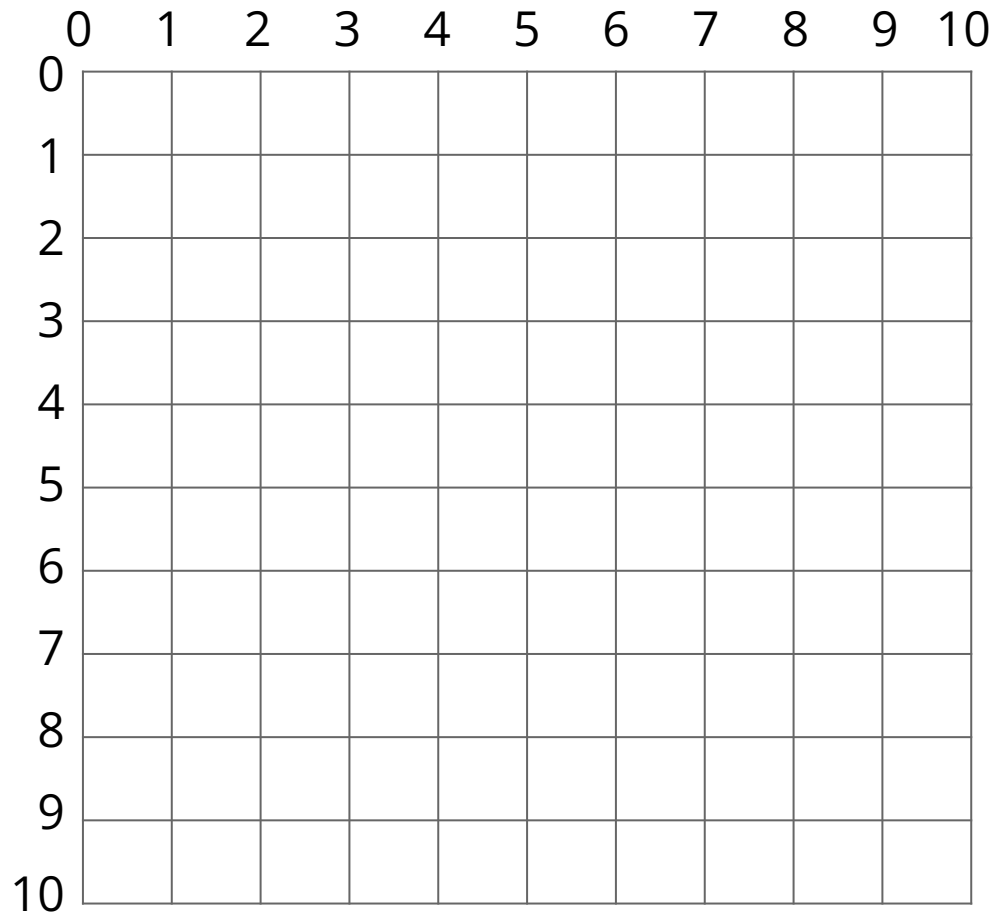
This line tells Pygame to update our display, which will show our new image

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

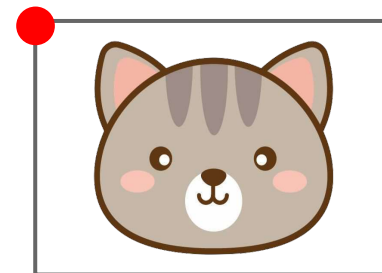
When you have an image in pygame like this:



Pygame thinks of it like a rectangle like this:

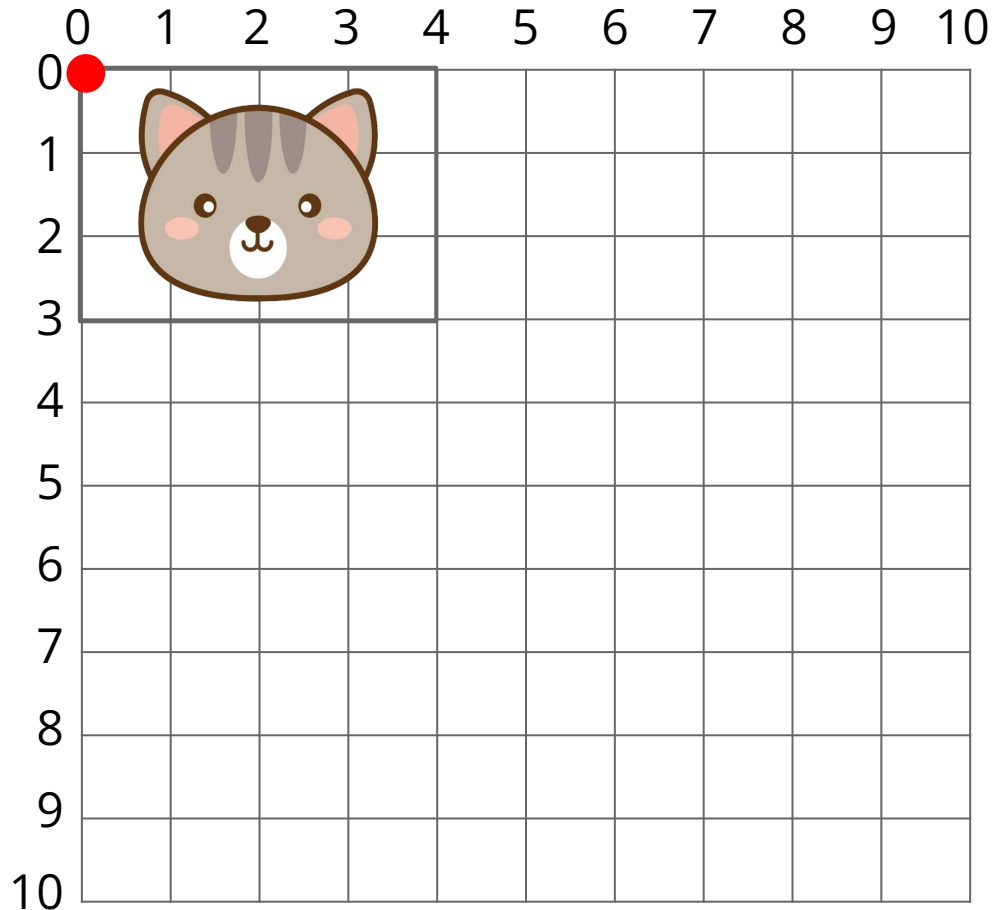


And the coordinates for the image are the top left corner like this:



# Pygame Coordinates

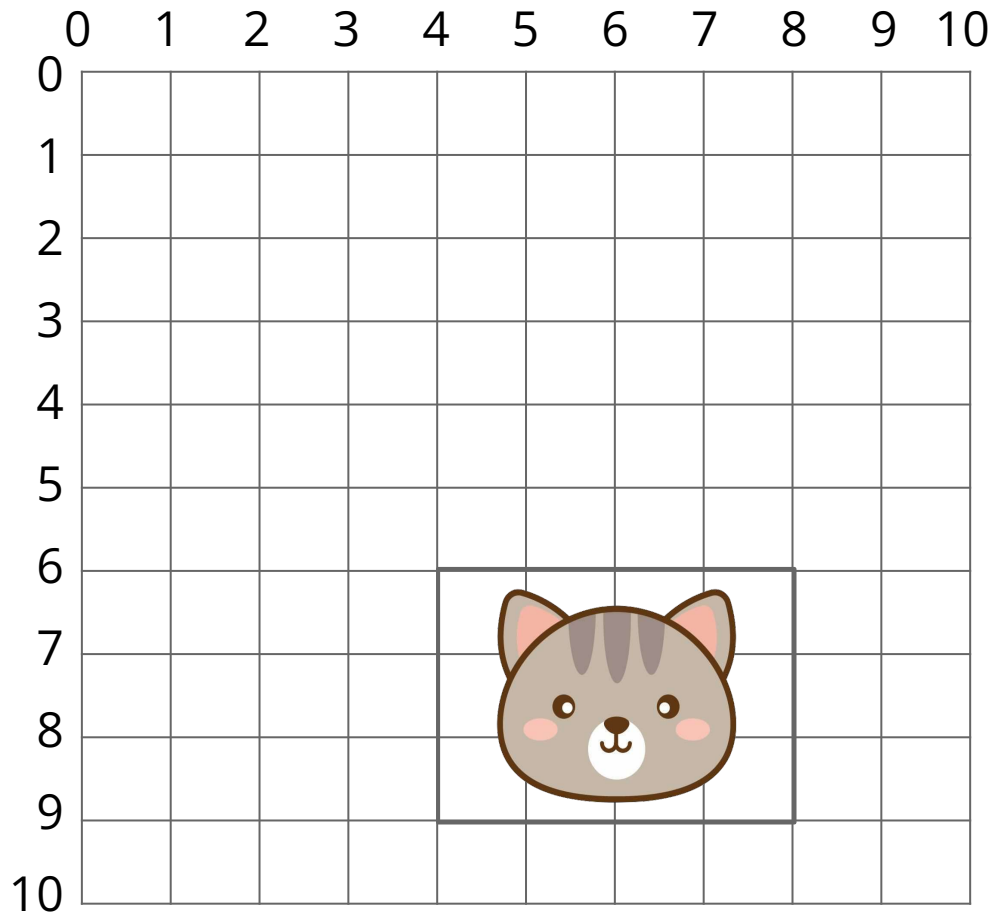
So to if we wanted our cat image to be on the top left we would use the coordinates (0, 0)



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?

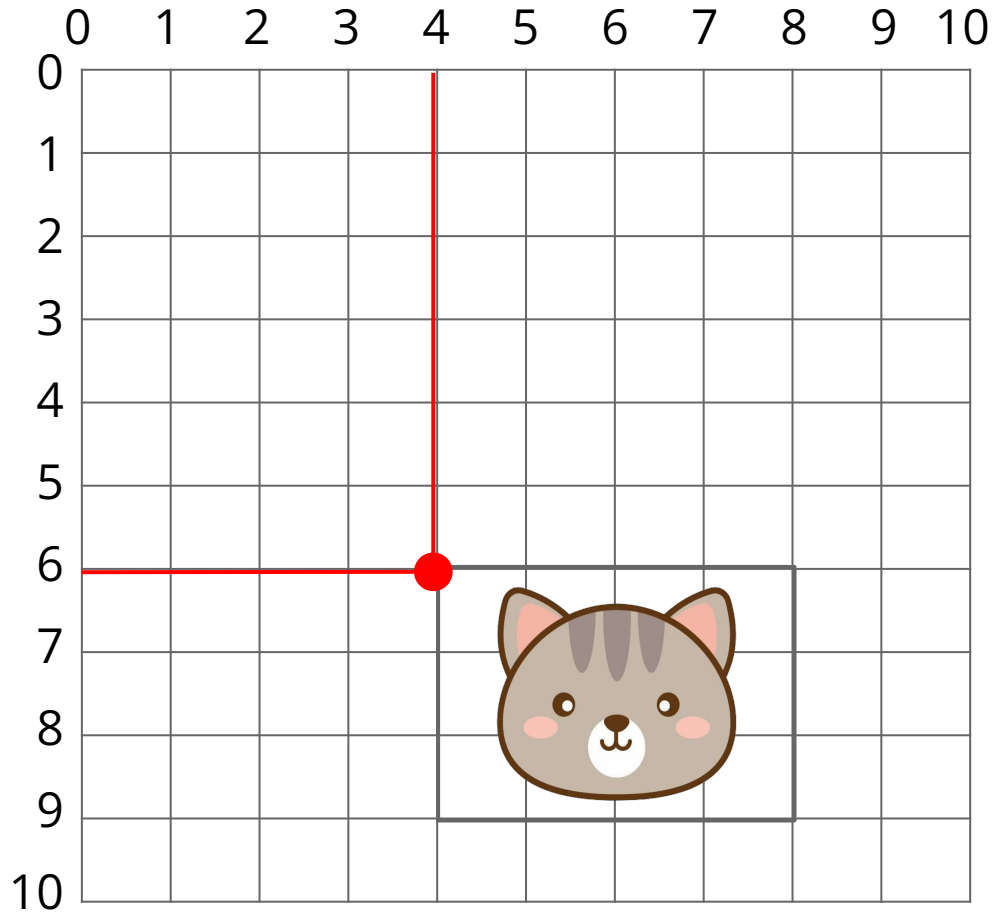


Remember that when we write coordinates we say the x (horizontal) number first and the y (vertical) number second

# Pygame Coordinates

What are the coordinates of our image now?

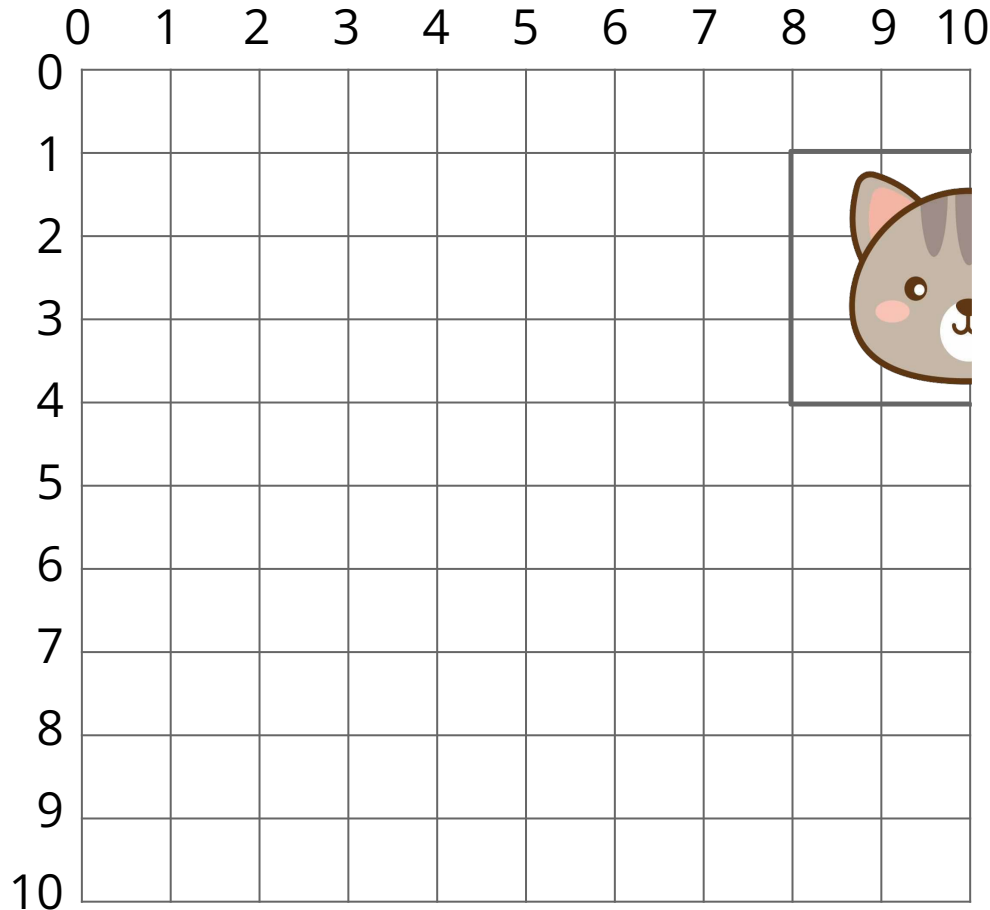
They are:  
(4, 6)



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?

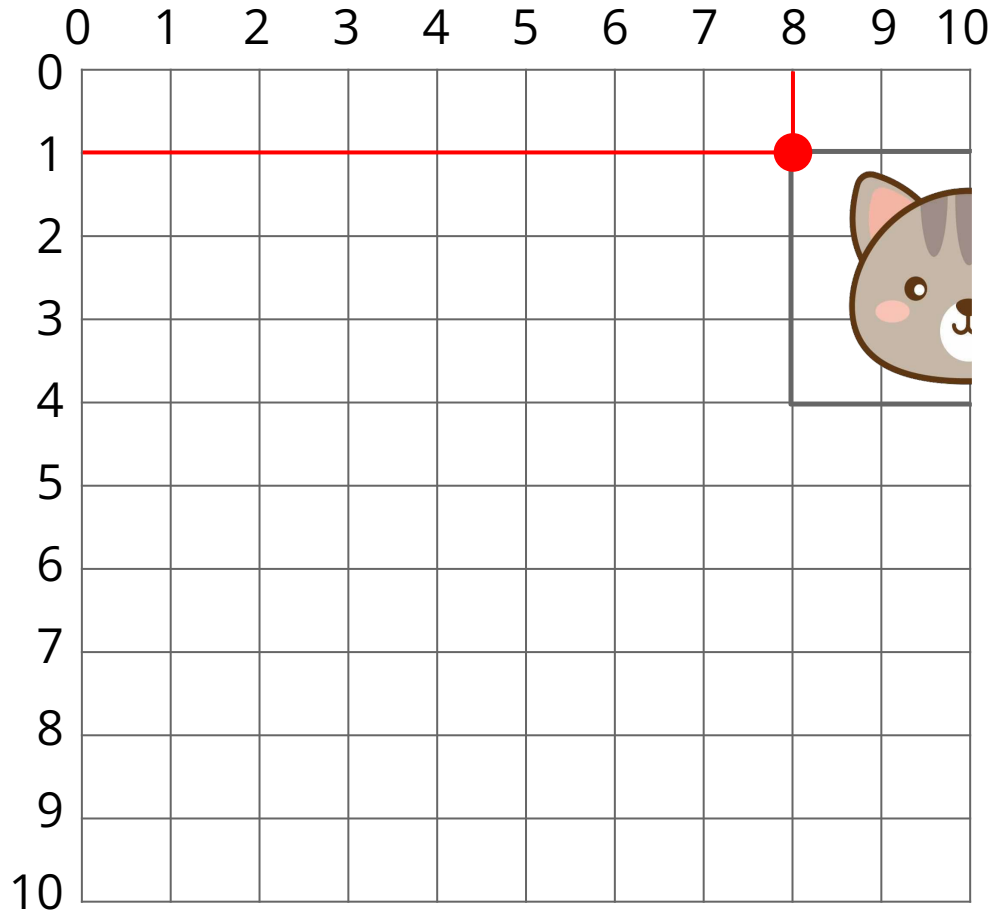


Sometimes we can put the image off the screen by giving coordinates that are far to the bottom or the right

# 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 **updated** 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()
```



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

# How do we use it?

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

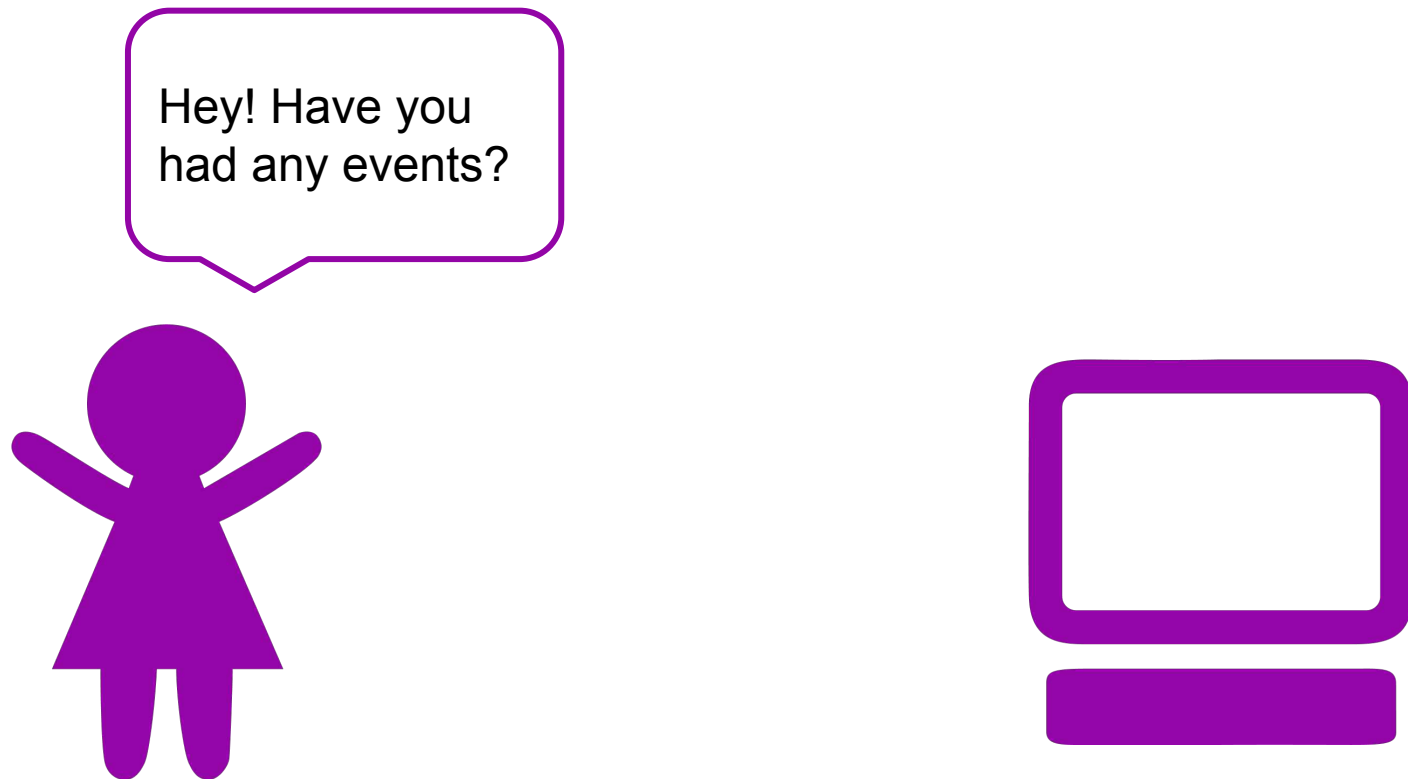
```
while True:  
    new_event = event.poll()
```

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

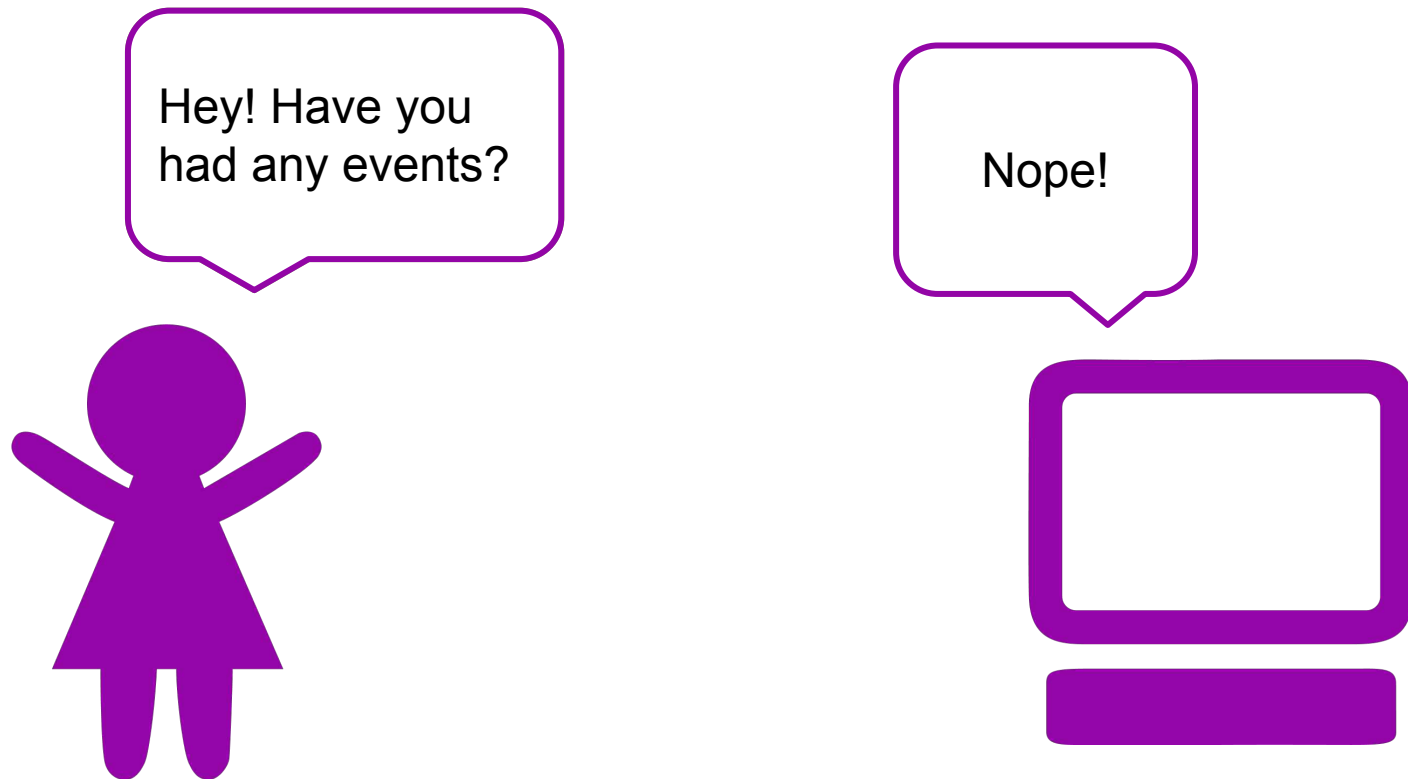
# Looking for Events

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



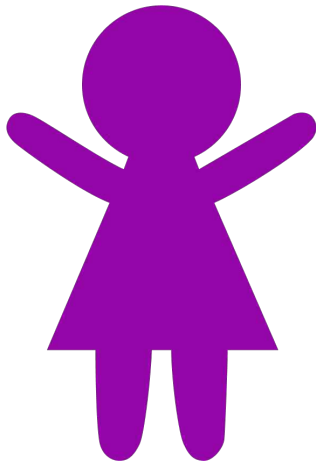
# Looking for Events

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





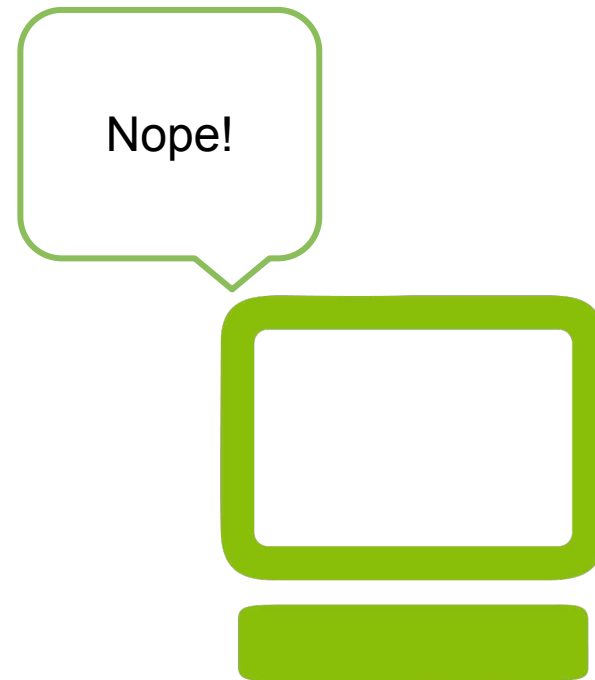
# Looking for Events

We need to keep asking over and over again!



# Looking for Events

We need to keep asking over and over again!



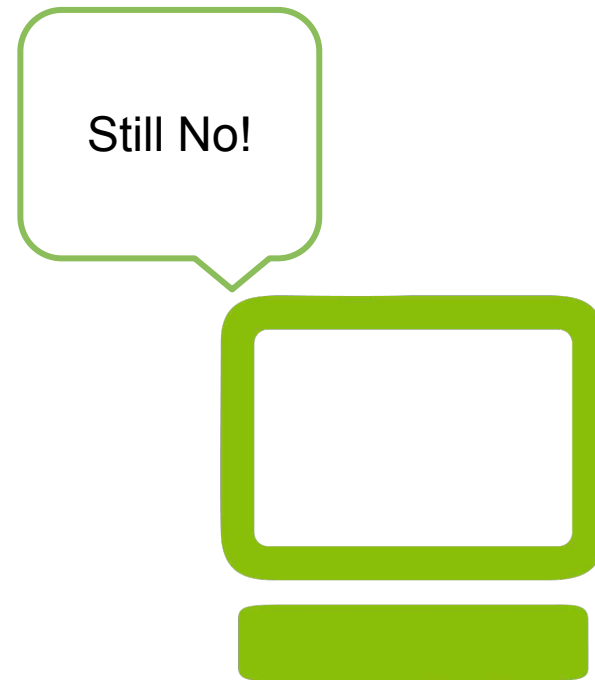
# Looking for Events

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!



Yes!  
Someone  
clicked the  
mouse!



# 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

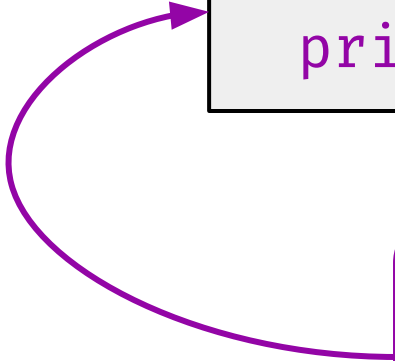
# Loops

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

Then we do whatever is **inside** the loop - we print "Hello"

# Loops

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



Then we go back to the top and see if we should do the loop again

# Loops

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

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

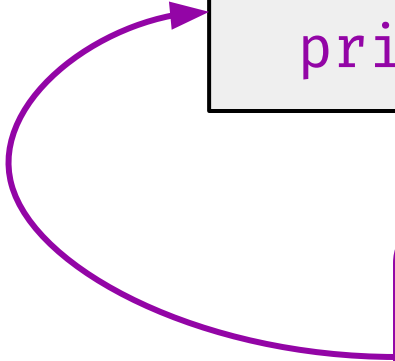
# Loops

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

Then we do whatever is **inside** the loop - we print "Hello"

# Loops

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



Then we go back to the top and see if we should do the loop again

# 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

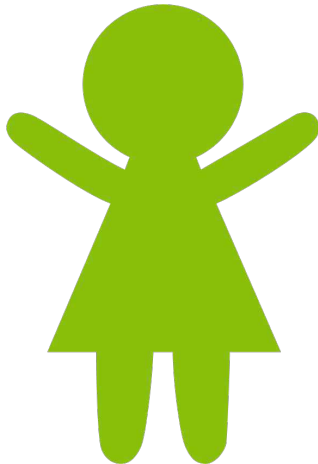


# Looking for Events

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

Nope!

Then we ask if  
there is a new  
event



# 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

Hey! Have  
you had 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

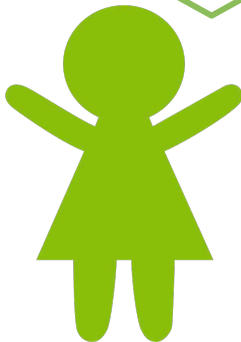


# Finding Events

```
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

You pressed  
a key!



# 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

Hey! Have  
you had 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

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

And we  
check  
what key  
was  
pressed



# 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

You pressed  
the space key!



# 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

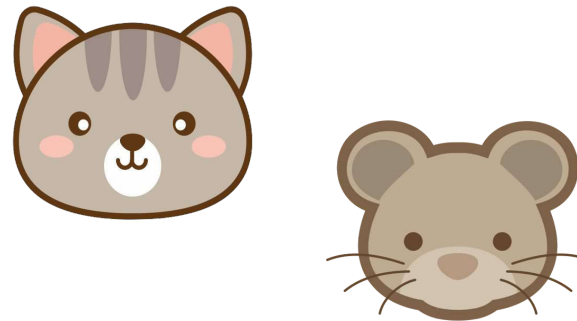
Let's have a look at this example where we have a game where the cat has to catch the mouse!





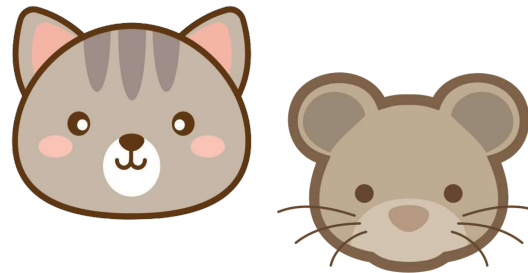
# Cat & Mouse

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



# Cat & Mouse

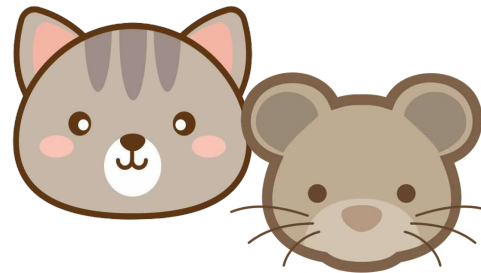
Let's have a look at this example where we have a game where the cat has to catch the mouse!



# Cat & Mouse

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

I win!



# 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!

# Collidect

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.collidect(mouse):
        print("I win!")
```

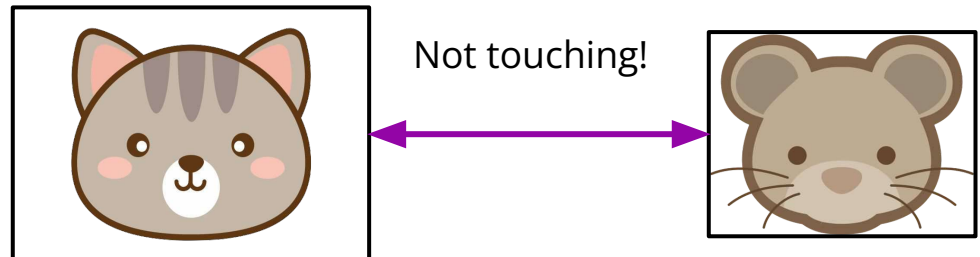
# Colliderect

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





# Collidect

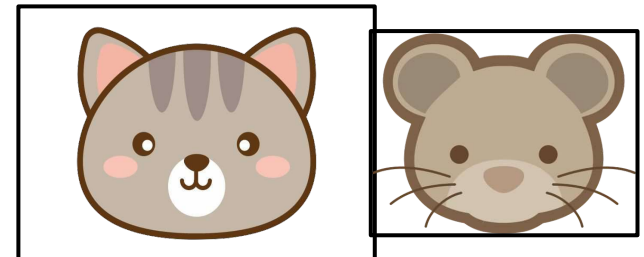
Let's take a closer look!

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

    if cat.collidect(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!

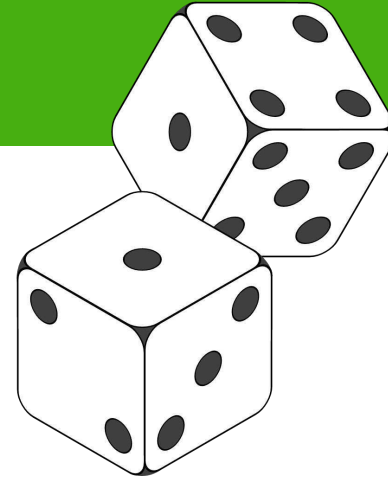
Random!

# That's so random!

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



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



We want the computer to be random sometimes!



# 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!

# 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 `"a`ppending"!



# Write to files!

You can also write over files!

```
f = open("newfile.txt", "w")  
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 **filed** 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 `"a`ppending"!