Welcome to GPN

Tech
Inclusion

Thank you to our Sponsors!

Platinum Sponsor:

Who are the tutors?

Tech
Inclusion

Who are you?

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!

Tech Inclusion

Tell us you're here!

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

Tech
Inclusion

Today's project!

Markov Chains!

What is a Markov Chain?

A Markov chain is a simple Artificial Intelligence!

Let's play a game with some cups to help explain it

Tech
Inclusion

Let's generate some text in the style of

Green Eggs & Ham by Dr Seuss

Do you like green eggs and ham?

I do not like them, Sam-I-am.

I do not like green eggs and ham.

Would you like them here or there?

I would not like them here or there.

I would not like them anywhere.

Let's play the cups game!

- Each cup is labelled with a word from Green Eggs and Ham

- Each cup contains the words that follow the label in Green Eggs and Ham

Let's play the cups game!

Read the outside of your cup!

If you hear someone shout the word on the outside of your cup:

1. Pick a piece of paper from inside your cup
2. Shout out the word on the piece of paper
3. Put the piece of paper back in your cup

Today we'll be making Markov Chains!

Markov chains are exactly what we just did with the cups!

Today we'll make the computer do it too to make some crazy stories!!

Here's one we made from some Shakespeare!

```
doth stay! All days when I compare thee to unseeing
eyes be blessed made By chance, or eyes can see, For
all the top of happy show thee in dark directed. Then
thou, whose shadow shadows doth stay! All days when I
compare thee in your self in inward worth nor outward
fair, Can make bright, How would thy shade Through
heavy sleep on the eye of life repair, Which this,
Time's pencil, or my pupil pen, Neither in the living
day, When in eternal lines of that fair from fair
thou grow'st, So should the lines to a summer's day?
```

Imagine if you used one of these to do your homework!!

Using the workbook!

The workbooks will help you put your project together!

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

Tasks - The parts of your project

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

Hints - Helpers for your tasks!

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

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

Task 6.2: Add a blah to your code!

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

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

Task 6.1: Make the thing do blah!

Make your project do blah ….

Hint

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

Tech Inclusion

Using the workbook!

The workbooks will help you put your project together!

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

Checklist - Am I done yet?

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

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

For Loops

★ BONUS 4.3: Do some extra!

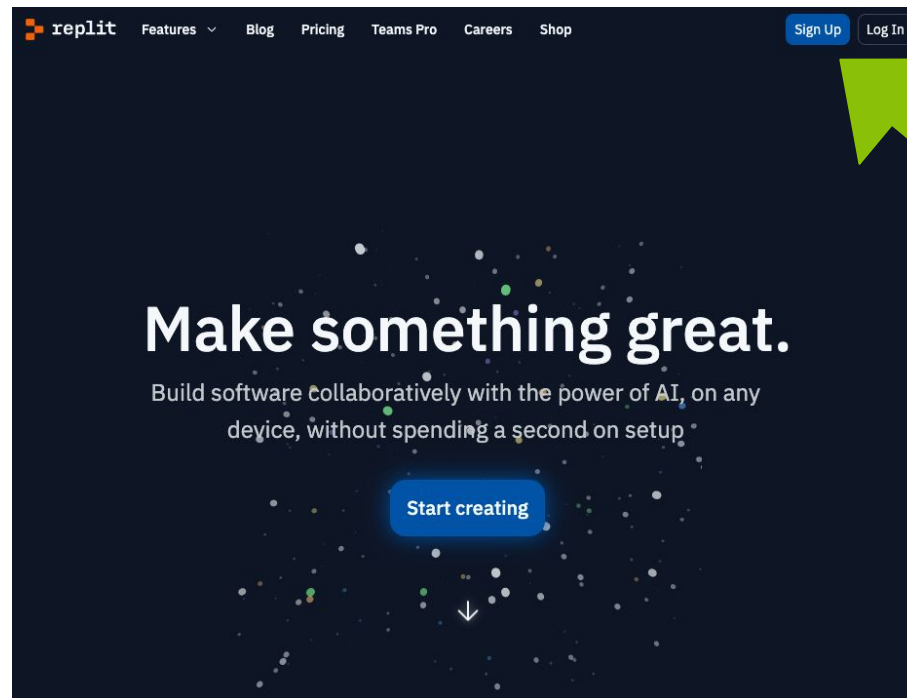Something to try if you have spare time before the next lecture!

Tech Inclusion

Intro to Python

Let's get coding!

Where do we program?

We'll use **_Repl It_** to make a Python project!

Go to replit.com in your web browser

Tech
Inclusion

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.

Tech Inclusion

Creating our **Repl It Project**
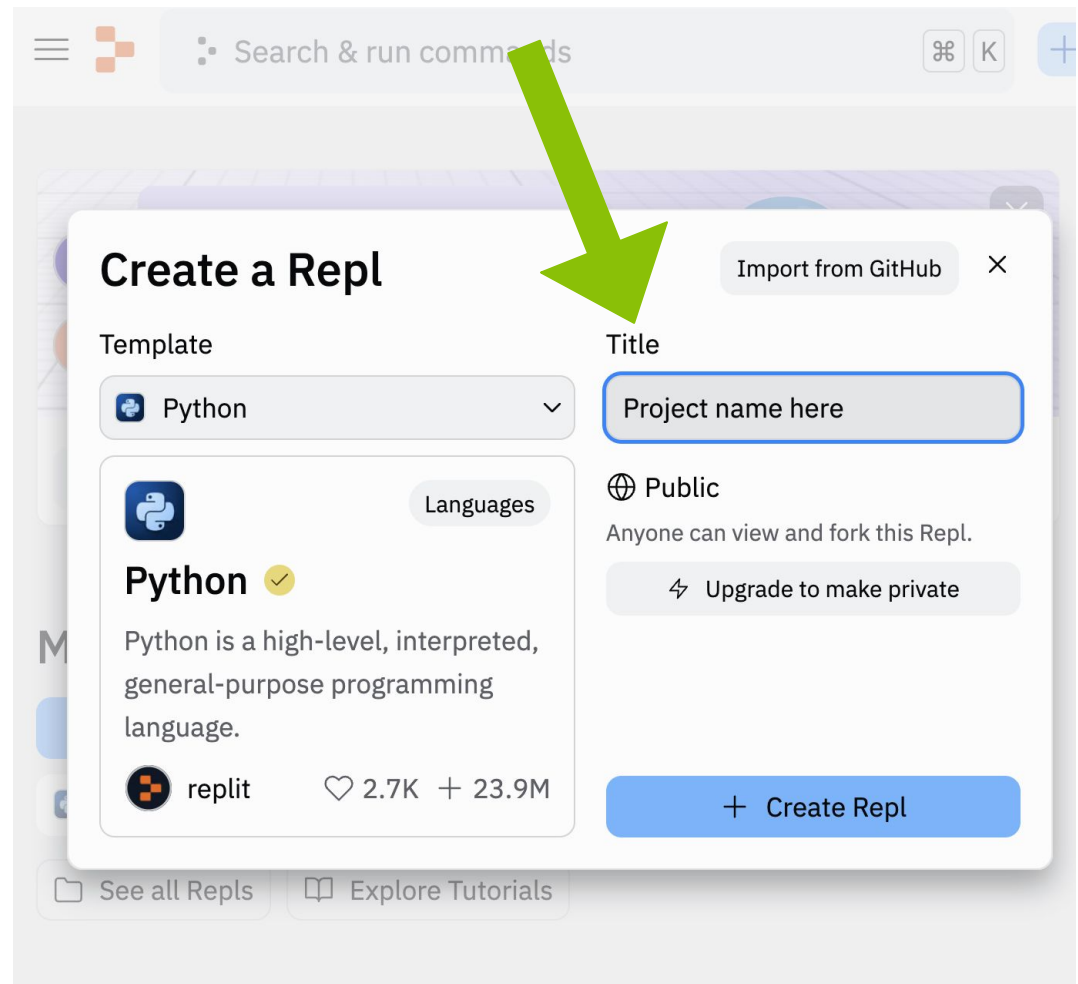
Let's create a new project
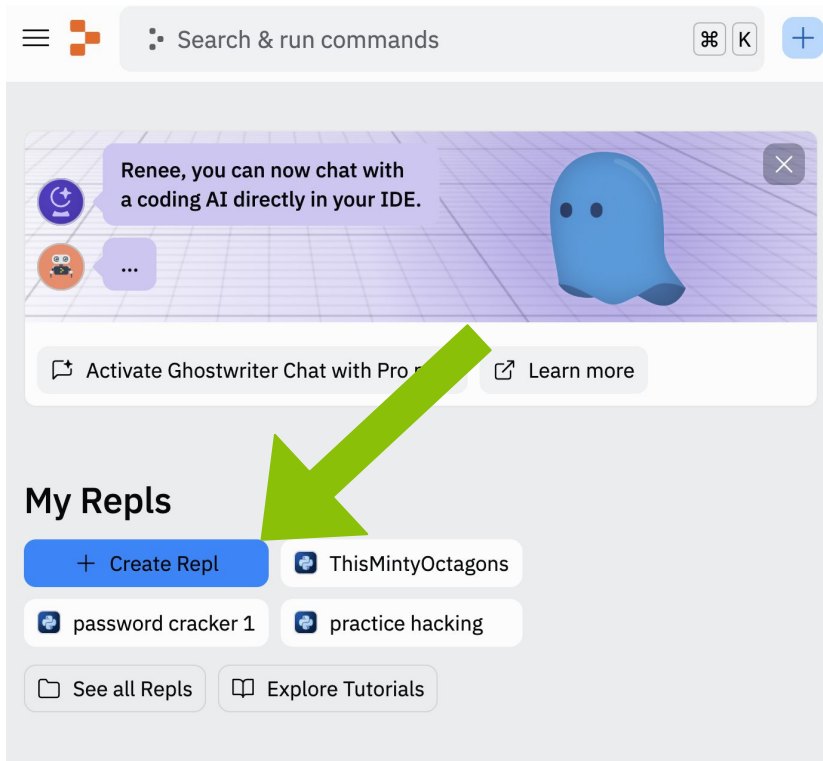
Select Python for the project template

Tech Inclusion
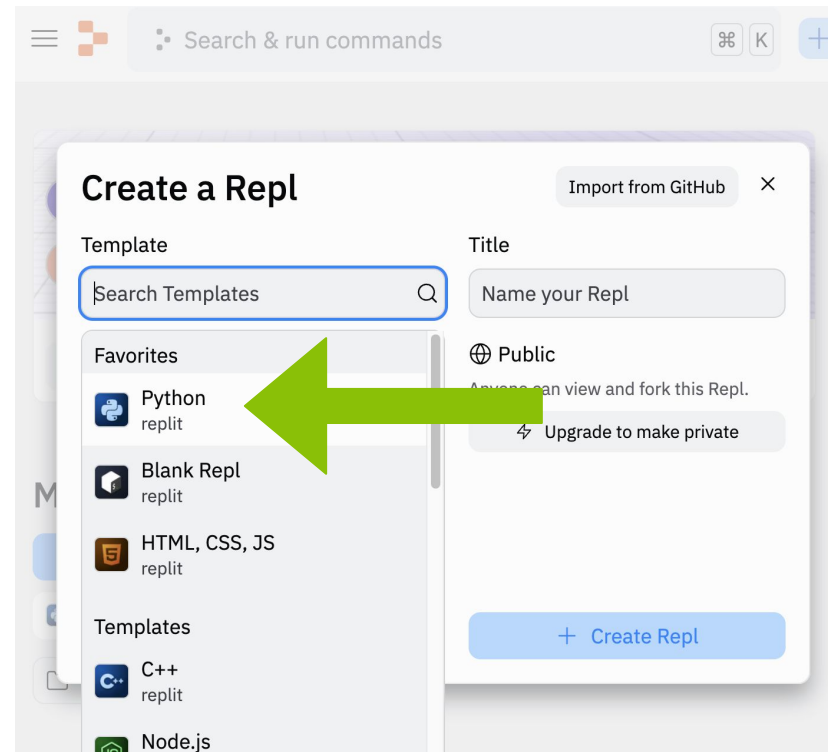
Creating our **Repl It Project**

Don't forget to give your project a name!
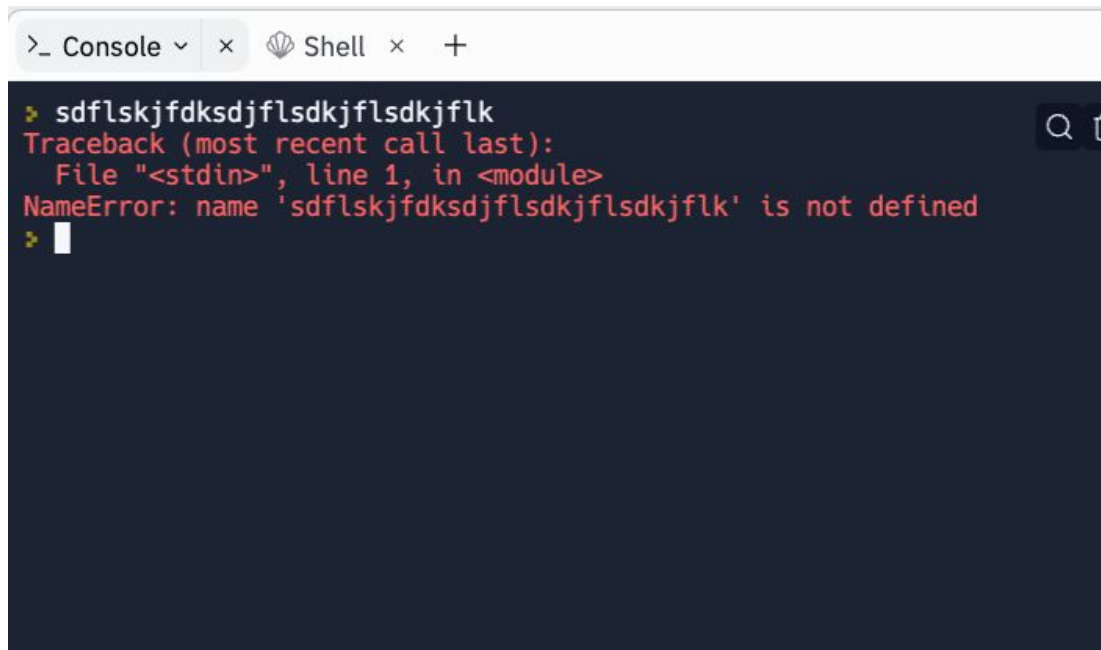
Name it after today's project!

Tech Inclusion

We're ready to code!

We'll write our project here in main.py

You can test out Python code in the console

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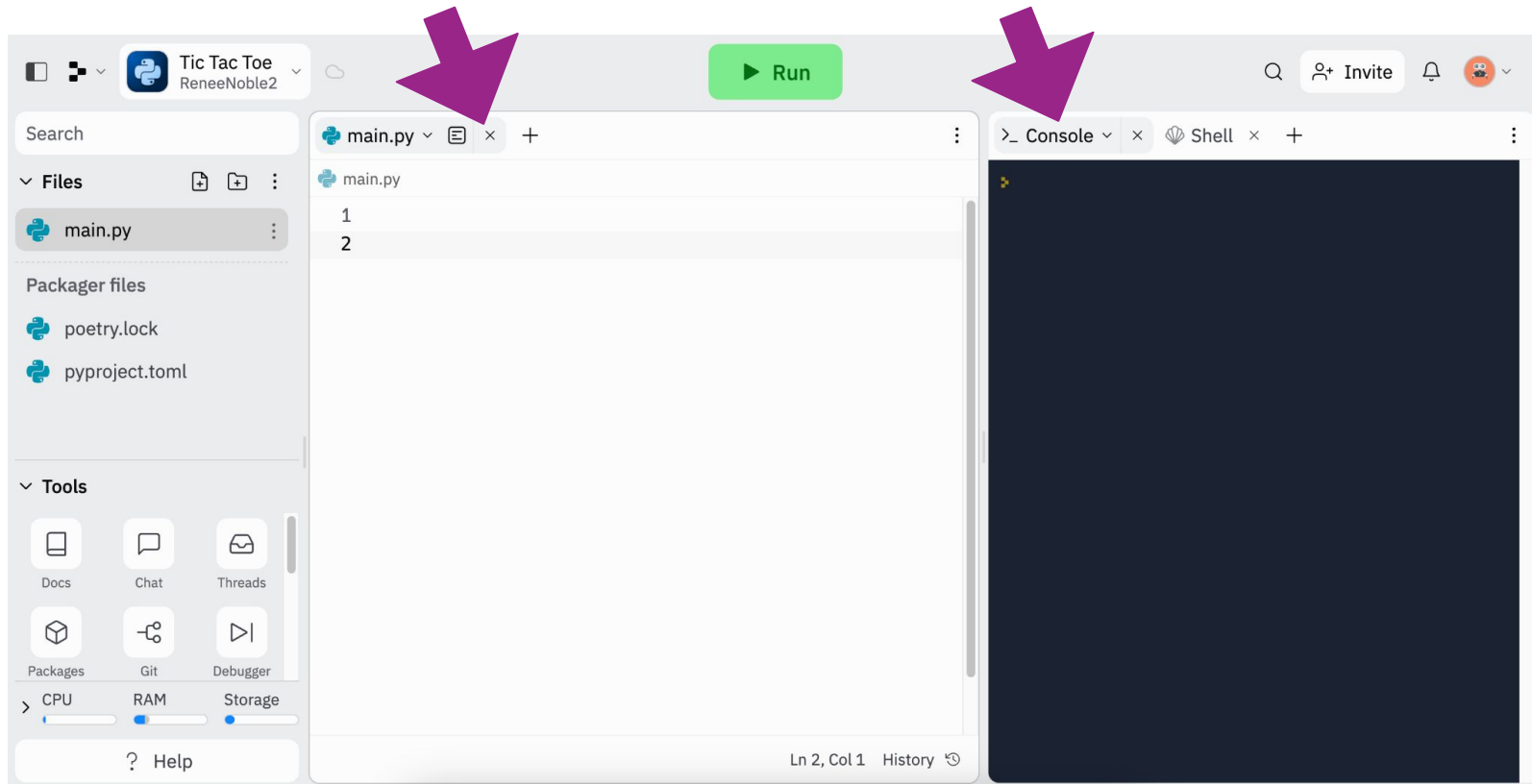
Test the **console**! Make a mistake!

Type by **button mashing** the keyboard!

Then press enter!

Did you get a big red error message?

Tech
Inclusion

Mistakes are great!

Good work you made an error!

SyntaxError: Invalid Syntax

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

Tech Inclusion

Write some code!!

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

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

Tech
Inclusion

Write some code!!

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

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

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

Tech
Inclusion

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
```

Asking a question!

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

Let's get the computer to ask us a question!

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

What do you think happens?

Asking a question!

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

Let's get the computer to ask us a question!

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

What do you think happens?

```
What is your name? Maddie
Hello Maddie
```

Tech Inclusion

Asking a question!

Writing input tells the computer to wait for a response

This is the question you want printed to the screen

```python
my_name = input('What is your name? ')
print('Hello ' + my_name)
```
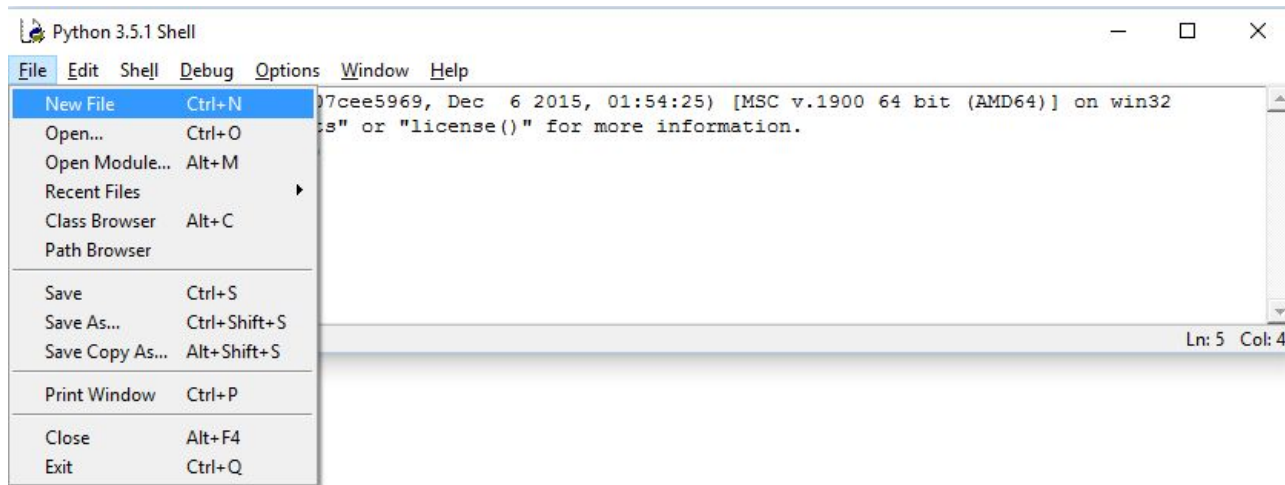
What do you think happens?

```
What is your name? Maddie
Hello Maddie
```

We can use the answer the user wrote that we then stored later!

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

Tech Inclusion

Adding a comment!

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

Use comments to write a note or explanation of our code

Comments make code easier for humans to understand

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

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

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

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 - Part 2

The tutors will be around to help!

If Statements and Lists

Tech
Inclusion

Conditions!

Conditions let us make decision.

First we test if the condition is met!

Then maybe we'll do the thing

If it's raining take an umbrella

Yep it's raining

...... take an umbrella

Tech
Inclusion

Conditions

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

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

Tech Inclusion

Conditions

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

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

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

>>>

Tech
Inclusion

Conditions

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

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

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

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

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???

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

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

Nothing!

Girls' Programming Network

Tech Inclusion

If statements

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

What happens?

If statements

```python
word = "GPN"
if word == "GPN":
  print("GPN is awesome!")
```
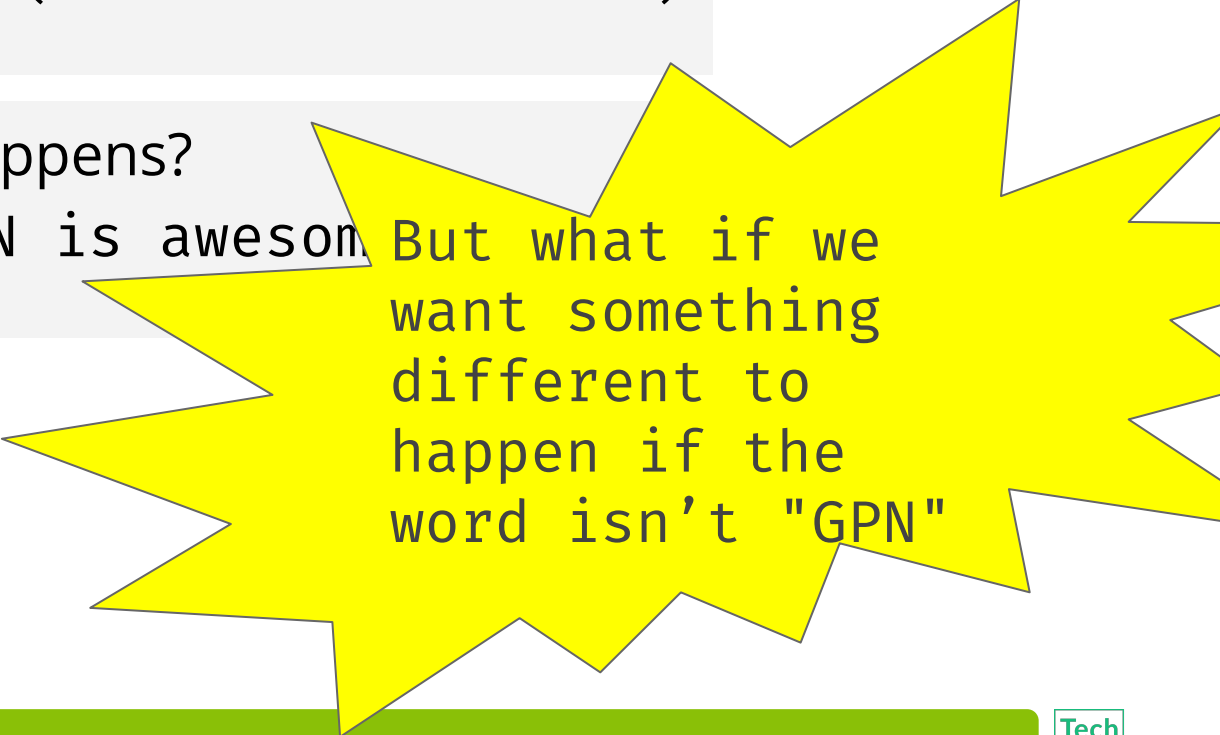
What happens?
```
>>> GPN is awesome!
```

If statements

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

What happens?
>>> GPN is awesome!

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

Tech Inclusion

Else statements

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

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

What happens?

Tech
Inclusion

Else statements

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

```python
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

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

What happens?

Tech
Inclusion

Elif statements

elif
Means we can give specific instructions for other words

```python
word = "Chocolate"
if word == "GPN":
    print("GPN is awesome!")
elif word == "Chocolate":
    print("YUMMM Chocolate!")
else:
    print("The word isn't GPN :(")
```
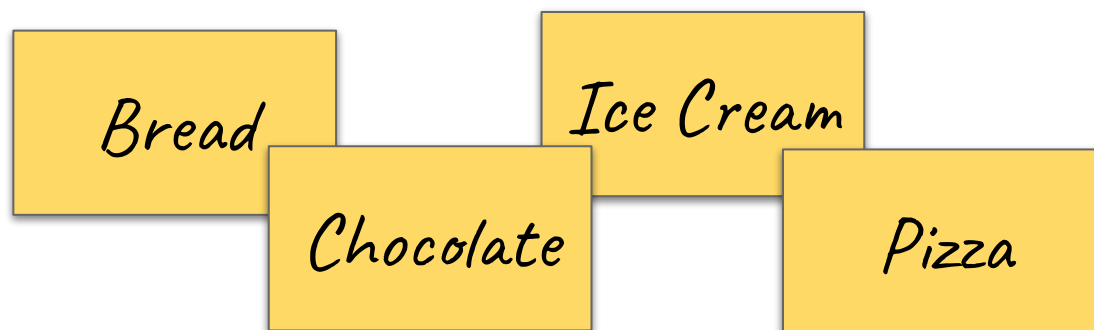
What happens?
>>> YUMM Chocolate!

Tech
Inclusion

Lists

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

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

Bread

Chocolate

Ice Cream

Pizza

We put it in one big shopping list!

- Bread
- Chocolate
- Ice Cream
- Pizza

Tech Inclusion

Lists

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

```
>>> shopping_item1 = "Bread"
>>> shopping_item2 = "Chocolate"
>>> shopping_item3 = "Ice Cream"
>>> shopping_item4 = "Pizza"
```
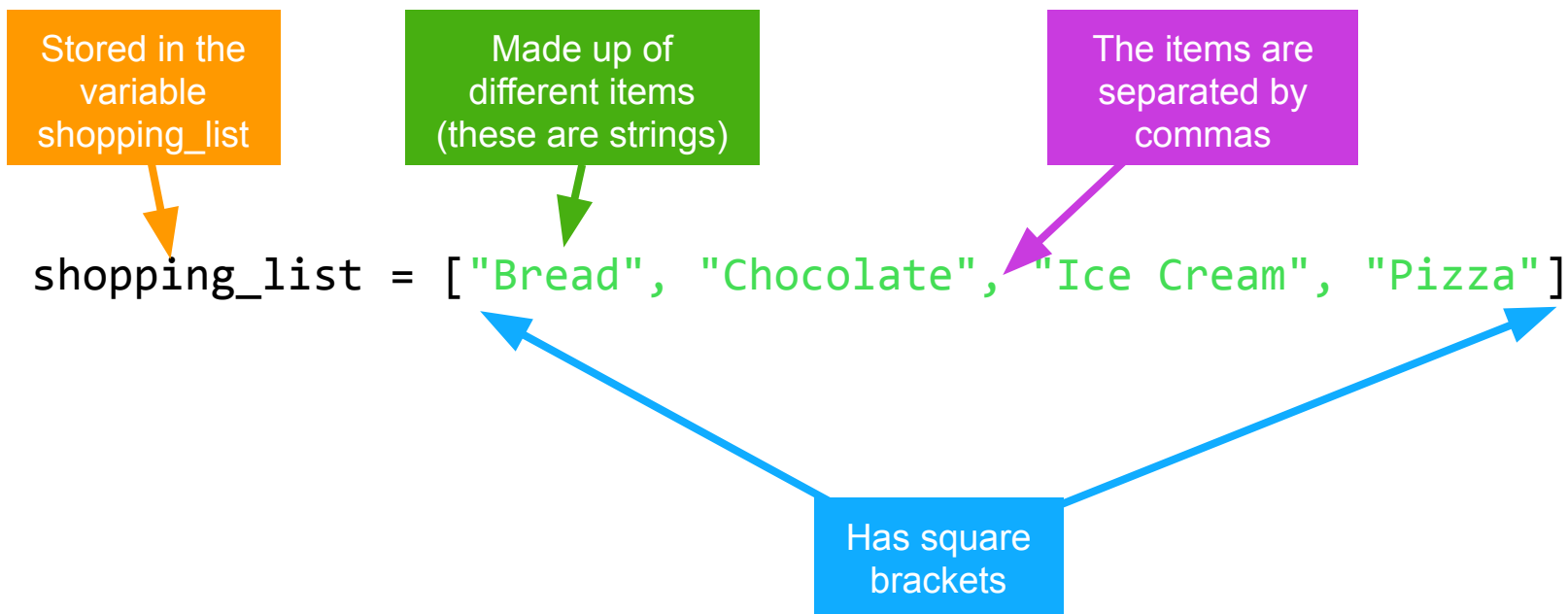
So much repetition!

Instead we use a python list!

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

List anatomy

Stored in the variable shopping_list

Made up of different items (these are strings)

The items are separated by commas

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

Has square brackets

Tech Inclusion

Project Time!

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

See if you can do Part 3

The tutors will be around to help!

Tech Inclusion
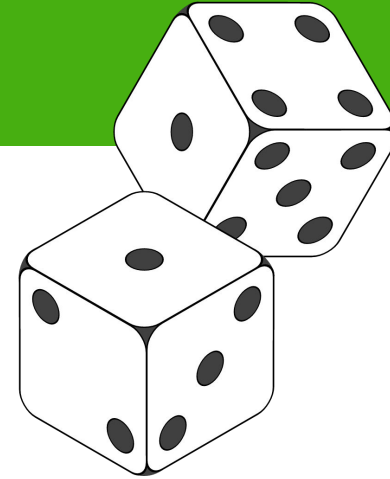
Random!

Tech
Inclusion

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)
   ```

Tech
Inclusion

Using the random module

You can also assign your random choice to a variable

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

Tech
Inclusion

Project Time!

Raaaaaaaaaandom! Can you handle that?

Let's try use it in our project!
Try to do Part 4

The tutors will be around to

For Loops

Tech
Inclusion

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

```python
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

Tech
Inclusion

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

Tech Inclusion

For Loops

This `i` is a temporary variable which will count how many times we have looped.

```python
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)

Tech
Inclusion

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:

```python
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 through a list:

```python
friends = 4
for i in range(friends):
    print("Hello friend!")
```

What's going to happen?

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

We do what's in the for loop as many times as what is in the "range"

Tech Inclusion

Project Time!

Now you know how to use a for loop!

Try to do Part 5
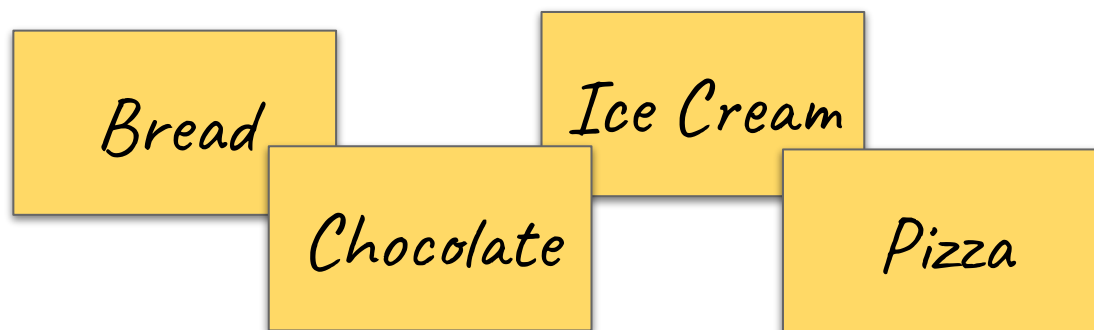**...if you are up `for` it!**

The tutors will be around to help!

Tech Inclusion

Lists and Dictionaries

Lists

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

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

Bread

Ice Cream

Chocolate

Pizza

We put it in one big shopping list!

- Bread
- Chocolate
- Ice Cream
- Pizza

Tech Inclusion

Lists

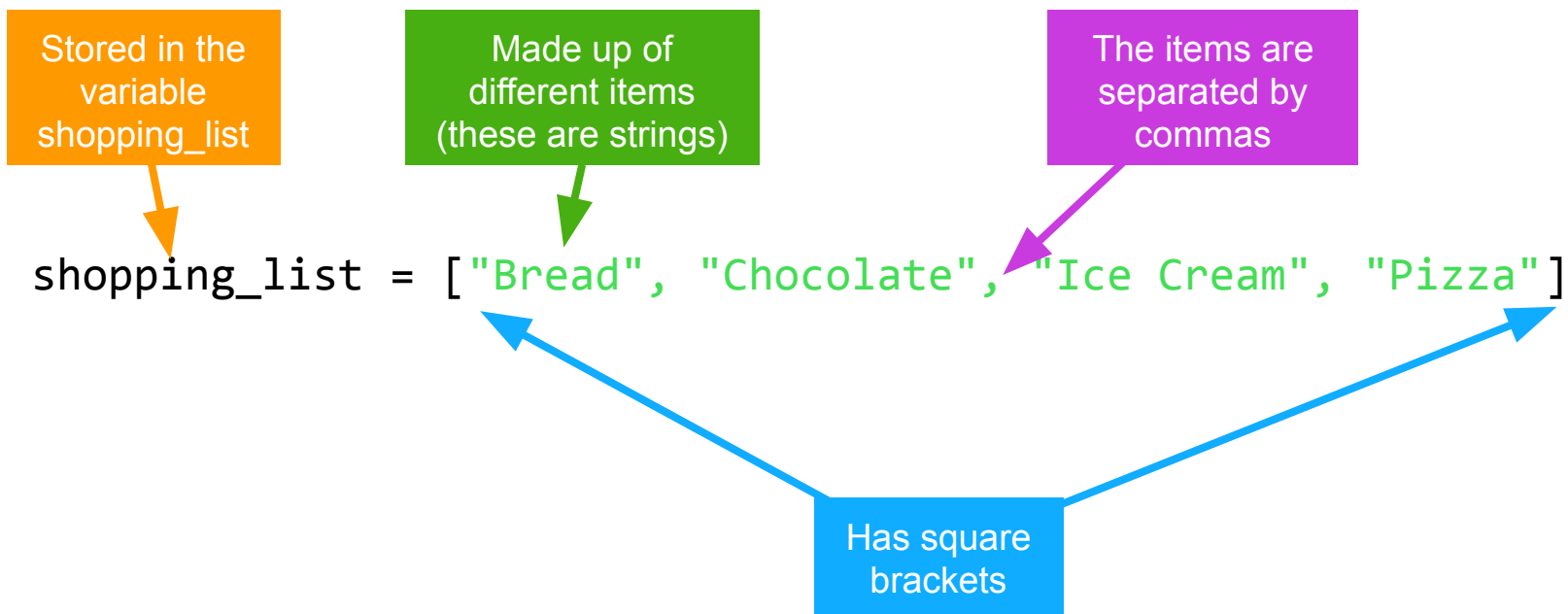**It would be annoying to store it separately when we code too!**

```python
shopping_item1 = "Bread"

shopping_item2 = "Chocolate"

shopping_item3 = "Ice Cream"

shopping_item4 = "Pizza"
```
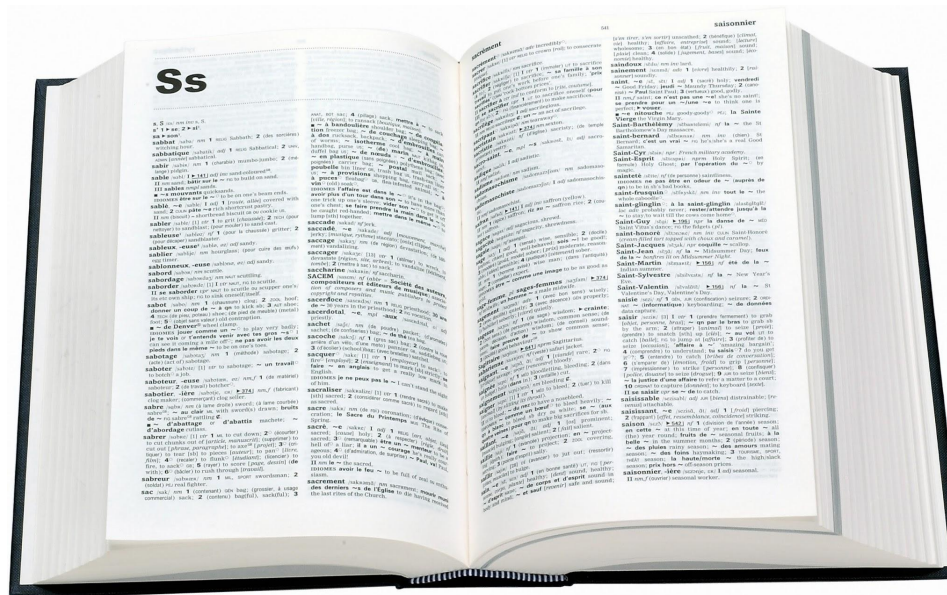
So much repetition!!

Instead we use a python list!

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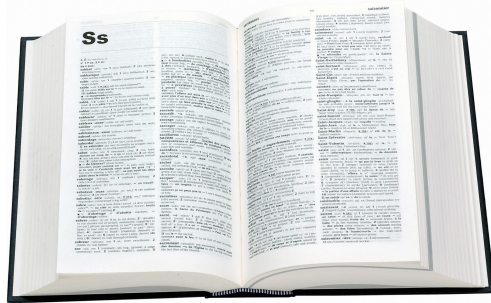
Tech
Inclusion

List anatomy

Stored in the variable shopping_list

Made up of different items (these are strings)

The items are separated by commas

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

Has square brackets
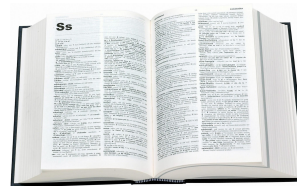
Tech Inclusion

Dictionaries!

Dictionaries!

You know dictionaries!

They're great at looking up thing by a word, not a position in a list!

Look up
Hello

Get back

A greeting (salutation) said when meeting someone or acknowledging someone's arrival or presence.

Tech Inclusion

Looking it up!

There are lots of times we want to look something up!

Competition registration

Team Name → List of team members

Phone Book

Name → Phone number

Vending Machine

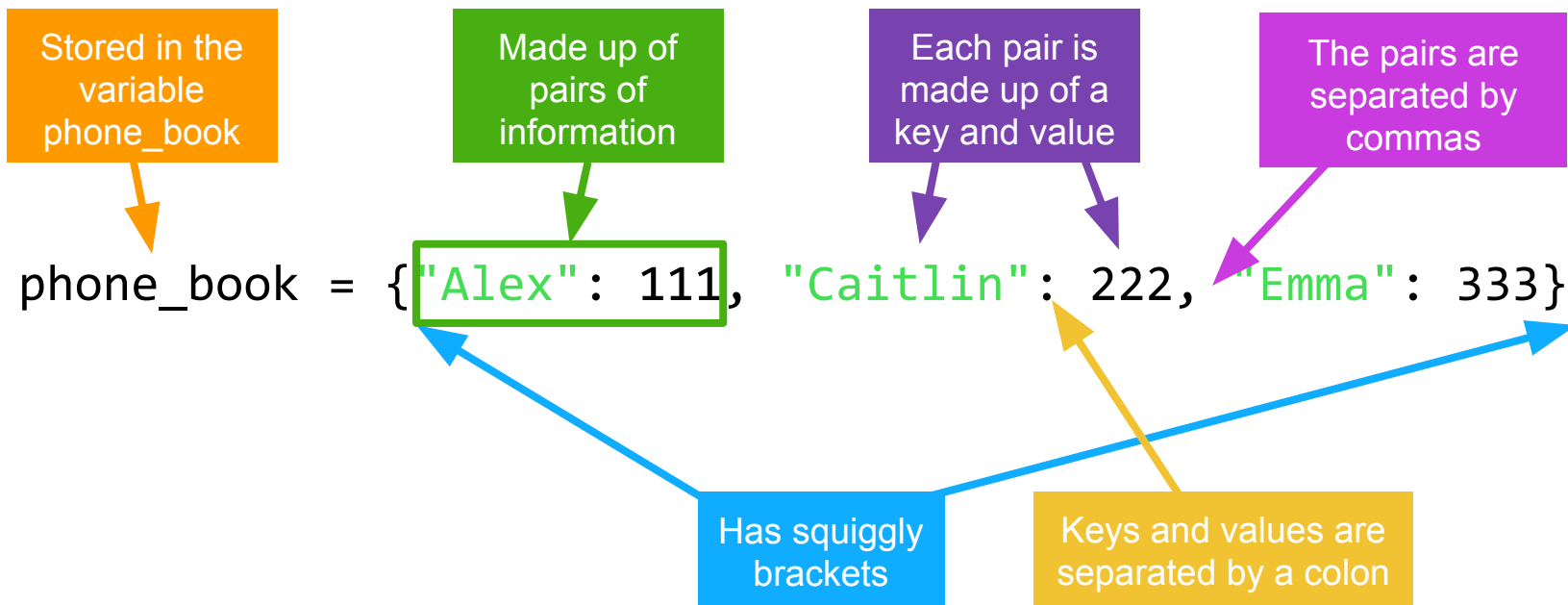Treat Name → Price

Tech Inclusion

Phone Book

Name → Phone number

Key

Value

**We can use a dictionary for anything with a key → value pattern!**

Dictionaries anatomy!

This is a python dictionary!

Stored in the variable phone_book

Made up of pairs of information

Each pair is made up of a key and value

The pairs are separated by commas

```
phone_book = {"Alex": 111, "Caitlin": 222, "Emma": 333}
```

Has squiggly brackets

Keys and values are separated by a colon

This dictionary has Alex, Caitlin and Emma's phone numbers

Tech Inclusion

Cups!!

Remember the cups activity from the start of the day?
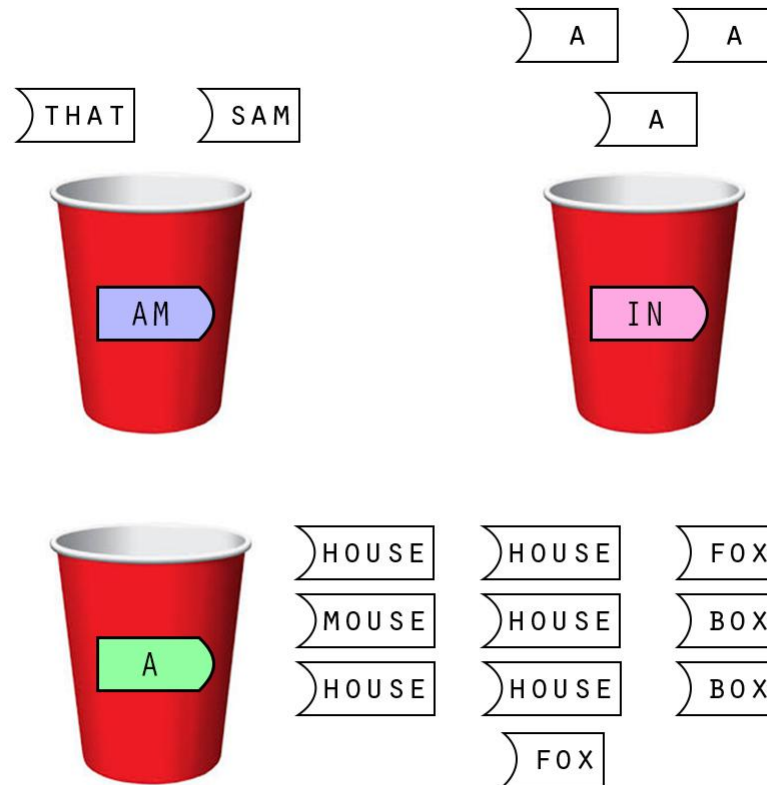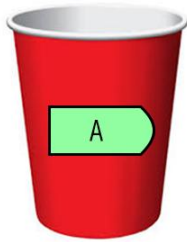
Tech
Inclusion

A Single Cup!

The word "A" **can be followed by** **Any of these words**

Key

Value

Tech Inclusion

A Single Cup!

The word "A" **can be followed by** **Any of these words**
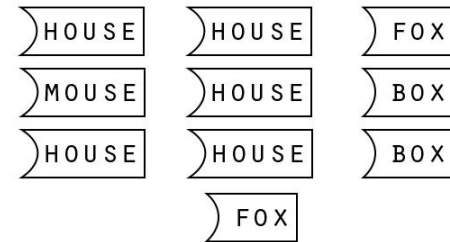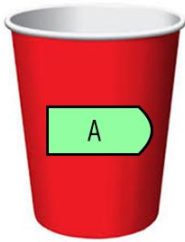
HOUSE	HOUSE	FOX
MOUSE	HOUSE	BOX
HOUSE	HOUSE	BOX
	FOX	
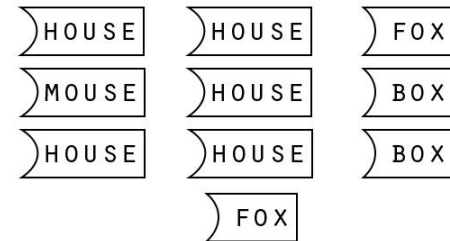
We can store the slips of paper as a python list!

```python
['house', 'mouse', 'house',
'mouse', 'box', 'fox', 'box',
'fox', 'house', 'mouse']
```

Tech Inclusion

A Single Cup!

The word "A" **can be followed by** **Any of these words**
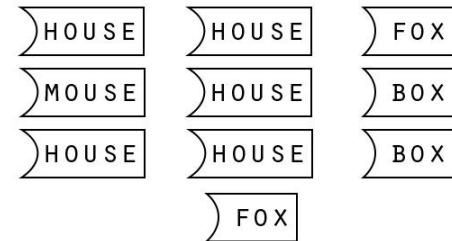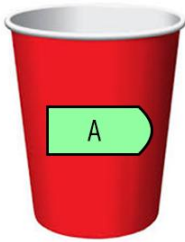

We want to look up the word "a" and get back the list!

```
{ 'a'  :  ['house', 'mouse', 'house',
           'mouse', 'box', 'fox', 'box',
           'fox', 'house', 'mouse']}
```

Tech Inclusion

A Single Cup!

So we get a Dictionary with a List value!

```
{ 'a' : ['house', 'mouse', 'house',
         'mouse', 'box', 'fox', 'box',}
         'fox', 'house', 'mouse']
```

Key

Value

If you look up "A" you get back a list of all the words that can follow "a"

Cups → Dictionary with lists!

Here's what it looks like for a few more cups!

```
cups = {'am': ['Sam', 'That'],
        'In': ['a', 'a', 'a'],
        'a' : ['house', 'mouse',
               'house', 'mouse',
               'box', 'fox', 'box',
               'fox', 'house',
               'Mouse']
        ....}
```
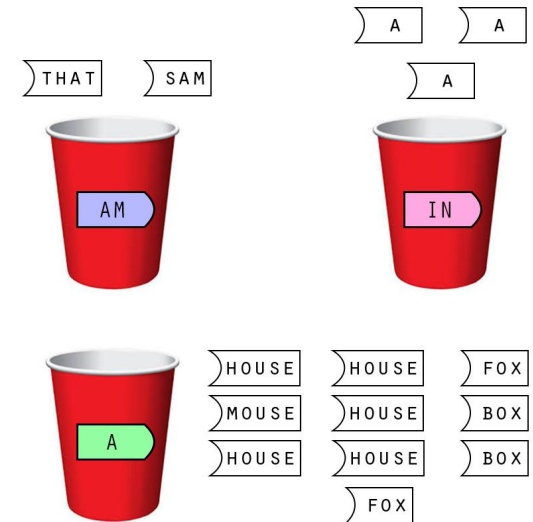
You can get the whole cup dictionary from today's website!

Project time!

You now know all about lists and dictionaries!

Let's put what we learnt into our project

Try to do Part 6

The tutors will be around to help!

Tech
Inclusion

More Dictionaries and Lists!

Before we start this lecture
Trying doing Part 0 in your second workbook!

Tech
Inclusion

Getting words from sample text

In order to be able to read in lots of text we need to be able to turn sentences into a list of words.

We can do this by using .split() on our text!

Using split

```
text = "a really cool sentence"

words = text.split()

print(words)
```

What do you think words will be?

Tech
Inclusion

Using split

```
text = "a really cool sentence"

words = text.split()

print(words)
```

What do you think words will be?

["a", "really", "cool", "sentence"]

Tech
Inclusion

More things you can do with lists!

There's lots of cool things we can do with lists! Like:

Getting the length of a list

```
words = ["a", "really", "cool", "sentence"]
print(len(words))
```

Adding new items to a list

```
words.append("yay")
print(words)
```

More things you can do with lists!

There's lots of cool things we can do with lists! Like:

Getting the length of a list

```
words = ["a", "really", "cool", "sentence"]
print(len(words))
 4
```

Adding new items to a list

```
words.append("yay")
print(words)
```

More things you can do with lists!

There's lots of cool things we can do with lists! Like:

Getting the length of a list

```
words = ["a", "really", "cool", "sentence"]
print(len(words))
 4
```

Adding new items to a list

```
words.append("yay")
print(words)
 ["a", "really", "cool", "sentence", "yay"]
```

Accessing Lists!

This favourites `list` holds four strings in order:

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

We can count out the items using index numbers!

0	1	2	3

Remember: Indices start from zero!

Tech
Inclusion

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?
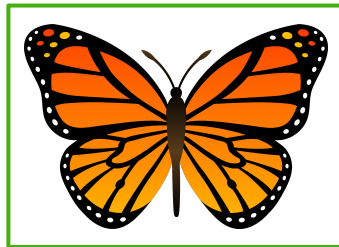
Accessing Lists

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 |

Tech Inclusion

Going Negative

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

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

What would `faves[-2]` return?

Tech Inclusion

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

Tech
Inclusion

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
```

Tech
Inclusion

Updating our dictionaries!

We've seen how to use dictionaries - but how do we update existing ones? Let's have a look at a phone book example!

```
>>> phone_book = {
        "Alex": 111, "Caitlin": 222, "Emma": 333
    }
```

- We met Rowena! Let's add her to our phone book

```
>>> phone_book["Rowena"] = 444
>>> phone_book
{ "Alex": 123, "Caitlin": 222, "Emma": 333,
"Rowena": 444 }
```

Tech
Inclusion

Lists in dictionaries!

We've been using lists as the values of our dictionary like this:

- Let's make some sports teams:

```
>>> team_members = {
        "Sydney": ["Pauline", "Srishti", "Amara"],
        "Perth": ["Crischell", "Ash", "Taylah"]
    }
```

- What happens if you do:

```
>>> team_members["Sydney"]
```

- What if we did this?

```
>>> team_members["Perth"].append("Priya")
```

Tech
Inclusion

Lists in dictionaries!

We've been using lists as the values of our dictionary like this:

- Let's make some sports teams:

```
>>> team_members = {
        "Sydney": ["Pauline", "Srishti", "Amara"],
        "Perth": ["Crischell", "Ash", "Taylah"]
    }
```

- What happens if you do:

```
>>> team_members["Sydney"]
 ["Pauline", "Srishti", "Amara"]
```

- What if we did this?

```
>>> team_members["Perth"].append("Priya")
```

Lists in dictionaries!

We've been using lists as the values of our dictionary like this:

- Let's make some sports teams:

```
>>> team_members = {
        "Sydney": ["Pauline", "Srishti", "Amara"],
        "Perth": ["Crischell", "Ash", "Taylah"]
    }
```

- What happens if you do:

```
>>> team_members["Sydney"]
 ["Pauline", "Srishti", "Amara"]
```

- What if we did this?

```
>>> team_members["Perth"].append("Priya")
  ["Pauline", "Srishti", "Amara", "Priya"]
```

Project Time!

Now you know even more about Dictionaries and Lists!

In your second workbook,

Try Parts 1 - 4

The tutors will be around to help!

Tech Inclusion