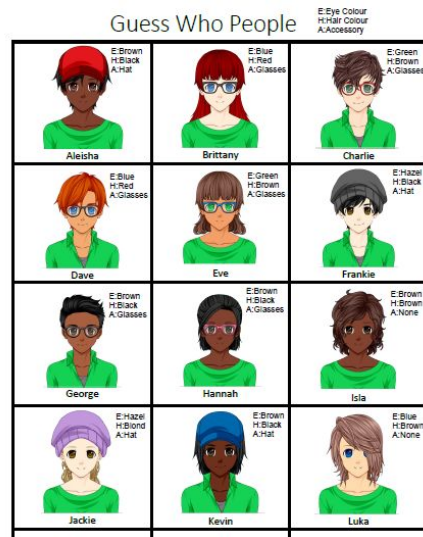


Guess Who!

Welcome to the Labs



Thank you to our Sponsors!

Platinum Sponsor:

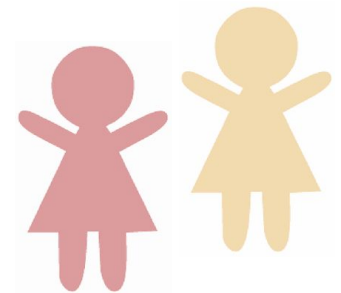
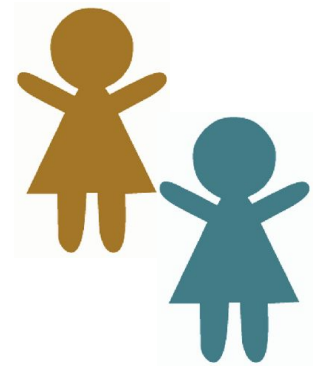


Who are the tutors?

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!

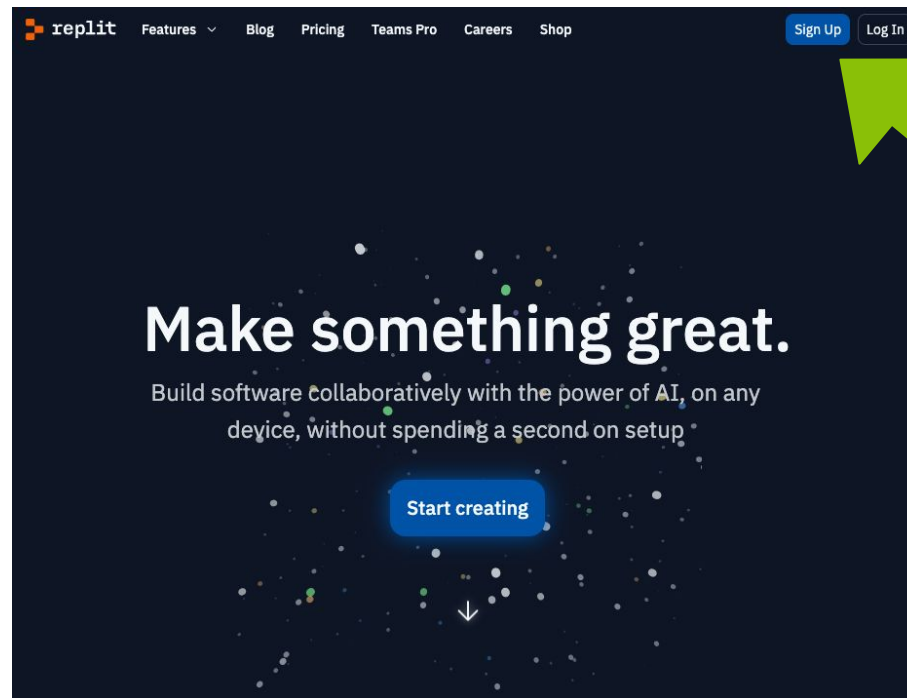
Something to try if you have spare time before the next lecture!

Intro to Python

Let's get coding!

Where do we program?

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



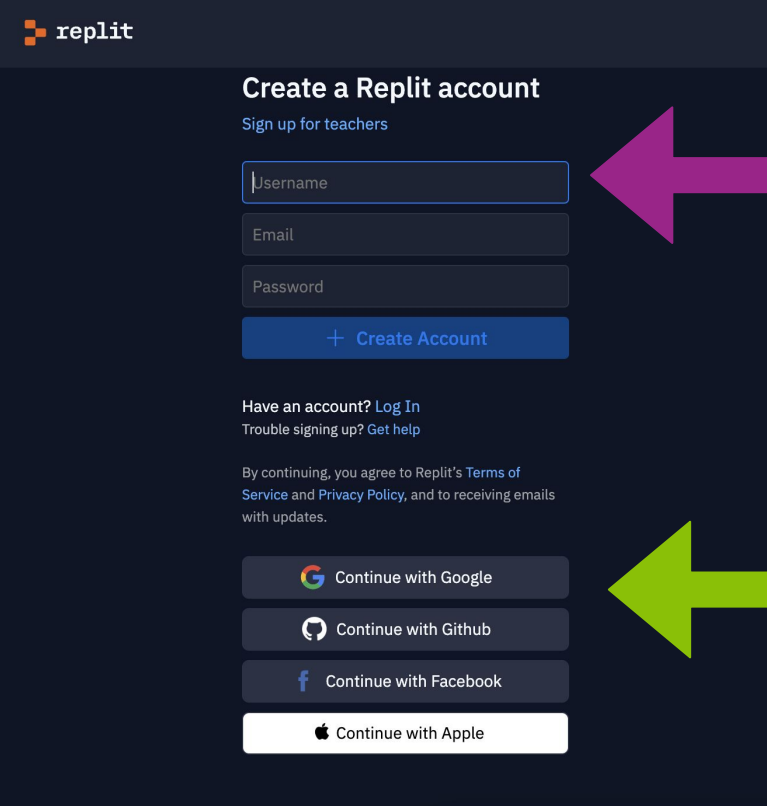
Go to replit.com in Google Chrome

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.

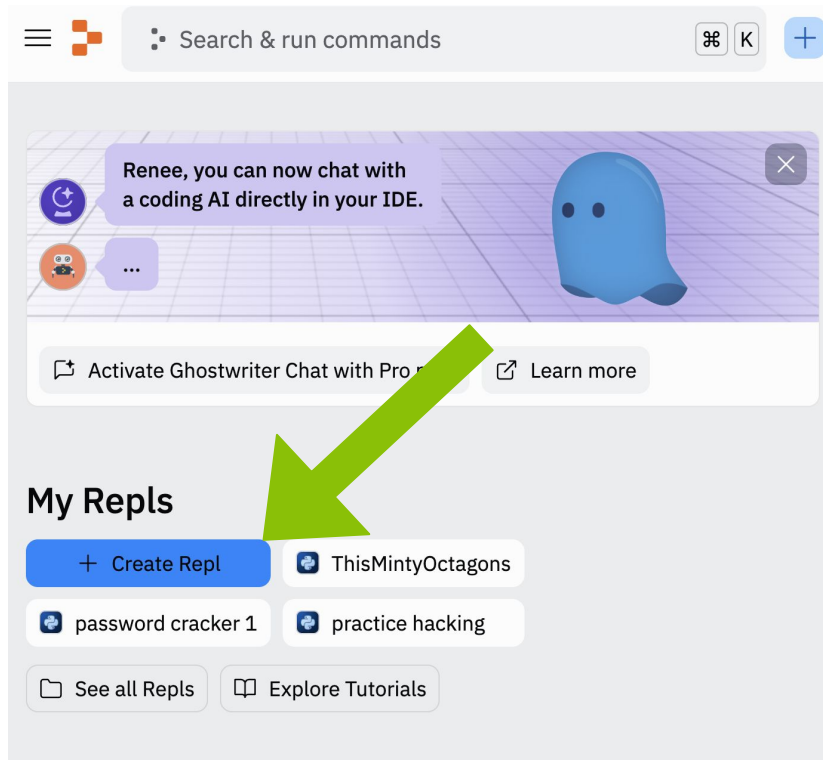
Or use an **email address** you are able to log into.



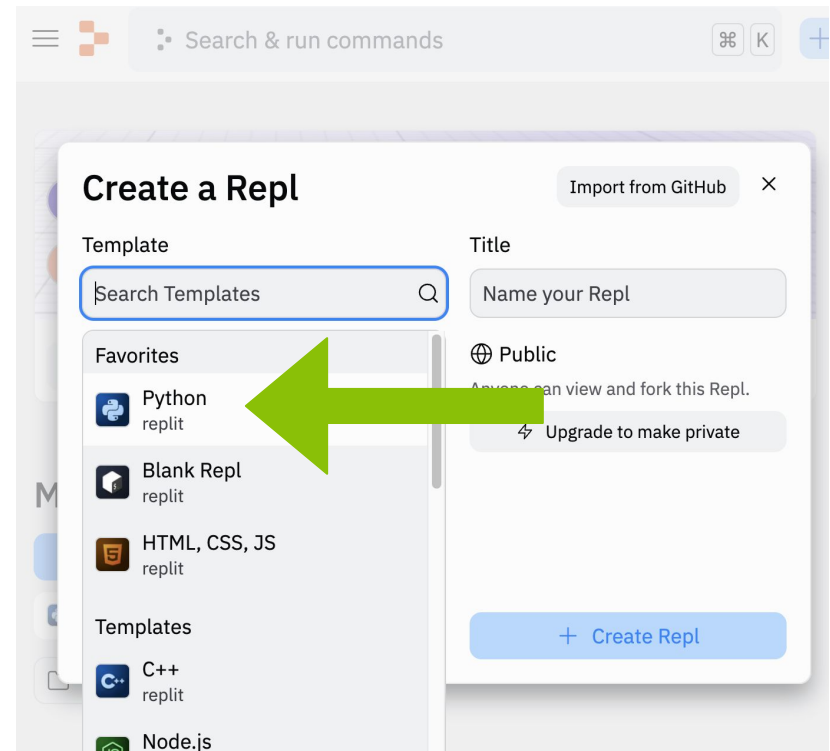
The screenshot shows the Replit website's account creation interface. At the top left is the 'replit' logo. The main heading is 'Create a Replit account', followed by a link 'Sign up for teachers'. Below these are three input fields: 'Username', 'Email', and 'Password'. A blue button with a plus icon and the text '+ Create Account' is positioned below the password field. A purple arrow points from the right edge of the slide to this button. Below the button, there is a link 'Have an account? Log In' and a smaller link 'Trouble signing up? Get help'. A paragraph of text states: 'By continuing, you agree to Replit's Terms of Service and Privacy Policy, and to receiving emails with updates.' Below this text are four social login buttons: 'Continue with Google' (with the Google logo), 'Continue with Github' (with the Github logo), 'Continue with Facebook' (with the Facebook logo), and 'Continue with Apple' (with the Apple logo). A green arrow points from the right edge of the slide to the 'Continue with Google' button.

Creating our Repl It Project

Let's create a new project



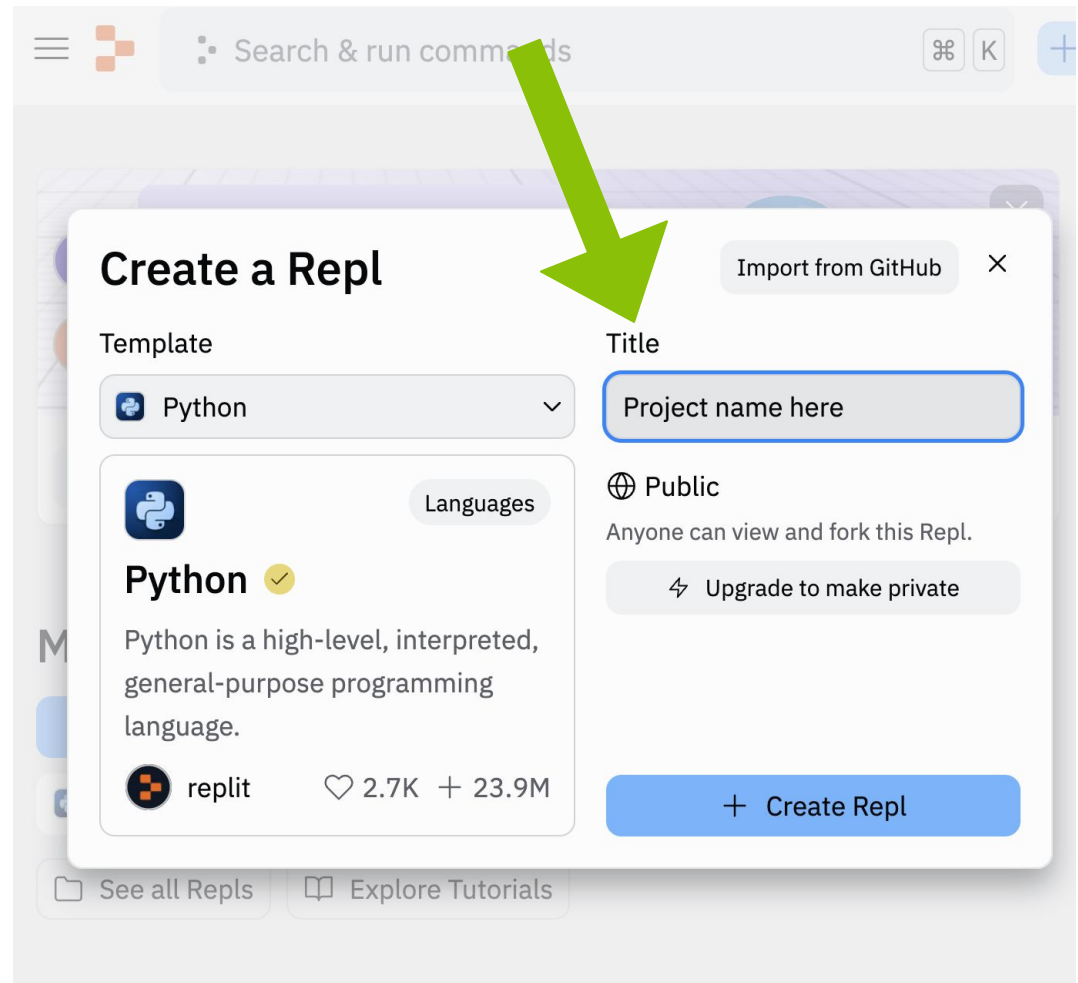
Select Python for the project template



Creating our Repl It Project

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

Name it after
today's project!

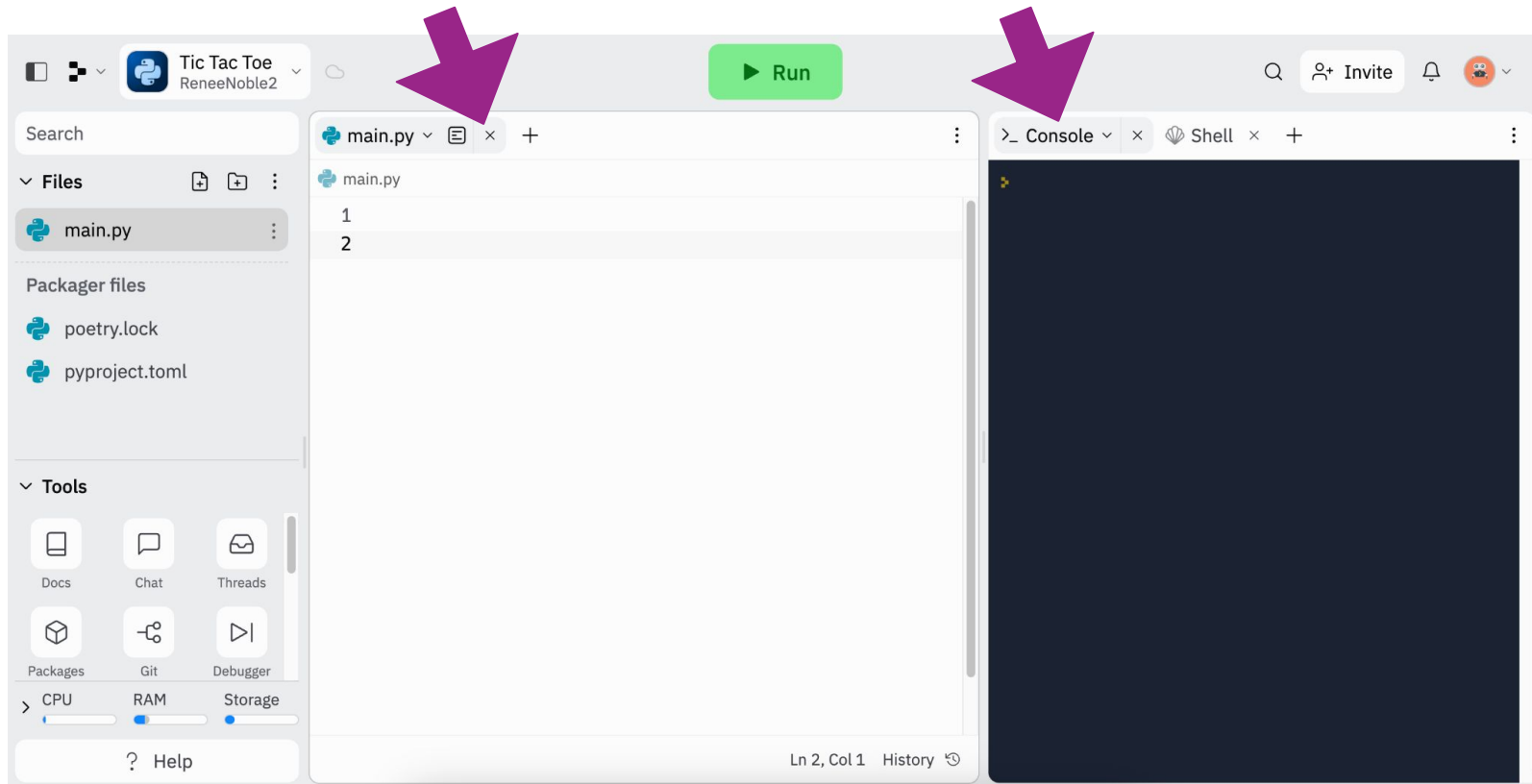


The screenshot shows the 'Create a Repl' dialog box in the Replit interface. A green arrow points to the 'Title' input field, which contains the placeholder text 'Project name here'. The dialog box has a title bar 'Create a Repl' and a close button 'X'. Below the title bar, there is a 'Template' section with a dropdown menu showing 'Python'. To the right of the dropdown is a 'Languages' button. Below the dropdown, there is a Python logo, the text 'Python' with a checkmark, and a description: 'Python is a high-level, interpreted, general-purpose programming language.' Below this, there is a Replit logo, the text 'replit', and a heart icon followed by '2.7K' and '+ 23.9M'. To the right of the 'Title' field, there is a 'Public' section with a globe icon, the text 'Public', and a description: 'Anyone can view and fork this Repl.' Below this, there is an 'Upgrade to make private' button. At the bottom right, there is a large blue button labeled '+ Create Repl'. At the bottom left, there are two buttons: 'See all Repls' and 'Explore Tutorials'.

We're ready to code!

**We'll write our project
here in main.py**

**You can test out Python
code in the console**



Tell me more!

We can `print` things in lots of different ways in python!

```
>>> print("Hello world!")
```

```
>>> print("Hello", "world!")
```

```
>>> print("Hello", "world", end="!")
```

Tell me more!

We can `print` things in lots of different ways in python!

```
>>> print("Hello world!")
```

```
Hello world!
```

```
>>> print("Hello", "world!")
```

```
>>> print("Hello", "world", end="!")
```

Tell me more!

We can `print` things in lots of different ways in python!

```
>>> print("Hello world!")
```

```
Hello world!
```

```
>>> print("Hello", "world!")
```

```
Hello world!
```

```
>>> print("Hello", "world", end="!")
```

Tell me more!

We can `print` things in lots of different ways in python!

```
>>> print("Hello world!")
```

```
Hello world!
```

```
>>> print("Hello", "world!")
```

```
Hello world!
```

```
>>> print("Hello", "world", end="!")
```

```
Hello world!
```

Note that this last one will not have a new line after it!

Tell me more!

We can `print` on many lines at once!

```
>>> print("""Hello world.
```

```
This is me!
```

```
Life should be fun for everyone""")
```

Tell me more!

We can `print` on many lines at once!

```
>>> print("""Hello world.
```

```
This is me!
```

```
Life should be fun for everyone""")
```

```
Hello world.
```

```
This is me!
```

```
Life should be fun for everyone
```

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!")
```

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!

Project time!

Guess Who!

**Now that we've had a Python refresher,
try and do Part0 and 1 !**

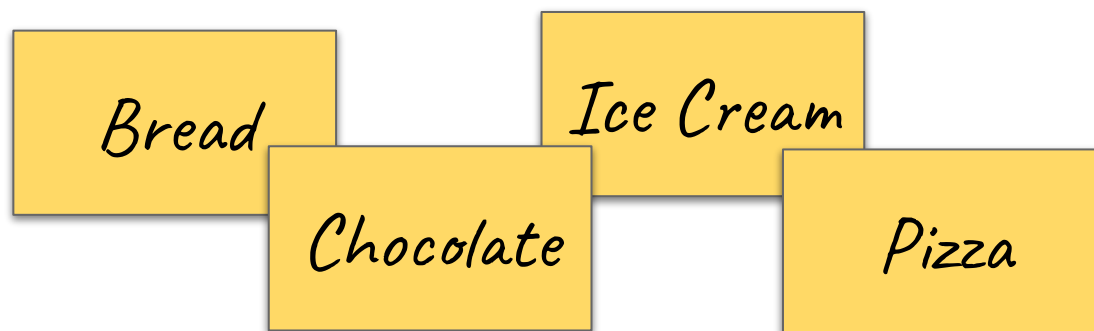
The tutors will be around to help!

Lists

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"]
```

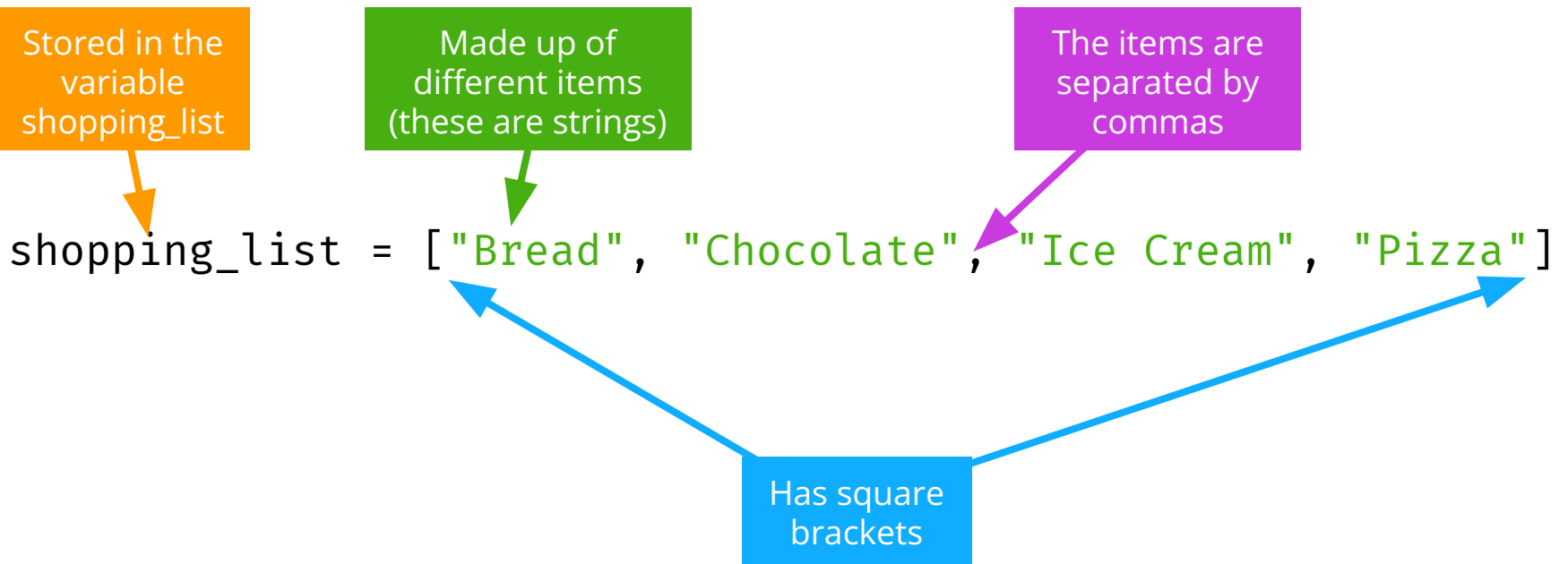
You can put (almost) anything into a list

- 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



Accessing Lists!

The favourites `list` holds four strings in order.

We can count out the items using index numbers!

0



1



2



3



Remember: Indices start from zero!

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'
```

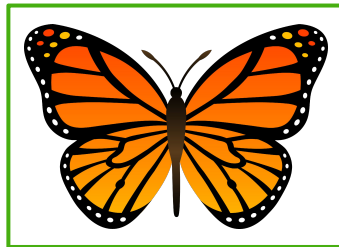
What code do you need to access the second item in the list?

```
>>> faves[1]  
'butterfly'
```

0



[1]



2



3



Going Negative

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

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

What would `faves[-2]` return?



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



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
```

Updating items!

We can also update things in a list:

```
>>> faves = ['books', 'butterfly',  
             'chocolate', 'skateboard']  
  
>>> faves[2]  
'chocolate'  
>>> faves[2] = 'lollipops'  
>>> faves
```



Updating items!

We can also update things in a list:

```
>>> faves = ['books', 'butterfly',  
             'chocolate', 'skateboard']  
  
>>> faves[2]  
'chocolate'  
>>> faves[2] = 'lollipops'  
>>> faves  
['books', 'butterfly', 'lollipops', 'skateboard']
```



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?

```
>>> faves
```

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

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

What does this list look like now?

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?

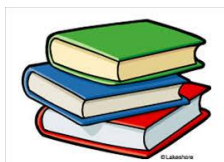
```
>>> faves
```

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

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

What does this list look like now?

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



List of lists!

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

We could use a list of lists to store different sports teams!

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

Get the first pair in the list

```
>>> first_pair = tennis_pairs[0]  
>>> ["Alex", "Emily"]
```

Now we have the first pair handy, we can get the first the first player of the first pair

```
>>> fist_player = first_pair[0]  
>>> "Alex"
```

Project time!

You now know all about 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



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>	True	<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

`5 < 10` **True**

`3 + 2 == 5` **True**

`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>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

5 < 10	True	"Dog" == "dog"	False
3 + 2 == 5	True	"D" in "Dog"	True
5 != 5	False	"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

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

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

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??

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 go 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 the next part**

The tutors will be around to help!

For Loops

Looping through lists!

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

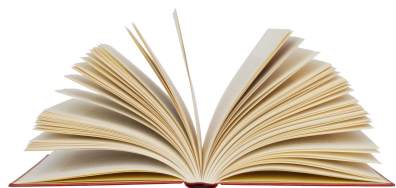
```
words = ['This', 'is', 'a', 'sentence']  
  
print(words[0])  
print(words[1])  
print(words[2])  
print(words[3])
```

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

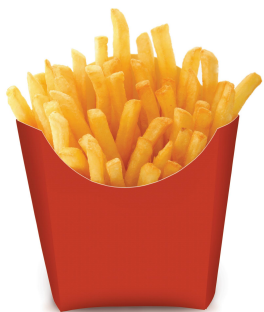
For Loops

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
```

How does it work??

Everything in the list gets to have a turn at being the dog variable





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

Let's set dog to to the **first** thing in the list!
dog is now 'apple'!

How does it work??

Everything in the list gets to have a turn at being the dog variable



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

Let's set dog to to the **first** thing in the list!

dog is now 'apple'!

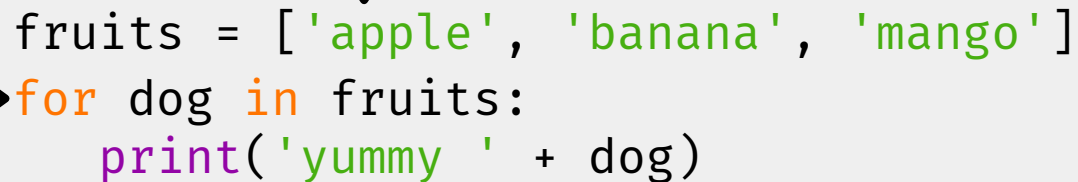
```
print('yummy ' + dog)
```

>>> Yummy apple



How does it work??

Everything in the list gets to have a turn at being the dog variable



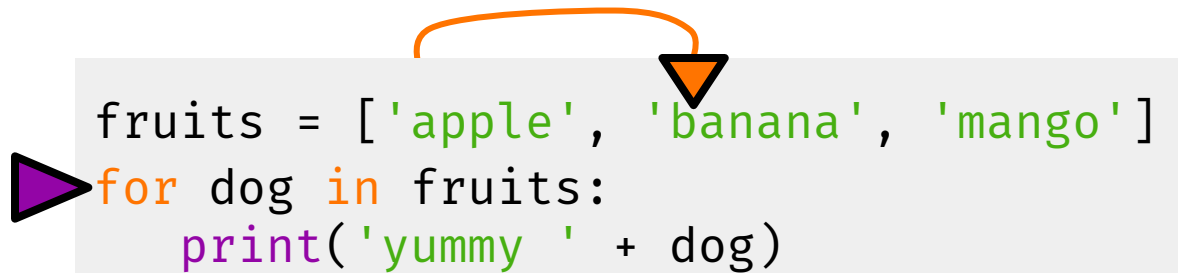
```
fruits = ['apple', 'banana', 'mango']  
for dog in fruits:  
    print('yummy ' + dog)
```

```
>>> Yummy apple
```

Let's set dog to to the **first** thing in the list!
dog is now 'apple'!
`print('yummy ' + dog)`
We're at the end of the loop body, back to the top!

How does it work??

Everything in the list gets to have a turn at being the dog variable



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

```
>>> Yummy apple
```

Let's set dog to to the **first** thing in the list!
dog is now 'apple'!
print('yummy ' + dog)
We're at the end of the loop body, back to the top!

Let's set dog to to the **next** thing in the list!
dog is now 'banana'!

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

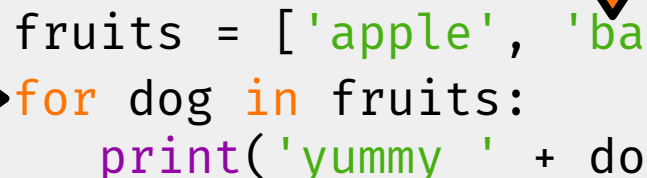
```
>>> Yummy apple  
>>> Yummy banana
```

Let's set dog to to the **first** thing in the list!
dog is now 'apple'!
print('yummy ' + dog)
We're at the end of the loop body, back to the top!

Let's set dog to to the **next** thing in the list!
dog is now 'banana'!
print('yummy ' + dog)

How does it work??

Everything in the list gets to have a turn at being the dog variable



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

```
>>> Yummy apple
```

```
>>> Yummy banana
```

Let's set dog to to the **first** thing in the list!

dog is now 'apple'!

```
print('yummy ' + dog)
```

We're at the end of the loop body, back to the top!

Let's set dog to to the **next** thing in the list!

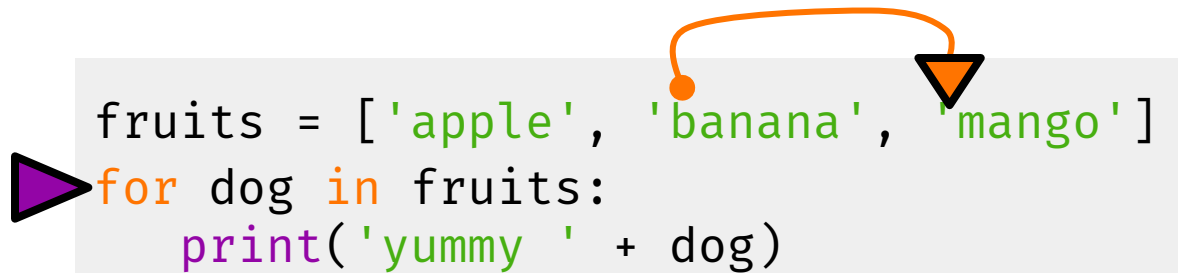
dog is now 'banana'!

```
print('yummy ' + dog)
```

Out of body, back to the top!

How does it work??

Everything in the list gets to have a turn at being the dog variable



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

```
>>> Yummy apple
```

```
>>> Yummy banana
```

Let's set dog to to the **first** thing in the list!

dog is now 'apple'!

```
print('yummy ' + dog)
```

We're at the end of the loop body, back to the top!

Let's set dog to to the **next** thing in the list!

dog is now 'banana'!

```
print('yummy ' + dog)
```

Out of body, back to the top!

Let's set dog to to the **next** thing in the list!

dog is now 'mango'!

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

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

Let's set dog to to the **first** thing in the list!
dog is now 'apple'!
print('yummy ' + dog)
We're at the end of the loop body, back to the top!

Let's set dog to to the **next** thing in the list!
dog is now 'banana'!
print('yummy ' + dog)
Out of body, back to the top!

Let's set dog to to the **next** thing in the list!
dog is now 'mango'!
print('yummy ' + dog)

How does it work??

Everything in the list gets to have a turn at being the dog variable

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

>>> Yummy apple

>>> Yummy banana

>>> Yummy mango



Let's set dog to to the **first** thing in the list!

dog is now 'apple'!

print('yummy ' + dog)

We're at the end of the loop body, back to the top!

Let's set dog to to the **next** thing in the list!

dog is now 'banana'!

print('yummy ' + dog)

Out of body, back to the top!

Let's set dog to to the **next** thing in the list!

dog is now 'mango'!

print('yummy ' + dog)

*Out of body, and out of list!!
We're done here!*

Project Time!

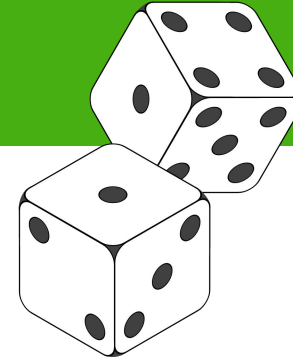
Now you know how to use a for loop!

**Try to do the next Parts
...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!



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

We want the computer to be random sometimes!



Using the random module

Let's choose something randomly from a list!

This is like drawing something out of a hat in a raffle!

Try this!

1. Import the random module!

```
>>> import random
```

2. Copy the shopping list into IDLE

```
>>> shopping_list = ["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!

Raaaaaaaaandom! Can you handle that?

Try to do the next Part!

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
```

MY VARIABLES

i = 0

Set the
variable

Introducing ... while loops!

One step at a time!

0 is less
than 3!

```
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!

Print !

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

i is 0

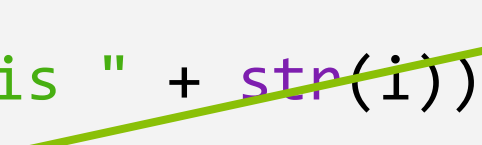
MY VARIABLES

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 = 1

UPDATE
TIME!

```
i is 0
```

Introducing ... while loops!

One step at a time!

Take it
from the
top!

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

```
i is 0
```

MY VARIABLES

```
i = 0
i = 1
```

Introducing ... while loops!

One step at a time!

1 is less
than 3!

```
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!

Print!

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

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


MY VARIABLES

```
i = 0
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
i = 1
i = 2
```

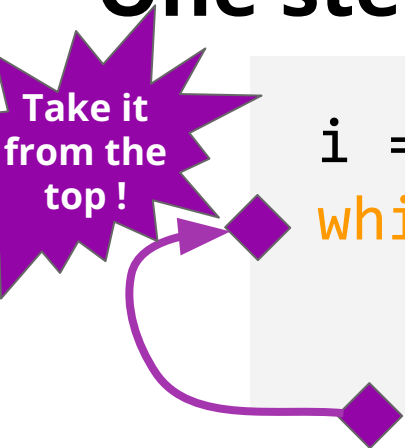
UPDATE
TIME!

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

Introducing ... while loops!

One step at a time!

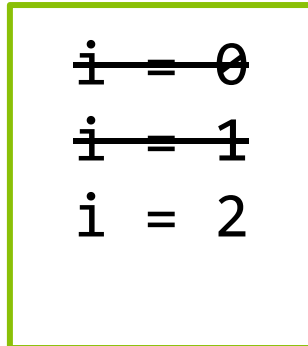
Take it
from the
top!



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

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

MY VARIABLES



```
i = 0
i = 1
i = 2
```

Introducing ... while loops!

One step at a time!

2 is less
than 3!

```
◆ 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!

Print!

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

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


MY VARIABLES

```
i = 0
i = 1
i = 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
```

UPDATE
TIME!

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

Introducing ... while loops!

One step at a time!

Take it
from the
top!

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

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

MY VARIABLES

```
i = 0
i = 1
i = 2
i = 3
```

Introducing ... while loops!

One step at a time!

3 IS NOT
less than
3!

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

MY VARIABLES

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

We are
are done
with this
loop!

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


Introducing ... while loops!

Initialise the loop variable

Loop condition

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

Code to repeat

Update the loop variable

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))
```

[illegible]

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?")
```

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?")
```

```
Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?  
Are we there yet?
```

Project Time!

while we're here:

Try to do the next Part

The tutors will be around to help!

Files

Filing it away!

What happens if we want to use different data in our program? What if that data is too big to write in with the keyboard?

We'd have to change our code!!

It would be better if we could keep all our data in a file and just be able to pick and choose what file we wanted to play today!

people.txt

```
Aleisha,brown,black,hat  
Brittany,blue,red,glasses  
Charlie,green,brown,glasses  
Dave,blue,red,glasses  
Eve,green,brown,glasses  
Frankie,hazel,black,hat  
George,brown,black,glasses  
Hannah,brown,black,glasses  
Isla,brown,brown,none  
Jackie,hazel,blonde,hat  
Kevin,brown,black,hat  
Luka,blue,brown,none
```


Opening files!

To get access to the stuff inside a file in python we need to **open** it!
That doesn't mean clicking on the little icon!

```
with open("test.txt", "r") as f:
```

You'll now be able to read the things in `f`

If your file is in the same location as your code you can just use the name!

A missing file causes an error

Here we try to open a file that doesn't exist:

```
with open("missing.txt", "r") as f:
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
IOError: [Errno 2] No such file or  
directory: 'missing.txt'
```

You can read in one line at a time

You can use a for loop to read 1 line at a time!

```
with open("haiku.txt", "r") as f:  
    for line in f:  
        print(line)
```

Wanna go outside.

Oh NO! Help! I got outside!

Let me back inside!

Why is there an extra blank line each time?

Chomping off the newline

The newline character is represented by '\n':

```
print('Hello\nWorld')  
Hello  
World
```

We can remove it from the lines we read with .strip()

```
x = 'abc\n'  
x.strip()  
'abc'
```

x.strip() is safe as lines without newlines will be unaffected

Reading and stripping!

```
with open("haiku.txt", "r") as f:  
    for line in f:  
        line = line.strip()  
        print(line)
```

Wanna go outside.
Oh NO! Help! I got outside!
Let me back inside!

No extra lines!

Project time!

I hope you **filed** that knowledge away

Use it in the next section of the project!

Try to do the next Part

The tutors will be around to help!

Tell us what you think!

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