

Welcome to the Labs

Scissors Paper Rock!

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

Click Content for your room. 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!

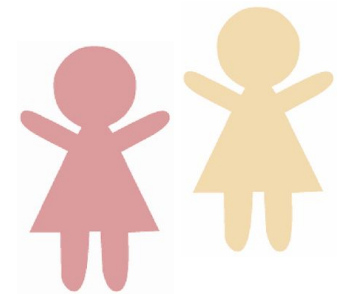
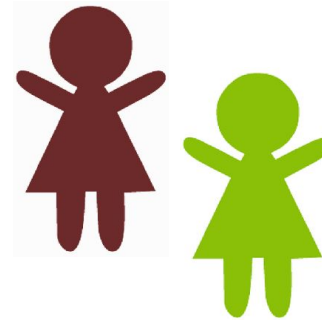
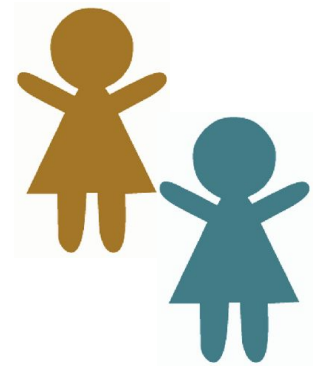
Today's project!

Scissors Paper Rock!

Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!

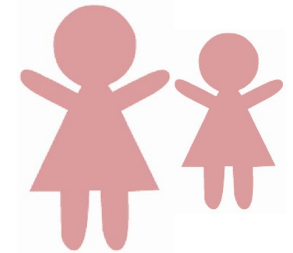
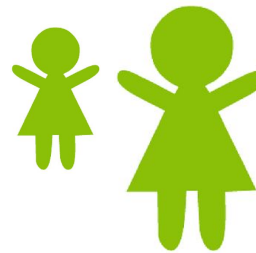
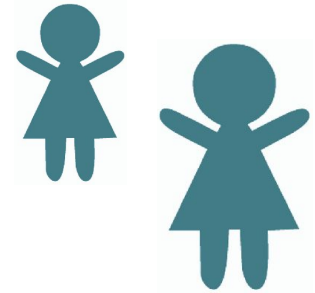
Who will be the champion?



Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

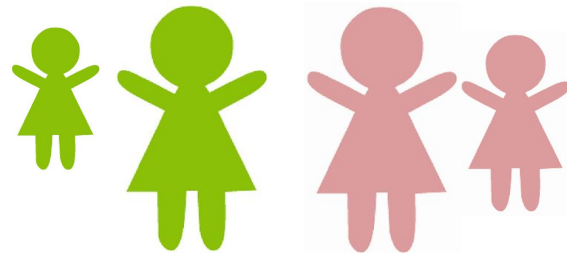
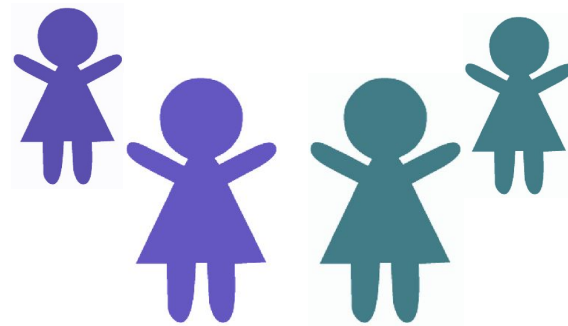
Who will be the champion?



Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

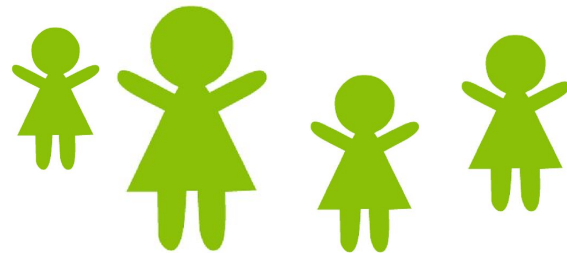
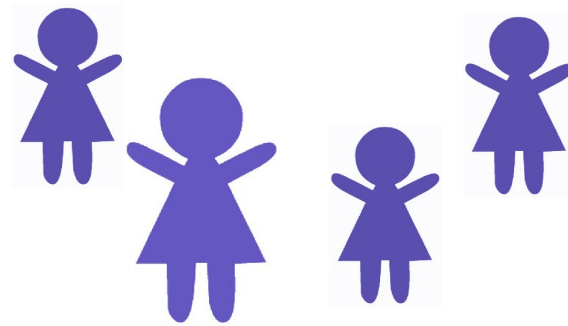
Who will be the champion?



Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

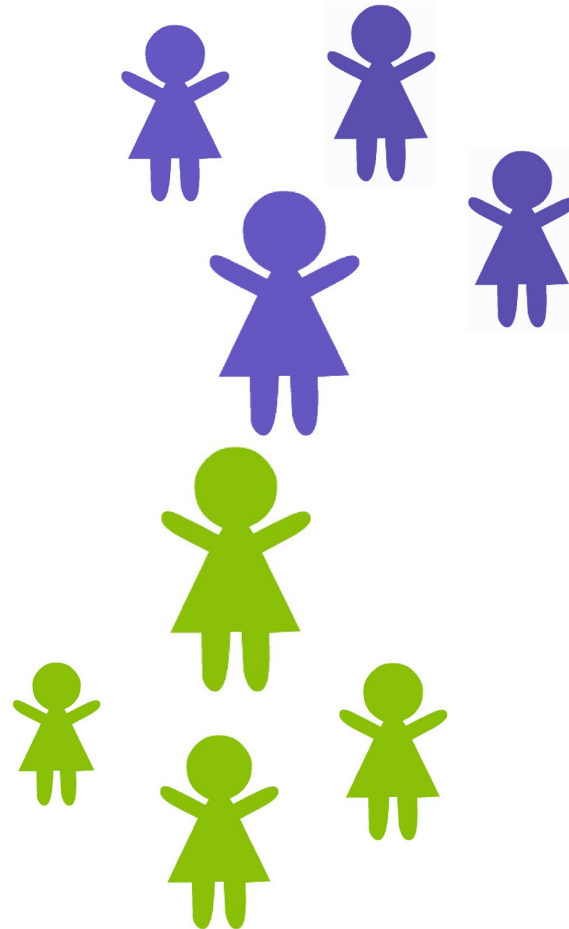
Who will be the champion?



Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

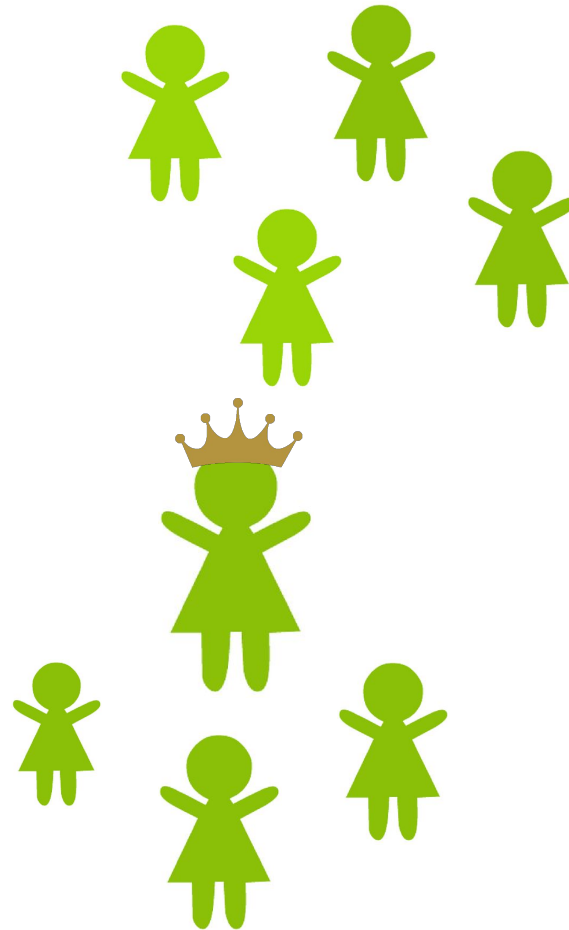
Who will be the champion?



Ultimate Scissors Paper Rock

1. Start with a partner
2. play scissors paper rock!
3. If you win they become your cheer squad!
And their squad becomes your squad!
4. Find a new partner!
5. Keep playing until there is only one person left!

Who will be the champion?



Scissors Paper Rock

How did you go? Did you win?

Some of the things that we need to do to play scissors paper rock include:

- We have to select a move (out of scissors, paper and rock)
- Our opponent has to select a move
- We need to know what combinations of move result in win, lose or tie.
- We need to compare our moves to see who won!
- We have to congratulate the winner!

We'll be programming these actions today! Our opponent is going to be the computer.

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?



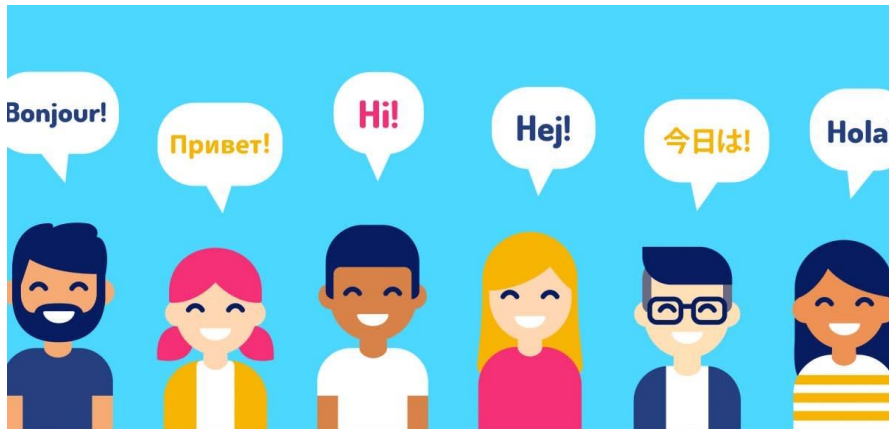
Programming is not a bunch of crazy numbers!

It's giving computers a set of instructions!



A special language

Humans have languages like English, French, Spanish, Mandarin



https://images.saymedia-content.com/.image/t_share/MTc0MTAyNzI3ODUxMjU1MjQx/how-to-easily-learn-a-language.jpg

And computers have languages like Python, Java, C and PHP



Problem solving

Programming is how we get computers to solve complicated problems for us, saving us both time and effort!

This might be solving maths problems or counting words in a paragraph!



People are smart, computers are dumb!

Computers do exactly what they're told. They follow instructions given to them in order, just like a cook following a recipe.



If the instructions are not in the correct order, we will end up with a mess!

Everyone/thing has strengths!



- Incomplete instructions are okay - we can fill in the blanks!
- Improves everyday



- Incomplete instructions are not okay
- Improves when you tell it how to

Intro to Python

Let's get coding!

Where do we program? In Replit!

Go to replit.com

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.
- If you don't have any of these, ask a tutor for one of our spare replit accounts to use today.

replit

Create a Replit account

Sign up for teachers

+ Create Account

Have an account? [Log In](#)
Trouble signing up? [Get help](#)

By continuing, you agree to Replit's [Terms of Service](#) and [Privacy Policy](#), and to receiving emails with updates.

Continue with Google

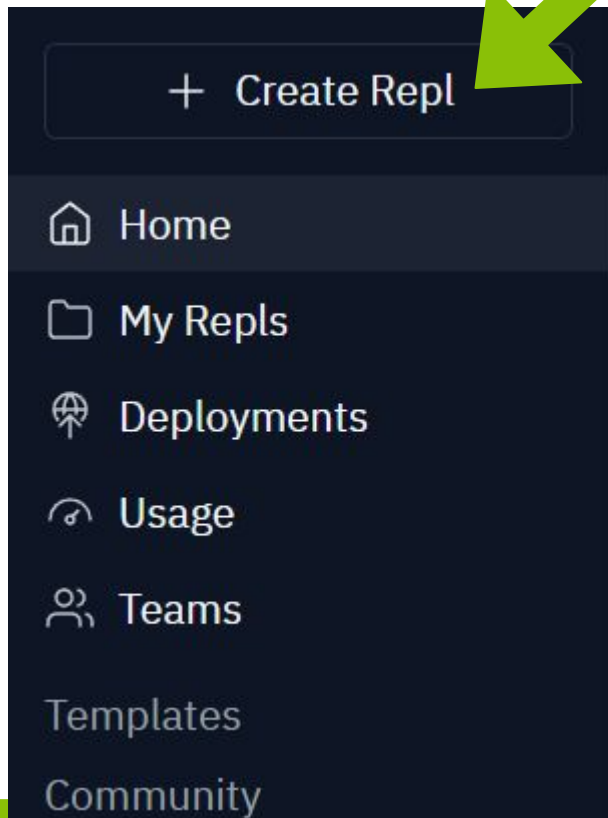
Continue with Github

Continue with Facebook

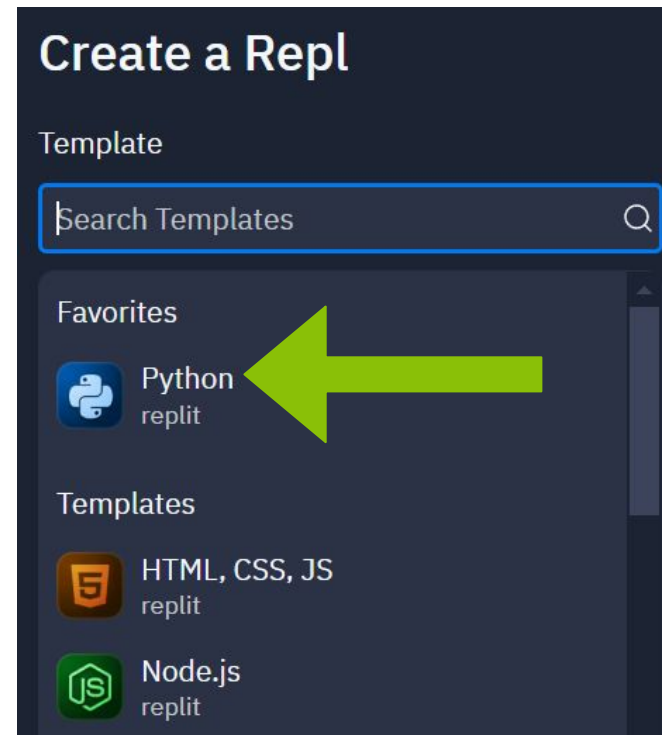
Continue with Apple

Creating our Repl It Project

Let's create a new project



Select Python for the project template

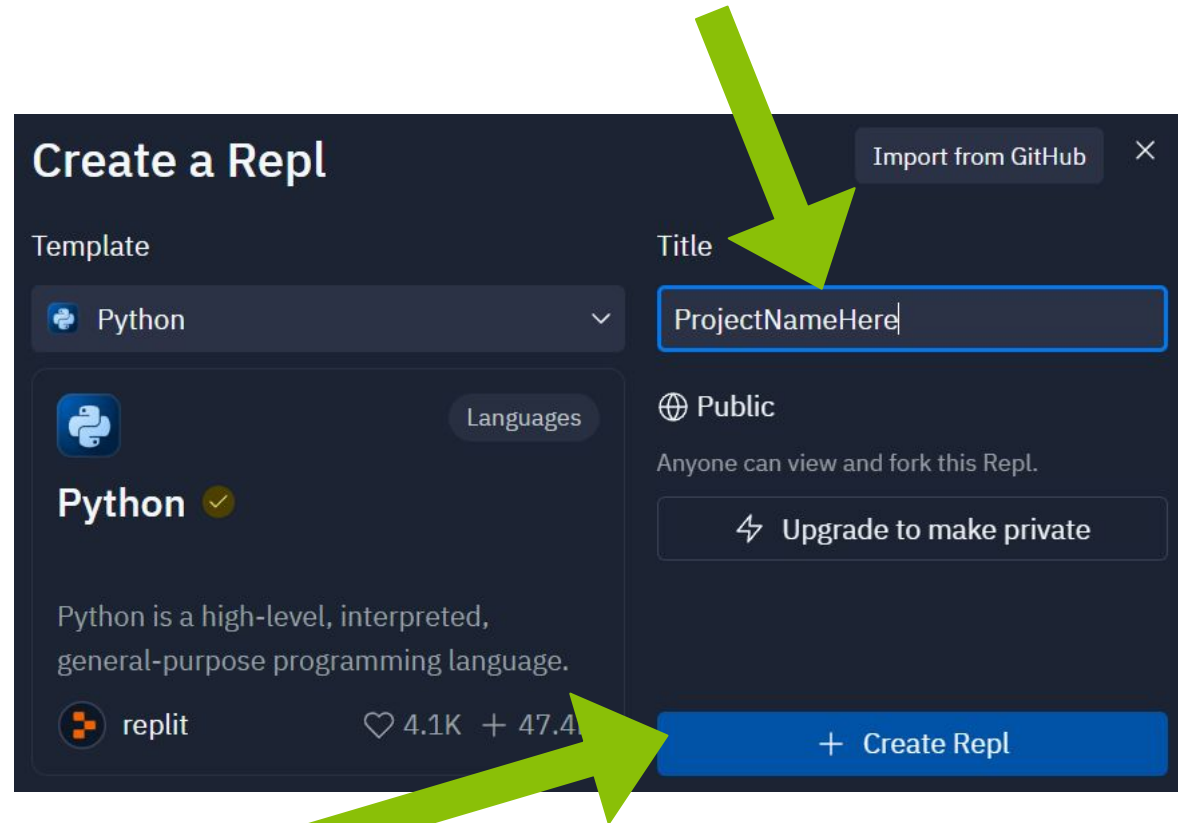


Creating our Repl It Project

**Don't forget to
give your
project a name!**

Name it after
today's project!

Click Create Repl



Setting our Repl It Project

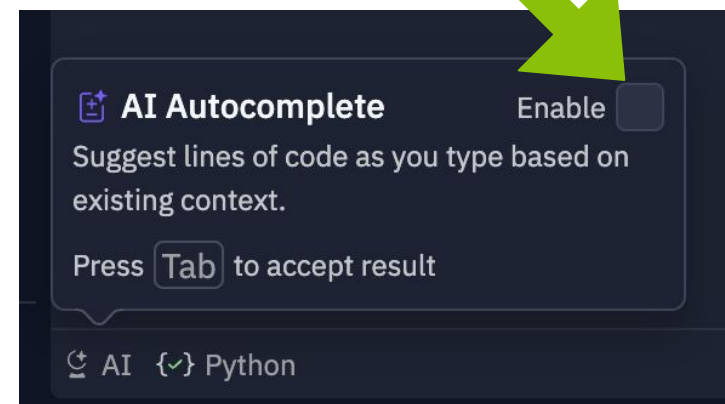
We can't learn if something else is doing all the work!

So we are going to disable AI Autocomplete for this project!



Click the small AI icon in the bottom left corner

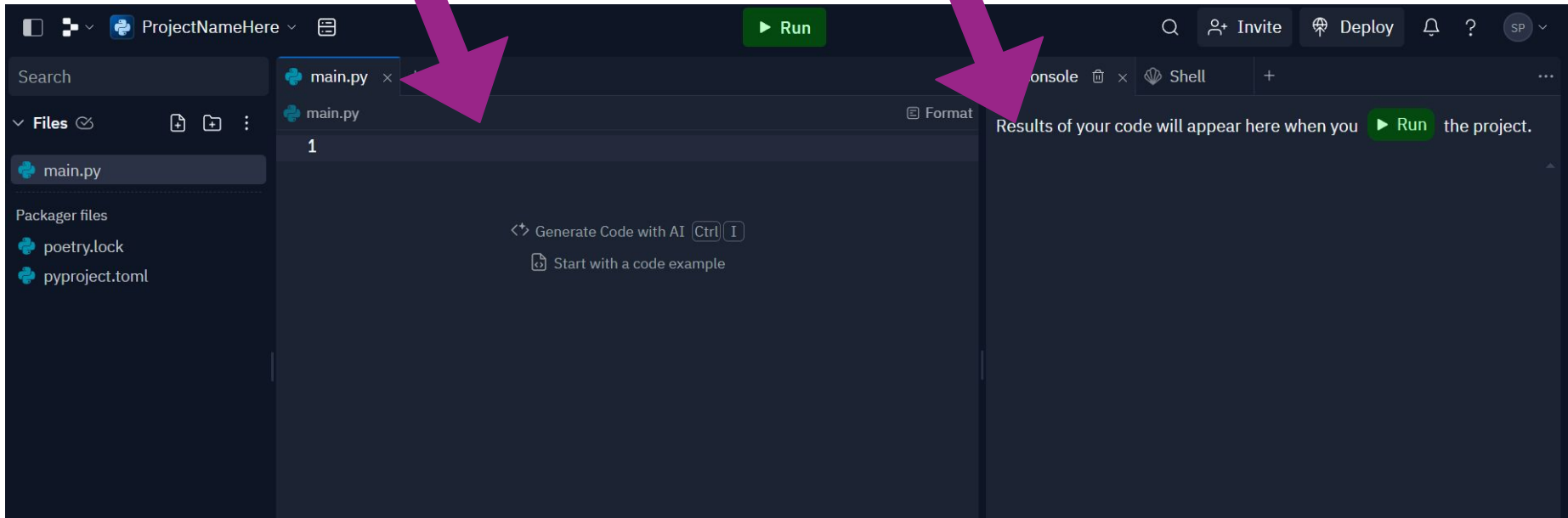
Then sure there is no tick in this box



We're ready to code!

We'll write our project here in main.py

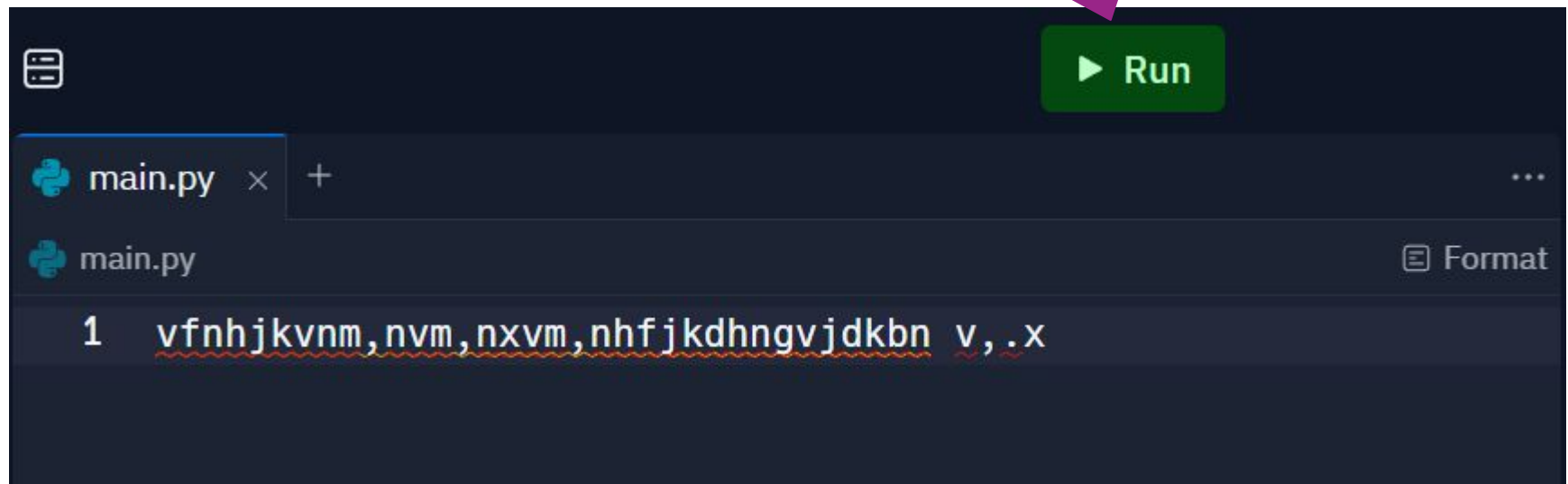
When you run your code, the results will display in the Console here



Run a test! Make a mistake!

Type by **button mashing** the keyboard!

Click Run



```
1 vfnhjkvnm,nvm,nxvm,nhfjkdhngvjdkbn v,.x
```

Did you get an error message in the Console?

Mistakes are great!

Good work you made an error!

*SyntaxError:
Invalid Syntax*

- 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!

*ImportError:
No module
named humour*



*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 code window

Then press Run!

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

Did it print:

hello world

???

A calculator for words!?

What do you think these bits of code do?

Try them and see!

```
print("cat" + "dog")
```

```
print("tortoise" * 3)
```


Calculator for... words!?

What do you think these bits of code do?

Try them and see!

```
print("cat" + "dog")
```

```
catdog
```

```
print("tortoise" * 3)
```

Calculator for... words!?

What do you think these bits of code do?

Try them and see!

```
print("cat" + "dog")
```

```
catdog
```

```
print("tortoise" * 3)
```

```
tortoisetortoisetortoise
```

No Storing is Boring!

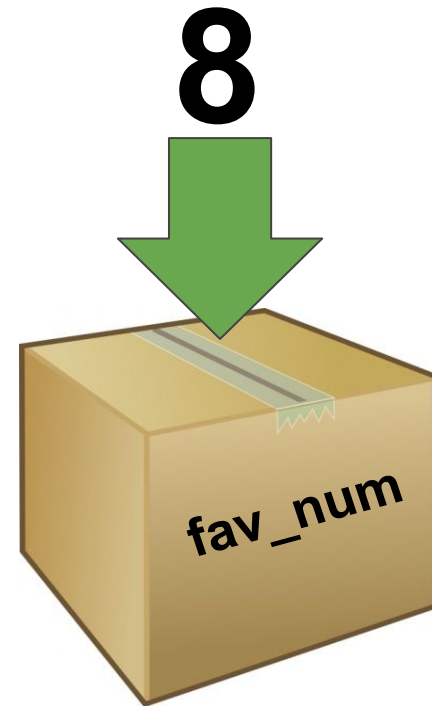
It's useful to be able to remember things for later!

Computers remember things in "**variables**"

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

Let's make our favourite number 8 today!

In our code we would write
`fav_num = 8`



Variables

Instead of writing the number 8, we can now use **fav_num** in our code.



Wherever the computer sees **fav_num**, it will use the **number 8**

fav_num - 6

=> **2**

fav_num * 2

=> **16**

fav_num + 21

=> **29**

fav_num / 2

=> **4**

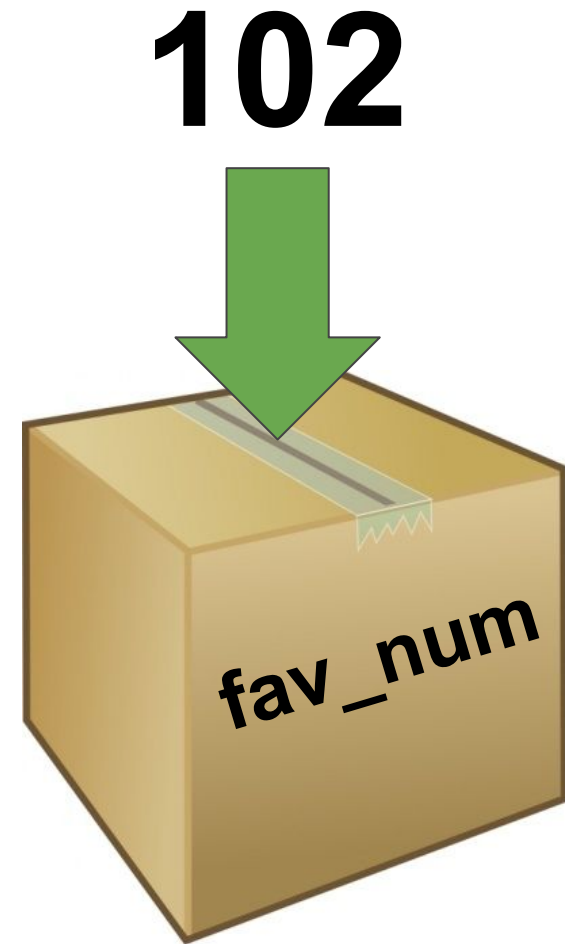
Variables

Variables are useful for storing things that change

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

What if we changed
fav_num to **102**.

fav_num = 102



Variables

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



$$\text{fav_num} - 6$$
$$\Rightarrow \mathbf{96}$$

$$\text{fav_num} + 21$$
$$\Rightarrow \mathbf{123}$$

$$\text{fav_num} * 2?$$
$$\Rightarrow \mathbf{204}$$

$$\text{fav_num} / 2?$$
$$\Rightarrow \mathbf{51}$$

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

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



No variables VS using variables



4
Changes

8 - 6	→	102 - 6
8 * 2	→	102 * 2
8 + 21	→	102 + 21
8 / 2	→	102 / 2



1
Change

fav_num = 8	→	fav_num = 102
fav_num - 6	→	fav_num - 6
fav_num * 2	→	fav_num * 2
fav_num + 21	→	fav_num + 21
fav_num / 2	→	fav_num / 2

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)
```

```
My favourite animal is a dog  
My favourite animal is a cat  
My favourite animal is a catdog
```



Variables

Your turn!

Can you guess what each `print` will do?

```
>>> 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?

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

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

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

Variables

Your turn!

Can you guess what each `print` will do?

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

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

Variables

Your turn!

Can you guess what each `print` will do?

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

Variables

Your turn!

Can you guess what each `print` will do?

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

Different data!

There are lots of types of data! Our main 4 ones are these:

Strings

Things in quotes used for storing text

```
"This is a string"
```

Ints

Whole numbers we can do maths with

```
a = 1  
b = 2  
print(a + b)
```

Floats

Decimal numbers for maths

```
a = 1.5  
b = 2.0  
print(a / b)
```

Booleans

For **True** and **False**

```
a = 5 > 3  
boring = False
```



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!

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? ')\nprint('Hello ' + my_name)
```

What do you think happens?

Asking a question!

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

What do you think happens?

What is your name? Maddie

Hello Maddie

Asking a question!

Store the answer
in the variable
my_name

Writing input tells
the computer to
wait for a response

This is the question
you want printed to
the screen

```
my_name = input('What is your name? ')\nprint('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!

Asking a question!

How would we ask somebody for their favourite type of cake?

How would we print their answer?

Give it a try on your own computer first!



```
What cake do you like? chocolate  
chocolate cake for you!
```

Asking a question!

How would we ask somebody for their favourite type of cake?

How would we print their answer?

Give it a try on your own computer first!

```
flavour = input("What cake do you like? ")
```

```
What cake do you like? chocolate  
chocolate cake for you!
```

Asking a question!

How would we ask somebody for their favourite type of cake?

How would we print their answer?

Give it a try on your own computer first!

```
flavour = input("What cake do you like? ")  
print(flavour + "cake for you!")
```

```
What cake do you like? chocolate  
chocolate cake for you!
```

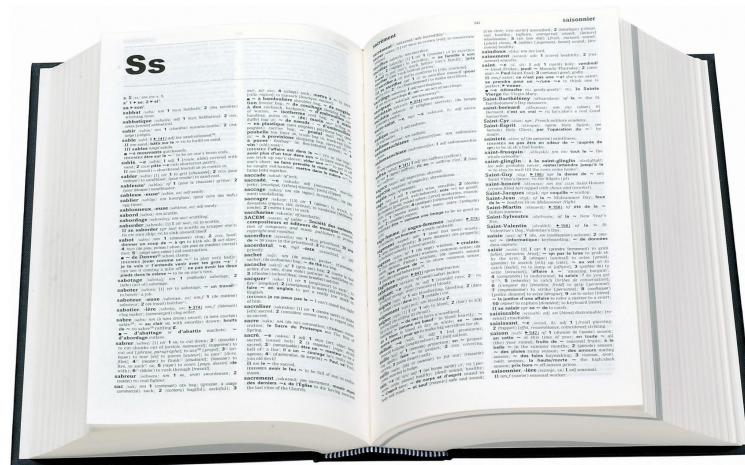
Project time!

You now know all about printing and variables!

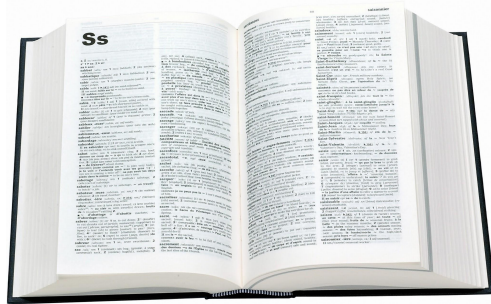
Let's put what we learnt into our project
Try to do Part 0 - Part 2

The tutors will be around to help!

Dictionaries



Dictionaries!

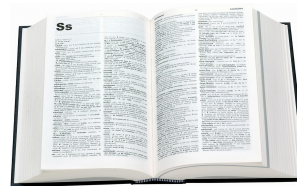


You know dictionaries!

**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.***



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

Looking it up!



Phone Book

Name → Phone number

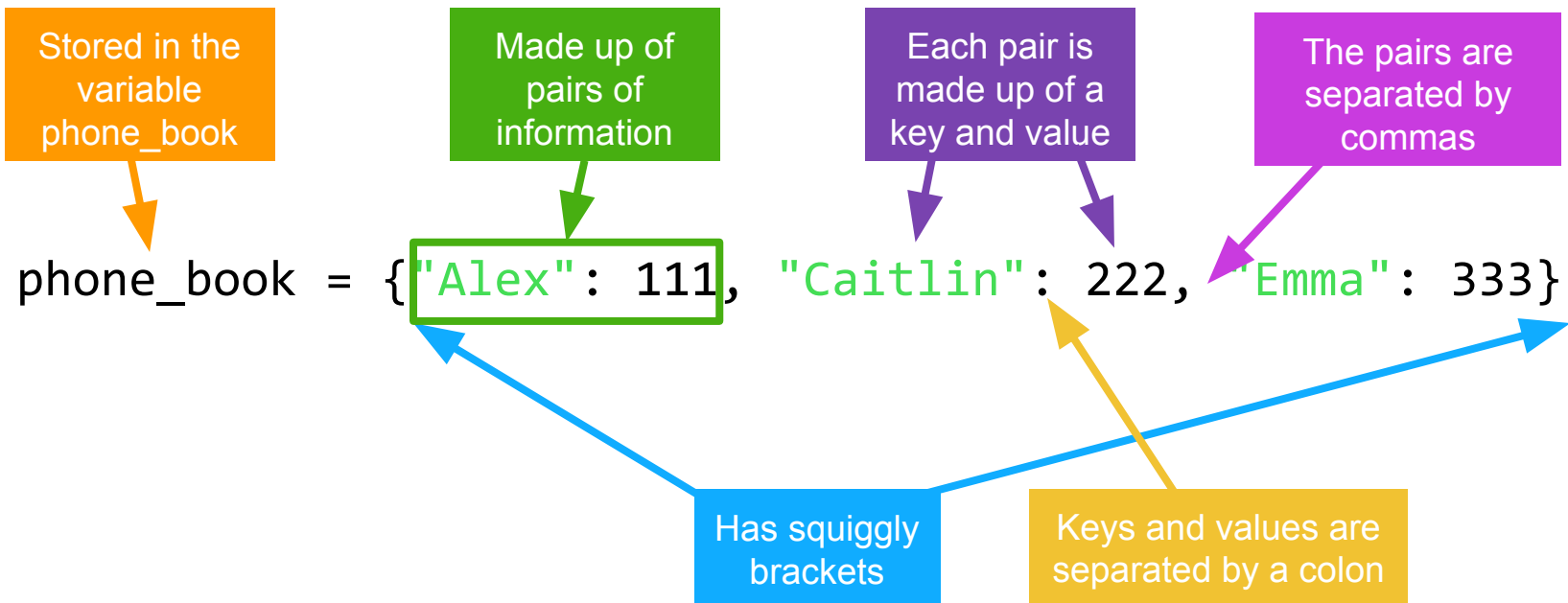
↑
Key

↑
Value

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

Dictionaries anatomy!

This is a python dictionary!



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

Playing with dictionaries!

Let's try using the phone book in Replit

1. Copy in the dictionary! Add your own made up phone number!

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

2. Try this: `print(phone_book["Alex"])`

3. How would you look up Emma's phone number?

4. Look up the name of someone who is not in the phone book? What happens?

Save it for later!

Sometimes we don't need the info right now.

Let's store it in a variable and use it later!

1. Look up Alex's phone number and store it in a variable

```
alexs_number = phone_book["Alex"]
```

2. Print out a message using alexs_number

```
print("Alexs number is: ", alexs_number)
```

3. Repeat task 1 and 2 for another person in the phone book!

Tuples!

Some data sticks together!

Tuples are like lists that you can't edit or add too!

It's a:

- **list of items**
- **in round brackets**
- **separated by commas**

Tuples are a way of grouping data!

`("January", "1st")`

`("December", "25th")`

`("April", "25th")`



Tuples in dictionaries!

We can use tuples as the key to a dictionary

1. Copy in the dictionary! Add your own made up date!

```
phone_book = {("January", "1st"): "New Years",  
              ("December", "25th"): "Christmas Day",  
              ("April", "25th"): "ANZAC Day" }
```

2. Try this: `phone_book[("January", "1st")]`
3. How would you look up what happens on the 25th of April
4. What happens if you we do: `phone_book[("25th", "December")]`

Project time!

You now know all about dictionaries!

Let's put what we learnt into our project
Try to do Part 3

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



If it's raining 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

`5 < 10`

`3 + 2 == 5`

`5 != 5`

`"Dog" == "dog"`

`"D" in "Dog"`

`"Q" not in "Cat"`



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

<code>5 < 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>
<code>3 + 2 == 5</code>		<code>"D" in "Dog"</code>
<code>5 != 5</code>		<code>"Q" not in "Cat"</code>

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

<code>5 < 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>
<code>5 != 5</code>		<code>"Q" not in "Cat"</code>

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

<code>5 < 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>
<code>5 != 5</code>	<code>False</code>	<code>"Q" not in "Cat"</code>

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

<code>5 < 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>	<code>False</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>	
<code>5 != 5</code>	<code>False</code>	<code>"Q" not in "Cat"</code>	

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

<code>5 < 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>	<code>False</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>	<code>True</code>
<code>5 != 5</code>	<code>False</code>	<code>"Q" not in "Cat"</code>	

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

<code>5 < 10</code>	<code>True</code>	<code>"Dog" == "dog"</code>	<code>False</code>
<code>3 + 2 == 5</code>	<code>True</code>	<code>"D" in "Dog"</code>	<code>True</code>
<code>5 != 5</code>	<code>False</code>	<code>"Q" not in "Cat"</code>	<code>True</code>

Booleans (True and False)

Python has some special comparisons for checking if something is **in** something else.

```
>>> "A" in "AEIOU"  
>>> "Z" in "AEIOU"  
>>> "a" in "AEIOU"
```

```
>>> animals = ["cat", "dog", "goat"]  
>>> "banana" in animals  
>>> "cat" in animals
```

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

Find out if it's **True**!

```
fave_num = 9000
if False:
    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 9000
- And it's not **True** that 9000 is less than 10
- So it is **False**!

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("that's a small number")
```



What do you think happens?

```
>>>
```



Nothing!

If statements

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

This line ...

... controls this line

If statements

Actually

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

This line ...

... controls anything below it
that is indented like this!

If statements

```
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

```
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 awesome!
```

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

Else statements

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

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

What happens??

Else statements

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

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

What happens??

```
>>> The word isn't GPN :(
```

Elif statements

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

elif
Means we can
give specific
instructions for
other words

What happens??

Elif statements

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

elif
Means we can
give specific
instructions for
other words

What happens??
>>>YUMMM Chocolate!

Simple Conditions!

We've learned about simple conditions like this one before.

They're really useful when you only want something to happen sometimes.



```
weather = "raining"  
if weather == "raining":  
    print("Take an umbrella!")
```

Complex Conditions!

But what if you want to only take an umbrella if it's raining and you're going outside?

You might do it like this:



```
weather = "raining"  
location = "outside"  
if weather == "raining":  
    if location == "outside":  
        print("Take an umbrella!")
```

Complex Conditions!

But what if you want to only take an umbrella if it's raining and you're going outside?

You might do it like this:



```
weather = "raining"  
location = "outside"  
if weather == "raining":  
    if location == "outside":  
        print("Take an umbrella!")
```

But that starts to get messy quickly.

AND

Instead you can do it like this!

```
weather = "raining"  
location = "outside"  
if weather == "raining" and location == "outside":  
    print("Take an umbrella!")
```

This is easier to read and stops things getting messy, especially if you have lots of conditions to check.



Project Time!

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

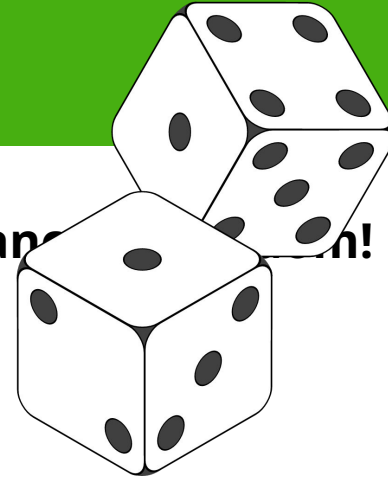
See **if you can do Part 4**

The tutors will be around to help!

Random!

That's so random!

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



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 5

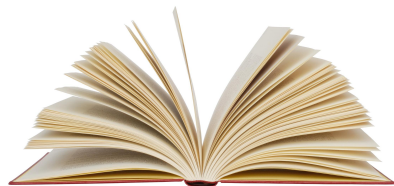
The tutors will be around to

For Loops

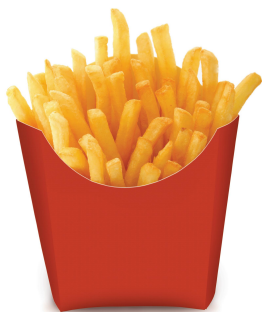
For Loops

For loops allow you to do something **for a number of times or for each item in a group**

There are many real world examples, like:



**For each page in this book:
Read**



**For each chip in this bag of chips:
Eat**

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?

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

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!
```

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

Asking a question with a number answer!

It's common to ask the user to enter a number

Input always gives us a string of text

We need to turn the **string** into a number before we can use it as a range in a for loop

We do this by using **int()**

```
no_of_turns = int(input("How many times: " ))  
for i in range(no_of_turns)  
    Do something
```

Project Time!

Now you know how to use a for loop!

Try to do Part 6

...if you are up **for it!**

And Extension parts 7-11

The tutors will be around to help!

Tell us what you think!

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