Welcome to GPN

Thank you to our Sponsors!

Platinum Sponsor:

Who are the tutors?

Who are you?

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!

Today's Project!

Flappy Bird!

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.

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.

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

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

☑ **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

★ **BONUS 4.3: Do some extra!**

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

Intro to Python

Let’s get coding!

Getting set up

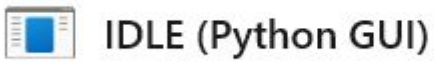
Go to your desktop and open the Flappy bird python

Flappy Bird Python

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

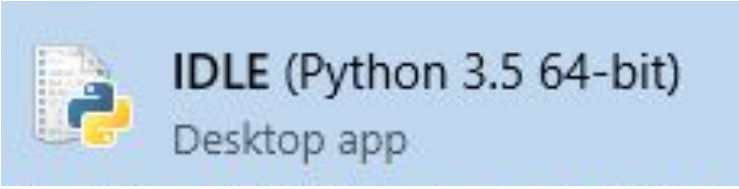
(This will download IDLE onto your desktop)

It should look like this

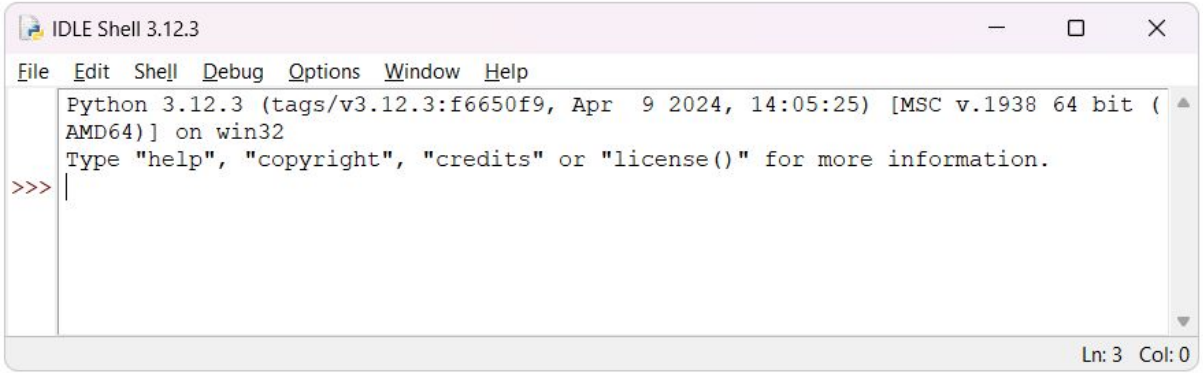


Where do we program? In IDLE

Once it's downloaded open IDLE.



You should get a screen that looks like this!



Make a mistake!

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!

SyntaxError: Invalid Syntax

Good work you made an error!

ImportError No module named humour

- 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
```

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!")
```

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?

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

It prints the words “hello world” onto the screen!

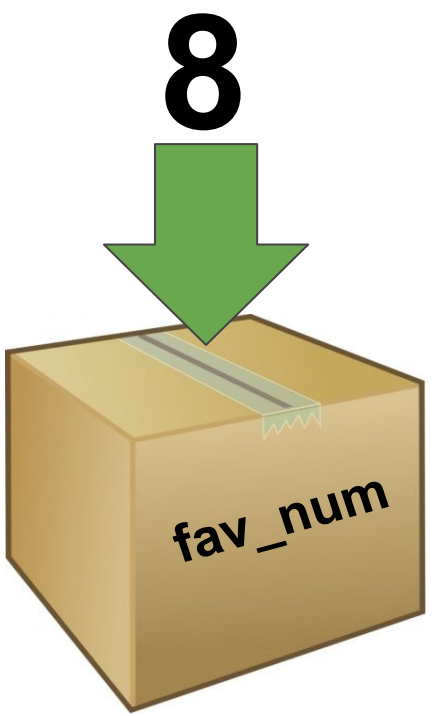
No Storing is Boring!

It's useful to be able to remember things for later!

Computers remember things in **"variables"**

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

Let's make our favourite number 8 today!



Variables

Instead of writing the number 8, we can write fav_num.


```
fav_num - 6
```

=> 2

```
fav_num + 21
```

=> 29

```
fav_num * 2
```

=> 16

```
fav_num / 2
```

=> 4

Variables

Instead of writing the number 8, we can write fav_num.


```
fav_num - 6
```

=> 2

```
fav_num + 21
```

=> 29

```
fav_num * 2
```

=> 16

```
f                2
```

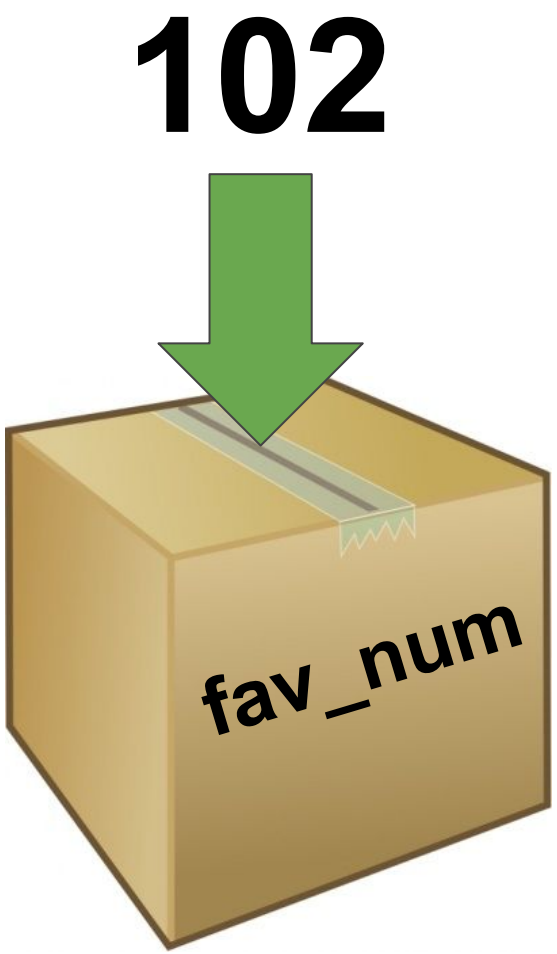
But writing 8 is much shorter than writing `fav_num`???

Variables

Variables are useful for storing things that change

(i.e. things that "vary" - hence the word "variable")

Try changing `fav_num` to **102**.



Variables

We're able to use our code for a new purpose, without rewriting everything:


```
fav_num - 6
```

=> 96

```
fav_num + 21
```

=> 123

```
fav_num * 2?
```

=> 204

```
fav_num / 2?
```

=> 51

Reusing variables

We can replace values in variables:

```
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:

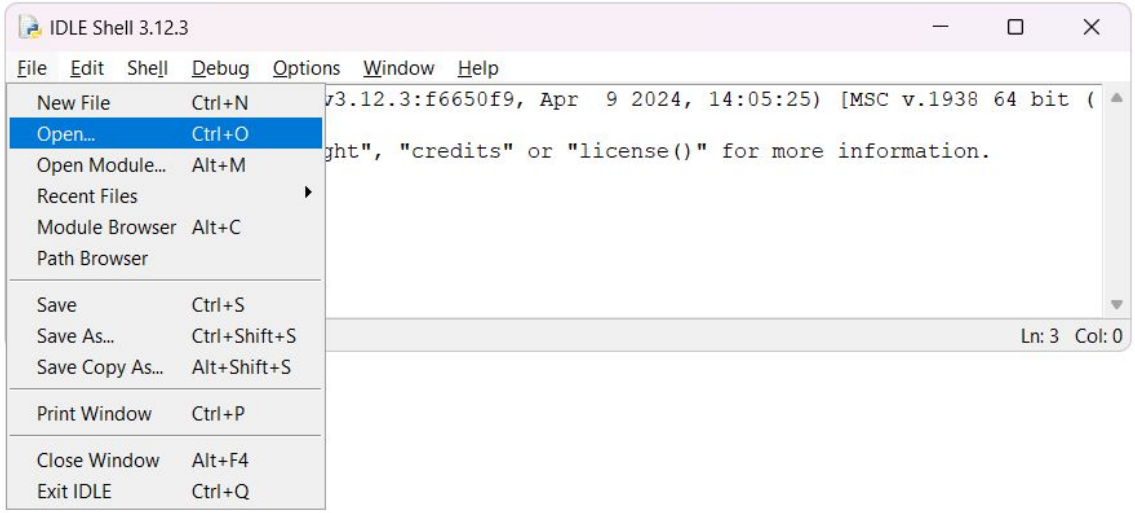
```
animal = "dog"
print("My favourite animal is a " + animal)
animal = "cat"
print("My favourite animal is a " + animal)
animal = animal + "dog"
print("My favourite animal is a " + animal)
```

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

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

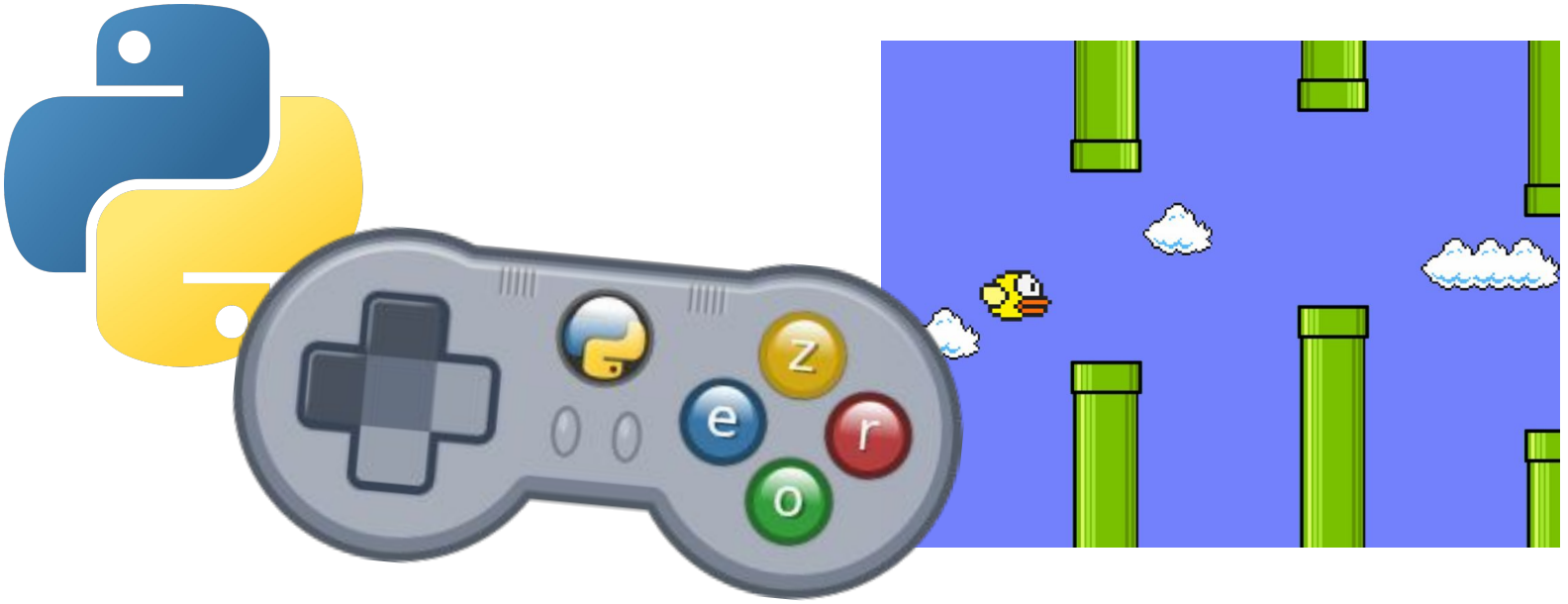
The tutors will be around to help!

Intro to PyGame Zero

Making it into a game!

What is Pygame Zero?

We use pygame zero to allow our code to do some cool things.



Pygame Zero Setup

The first thing we need to do to use pygame zero is to write this at the top of your file

```
>>> import pgzrun
```

Pygame Zero Setup

The first thing we need to do to use pygame zero is to write this at the top of 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

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!

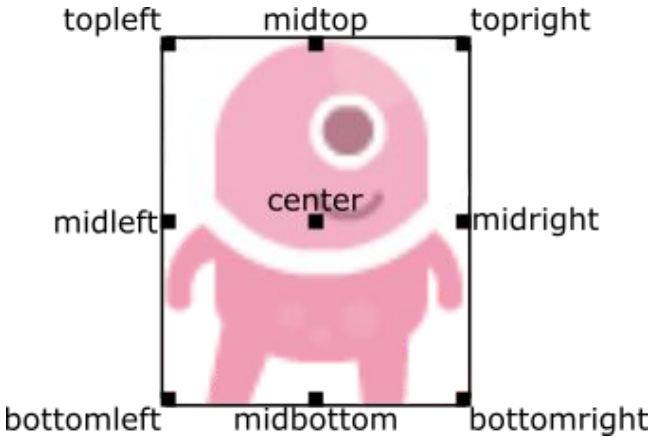
PyGame Zero images

Adding things to our screen!

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.

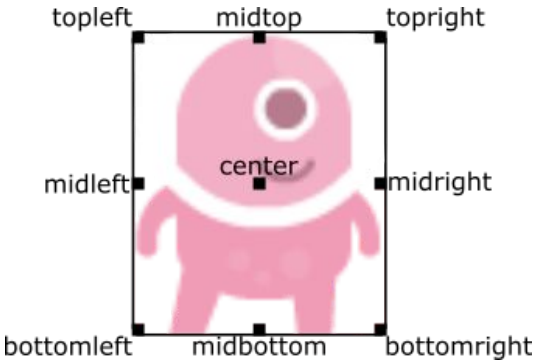


How to make an actor

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



How to make an actor

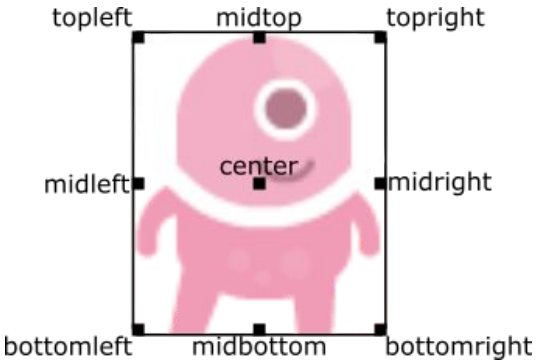
To make a new actor and tell Pygame zero what image it is you need to write the code:

```
>>> 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
```



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:

```
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
```

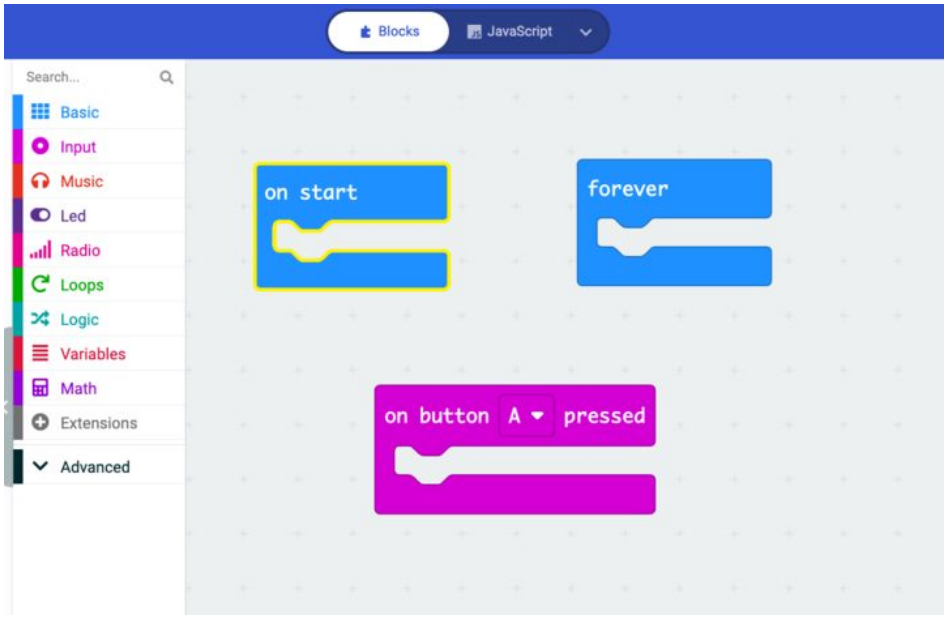
What is a function?

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.

Functions in blockly

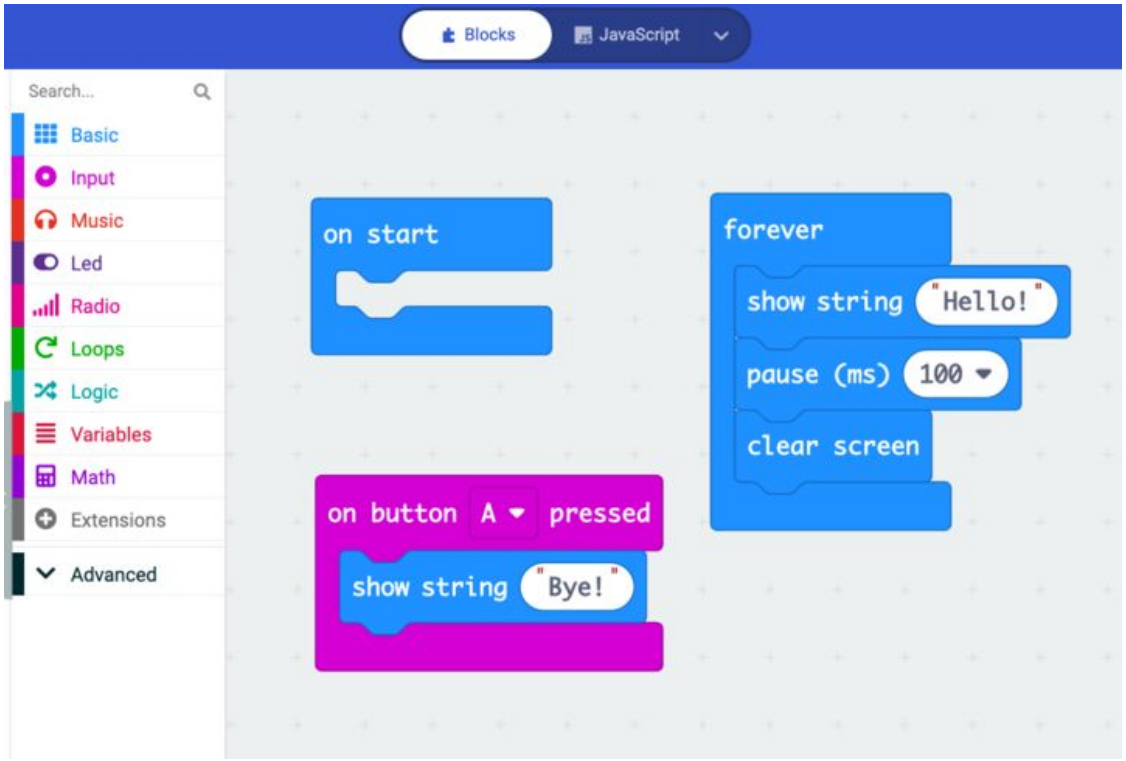
Here are some functions in blockly - maybe they seem familiar from school



Here any code you put in these boxes will get run every time they do

Functions in blockly

This is what the functions can look like with code in it...



We can do the same thing with code!

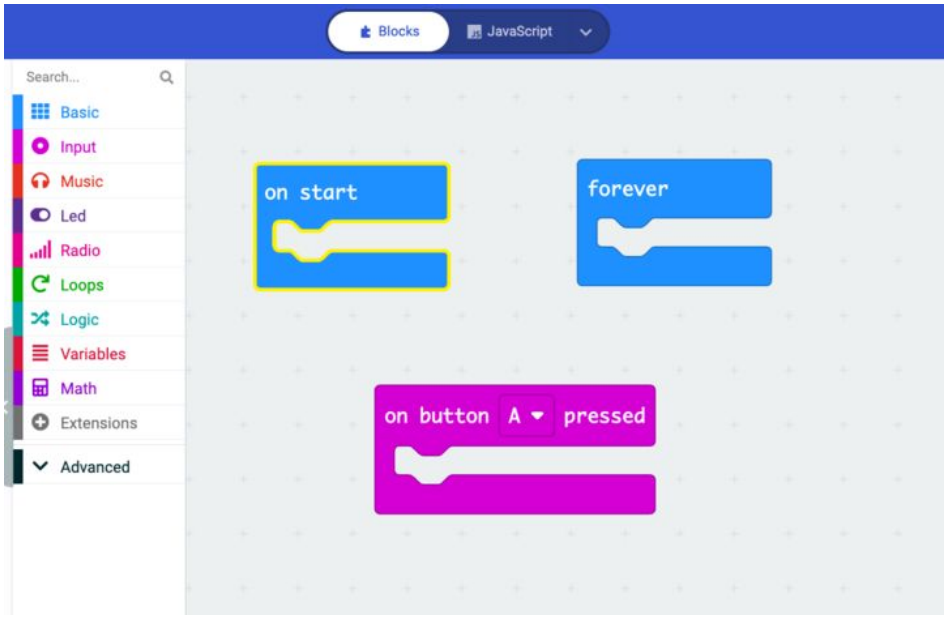
Some important code

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

```
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
```

Functions in blockly

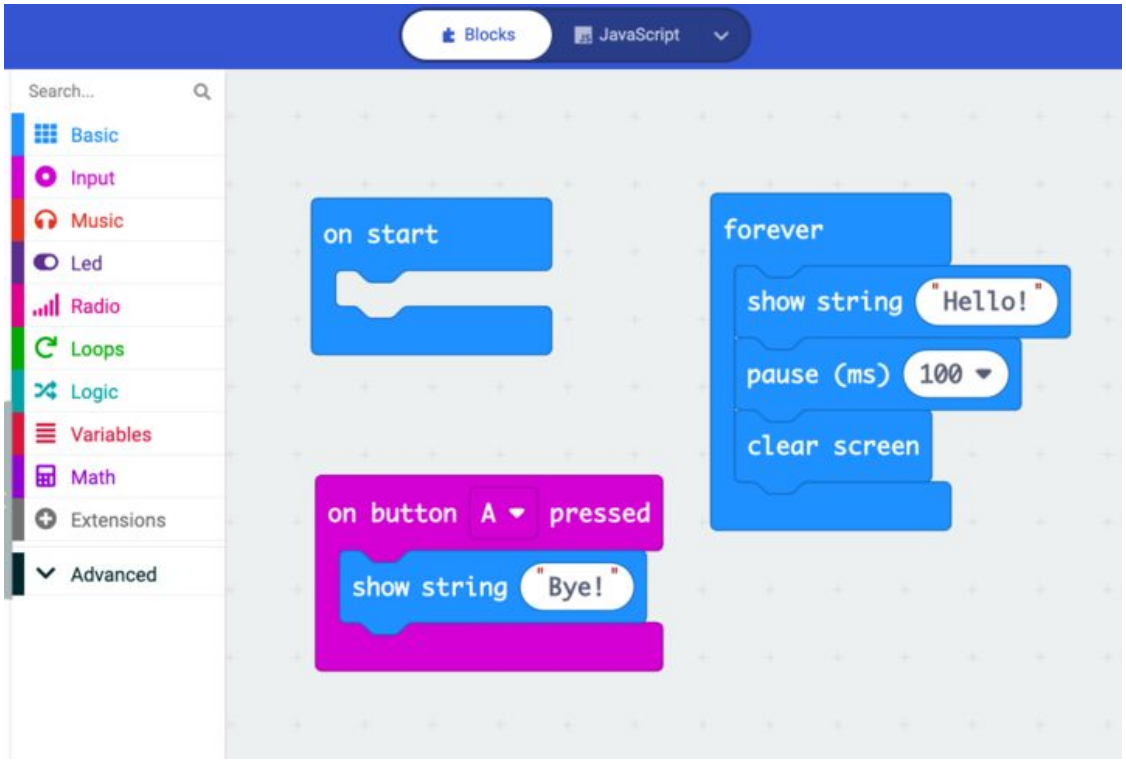
Here are some functions in blockly - maybe they seem familiar from school



Here any code you put in these boxes will get run every time they do

Functions in blockly

This is what the functions can look like with code in it...



We can do the same thing with code!

Some important code

Our special Pygame Zero functions are just like the blocks!

```
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()
```

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
```

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")
```

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!

Events and If Statements

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

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 small number")
```

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 small number")
```

That's the condition!

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 small number")
```

That's the condition!

Is it `True` that fave_num is less than 10?

- Well, fave_num is 5
- And it's `True` that 5 is less than 10
- So it is `True`!

Conditions

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

```
fave_num = 5
if True:
    print("That's a small number")
```

Put in the answer to the question

Is it `True` that fave_num is less than 10?

- Well, fave_num is 5
- And it's `True` that 5 is less than 10
- So it is `True`!

Conditions

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

```
fave_num = 5
if True:
    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`!

```
fave_num = 5
if True:
    print("That's a small number")
```

What do you think happens?

```
>>> That's a small number
```

Conditions

How about a different number???

```
fave_num = 9000
if fave_num < 10:
    print("That's a small number")
```

Conditions

How about a different number???

```
fave_num = 9000
if fave_num < 10:
    print("That's a small number")
```

What do you think happens?

```
>>>
```

Conditions

How about a different number???

```
fave_num = 9000
if fave_num < 10:
    print("What's a small number")
```

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

Nothing!

Elif statements

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

```
fav_number = 90
if fav_number < 10:
  print("That's a small number")
else:
  print("That's a big number")
```

What happens?

```
>>>
```

Elif statements

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

```
fav_number = 90
if fav_number < 10:
    print("That's a small number")
else:
    print("That's a big number")
```

What happens?

```
>>> That's a big number
```

Elif statements

elif statements means we can give specific instructions for other scenarios

```
fav_number = 90
if fav_number < 10:
  print("That's a small number")
elif fav_number > 10:
  print("That's a big number")
else:
  print("That number is just right!")
```

What happens?

```
>>>
```

Elif statements

elif statements means we can give specific instructions for other scenarios

```
fav_number = 90
if fav_number < 10:
  print("That's a small number")
elif fav_number > 10:
  print("That's a big number")
else:
  print("That number is just right!")
```

What happens?

```
>>> That's a big number
```

Elif statements

How about a different number???

elif statements means we can give specific instructions for other scenarios

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

What happens?

```
>>>
```

Elif statements

How about a different number???

elif statements means we can give specific instructions for other scenarios

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

What happens?

```
>>> That number is just right!
```

Project Time!

You now know all about `if` and lists!

See `if` you can do Part 3

The tutors will be around to help!

For Loops and Lists

For Loops

For loops allow you to do something a certain number of times.

We use them when we know exactly how many times we want to do something!

For Loops

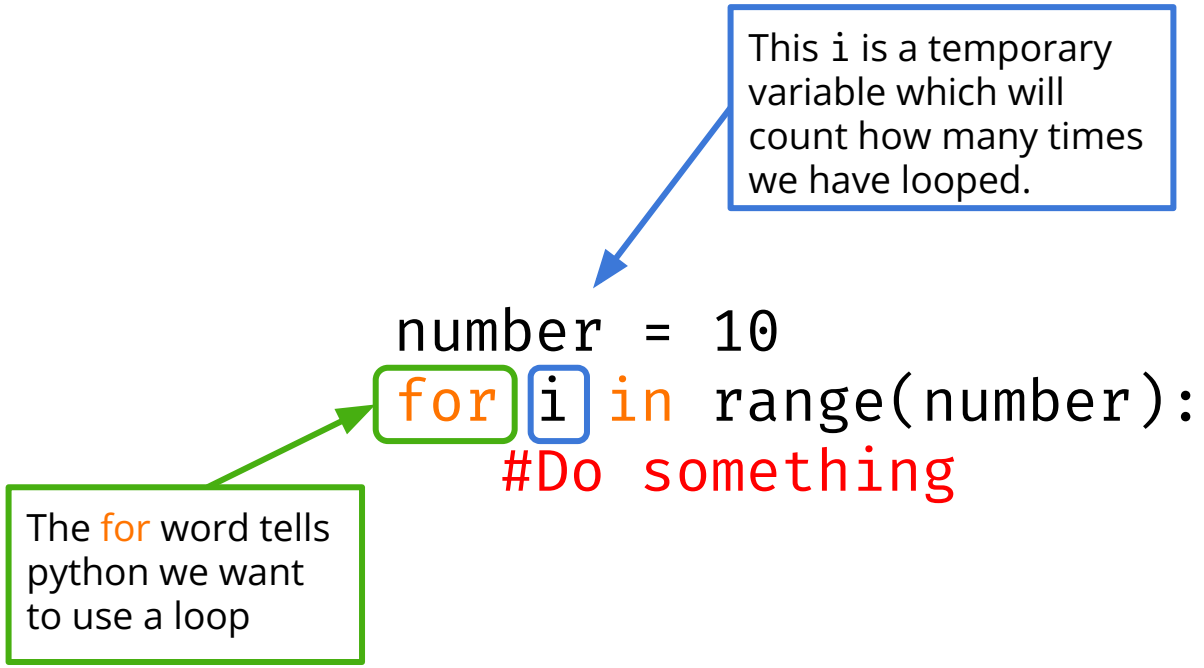
```
number = 10
for i in range(number):
    #Do something
```

For Loops

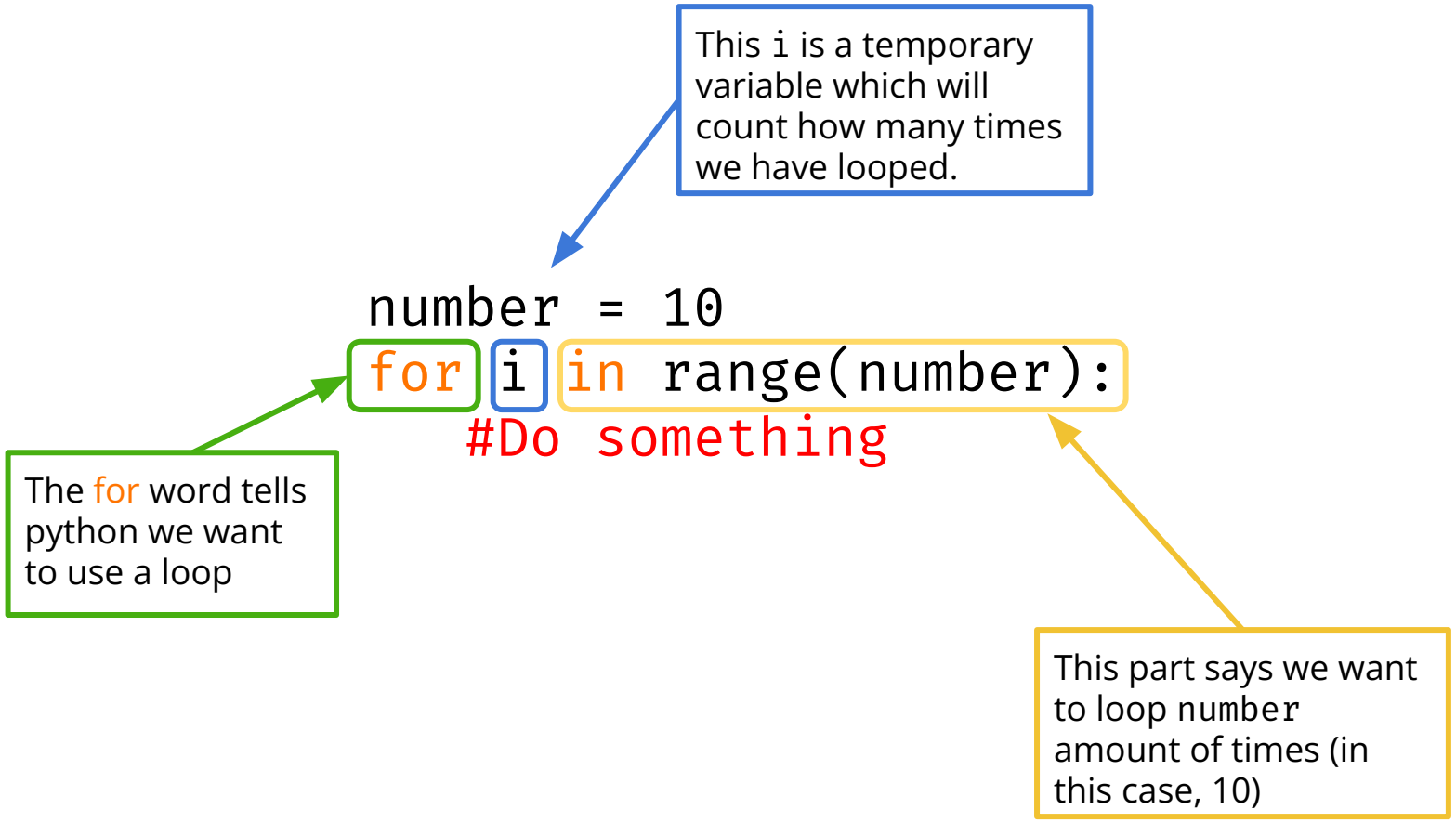
```
number = 10
for i in range(number):
    #Do something
```

The for word tells python we want to use a loop

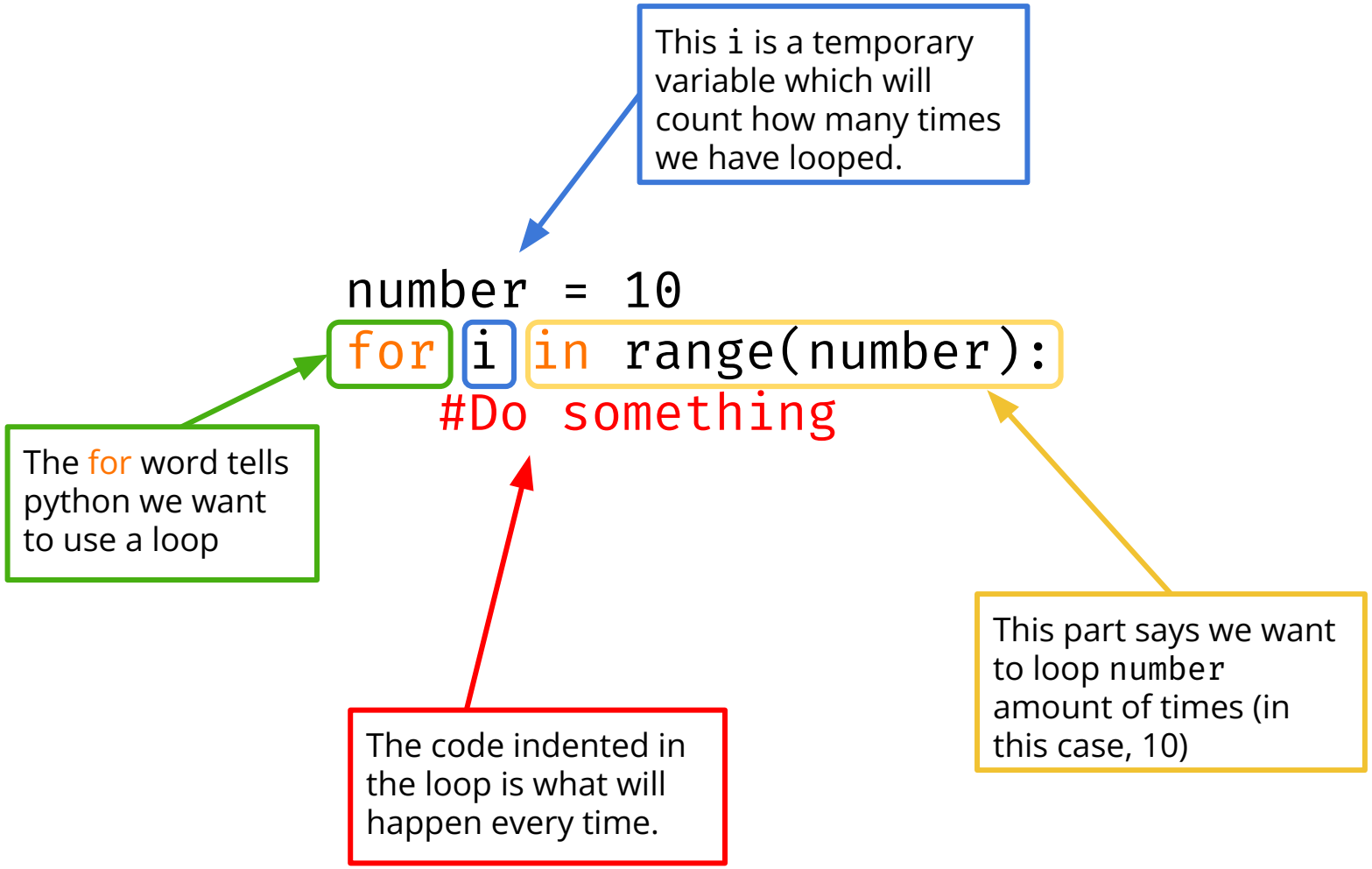
For Loops


This i is a temporary variable which will count how many times we have looped.
number = 10
for i in range(number):
#Do something
The for word tells python we want to use a loop

For Loops


This i is a temporary variable which will count how many times we have looped.
number = 10
for i in range(number):
#Do something
The for word tells python we want to use a loop
This part says we want to loop number amount of times (in this case, 10)

For Loops


This i is a temporary variable which will count how many times we have looped.
number = 10
for i in range(number):
#Do something
The for word tells python we want to use a loop
The code indented in the loop is what will happen every time.
This part says we want to loop number amount of times (in this case, 10)

Looping how many times?

We can loop through a list:

```
friends = 4
for i in range(friends):
    print(“Hello friend!”)
```

What’s going to happen?

Looping how many times?

We can loop through a list:

```
friends = 4
for i in range(friends):
    print(“Hello friend!”)
```

What’s going to happen?

```
>>> Hello friend!
>>> Hello friend!
>>> Hello friend!
>>> Hello friend!
```

Looping how many times?

We can loop any number of times:

```
friends = 4
for i in range(friends):
    print(“Hello friend!”)
```

We do what’s in the for loop as many times as what is in the “range”

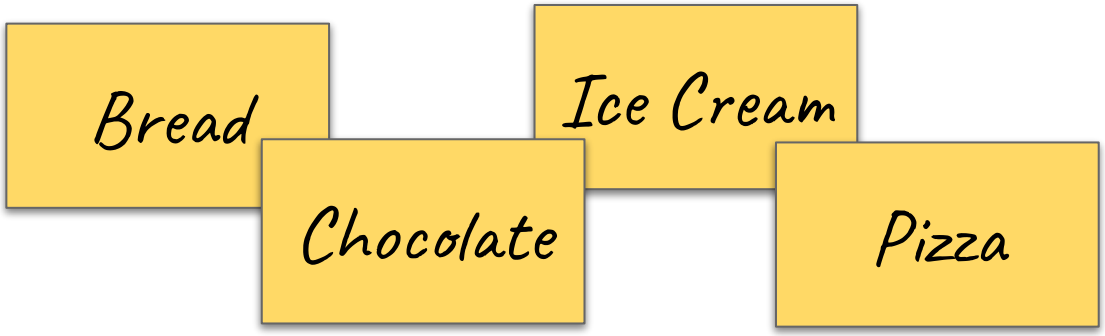
What’s going to happen?

```
>>> Hello friend!
>>> Hello friend!
>>> Hello friend!
>>> Hello friend!
```

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

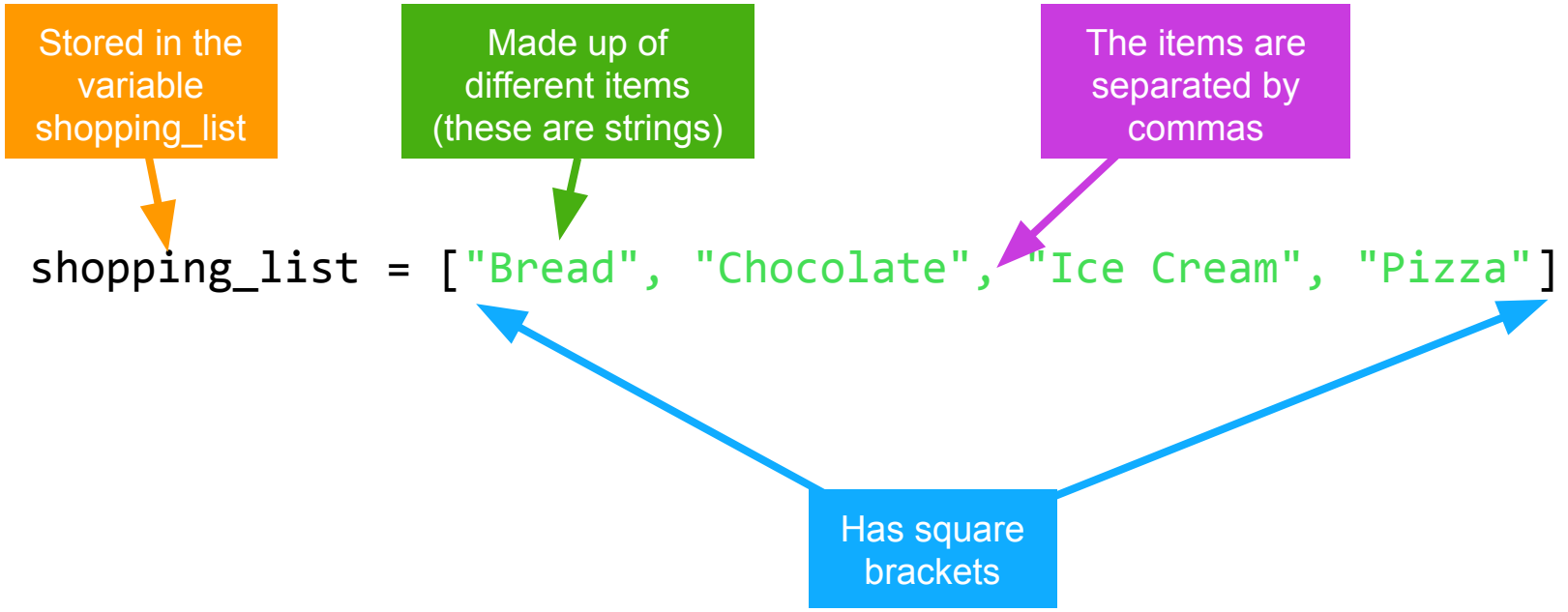
```
>>> primes = [1, 2, 3, 5, 11]
```

- 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



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!

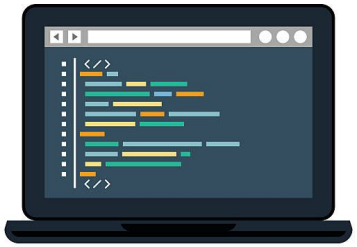
What if we decided that we also liked programming?

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

What does this list look like now?

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



Looping through lists!

What would we do if we wanted to print out this list, one word at a time?

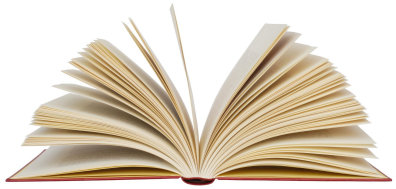
```
words = ['This', 'is', 'a', 'sentence']


print(words[0])
print(words[1])
print(words[2])
print(words[3])
```

What if it had a 100 items??? That would be **BORING!**

For Loops

For loops allow you to do something for **each** item in a **group** of things

There are many real world examples, like:



For each page in this book:
    Read page

For each chip in this bag of chips:
    Eat chip

Looping over a list of ints

We can loop through a list:

```
numbers = [1, 2, 3, 4]
for i in numbers:
    print(i)
```

What's going to happen?

Looping over a list of ints

We can loop through a list:

```
numbers = [1, 2, 3, 4]
for i in numbers:
    print(i)
```

What's going to happen?

```
>>> 1
>>> 2
>>> 3
>>> 4
```

- Each item of the list takes a turn at being the variable i
- Do the body once for each item
- We're done when we run out of items!

Looping over a list of ints

Strings are lists of letters!

```
word = "cat"
for i in word:
    print(i)
```

What's going to happen?

Looping over a list of ints

Strings are lists of letters!

```
word = "cat"
for i in word:
    print(i)
```

What's going to happen?

```
>>> c
>>> a
>>> t
```

How does it work??

Somehow it knows how to get one fruit out at a time!!

It's like it knows english!

```
fruits = ['apple', 'banana', 'mango']
for fruit in fruits:
    print('yummy ' + fruit)
```

But fruit is just a variable! We could call it anything! Like dog!

```
fruits = ['apple', 'banana', 'mango']
for dog in fruits:
    print('yummy ' + dog)
```

```
>>> Yummy apple
>>> Yummy banana
>>> Yummy mango
```

Project Time!

Now you know how to use a for loop!

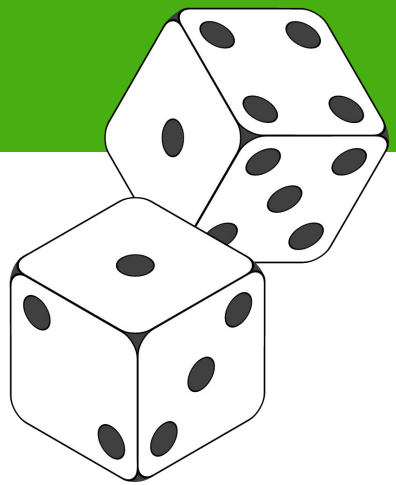
Try to do Part 4 - 5

...if you are up for it!

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 = ["eggs", "bread", "apples", "milk"]
```

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 = ["eggs", "bread", "apples", "milk"]
>>> 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 Part 6 - 7

The tutors will be around to