Guess Who!

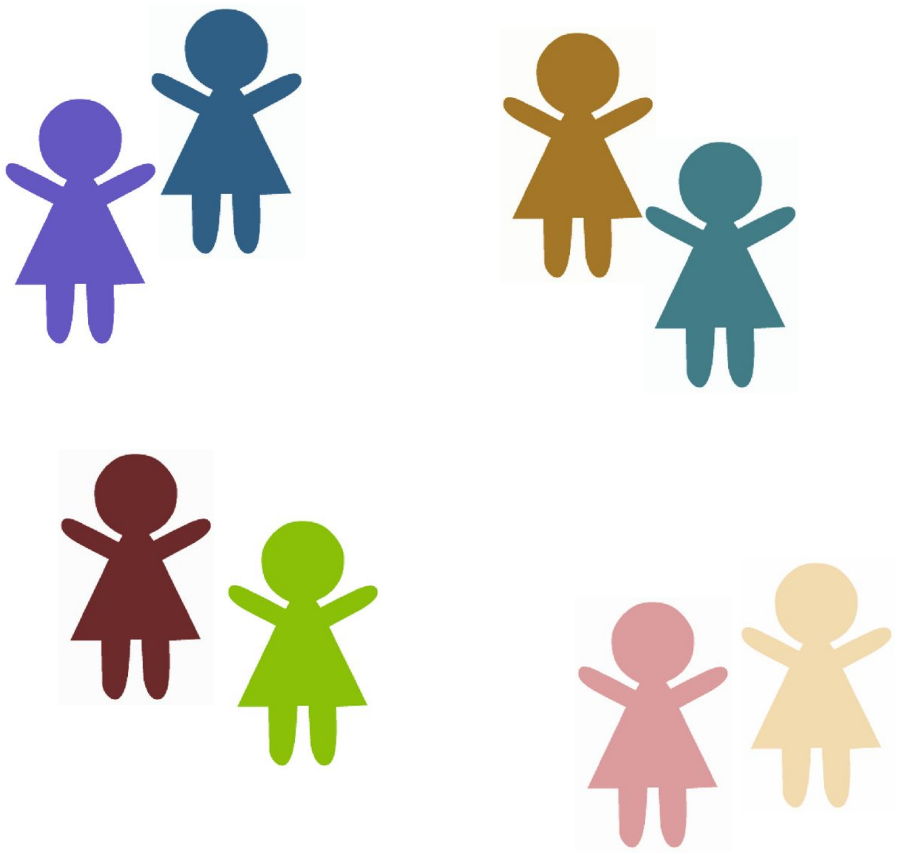
Welcome to the labs!

Who are the tutors?

Who are you?

Introduce your partner

1. Find a partner (someone you've never met before)
2. Find out:
   a. Their name
   b. What (school) year they are in
   c. A fun fact about them!
3. Introduce them to the rest of the group!



Log on

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!

Guess Who?

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.

```
print('This example is not part of the project' )
```

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 Programming

What is programming?

Programming is not a bunch of crazy numbers!

It's giving computers a set of instructions!

A Special Language

A language to talk to dogs!

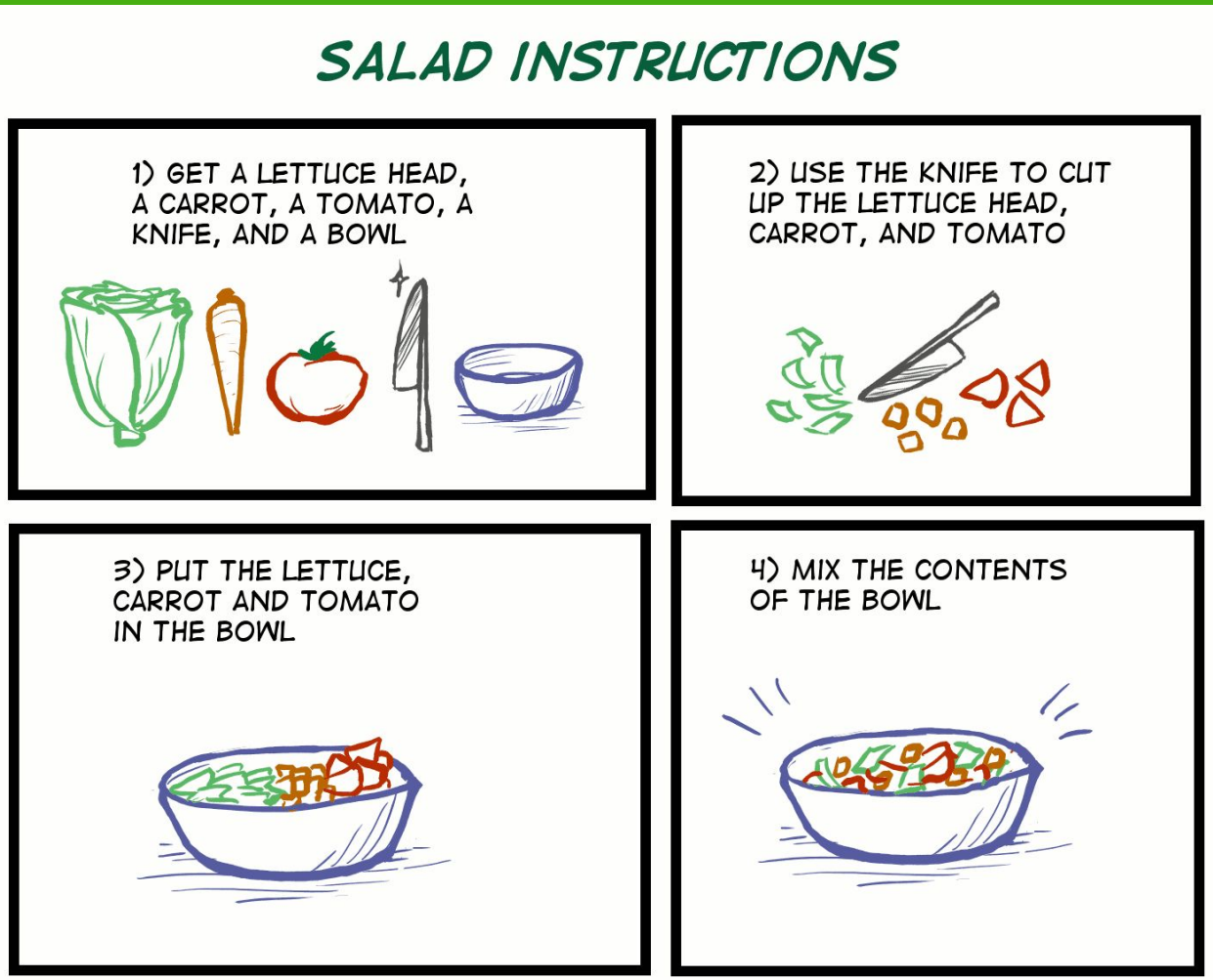
Programming is a language to talk to computers

People are smart! Computers are dumb!

Programming is like a recipe!

Computers do EXACTLY what you say, every time.

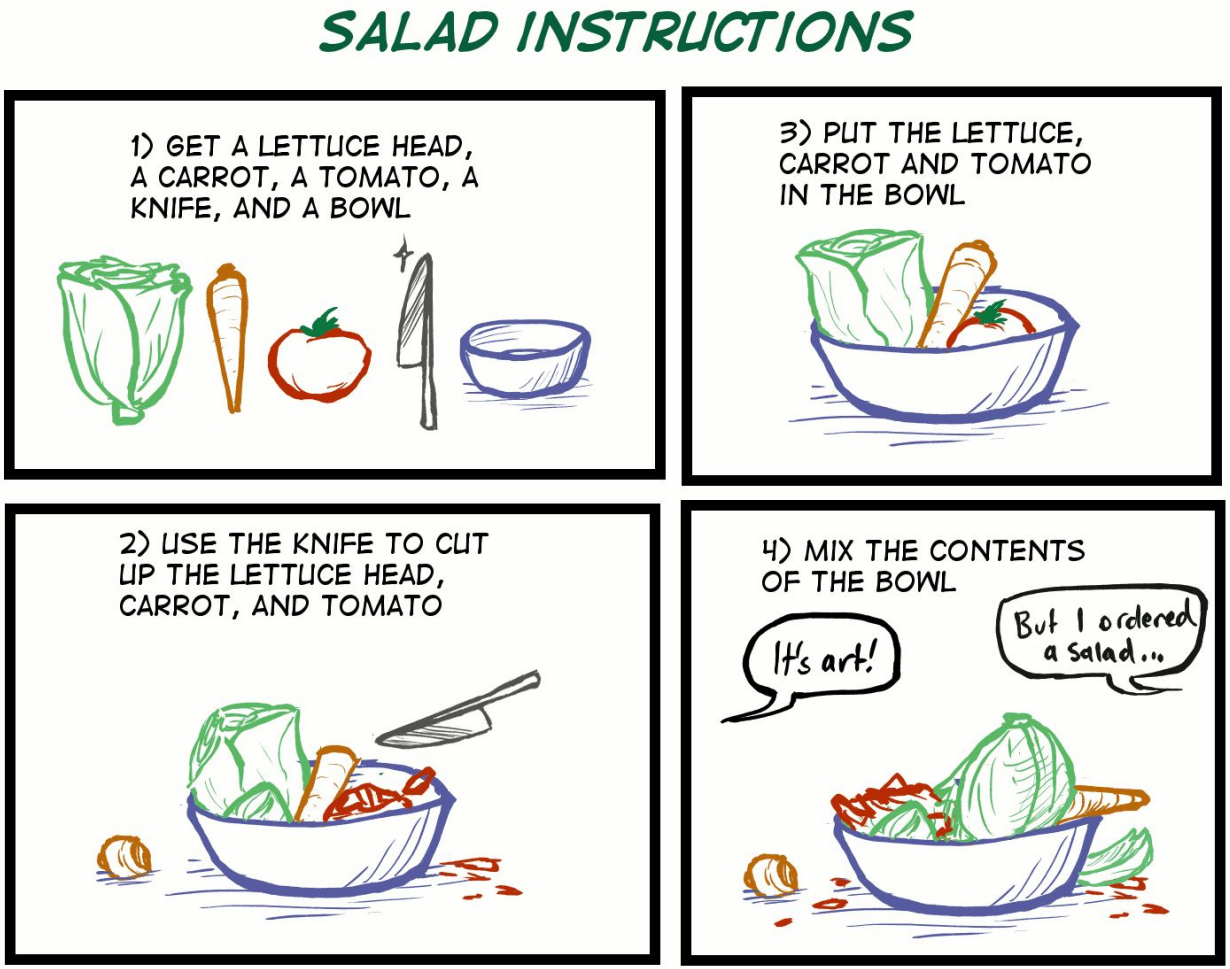
Which is great if you give them a good recipe!



People are smart! Computers are dumb!

But if you get it out of order….

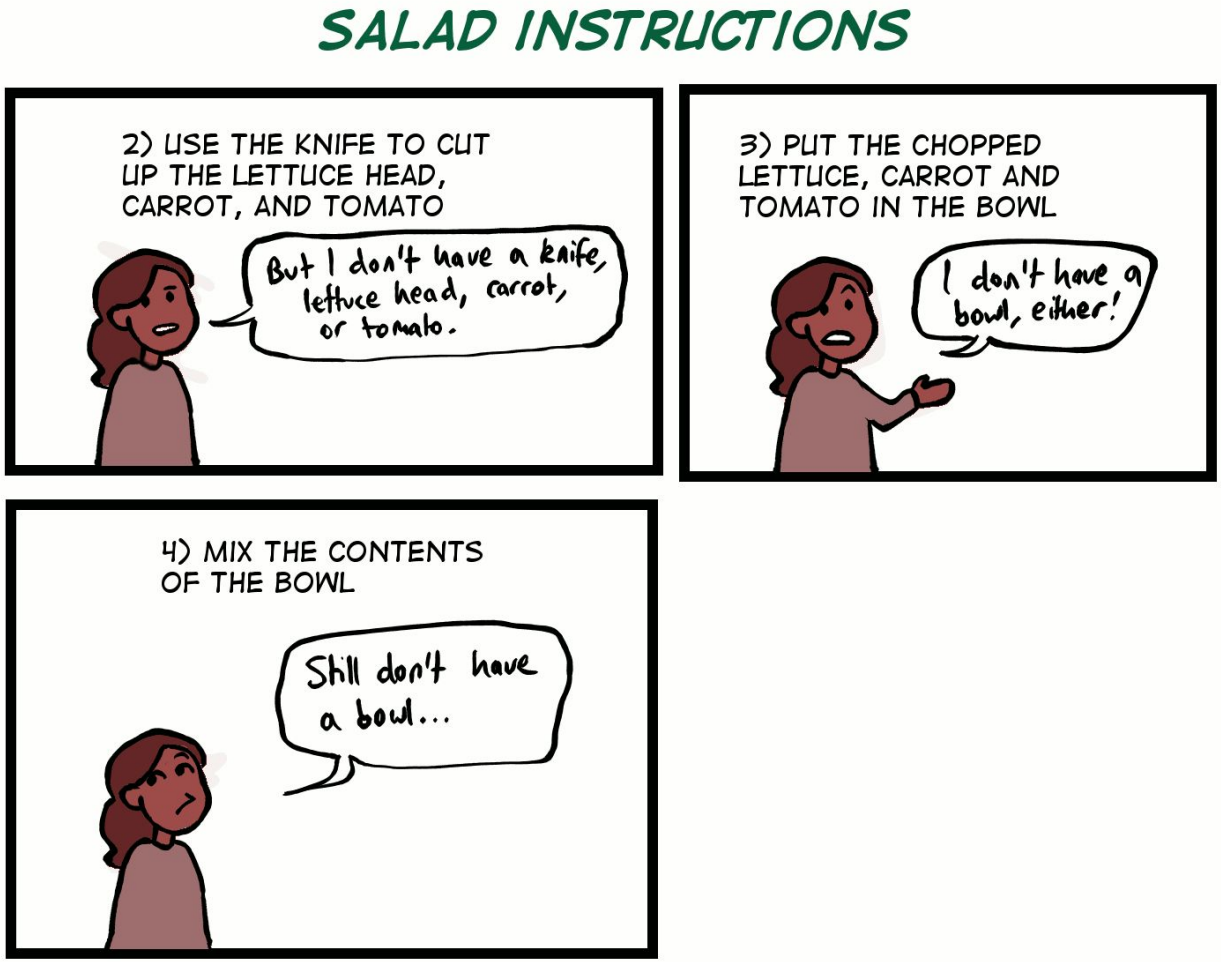
A computer wouldn't know this recipe was wrong!



People are smart! Computers are dumb!

Computers are bad at filling in the gaps!

A computer wouldn't know something was missing, it would just freak out!



Everyone/thing has strengths!

- Understand instructions despite:
  - Spelling mistakes
  - Typos
  - Confusing parts
- Solve problems
- Tell computers what to do
- Get smarter every day

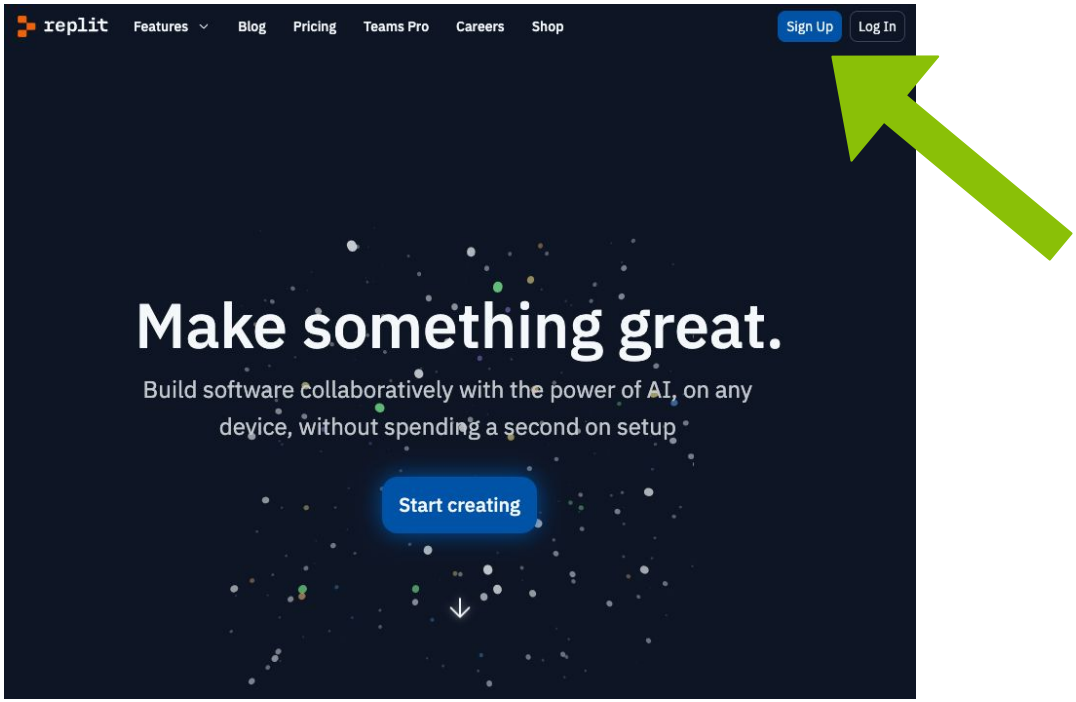
- Does exactly what you tell it
- Does it the same every time
- Doesn't need to sleep!
- Will work for hours on end!
- Get smarter when you tell them how

Intro to Python

Let’s get coding!

Where do we program?

We'll use ***Repl It*** to make a Python project!



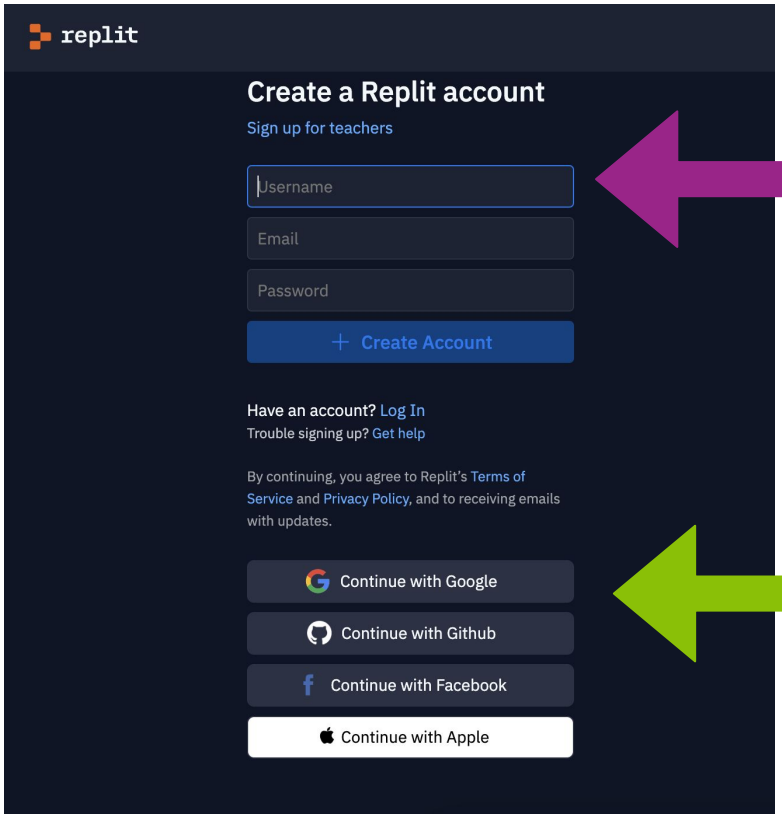
Go to replit.com in Google Chrome

Where do we program?

You need to sign up or sign in to start coding

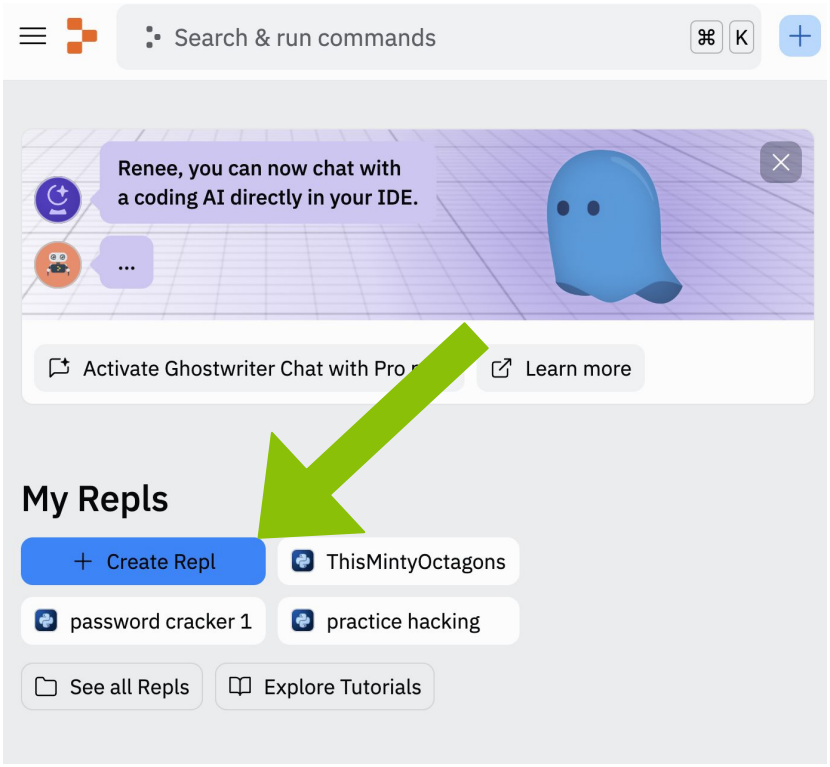
If you have a **Google** or **Apple account** it's easiest to use that.

Or use an **email address** you are able to log into.

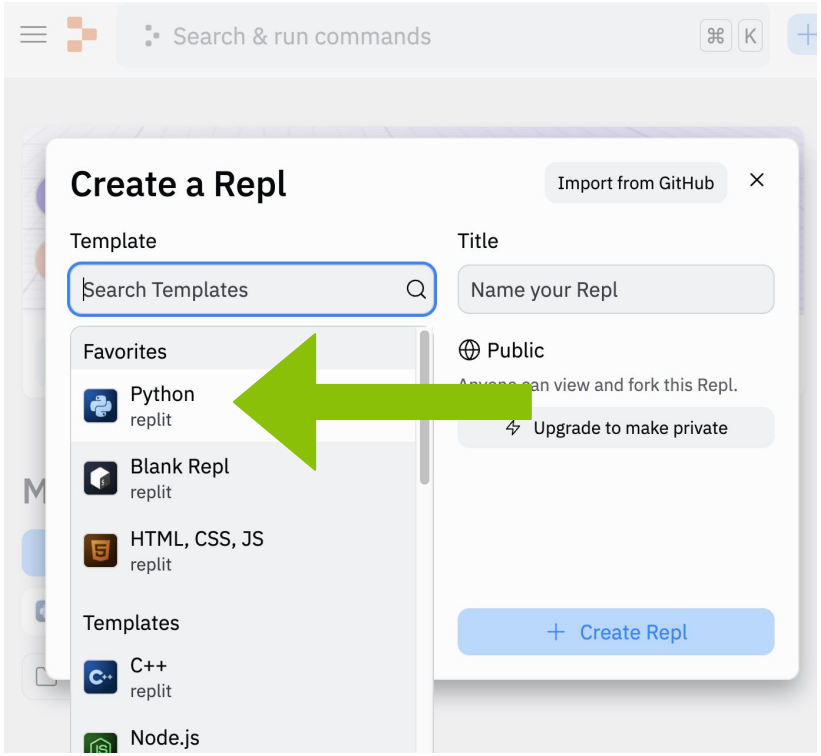


Creating our **Repl It Project**

Let's create a new project



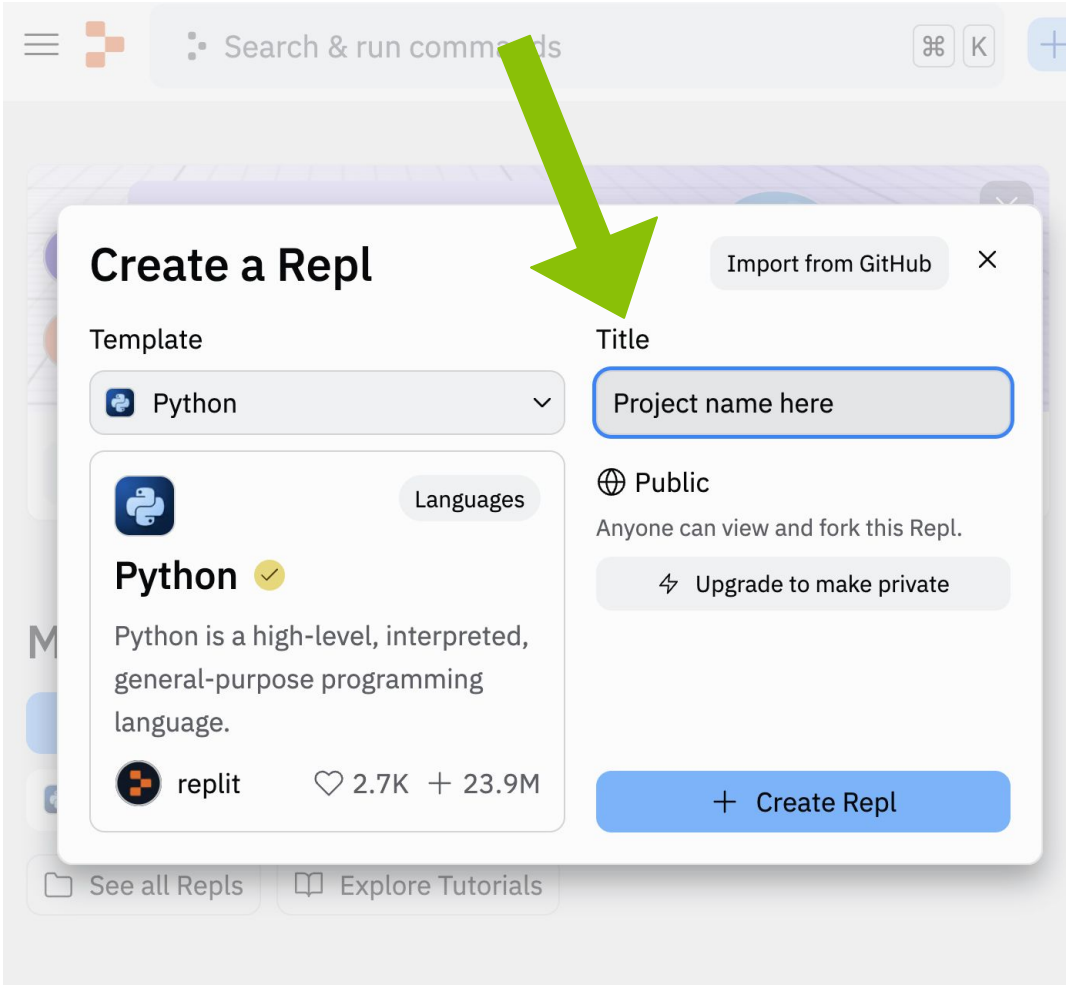
Select Python for the project template



Creating our **Repl It Project**

Don’t forget to give your project a name!

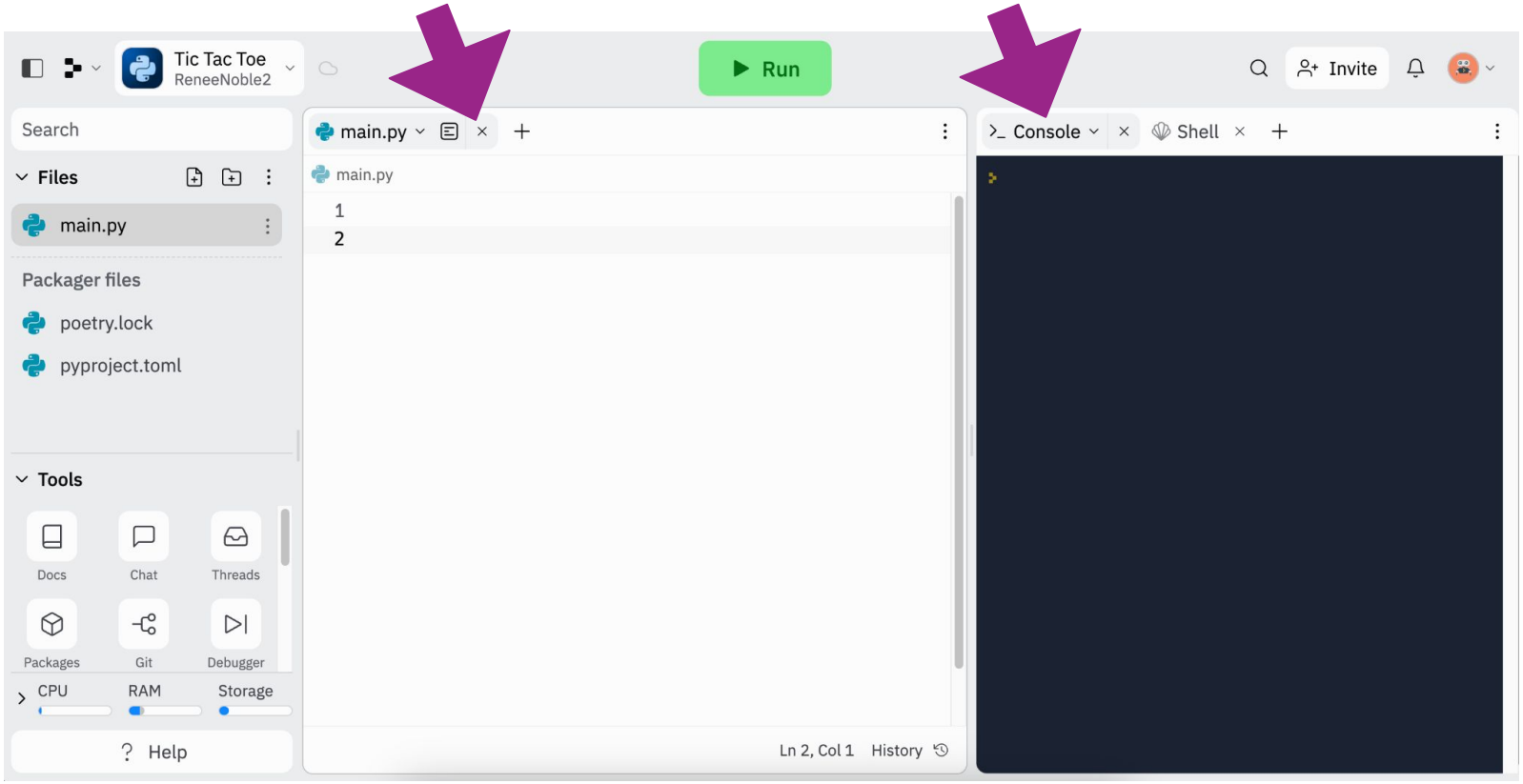
Name it after today’s project!



We're ready to code!

We'll write our project here in main.py

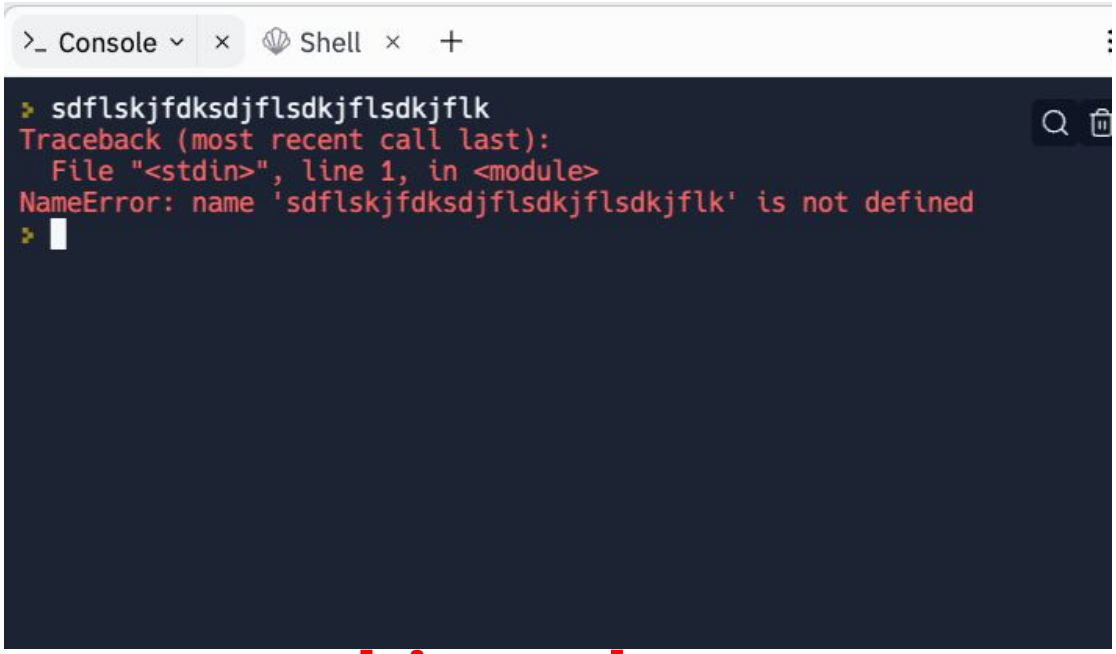
You can test out Python code in the console



Test the **console**! Make a mistake!

Type by **button mashing** the keyboard!

Then press enter!



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
```

Write some code!!

Type this into the window

Then press enter!

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

Did it print:

```
hello world
```

???

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5

>>> 2 - 7

>>> 2 * 8

>>> 12/3
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
6
>>> 2 - 7


>>> 2 * 8


>>> 12/3
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
6
>>> 2 - 7
-5
>>> 2 * 8

>>> 12/3
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
6
>>> 2 - 7
-5
>>> 2 * 8
16
>>> 12/3
```

Python the calculator!

Try writing some maths into python!

```
>>> 1 + 5
6
>>> 2 - 7
-5
>>> 2 * 8
16
>>> 12/3
4
```

A calculator for words!

What do you think these bits of code do?

Try them and see!

```
>>> "cat" + "dog"
```

```
>>> "tortoise" * 3
```

A calculator for words!

What do you think these bits of code do?

Try them and see!

```
>>> "cat" + "dog"
```

catdog

```
>>> "tortoise" * 3
```

A calculator for words!

What do you think these bits of code do?
Try them and see!

```
>>> "cat" + "dog"
```

catdog

```
>>> "tortoise" * 3
```

tortoisetortoisetortoise

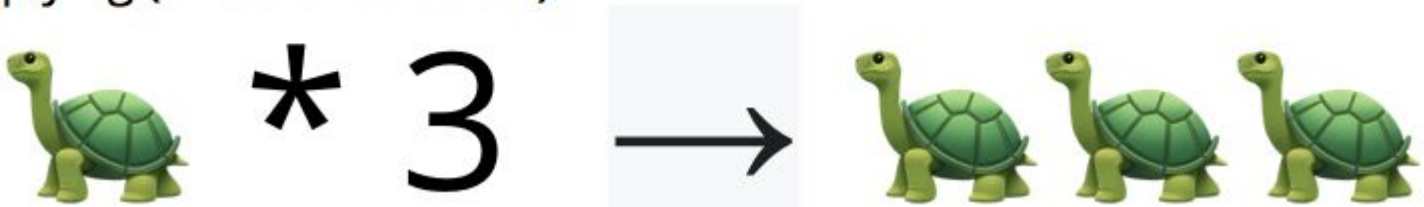
Strings!

Strings are things with `"quotes"`

To python they are essentially just a bunch of pictures!

Adding :

Multiplying (3 lots of tortoise!):



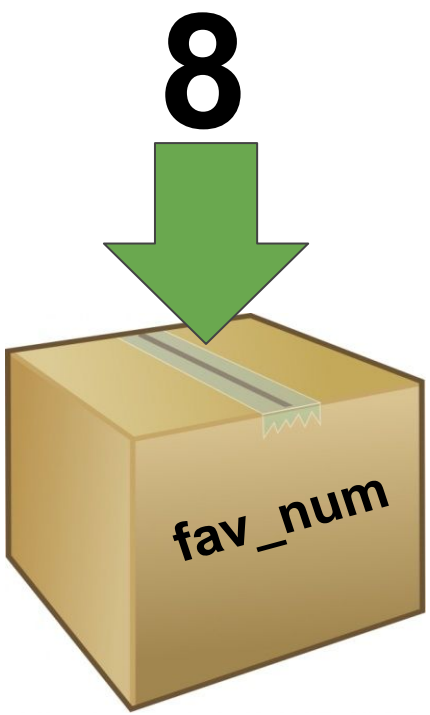
No Storing is Boring!

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

2

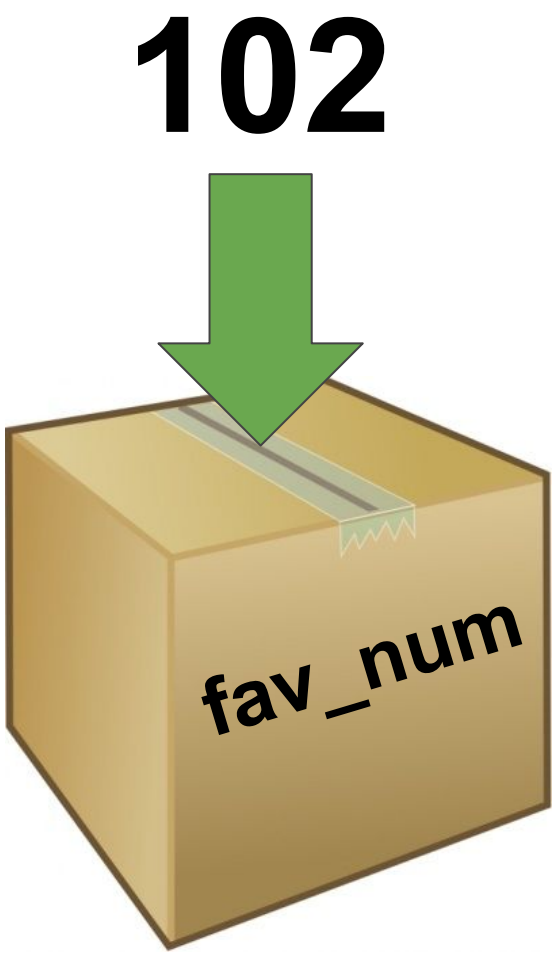
But writing 8 is much shorter than writing `fav_num`???

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

No variables VS using variables

4
Changes

1
Change

8 - 6
8 * 2
8 + 21
8 / 2

102 - 6
102 * 2
102 + 21
102 / 2

int fav_num = **8**
fav_num - 6
fav_num * 2
fav_num + 21
fav_num / 2

int fav_num = **102**
fav_num - 6
fav_num * 2
fav_num + 21
fav_num / 2

Variables

Your turn!

Can you guess what each `print` will do?

Type the code into IDLE to check your guesses

```
>>> x = 3
>>> print(x)

>>> print(x + x)

>>> y = x
>>> print(y)

>>> y = y + 1
>>> print(y)
```

Variables

Your turn!

Can you guess what each `print` will do?

Type the code into IDLE to check your guesses

```
>>> x = 3
>>> print(x)
3
>>> print(x + x)
6
>>> y = x
>>> print(y)
3
>>> y = y + 1
>>> print(y)
4
```

Switcharoo - Making copies!

Set some variables!

```
>>> x = 3
>>> y = x
>>> x = 5
```

What do x and y contain now?

Let's find out together!

Switcharoo - Making copies!

Set some variables!

```
>>> x = 3
>>> y = x
>>> x = 5
```

What do x and y contain now?

```
>>> x
5
>>> y
3
```

y hasn't changed because it has a copy of x in it!

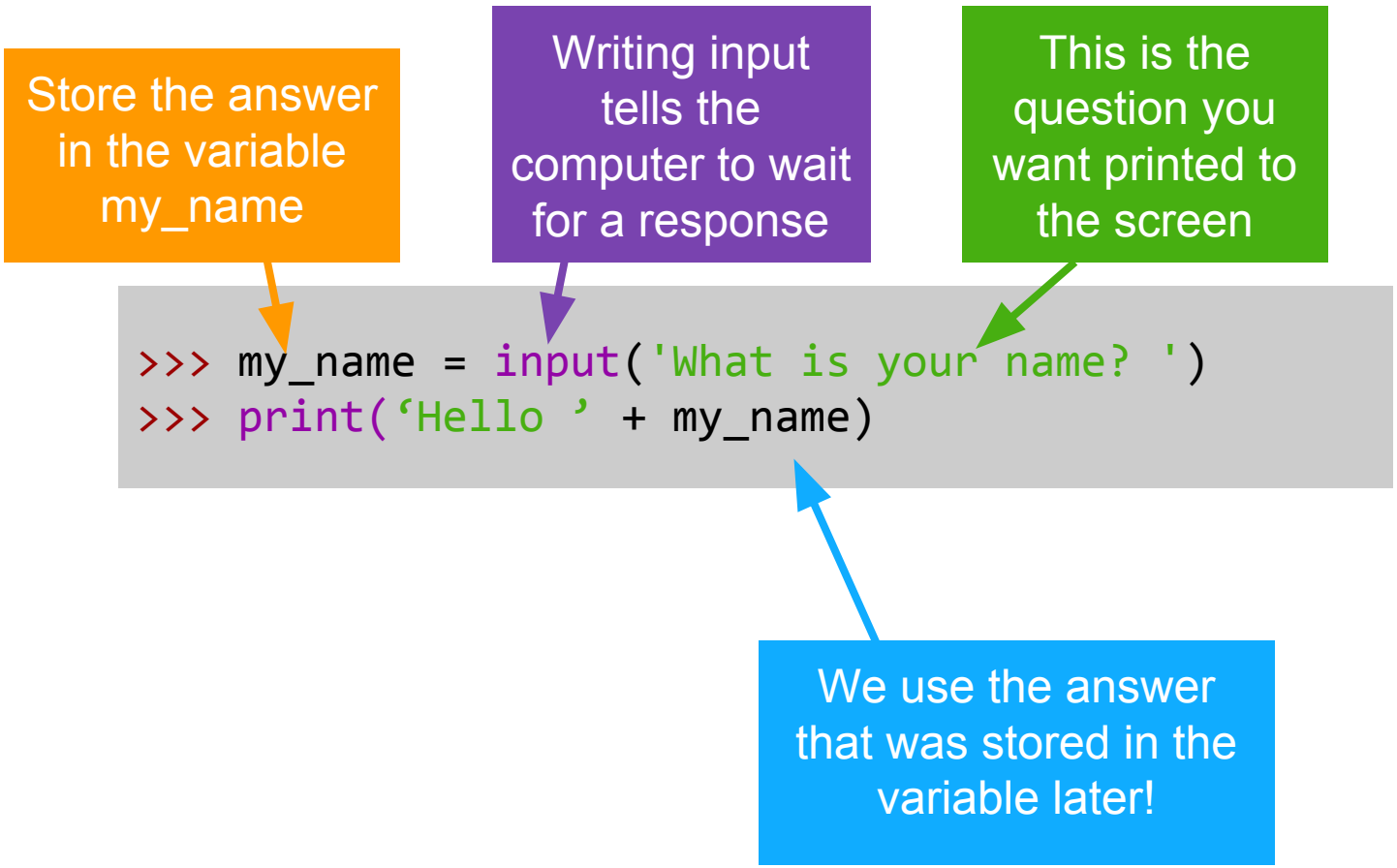
Asking a question!

It's more fun when we get to interact with the computer!

Try out this code to get the computer to ask you a question!

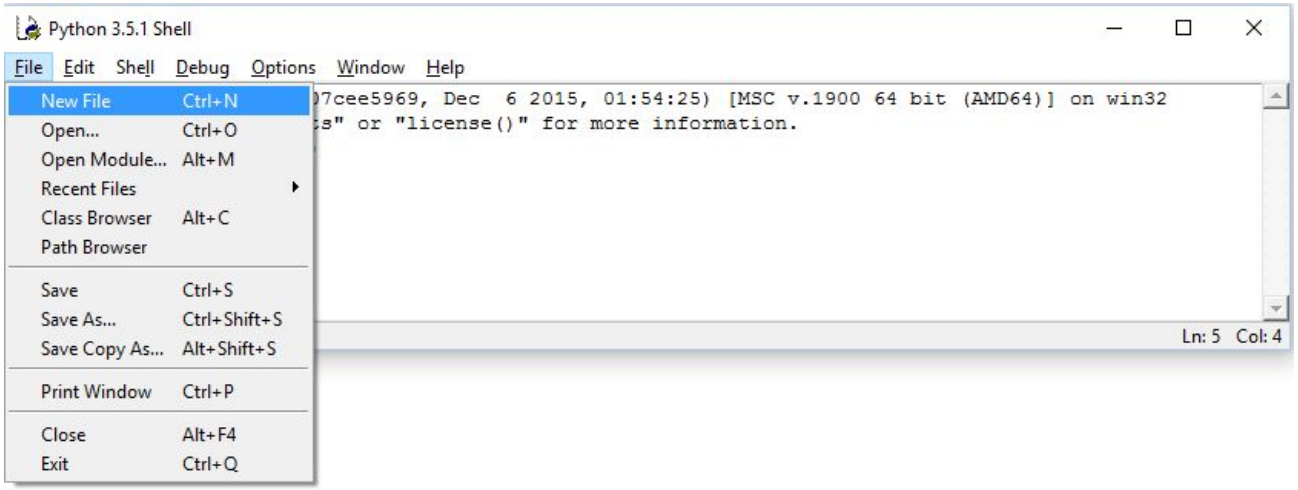
```
>>> my_name = input('What is your name? ')
>>> print('Hello ' + my_name)
```

How input works!



Coding in a file!

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



1. Make a new file called hello.py, like the picture
2. Put your `print('hello world')` code in it
3. Run your file using the F5 key

Adding a comment!

Sometimes we want to write things in our file that the computer doesn't look at! **We can use "Comments" for that!**

Sometimes we want to write a note for a people to read

```
# This code was written by Vivian
```

And sometimes we want to not run some code (but don't want to delete it!)

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

Try it!

1. Add a comment to your hello.py file!
2. Run your code to make sure it doesn't do anything extra

Project time!

Now you can give the computer variables!

Let's put what we learnt into our project

Try to do Part 0 - 1

The tutors will be around to help!

Lists

Storing groups of things in variables

- We know how to store individual things, but if we have a group of things?
- We can try to do this with variables

```
>>> day1 = 'Monday'
>>> day2 = 'Tuesday'
>>> day3 = 'Wednesday'
>>> day4 = 'Thursday'
>>> day5 = 'Friday'
>>> day6 = 'Saturday'
>>> day7 = 'Sunday'
```

- But this can get long and hard to deal with *really* quickly...

Lists can store multiple things

- It's better to create a `list`. A `list` is a data type, like `int`eger and `str`ing, but cooler!

- A `list` is an ordered group of related items, all in the same variable

- So instead of using 7 variables to store the days, we can use one:

```
>>> days = ['Monday', 'Tuesday',
'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']
```

Creating lists

- A `list` is created using square brackets in Python
- An example

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

- Think of your four favourite things.....what are they?

- How could we store them in a list?




Your Favourite Things!

```
>>> favourites = [‘books', ‘butterfly', ‘chocolate', 'skateboard']
```

You can put anything into a list

- You can have a list of integers

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

- You can have a lists of strings

```
>>> colours = ['red', 'blue', 'green']
```

Try this!

1. Make a list of your favourite foods

```
>>> fave_foods = ['mango', 'pie', 'pizza']
```

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

```
>>> print(fave_foods)
 ['mango', 'pie', 'pizza']
```

Accessing Lists!

- The favourites `list` holds four strings in order.

```
favourites = [‘books', ‘butterfly',
‘chocolate', 'skateboard']
```

- We can count out the items using index numbers!

0

1

2

3

- **Indices start from zero!**

Accessing Lists

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

```
>>> favourites[0]
'Books'
```

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

Falling off the edge

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

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


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

Updating items!

- We can also update what is stored in a list, but we need to know what item we are updating.

- What if we decided that we didn't like chocolate anymore, but loved lollypops?

- `>>> favourites[1] = 'lollypops'`

Updating items




- `favourites[1] = 'lollypops'`




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?

```
>>> favourites.remove('butterfly')
```

- What does this list look like now?

List of lists!

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

We could use a list of lists to store tennis partners.!

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

Project time!

Now you can use lists!

Let's put what we learnt into our project

Try to do Part 2

The tutors will be around to help!

If Statements

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

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

To figure out if something is `True` or `False` we do a comparison

Try typing these into IDLE!	
`5 < 10`	`"Dog" == "dog"`
`3 + 2 == 5`	`"D" in "Dog"`
`5 != 5`	`"Q" not in "Cat"`

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

It's `False`!

```
fave_num = 9000
if False:
    print("that's a small number")
```

Put in the answer to the question

Conditions

It's `False`!

```
fave_num = 9000
if False:
    print("that's a small number")
```

What do you think happens?

```
>>>
```

Conditions

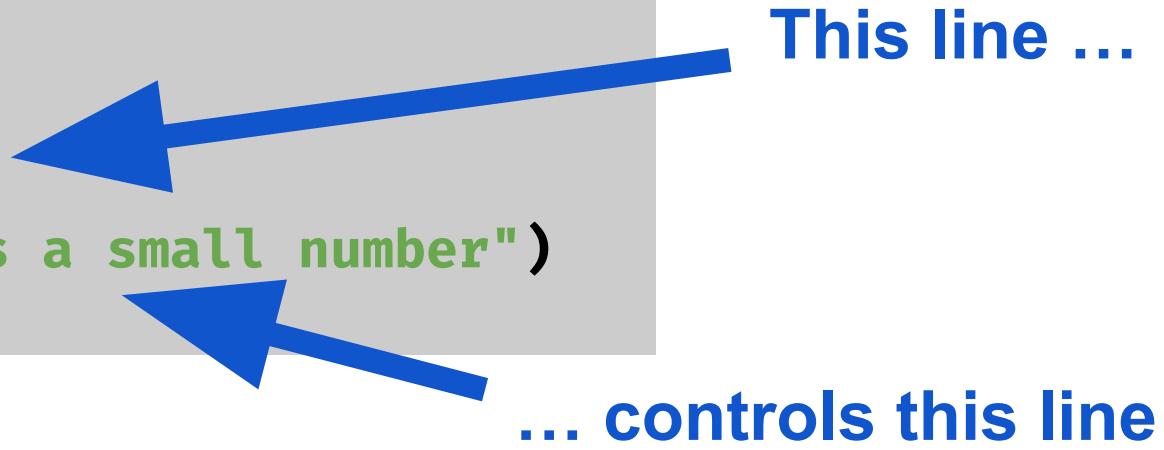
```
fave_num = 9000
if False:
    print("that's a small number")
```

What do you think happens?

```
>>>
```

If statements

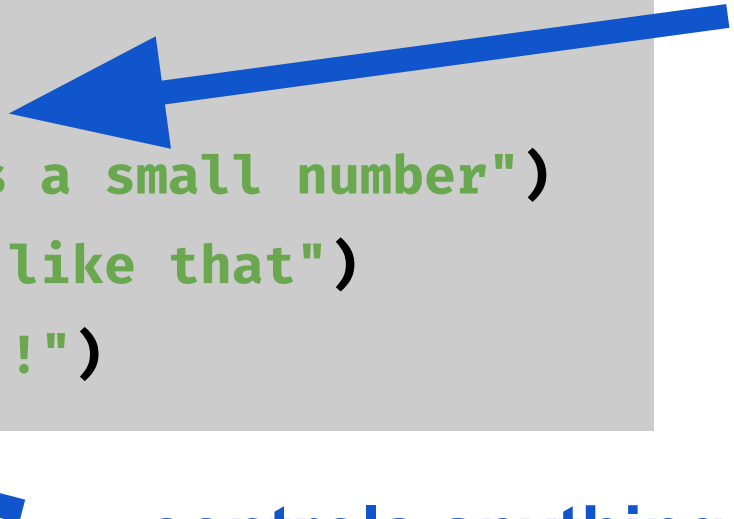
```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
```



If statements

Actually .....

```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
    print("and I like that")
    print("A LOT!!")
```



If statements

What do you think happens?

```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
    print("and I like that")
    print("A LOT!!")
```

What do you think happens?

If statements

What do you think happens?

```
fave_num = 5
if fave_num < 10:
    print("that's a small number")
    print("and I like that")
    print("A LOT!!")
```

```
>>> that's a small number
>>> and I like that
>>> A LOT!!
```

If statements

```
word = "GPN"
if word == "GPN":
  print("GPN is awesome!")
```

```
What happens??
```

If statements

```
word = "GPN"
if word == "GPN":
  print("GPN is awesome!")
```

```
What happens??
>>> GPN is awesome!
```

Else statements

```
word = "GPN"
if word == "GPN":
  print("GPN is awesome!")
```

```
What happens??
>>> GPN is awesom
```

But what if we want something different to happen if the word isn't "GPN"

Else statements

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

```
word = "Chocolate"
if word == "GPN":
  print("GPN is awesome!")
else:
  print("The word isn't GPN :(")
```

What happens??

Else statements

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

```
word = "Chocolate"
if word == "GPN":
  print("GPN is awesome!")
else:
  print("The word isn’t GPN :(")
```

```
What happens??
>>> The word isn’t GPN :(
```

Elif statements

elif
Means we can give specific instructions for other words

```
word = "Chocolate"
if word == "GPN":
  print("GPN is awesome!")
elif word == "Chocolate":
  print("YUMMM Chocolate!")
else:
  print("The word isn’t GPN :(")
```

```
What happens??
```

Project Time!

You now know all about `if` and `else`!

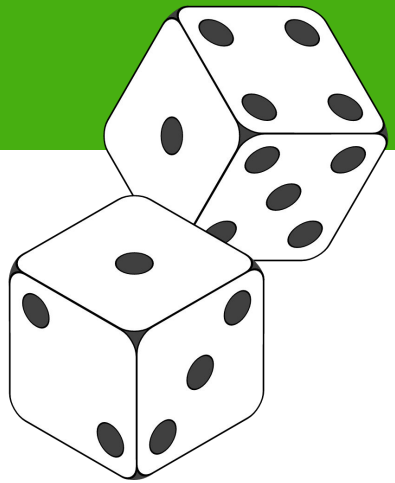
Let's put what we learnt into our project

Try to do Parts 3 and 4

The tutors will be around to help!

Random!

That's so random!



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

We'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 = ["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!

You now know all about `if` and `else`!

Let's put what we learnt into our project

Try to do Parts 3 and 4

The tutors will be around to help!

Project Time!

Raaaaaaaaaandom! Can you handle that?

Let's put what we learnt into our project

Try to do Part 5

The tutors will be around to help!

While Loops

Loops

We know how to do things on repeat!

Sometimes we want to do some code on repeat!

Introducing ... `while` loops!

What do you think this does?

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

Introducing ... `while` loops!

What do you think this does?

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

```
i is 0
i is 1
i is 2
>>>
```

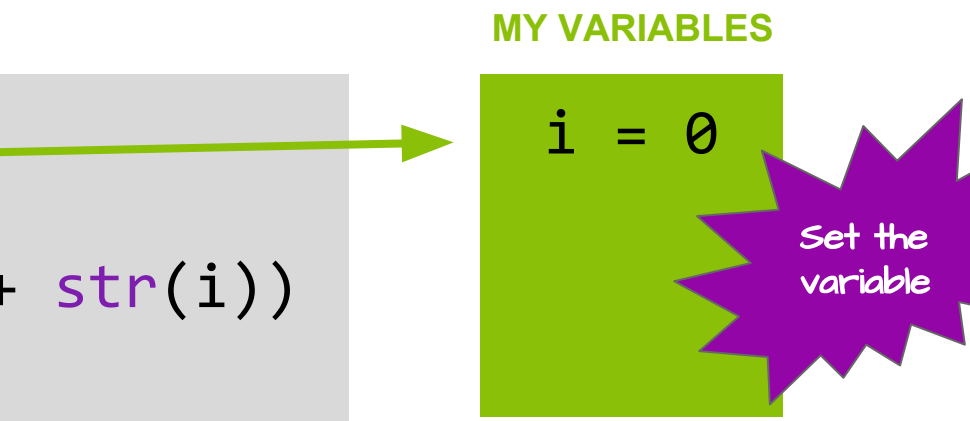
Introducing … `while` loops!

Stepping through a while loop…

Introducing … `while` loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```



Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

```
i = 0
```

Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

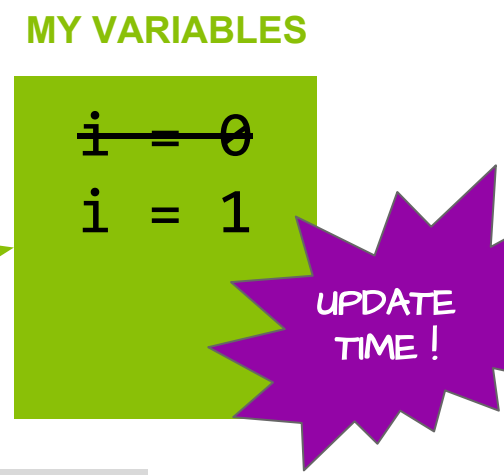
```
i = 0
```

```
i is 0
```

Introducing … `while` loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```




```
i is 0
```

Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~
i = 1

```
i is 0
```

Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~
i = 1

```
i is 0
```

Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~

i = 1

```
i is 0
i is 1
```

Introducing ... `while` loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2

UPDATE TIME !

```
i is 0
i is 1
```

Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2

```
i is 0
i is 1
```

Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2

```
i is 0
i is 1
```

Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~

~~i = 1~~

i = 2

```
i is 0
i is 1
i is 2
```

Introducing ... `while` loops!

One step at a time!

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~
~~i = 1~~
~~i = 2~~
i = 3


```
i is 0
i is 1
i is 2
```

Introducing … `while` loops!

One step at a time!


```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~
~~i = 1~~
~~i = 2~~
i = 3

```
i is 0
i is 1
i is 2
```

Introducing ... `while` loops!

One step at a time!

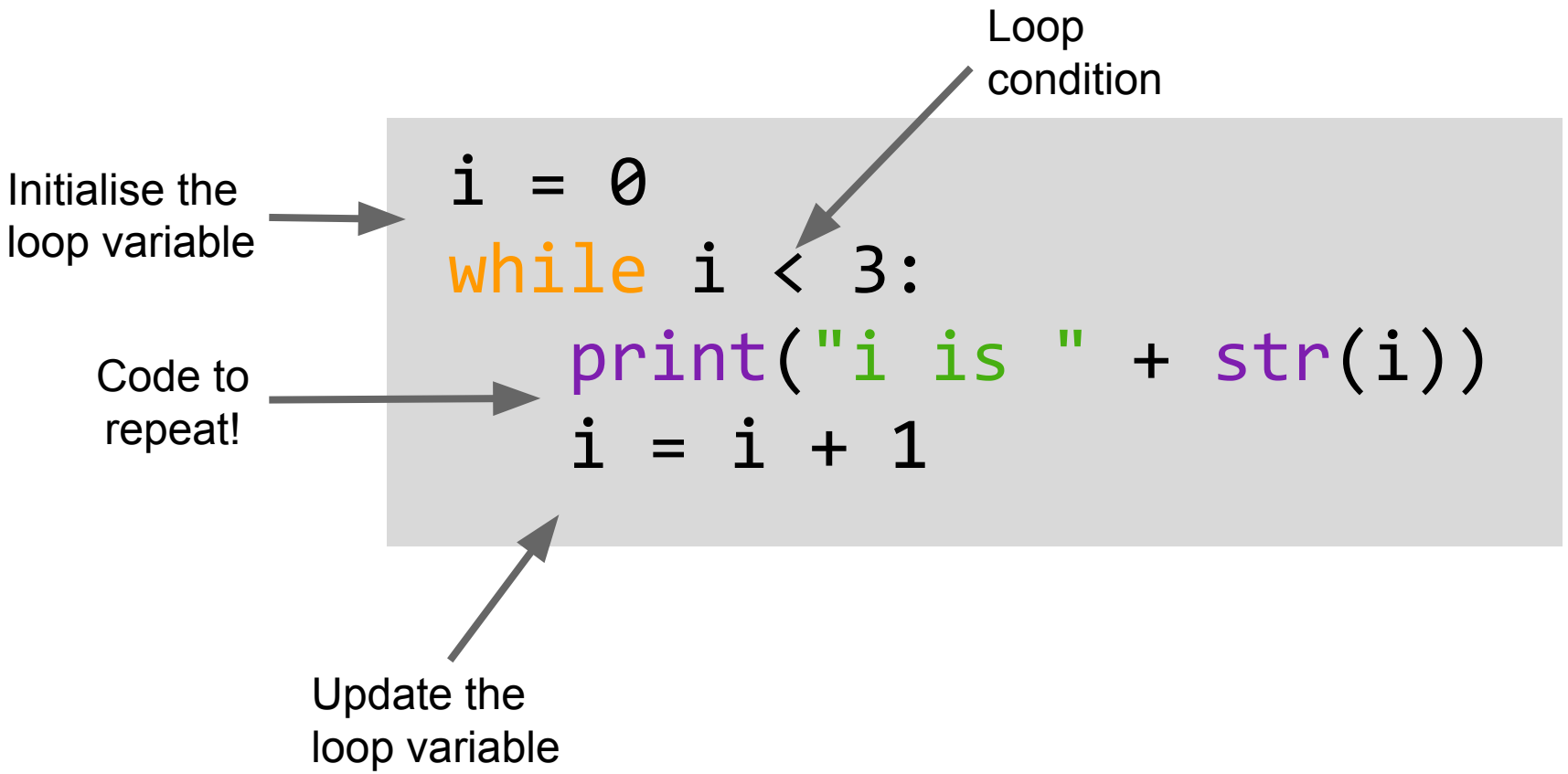

```
i = 0
while i < 3:
    print("i is " + str(i))
    i = i + 1
```

MY VARIABLES

~~i = 0~~
~~i = 1~~
~~i = 2~~
i = 3


```
i is 0
i is 1
i is 2
```

Introducing … `while` loops!



What happens when.....

What happens if we forget to update the loop variable?

```
i = 0
while i < 3:
    print("i is " + str(i))
```

What happens when.....

What happens if we forget to update the loop variable?

```
i = 0
while i < 3:
    print("i is " + str(i))
```

```
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
i is 0
```

Infinite loop!

Sometimes we want our loop to go forever!

So we set a condition that is always True!

We can even just write True!

```
while True:
    print("Are we there yet?")
```

Not-so-infinite loop!

But we might want the loop to stop!

To break out of the loop, we can use **break**:

```
while True:
    print("Are we there yet?")
    break
```

Project Time!

Raaaaaaaaaandom! Can you handle that?

Let's put what we learnt into our project

Try to do Part 5

The tutors will be around to help!

Project Time!

while we're here:

Let's put what we learnt into our project

Try to do Part 6!

Then try Extension Parts 7 - 10

The tutors will be around to help!

Tell us what you think!

Click on the
End of Day Form
and fill it in now!