

Welcome to the labs!

Cryptography

Who are the tutors?

Else statements

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

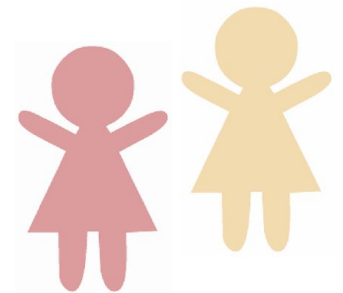
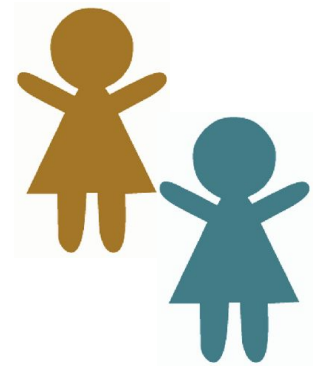
```
word = "Chocolate"  
if word == "GPN":  
    print("GPN is awesome!")  
else:  
    print("The word isn't GPN :(")
```

What happens?

Who are you?

Two Truths and a Lie

1. Get in a group of 3-5 people
2. Tell them three things about yourself:
 - a. Two of these things should be true
 - b. One of these things should be a lie!
3. The other group members have to guess which is the lie



Log on

Log on and jump on the GPN website

girlsprogramming.network/workshop

You can see:

- These **slides** (to take a look back or go on ahead).
- A digital copy of your **workbook**.
- Help bits of text you can **copy and paste**!

There's also links to places where you can do more programming!

Tell us you're here!

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

Today's project!

Cryptography

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 Caesar Ciphers

Let's get encrypting!

What is a cipher?

A cipher is a way to write a message so that no one else can read it!

Unless they know the secret key!



Examples of ciphers

If you've ever made up your own secret language or made notes to your friends so that other people can't read them, you've made a cipher!

For example:

gnidoc evol i

Can you figure out what this says?

Examples of ciphers

If you've ever made up your own secret language or made notes to your friends so that other people can't read them, you've made a cipher!

For example:

gnidoc evol i

Can you figure out what this says?

It says **I love coding** backwards!

Caesar Cipher

So what's a Caesar Cipher?

It's a cypher that Julius Caesar used in ancient Rome to send secret messages to his armies!

Let's learn how it works!

Make a Cipher Wheel

- Cut out green circle
- Cut out purple circle
- Put small circle on top of big circle matching centres
- Secure together with centre split pin
- Spin inside circle of letters around



Caesar Cipher Wheel template in Workshop Material folder

Shifting letters

A Caesar Cipher works by shifting letters in the alphabet so that they line up with new letters.

For example if we were to shift everything by 3 it would look like this:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Line up the 'a' on both wheels and then turn the inside wheel 3 letters **anti-clockwise** so that you have your letters lining up like this!

Encrypting

Now, let's encrypt **I love coding** using the wheel

For our Caesar Cipher we take each letter and replace it with the 'shifted' letter

So, let's start with the letter 'i'
What new letter should we use to replace it?



>>> Find letter *i* on the **outside** wheel and replace it with its matching letter on the **inside** wheel = the letter 'h'

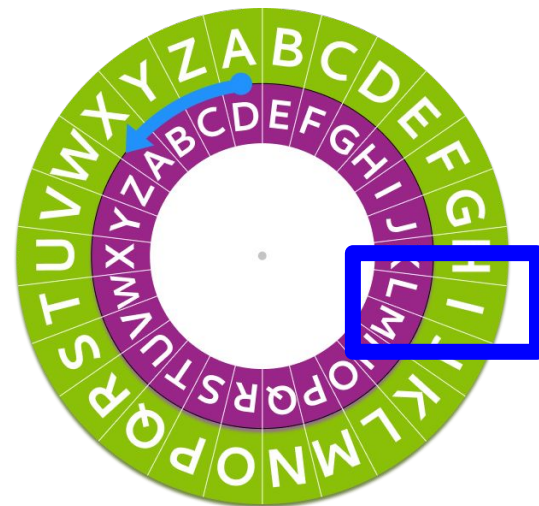
Encrypting

Now, let's encrypt **I love coding** using the wheel

For our Caesar Cipher we take each letter and replace it with the 'shifted' letter

So, let's start with the letter 'i'

What new letter should we use to replace it?



>>> Find letter *i* on the **outside** wheel and replace it with its matching letter on the **inside** wheel = the letter 'j'

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	
o	Is replaced with	
v	Is replaced with	
e	Is replaced with	
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	
o	Is replaced with	
v	Is replaced with	
e	Is replaced with	
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	
e	Is replaced with	
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	g
i	Is replaced with	
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	g
i	Is replaced with	I
n	Is replaced with	
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	g
i	Is replaced with	l
n	Is replaced with	q
g	Is replaced with	

Writing the whole message!

Let's do the rest of the message together

I love coding

I	Is replaced with	o
o	Is replaced with	r
v	Is replaced with	y
e	Is replaced with	h
c	Is replaced with	f
o	Is replaced with	r
d	Is replaced with	g
i	Is replaced with	l
n	Is replaced with	q
g	Is replaced with	j

Secret Message

**So our secret encrypted message is
L oryh frglqj**

That's a lot harder to figure out than it just being
backwards!

Encrypt your own name!

Using a key of minus 1 (so A=Z) (Jessica = ldr rhbz)

Write your name on the blank tag in name badge!

Decrypting

Writing secret messages isn't any fun if you can't figure out what they say!

Luckily you can also use your cipher wheel to *decrypt* a secret message.

How do you think we can do that?

What information do we need to know in order to decrypt a secret message?

It's the key!

To decrypt a secret message **we need to know** the amount that we shifted the wheel when we encrypted it. That number is called **the key**!

Once we know the key we can just turn our wheel and read the wheel from the inside out!

*Find the letter on the **inside** wheel and replace it with it's matching letter on the **outside** wheel*

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with



Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o
d

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o
d
i

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o
d
i
n

Let's check it works!

l	Is replaced with
o	Is replaced with
r	Is replaced with
y	Is replaced with
h	Is replaced with
f	Is replaced with
r	Is replaced with
g	Is replaced with
l	Is replaced with
q	Is replaced with
j	Is replaced with

i
l
o
v
e
c
o
d
i
n
g



Another way to decrypt



- Another way to decrypt a message is to change the key value to become the negative of the encryption key value
- We will use this method in our code
- This is because to decrypt a message we need to shift the alphabet the opposite way.
- A negative key value means you turn your inner purple wheel to the right (clockwise)

Your Turn!

**Try doing Part 0 of Workbook 1
using your Caesar Cipher wheels!**

Your tutors are here to help you if you get
stuck

Intro to Programming

What is programming?



Programming is not a bunch of crazy numbers!

It's giving computers a set of instructions!



A Special Language

A language to talk
to dogs!



Programming is a
language to talk to
computers

People are smart! Computers are dumb!

SALAD INSTRUCTIONS

Programming is like a recipe!

Computers do EXACTLY what you say, every time.

Which is great if you give them a good recipe!

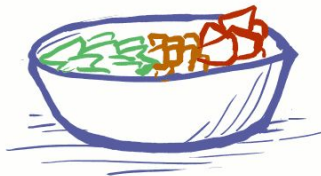
1) GET A LETTUCE HEAD, A CARROT, A TOMATO, A KNIFE, AND A BOWL



2) USE THE KNIFE TO CUT UP THE LETTUCE HEAD, CARROT, AND TOMATO



3) PUT THE LETTUCE, CARROT AND TOMATO IN THE BOWL



4) MIX THE CONTENTS OF THE BOWL



People are smart! Computers are dumb!

But if you get it
out of order....

A computer
wouldn't know
this recipe was
wrong!

SALAD INSTRUCTIONS

1) GET A LETTUCE HEAD,
A CARROT, A TOMATO, A
KNIFE, AND A BOWL



3) PUT THE LETTUCE,
CARROT AND TOMATO
IN THE BOWL



2) USE THE KNIFE TO CUT
UP THE LETTUCE HEAD,
CARROT, AND TOMATO



4) MIX THE CONTENTS
OF THE BOWL



People are smart! Computers are dumb!

Computers are bad at filling in the gaps!

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

SALAD INSTRUCTIONS



Everyone/thing has strengths!



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



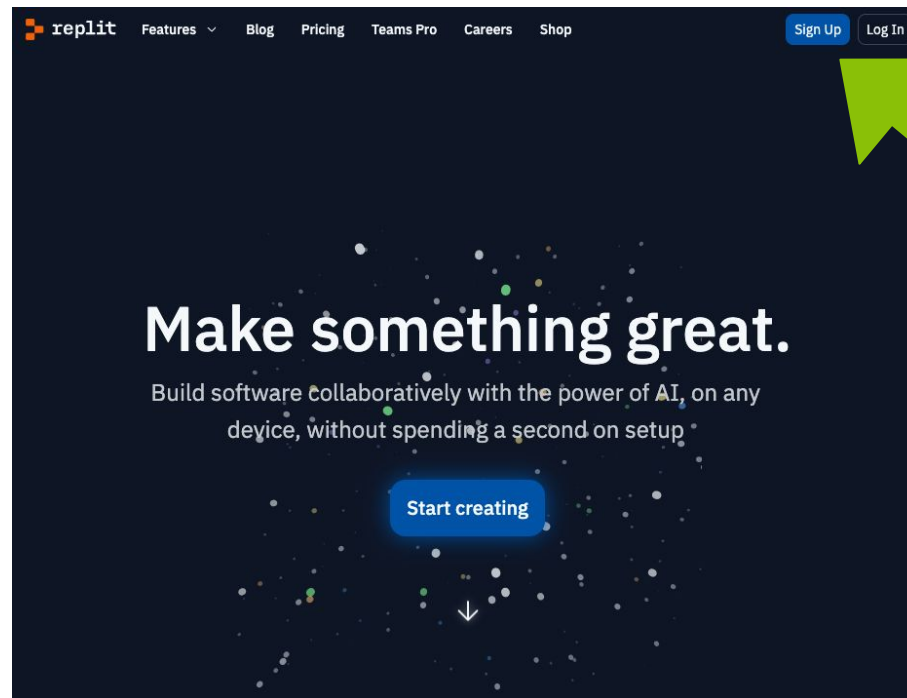
- Does exactly what you tell it
- Does it the same every time
- Doesn't need to sleep!
- Will work for hours on end!
- Get smarter when you tell it how

Intro to Python

Let's get coding!

Where do we program?

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



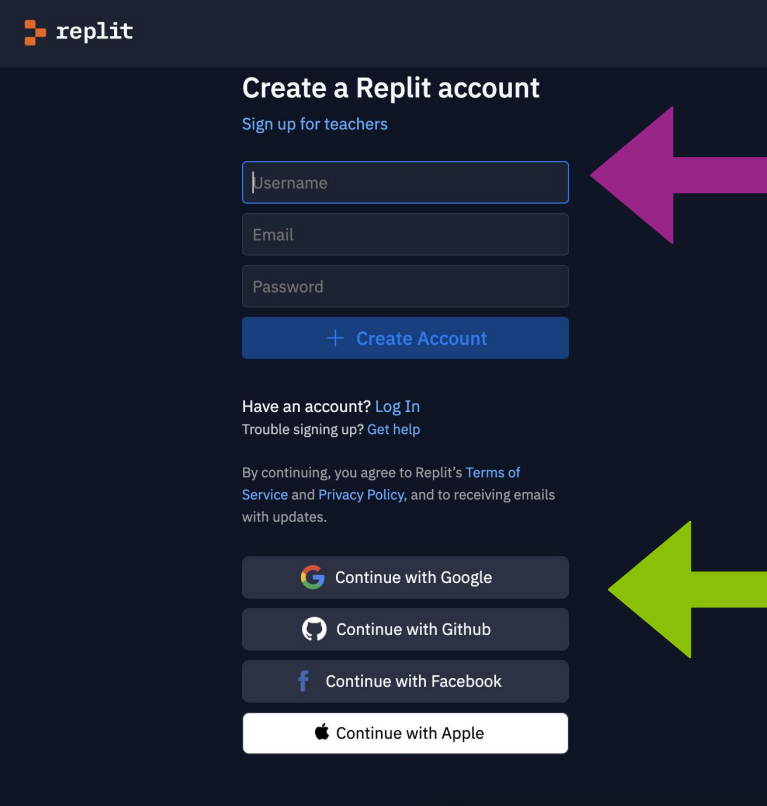
Go to replit.com in your web browser

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.

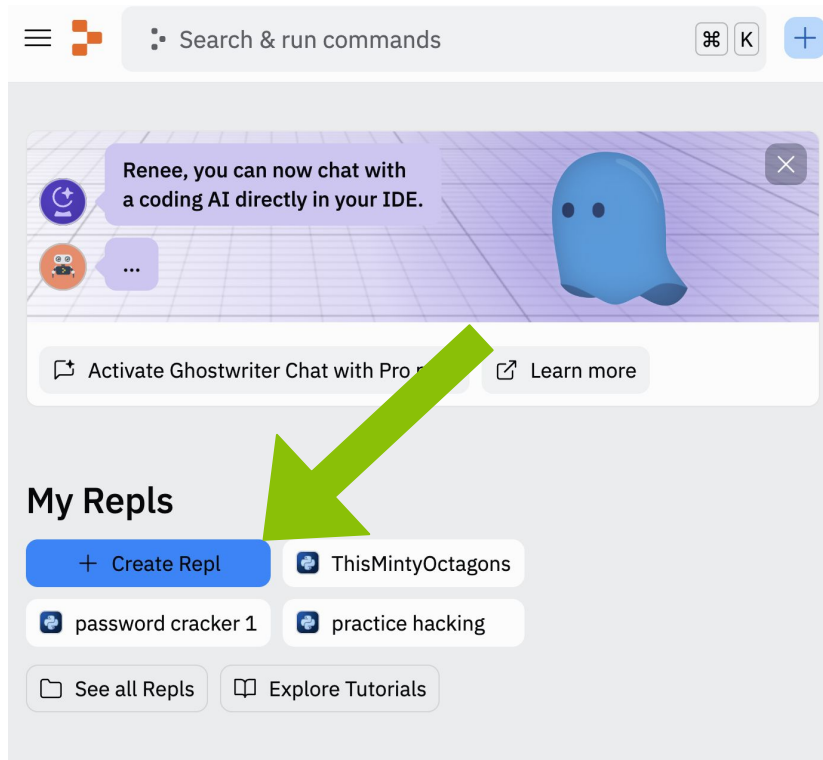
Or use an **email address** you are able to log into.



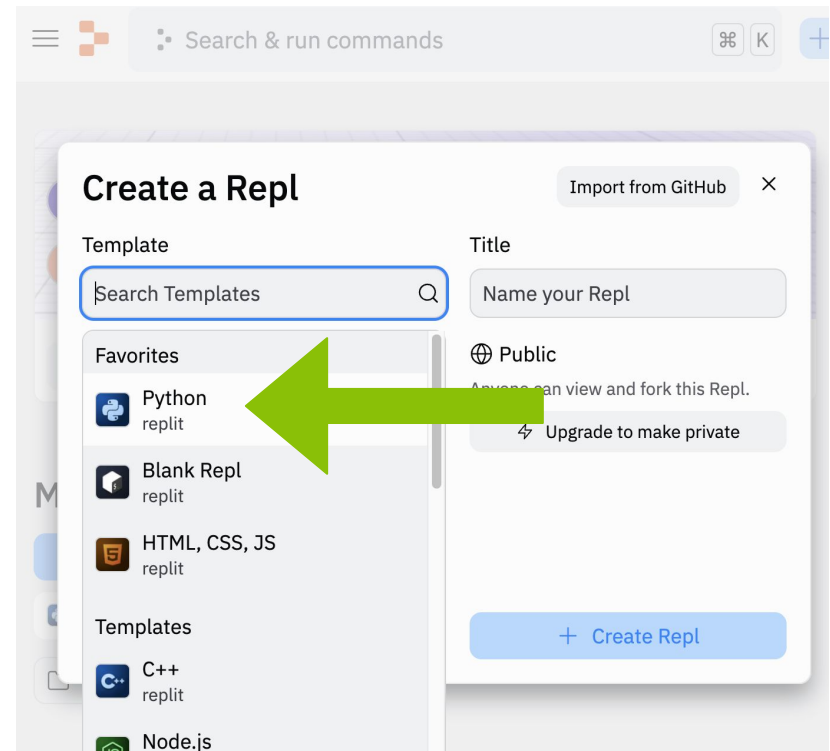
The screenshot shows the Replit website's account creation interface. At the top left is the 'replit' logo. The main heading is 'Create a Replit account', followed by a link 'Sign up for teachers'. Below these are three input fields: 'Username', 'Email', and 'Password'. A blue button with a plus icon 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 terms and conditions follows. At the bottom, there are four buttons for social login: 'Continue with Google' (with a Google logo), 'Continue with Github' (with a Github logo), 'Continue with Facebook' (with a Facebook logo), and 'Continue with Apple' (with an 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



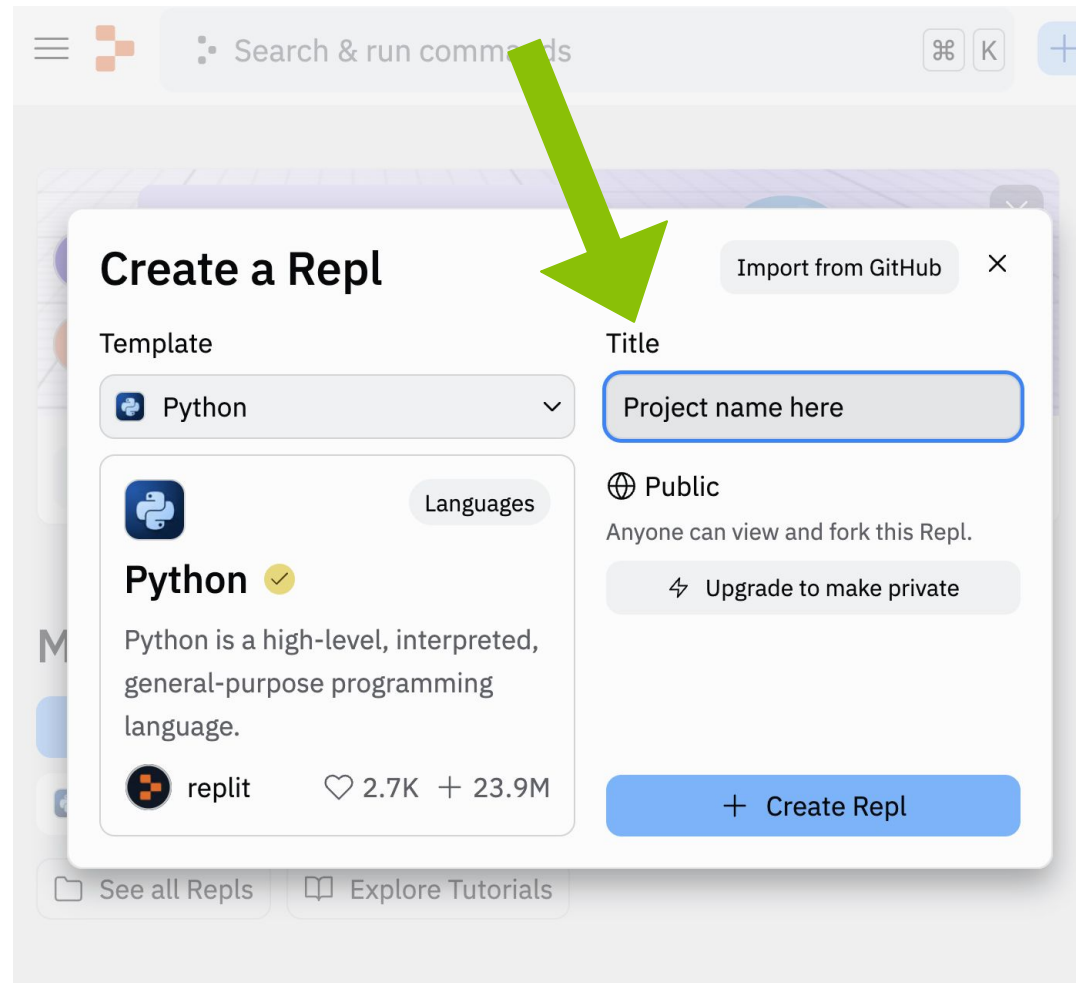
Select Python for the project template



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' modal in the Replit web interface. A green arrow points to the 'Title' input field, which contains the placeholder text 'Project name here'. The modal includes a 'Template' dropdown set to 'Python', a description of Python, and a 'Public' visibility setting. A blue '+ Create Repl' button is at the bottom right.

Search & run commands

⌘ K

Import from GitHub

Create a Repl

Template

Python

Python

Python is a high-level, interpreted, general-purpose programming language.

replit 2.7K + 23.9M

Title

Project name here

Public

Anyone can view and fork this Repl.

Upgrade to make private

+ Create Repl

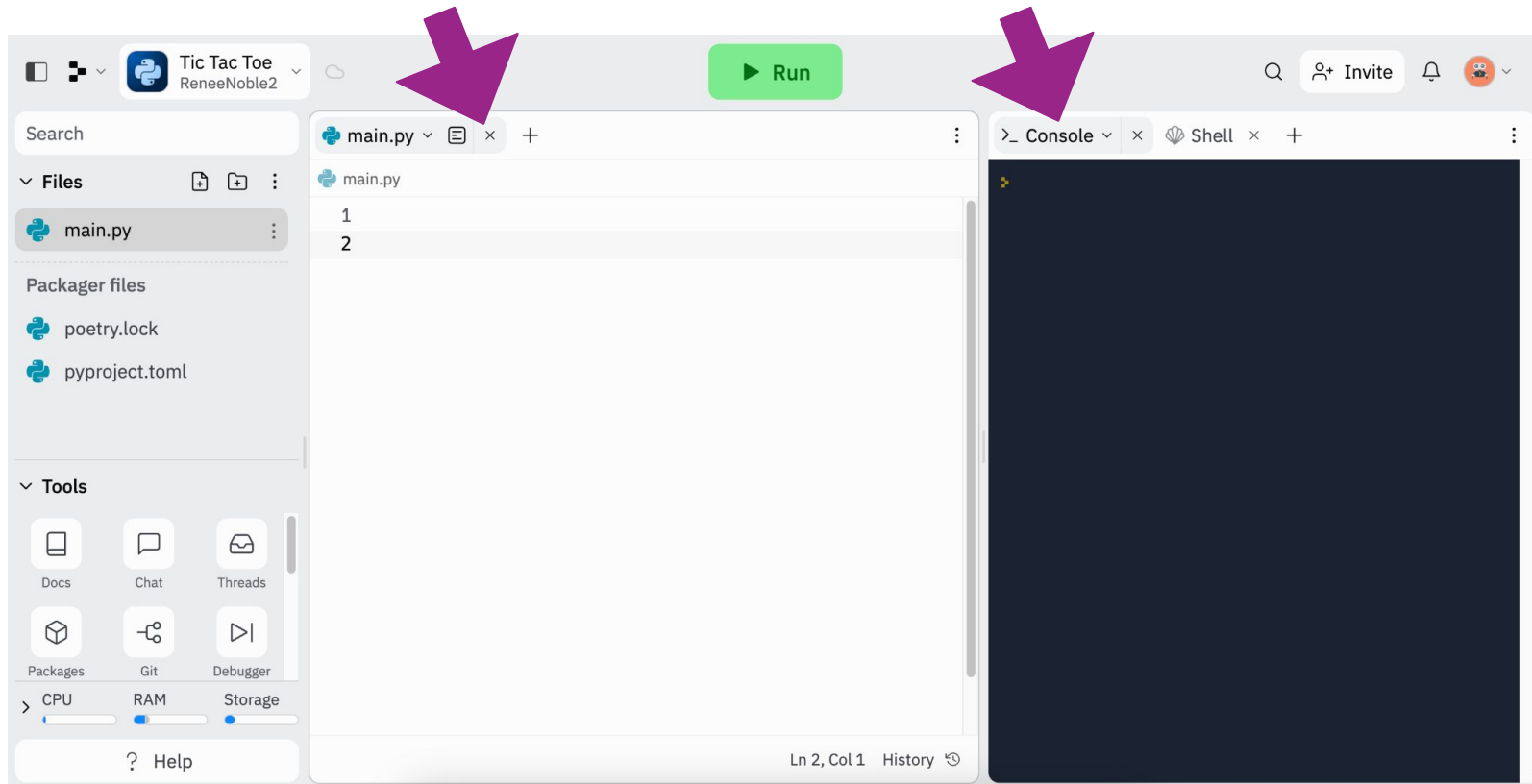
See all Repls

Explore Tutorials

We're ready to code!

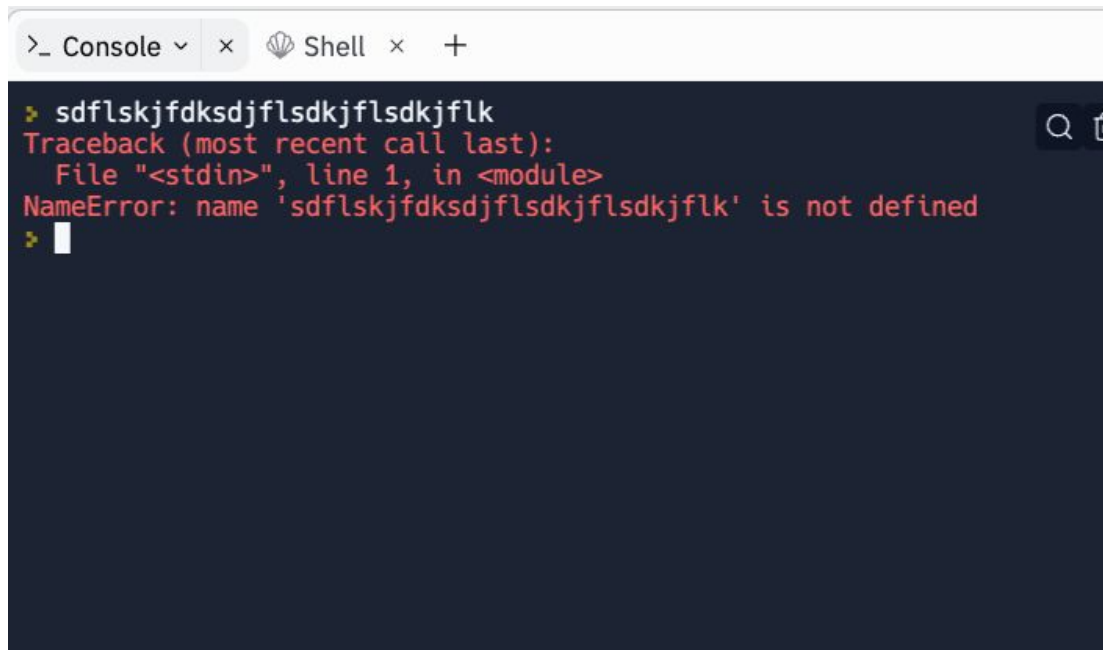
**We'll write our project
here in main.py**

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



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

Type by **button mashing** the keyboard!
Then press enter!

A screenshot of a web-based console window. The window has a title bar with tabs for 'Console' and 'Shell'. The console area is dark blue with light blue text. It shows a prompt '>' followed by the text 'sdflskjfdksdjflsdkjflsdkjflk'. Below this, a red error message is displayed: 'Traceback (most recent call last):', 'File "<stdin>", line 1, in <module>', and 'NameError: name 'sdflskjfdksdjflsdkjflsdkjflk' is not defined'. A cursor is visible on the line following the error message.

```
>_ Console x Shell x +
> sdflskjfdksdjflsdkjflsdkjflk
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sdflskjfdksdjflsdkjflsdkjflk' is not defined
> 
```

Did you get a big red error message?

Mistakes are great!

*SyntaxError:
Invalid Syntax*

Good work you made an error!

*ImportError:
No module
named humour*

- Programmers make A LOT of errors!
- Errors give us hints to find mistakes
- Run your code often to get the hints!!
- Mistakes won't break computers!



*KeyError:
'Hairy Potter'*

*AttributeError:
'NoneType' object
has no attribute
'foo'*

*TypeError: Can't
convert 'int' object
to str implicitly*



We can learn from our mistakes!

Error messages help us fix our mistakes!
We read error messages from bottom to top

Traceback (most recent call last):

File "C:/Users/Madeleine/Desktop/tmp.py", line 9, in <module>
 print("I have " + 5 + " apples")

TypeError: can only concatenate str (not "int") to str

1. What went wrong

2. What code didn't work

3. Where that code is

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!")
```


Write some code!!

Watch a Tutor type this into the window
Then press enter!

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

Did it print:

hello world

???

A calculator for words!

What do you think these bits of code do?

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

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

A calculator for words!

What do you think these bits of code do?

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

```
catdog
```

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

```
.....
```

A calculator for words!

What do you think these bits of code do?

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

```
catdog
```

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

```
tortoisetortoisetortoise
```

Strings!

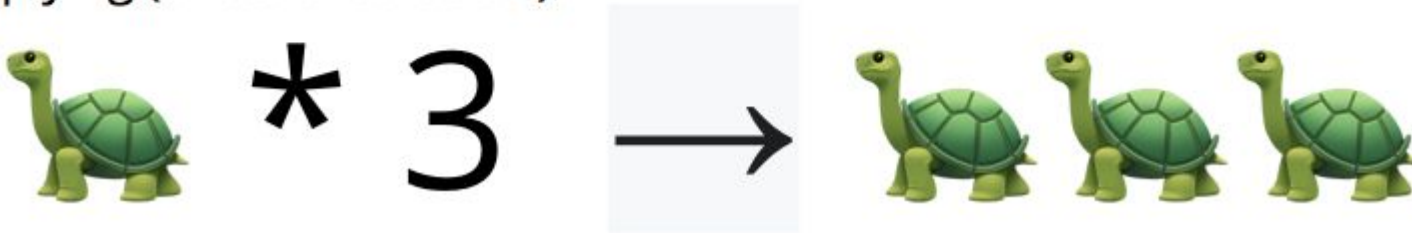
Strings are things with "quotes"

To python they are essentially just a bunch of pictures!

Adding :



Multiplying (3 lots of tortoise!):



Strings!

Strings can have any letters in them, even just spaces!

```
"Hello, world!"
```

```
"bla bla bla"
```

```
":)"
```

```
" "
```

```
'I can use single quotes too!'
```

```
"~\_(\ツ)\_/~"
```

```
"asdfghjklqwertyuiopzxcvbnm"
```

```
"DOGS ARE AWESOME!"
```

```
"!@#$%^&*()_+--=[ ]|\:;'<>,./?"
```

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings

```
>>> 5 + "5"
```

We can turn an integer into a string using `int()`

```
>>> 5 + int("5")
```

Similarly, we turn an `int` into a string using `str()`

```
>>> str(5) + "5"
```

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings

```
>>> 5 + "5"
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

We can turn an integer into a string using `int()`

```
>>> 5 + int("5")
```

Similarly, we turn an `int` into a string using `str()`

```
>>> str(5) + "5"
```


Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings

```
>>> 5 + "5"
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

We can turn an integer into a string using `int()`

```
>>> 5 + int("5")
```

```
10
```

Similarly, we turn an `int` into a string using `str()`

```
>>> str(5) + "5"
```

Strings and Ints!

Integers are numbers in python.

We can do maths with integers but not strings

```
>>> 5 + "5"
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

We can turn an integer into a string using `int()`

```
>>> 5 + int("5")
```

```
10
```

Similarly, we turn an `int` into a string using `str()`

```
>>> str(5) + "5"
```

```
'55'
```

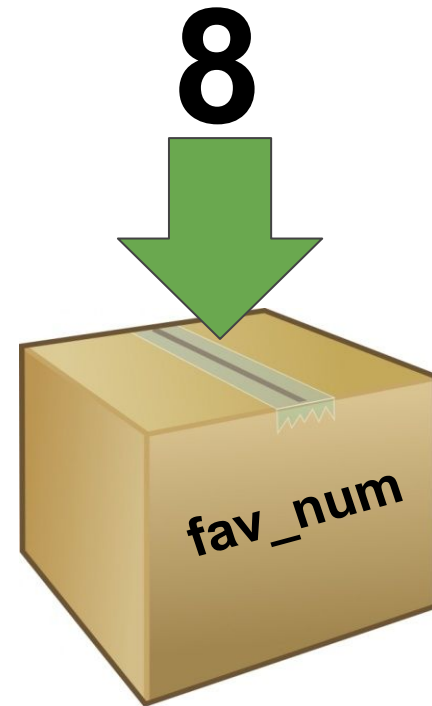
No Storing is Boring!

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

Computers remember things in "**variables**"

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

Let's make our favourite number 8 today!



Variables

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



$$\text{fav_num} - 6 \\ \Rightarrow \mathbf{2}$$

$$\text{fav_num} + 21 \\ \Rightarrow \mathbf{29}$$

$$\text{fav_num} * 2 \\ \Rightarrow \mathbf{16}$$

$$\text{fav_num} / 2 \\ \Rightarrow \mathbf{4}$$



Variables

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



fav_num - 6
=> 2

fav_num + 21
=> 29

fav_num * 2
=> 16

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

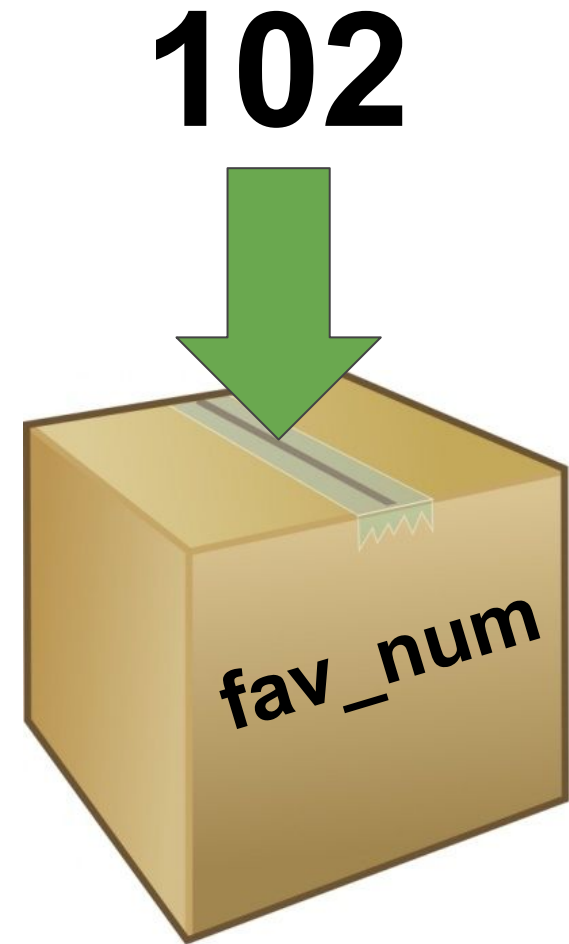


Variables

**Variables are useful
for storing things
that change**

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

Try changing `fav_num` to
102.



Variables

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



`fav_num - 6`
=> 96

`fav_num + 21`
=> 123

`fav_num * 2?`
=> 204

`fav_num / 2?`
=> 51



No variables VS using variables



4
Changes

8 - 6

8 * 2

8 + 21

8 / 2



102 - 6

102 * 2

102 + 21

102 / 2



1
Change

fav_num = 8

fav_num - 6

fav_num * 2

fav_num + 21

fav_num / 2



fav_num = 102

fav_num - 6

fav_num * 2

fav_num + 21

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
```

What can we store?

We can put any value in a variable:

```
apples = 5 + 5
print(apples)
apples = apples - 1
print(apples)
apples = "Delicious"
print(apples)
```

What will this output?

What can we store?

We can put any value in a variable:

```
apples = 5 + 5
print(apples)
apples = apples - 1
print(apples)
apples = "Delicious"
print(apples)
```

10

9

Delicious

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
```

Switcharoo - Making copies!

Set some variables!

```
>>> x = 3
```

```
>>> y = x
```

```
>>> x = 5
```

What do x and y contain now?

Let's find out together!

Switcharoo - Making copies!

Set some variables!

```
>>> x = 3
```

```
>>> y = x
```

```
>>> x = 5
```

What do x and y contain now?

```
>>> x
```

```
5
```

```
>>> y
```

```
3
```

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

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
```

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? ')\n>>> print('Hello ' + my_name)
```

How input works!

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

We use the answer
that was stored in the
variable later!

Adding a comment!

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

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

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

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

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

Try it!

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

Project time!

You now know all about the building blocks
of Python!

Let's put what we learnt into our project
Try to do the next Part!

The tutors will be around to help!

Strings, Ints & Modulo



Strings!

Strings are a sequence of characters in python.

Strings are created by enclosing characters inside
"quotes"

>>> `alphabet = 'abcdefghijklmnopqrstuvwxyz'` creates a string variable that contains the letters of the alphabet

We can add strings together

>>> `"abc" + "def" = "abcdef"`

Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
>>> yum[5]
```

```
>>> yum[-1]
```

```
>>> yum[500]
```

Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
'c'
```

Computers start counting from 0, not 1!

```
>>> yum[5]
```

```
>>> yum[-1]
```

```
>>> yum[500]
```

Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
'c'
```

Computers start counting from 0, not 1!

```
>>> yum[5]
```

```
'l'
```

```
>>> yum[-1]
```

```
>>> yum[500]
```

Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
'c'
```

Computers start counting from 0, not 1!

```
>>> yum[5]
```

```
'l'
```

```
>>> yum[-1]
```

```
'e'
```

```
>>> yum[500]
```

Strings

We can get individual letters from a **string** using indexes.

```
>>> yum = "chocolate"
```

```
>>> yum[0]
```

```
'c'
```

Computers start counting from 0, not 1!

```
>>> yum[5]
```

```
'l'
```

```
>>> yum[-1]
```

```
'e'
```

```
>>> yum[500]
```

```
IndexError: string index out of range
```

Searching Strings

If we want to find where a letter is in a **string**, we look it up using **index()**

```
>>> yum = "chocolate"
```

```
>>> yum.index('h')
```

```
>>> yum.index('o')
```

```
>>> yum.index('z')
```


Searching Strings

If we want to find where a letter is in a **string**, we look it up using **index()**

```
>>> yum = "chocolate"
```

```
>>> yum.index('h')
```

```
1
```

```
>>> yum.index('o')
```

```
>>> yum.index('z')
```

Searching Strings

If we want to find where a letter is in a **string**, we look it up using **index()**

```
>>> yum = "chocolate"
```

```
>>> yum.index('h')
```

```
1
```

```
>>> yum.index('o')
```

```
2
```

Only the index of the first 'o' is returned!

```
>>> yum.index('z')
```

Searching Strings

If we want to find where a letter is in a **string**, we look it up using **index()**

```
>>> yum = "chocolate"
```

```
>>> yum.index('h')
```

```
1
```

```
>>> yum.index('o')
```

```
2
```

Only the index of the first 'o' is returned!

```
>>> yum.index('z')
```

```
ValueError: substring not found
```

Test if character in string

We can test if a character is in a string!

```
>>> yum = "chocolate"  
>>> if 'a' in yum:
```

Maths on Indexes!

We can use any sort of **int** as an index, including the result of an expression or maths equation!

```
>>> yum = "chocolate"
```

```
>>> len(yum)
```

```
>>> yum[9 - 1]
```

Maths on Indexes!

We can use any sort of **int** as an index, including the result of an expression or maths equation!

```
>>> yum = "chocolate"
```

```
>>> len(yum)
```

```
9
```

```
>>> yum[9 - 1]
```

Maths on Indexes!

We can use any sort of **int** as an index, including the result of an expression or maths equation!

```
>>> yum = "chocolate"
```

```
>>> len(yum)
```

```
9
```

```
>>> yum[9 - 1]
```

```
'e'
```

Modulo %

Modulo % is a maths operation

% gives the **remainder** of a division

You'll need to use it in your code!

- $10 \% 8 = 2$ (10 divided by 8 is 1 with remainder 2)
- $20 \% 7 = 6$ (20 divided by 7 is 2 with remainder 6)
- $5 \% 6 = 5$ (5 divided by 6 is 0 with remainder 5)

Project time!

You now know all about strings, ints and modulo!

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

The tutors will be around to help!

For Loops

Looping through a string!

What would we do if we wanted to print out this string one character at a time?

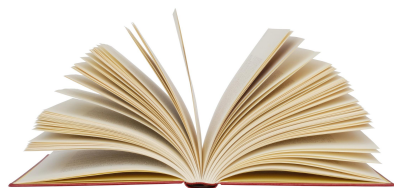
```
word = 'cars'  
  
print(word[0])  
print(word[1])  
print(word[2])  
print(word[3])
```

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

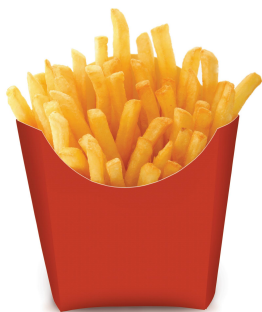
For Loops

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



For each chip in this bag of chips:
Eat

Looping through a string

Strings are a group of characters!


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

What's going to happen?


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

How does it work??

Every character in the string gets to have a turn at being the *i* variable



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

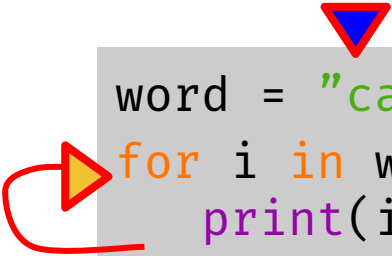


Let's set *i* to to the **first** character in the string!
i is now 'c'
`print(i)`

```
>>> c
```



How does it work??



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

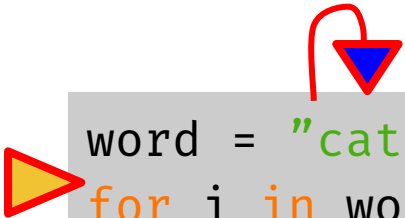
```
>>> c
```

Now we're at the end of the loop body, so go back to the start

Let's set `i` to to the **first** thing in the string!
`i` is now `'c'`
`print(i)`

How does it work??

Every character in the string gets to have a turn at being the `i` variable, so we now set `i` to the next character




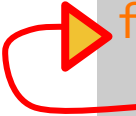
```
word = "cat"  
for i in word:  
    print(i)
```

```
>>> c
```

```
>>> a
```

Let's set `i` to to the **next** charcater in the string!
`i` is now 'a'!
`print(i)`

How does it work??

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

```
>>> c
```

```
>>> a
```


Now we're at the end of the loop body AGAIN, so go back to the start

Let's set *i* to the **next** thing in the string!

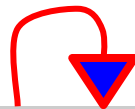
i is now 'a'
`print(i)`

How does it work??

Every character in the string gets to have a turn at being the `i` variable, so we now set `i` to the next character



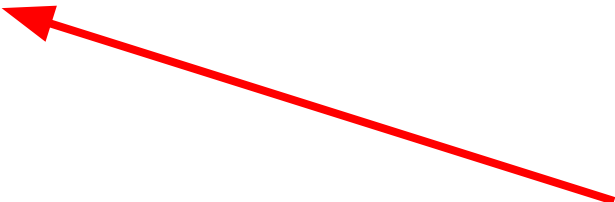
```
word = "cat"
for i in word:
    print(i)
```



```
>>> c
```

```
>>> a
```

```
>>> t
```



Let's set `i` to to the **next** thing in the string!
`i` is now `'t'`!
`print(i)`

How does it work??

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



```
>>> c
```

```
>>> a
```

```
>>> t
```

Now we're at the end of the loop body AGAIN but we have been through all the characters in the string so we exit the for loop

Let's set `i` to to the **next** thing in the string!
`i` is now 't'!
`print(i)`

Project Time!

Now you know how to use a for loop!

Try to do Part 4
...if you are up **for it!**

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` **True**

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

`5 != 5` **False**

`"Dog" == "dog"` **False**

`"D" in "Dog"` **True**

`"Q" not in "Cat"` **True**

Booleans (True and False)

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

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

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


Booleans (True and False)

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

True

"A" in "AEIOU"

False

"Z" in "AEIOU"

False

"a" in "AEIOU"

False

"banana" in animals

True

"cat" in animals

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

Conditions

So to know whether to do something, 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, 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, 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, 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, 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

```
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

What do you think happens?

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

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

If statements

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

What happens?

If statements

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

What happens?

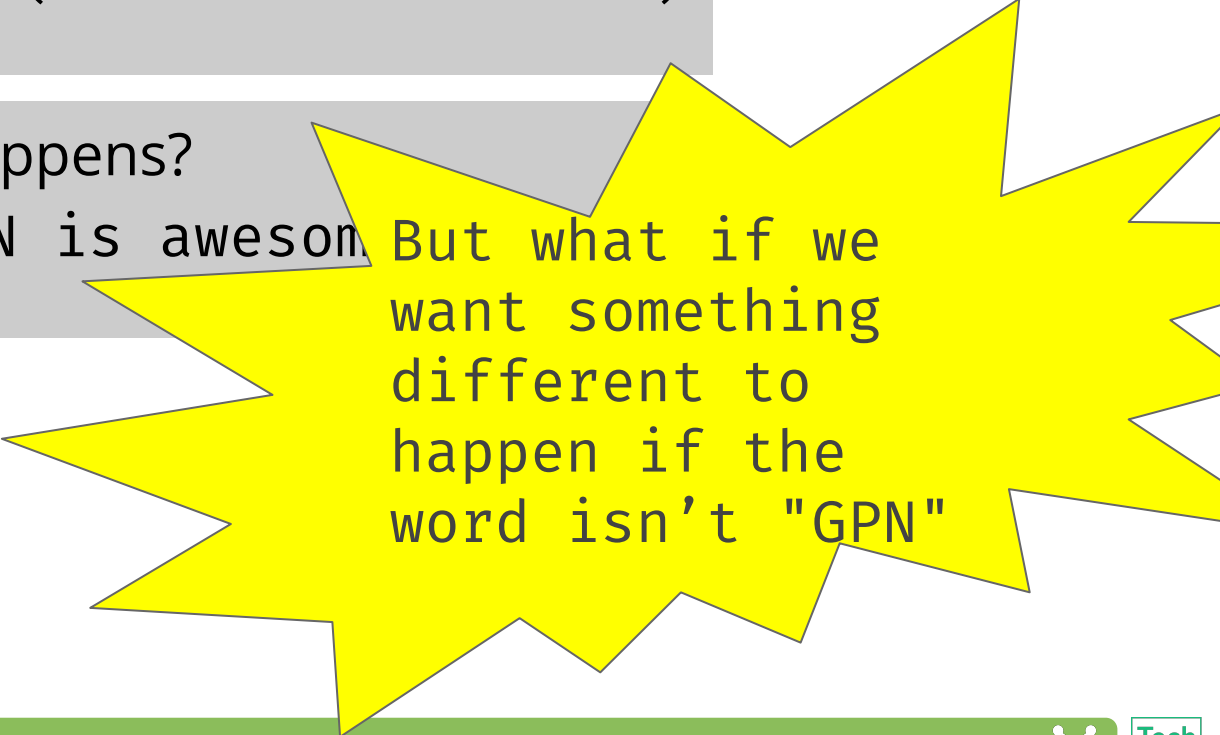
```
>>> GPN is awesome!
```


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"

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?

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?

```
>>> YUMM Chocolate!
```

Project Time!

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

See **if** you can do
Part 5

The tutors will be around to help!