Password Cracker

Welcome to the labs!

Thank you to our Sponsors!

Platinum Sponsor:

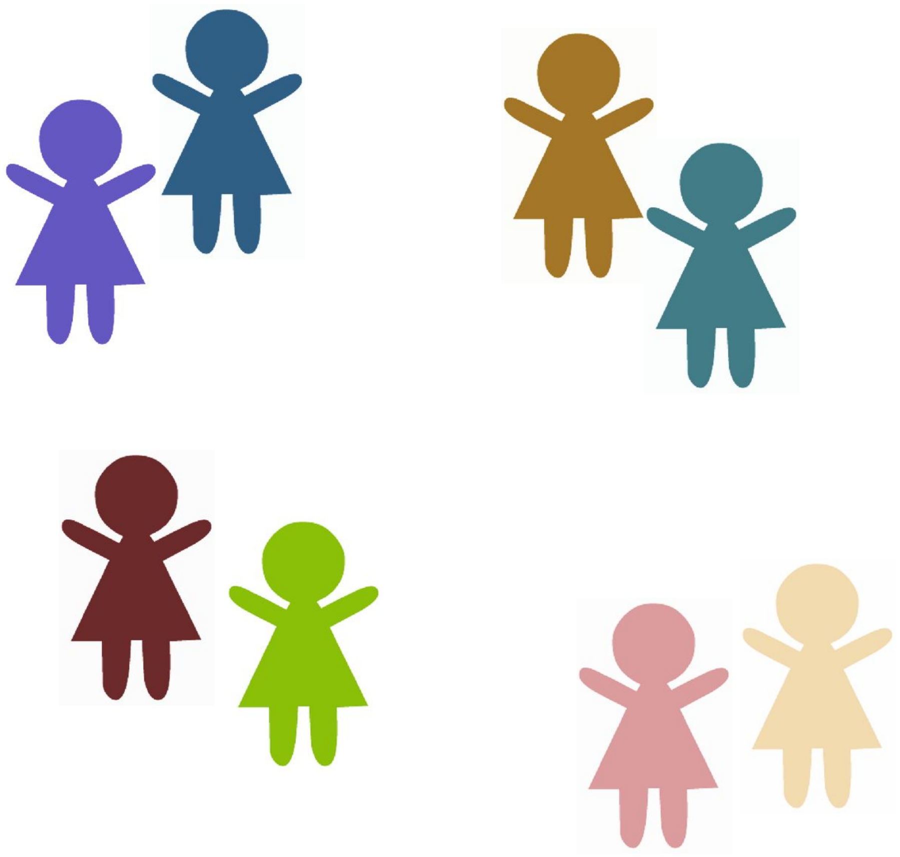
ATLASSIAN

Who are the tutors?

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

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!

Password Cracker!

Today's project!

Password Cracker!

Today we are going to build a program that can make passwords more secure using encoding, then compare an entered password with the actual password to see if it matches!

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!

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**

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 6.1: 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.

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

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.

★ BONUS 4.3: Do some extra!

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

Intro to Programming

What is programming?

Programming is not a bunch of crazy numbers!

It's giving computers a set of instructions!

A Special Language

A language to talk to dogs!

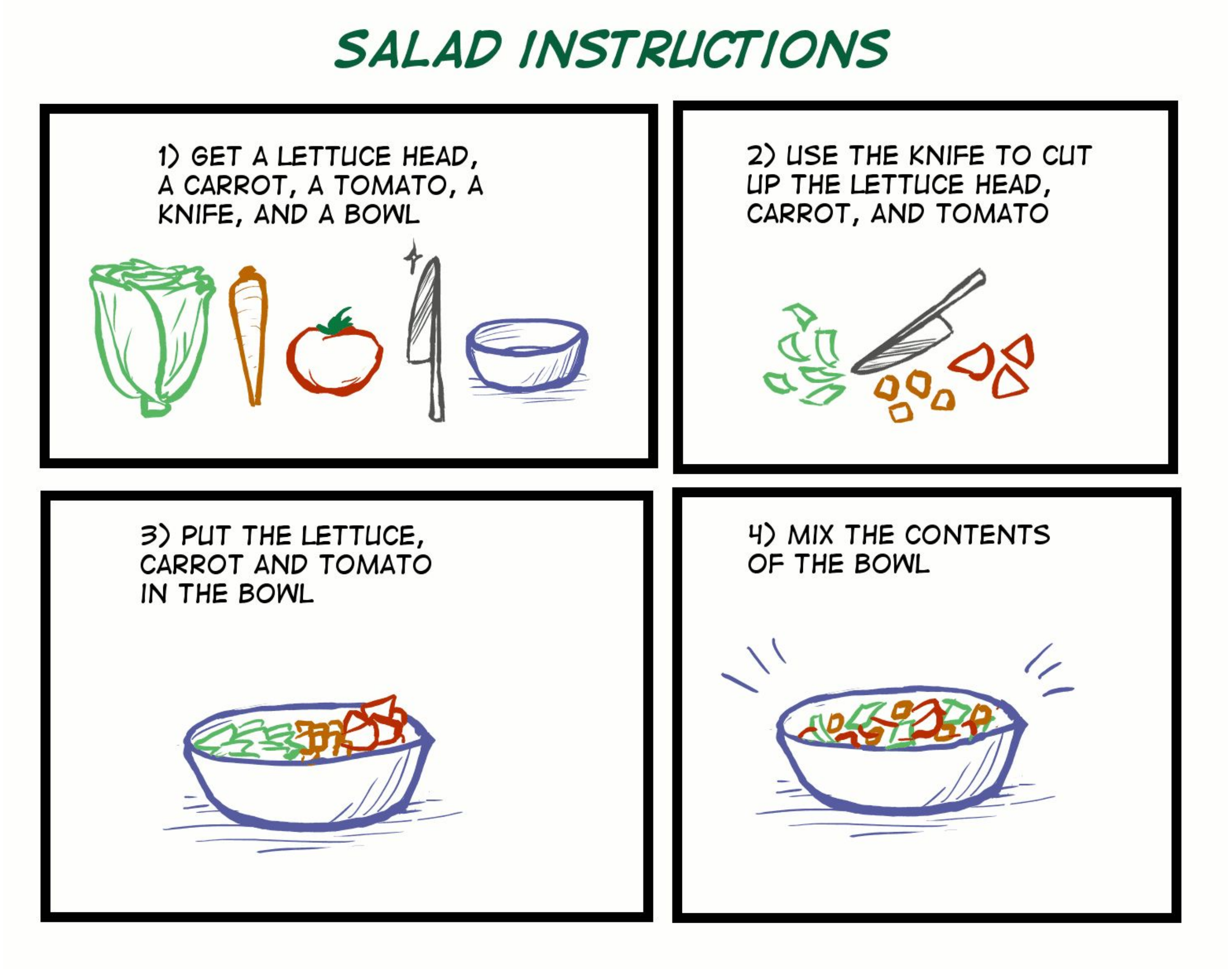
Programming is a language to talk to computers

People are smart! Computers are dumb!

Programming is like a recipe!

Computers do EXACTLY what you say, every time.

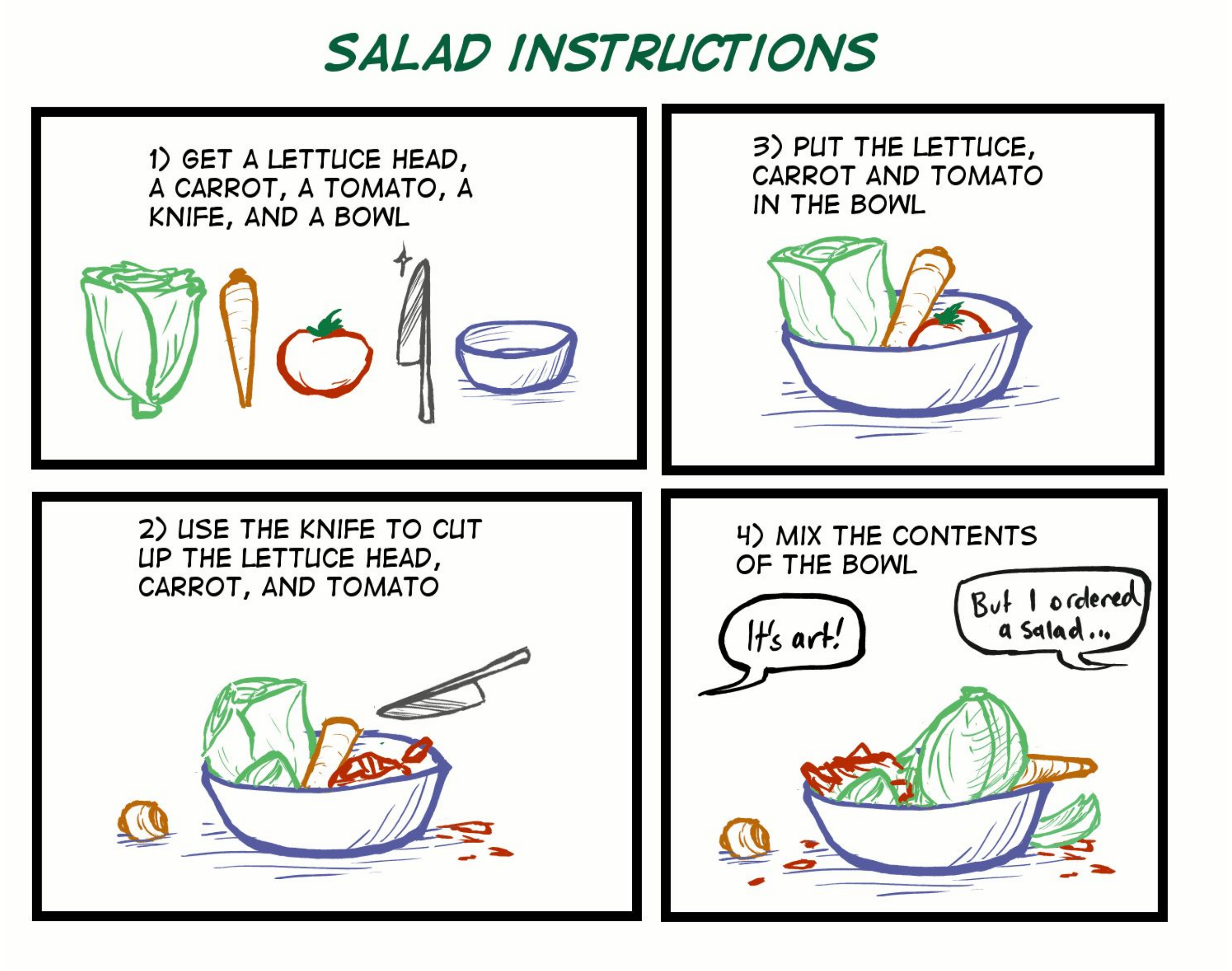
Which is great if you give them a good recipe!


SALAD INSTRUCTIONS
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….

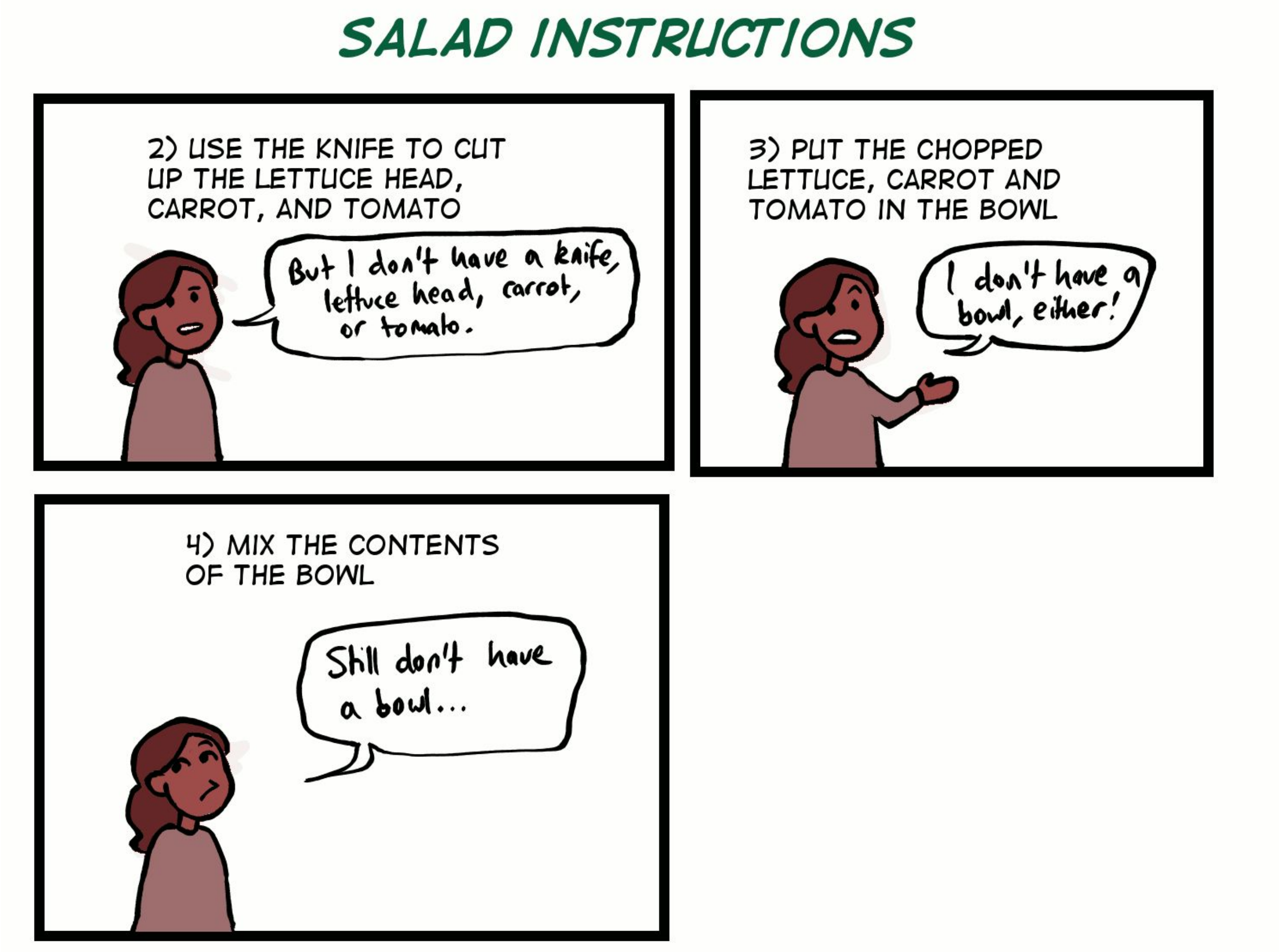
A computer wouldn't know this recipe was wrong!


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
It's art!
But I ordered a salad...

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
2) USE THE KNIFE TO CUT UP THE LETTUCE HEAD, CARROT, AND TOMATO
But I don't have a knife, lettuce head, carrot, or tomato.
3) PUT THE CHOPPED LETTUCE, CARROT AND TOMATO IN THE BOWL
I don't have a bowl, either!
4) MIX THE CONTENTS OF THE BOWL
Still don't have a bowl...

Everyone/thing has strengths!

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

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

Intro to Python

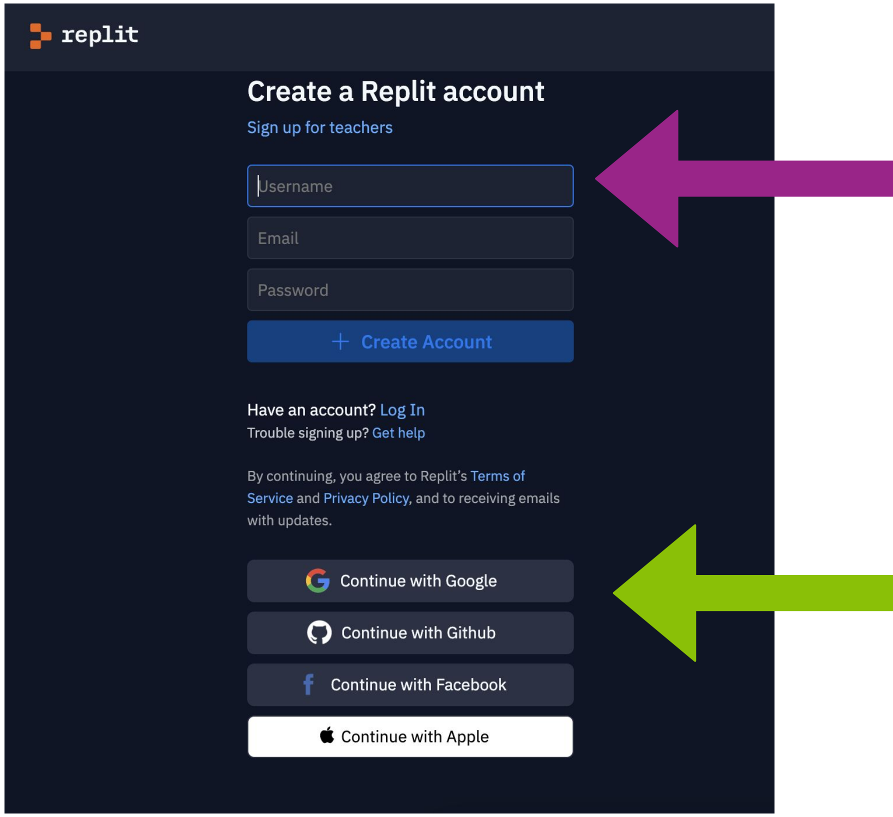
Let’s get coding!

Where do we program? 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.

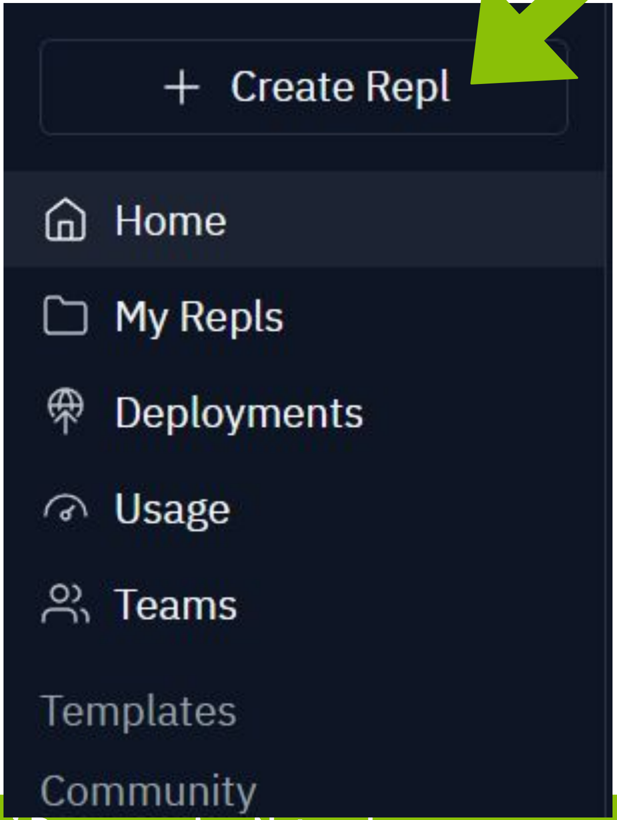


Have an account? Log In
Trouble signing up? Get help

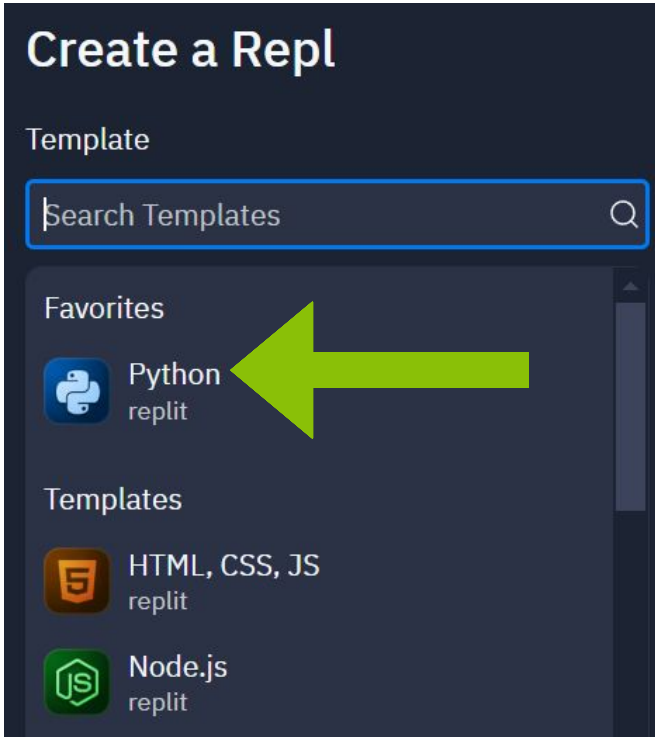
By continuing, you agree to Replit's Terms of Service and Privacy Policy, and to receiving emails with updates.

Creating our **Repl It Project**

Let's create a new project



Select Python for the project template

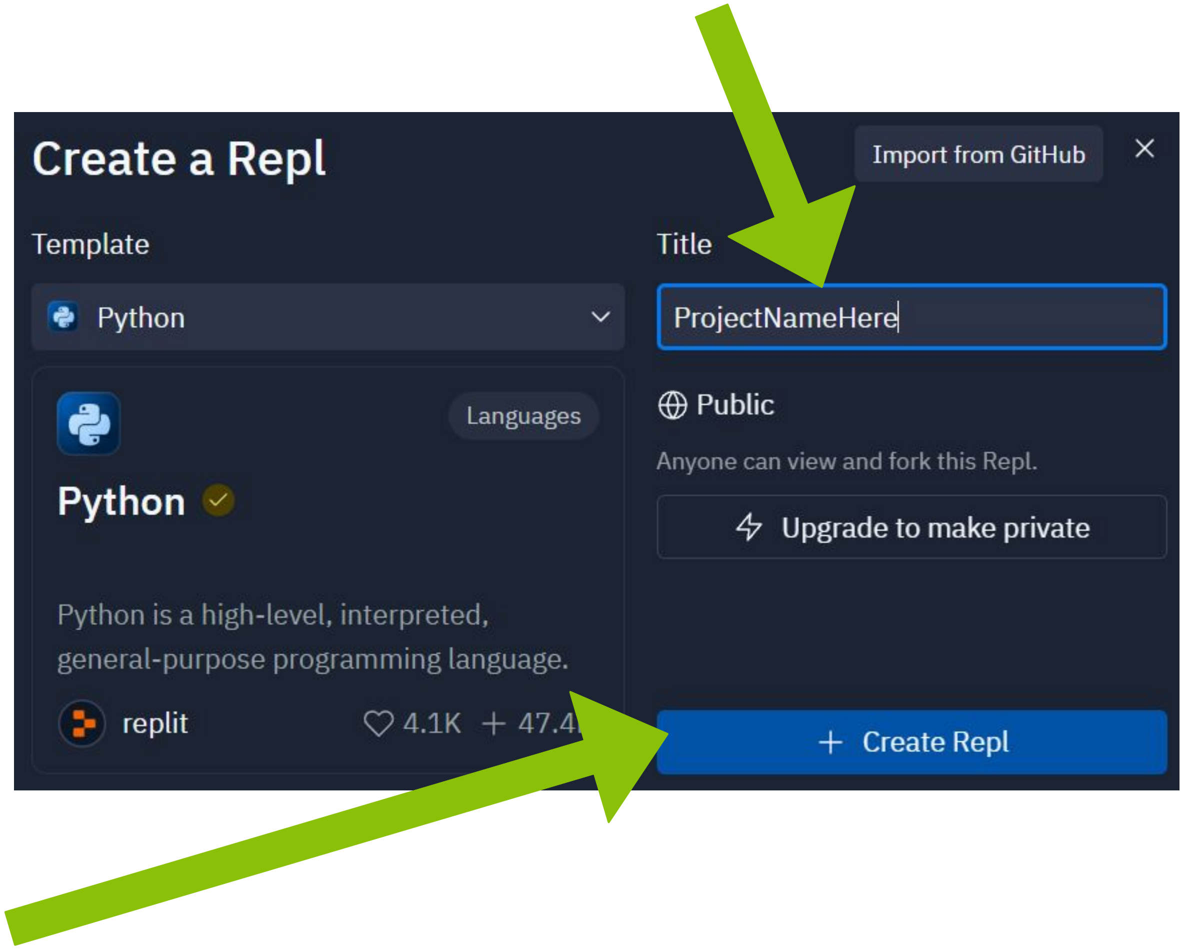


Creating our **Repl It Project**

Don't forget to give your project a name!

Name it after today's project!

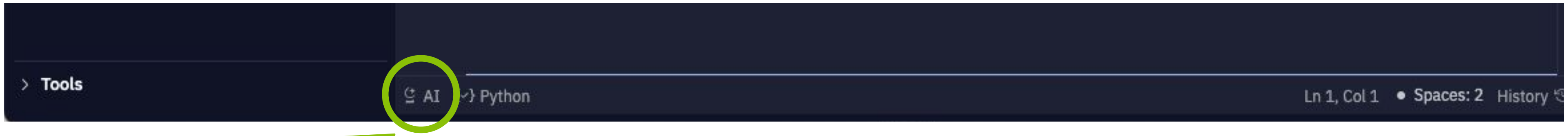
Click Create Repl



Setting our **Repl It Project**

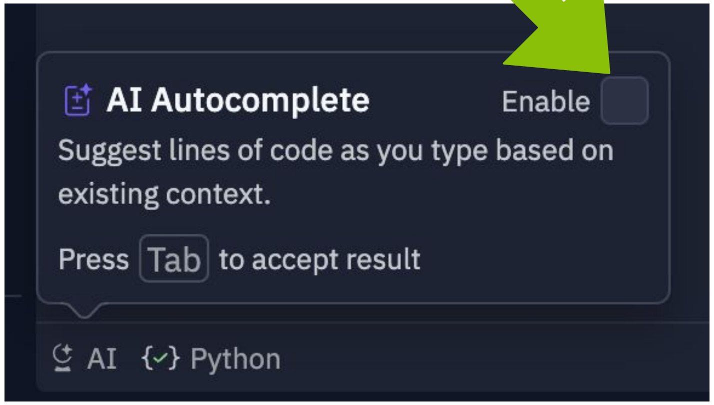
We can't learn if something else is doing all the work!

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


AI Autocomplete Enable

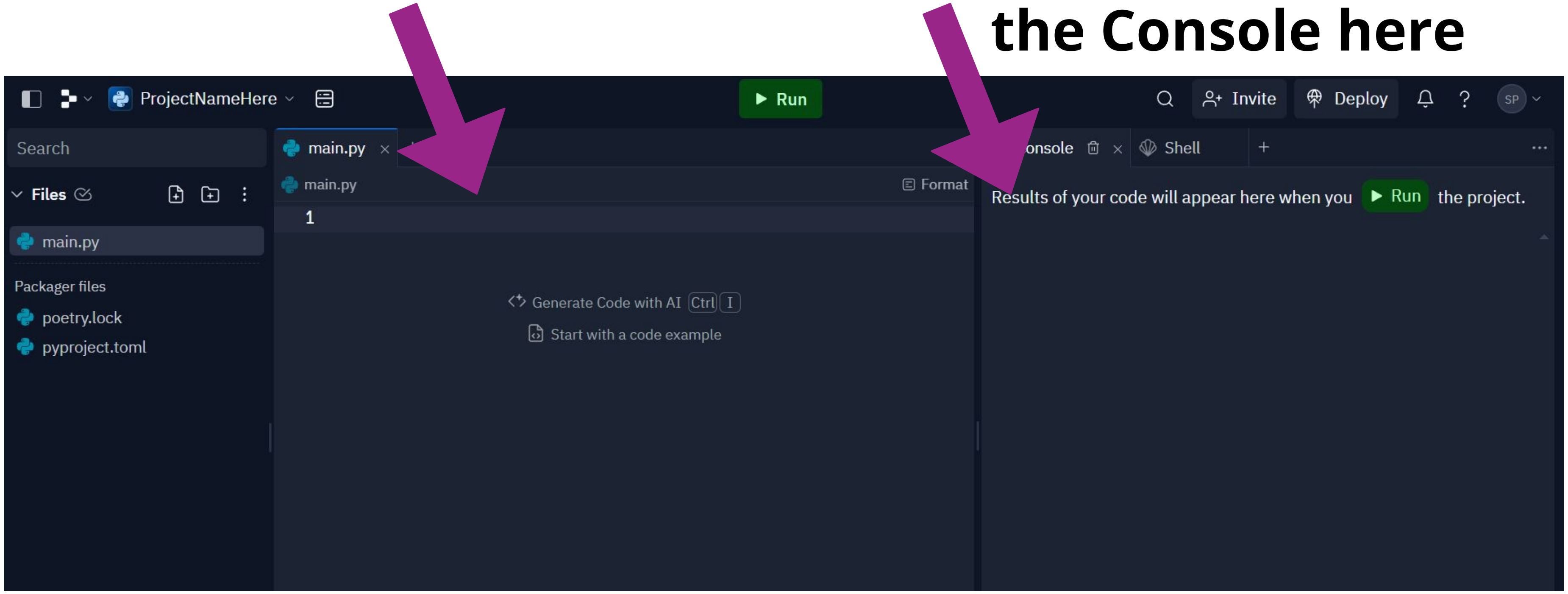
Suggest lines of code as you type based on existing context.

Press Tab to accept result

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

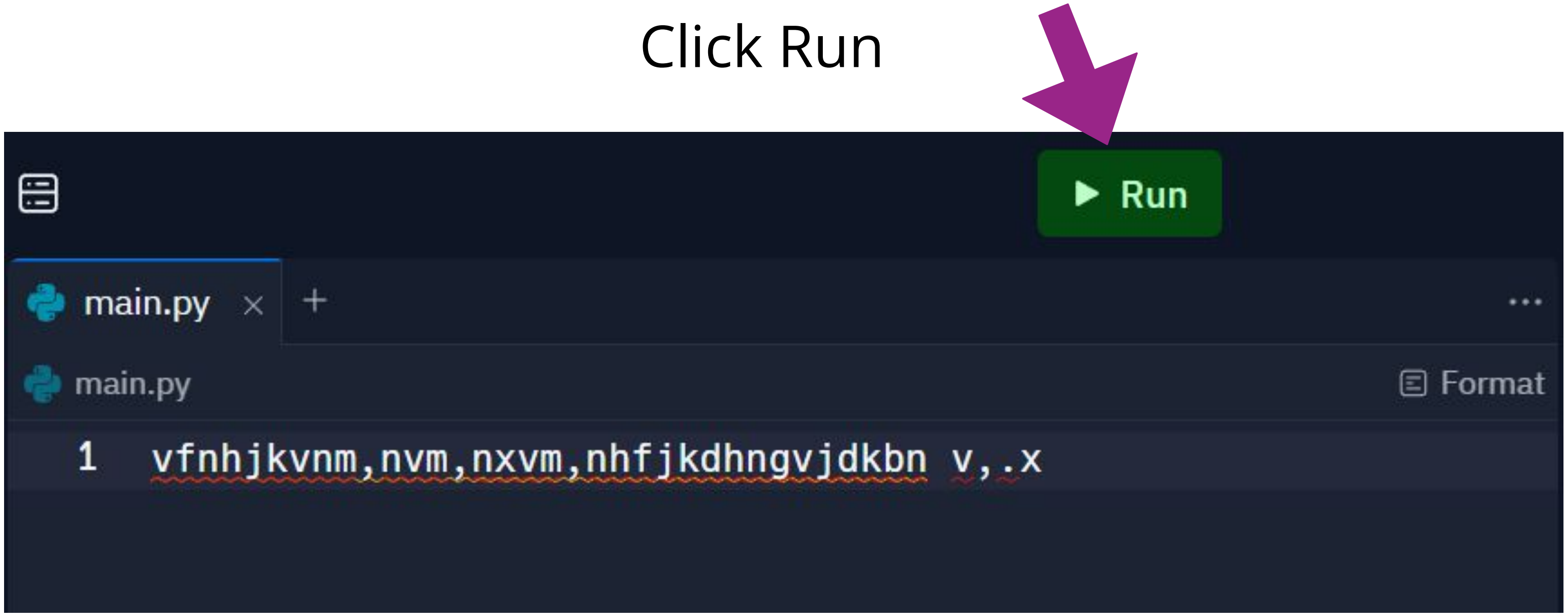


Results of your code will appear here when you ▶ Run the project.

Run a test! Make a mistake!

Type by **button mashing** the keyboard!

Click Run



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

Did you get an error message in the Console?

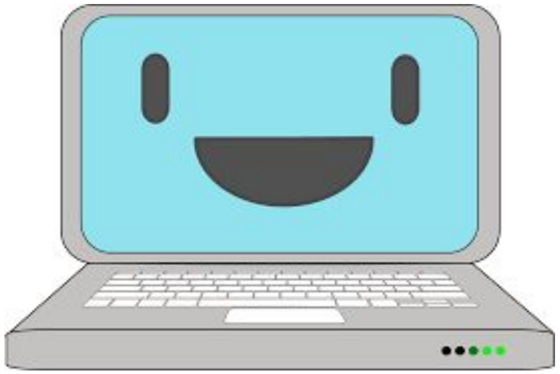
Mistakes are great!

SyntaxError: Invalid Syntax

Good work you made an error!

ImportError: No module named humour

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



KeyError: 'Hairy Potter'

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

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

Write some code!!

Type this into the window

Then press enter!

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

Did it print:

```
hello world
```

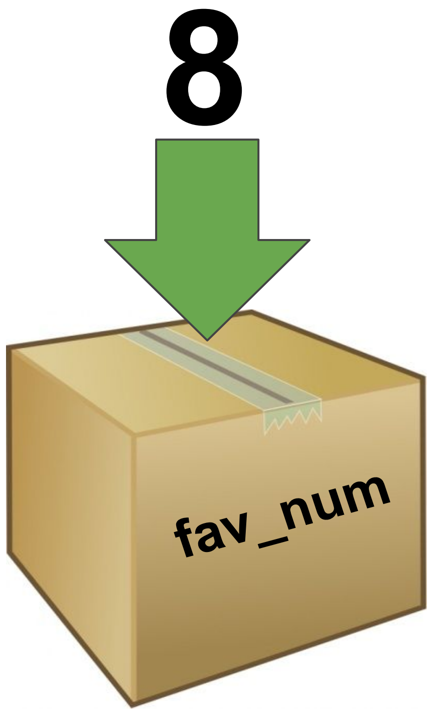
???

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!



Math operators in Python

Before we dive into some examples, let's learn some math operators in Python!

Plus	+
Minus	-
Multiply	*
Divide	/

Variables

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

```
fav_num - 6
   => _


fav_num * 2
   => __
```

Variables

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

```
fav_num - 6
   => 2

fav_num * 2
  => __
```

Variables

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

```
fav_num - 6
   => 2


fav_num * 2
   => 16
```

Variables

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

```
fav_num - 6
   => 2

fav_num * 2
   => 16
```

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

No variables VS using variables

`Imagine we want to change the operating number from 8 to 102:`

4 Changes

1 Change

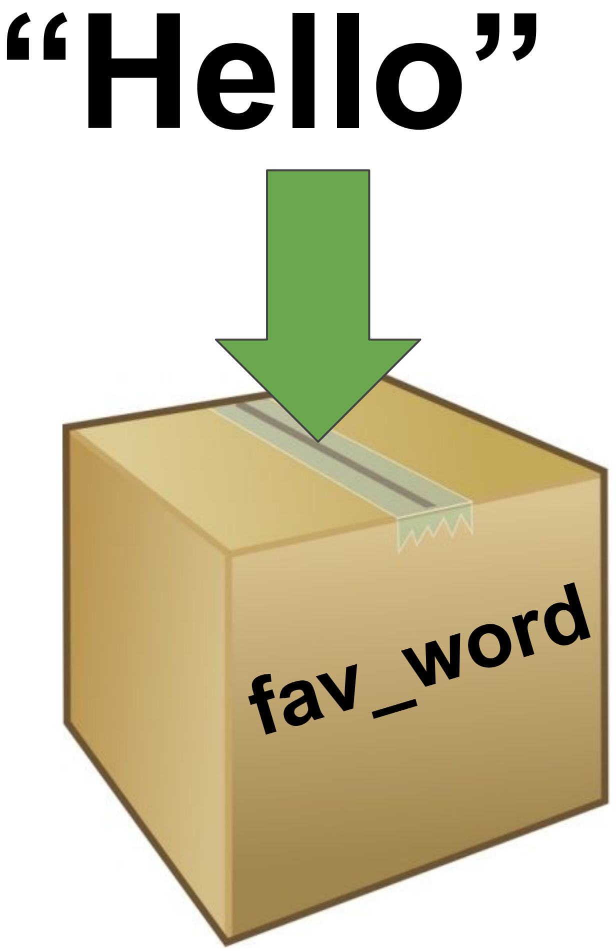
8 - 6	**102** - 6
8 * 2	**102** * 2
8 + 21	**102** + 21
8 / 2	**102** / 2

int fav_num = **8**	int 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

Variables

Variables can store more than numbers

Try store a string in `fav_word`



Variables

Instead of writing the string “Hello”, we can write fav_word:

```
 fav_word + “World”
          => __

    fav_word * 2?
          => ___
```

Variables

Instead of writing the string "Hello", we can write fav_word:

```
fav_word + "World"
  => "HelloWorld"


    fav_word * 2?
        => ___
```

Variables

Instead of writing the string “Hello”, we can write fav_word:

```
fav_word + “World”
  => “HelloWorld”


    fav_word * 2?
  => “HelloHello”
```

Asking a question!

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

Let’s learn about input!

```
>>> my_name = input('What is your name? ')
>>> 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? ')
>>> 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 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!")
```

Project time!

Now you can give the computer variables!

Let's put what we learnt into our project

Try to do Part 0 - 1

The tutors will be around to help!

If Statements

Conditions!

Conditions let us make decision.

First we test if the condition is met!

Then maybe we'll do the thing

<u>If it's raining</u> take an umbrella

Yep it's raining

…… take an umbrella

Booleans (True and False)

Computers store whether a condition is met in the form of

`True` and `False`

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

What do you think these are? True or False?

```
5 < 10          "Dog" == "dog"
5 != 5          "D" in "Dog"
```

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

What do you think these are? True or False?

5 < 10	True	**"Dog" == "dog"**
5 != 5		**"D" in "Dog"**

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

What do you think these are? True or False?

5 < 10	True	**"Dog" == "dog"**
5 != 5	False	**"D" in "Dog"**

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

What do you think these are? True or False?

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

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

What do you think these are? True or False?

5 < 10	True	**"Dog" == "dog"**	False
5 != 5	False	**"D" in "Dog"**	True

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 True      :
    print("that's a small number")
```

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")
```

```
The condition is True, and what happen?
```

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")
```

```
The condition is True, we run the code!
>>> that's a small number
```

Conditions

How about a different number???

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

That's the condition!

Conditions

It’s False!

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

Conditions

It's False!

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

The condition is False, and what happen?

Conditions

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

The condition is False, and wha

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

What do you think happens?

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

What do you think will happen?

If statements

What do you think happens?

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

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

If statements

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

```
What happens??
```

If statements

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

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

Else statements

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

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

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

Else statements

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

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

```
What happens??
```

Else statements

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

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

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

Project Time!

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

Let’s put what we learnt into our project

Try to do Part 2

The tutors will be around to help!

Hashing

Encoding!

Now before we actually start hashing, we need to learn about the concept *encoding*.

Have you heard of it before? Any guesses on how this might be different from hashing?

I'VE FINALLY FOUND IT... AFTER 15 YEARS
THE SCROLL OF TRUTH!
Encoding, Encryption and Hashing are all the same
NYEHHH
DEVELOPERS
Robotatertotcomics
imgflip.com

https://medium.com/swlh/the-difference-between-encoding-encryption-and-hashing-878c606a7aff#:~:text=%2D%20Encoding%20is%20a%20process%20of,into%20a%20fixed%2Dlength%20string.

What is Encoding?

Encoding is the process of making a word (or character, sentence etc.) readable by a computer.

There are different ways we can store things in a computer, such as utf-8 where the letter `a` is encoded to `01100001` which a computer can understand.

https://medium.com/swlh/the-difference-between-encoding-encryption-and-hashing-878c606a7aff#:~:text=%2D%20Encoding%20is%20a%20process%20of,into%20a%20fixed%2Dlength%20string.

What is Hashing?

Hashing is the process of making a character, word, etc. **unreadable** by a human, which makes it more secure.

The value that has been hashed is called a hash!

What is Hashing?

How does it work?

We take a readable word or phrase (this is called plaintext) like this:

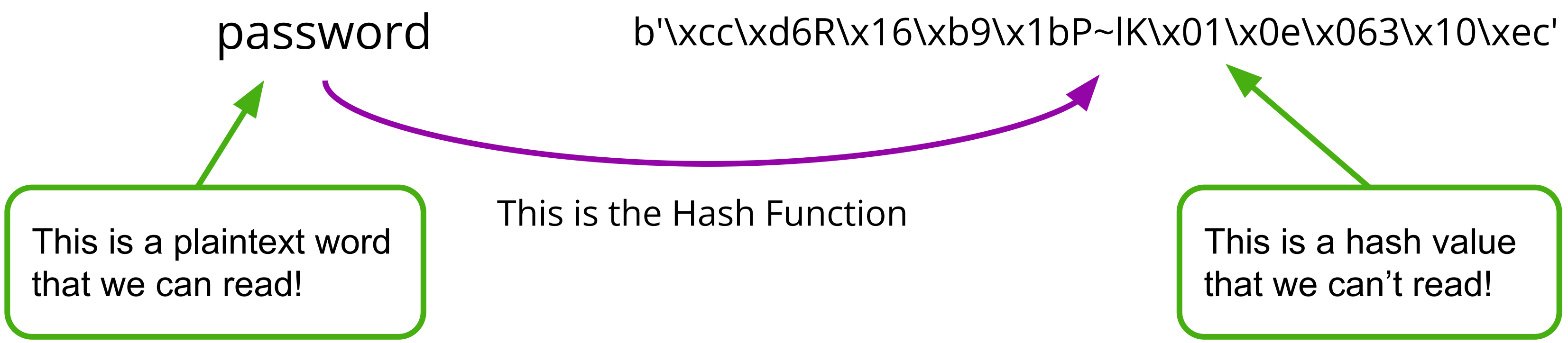
password

This is a plaintext word that we can read!

What is Hashing?

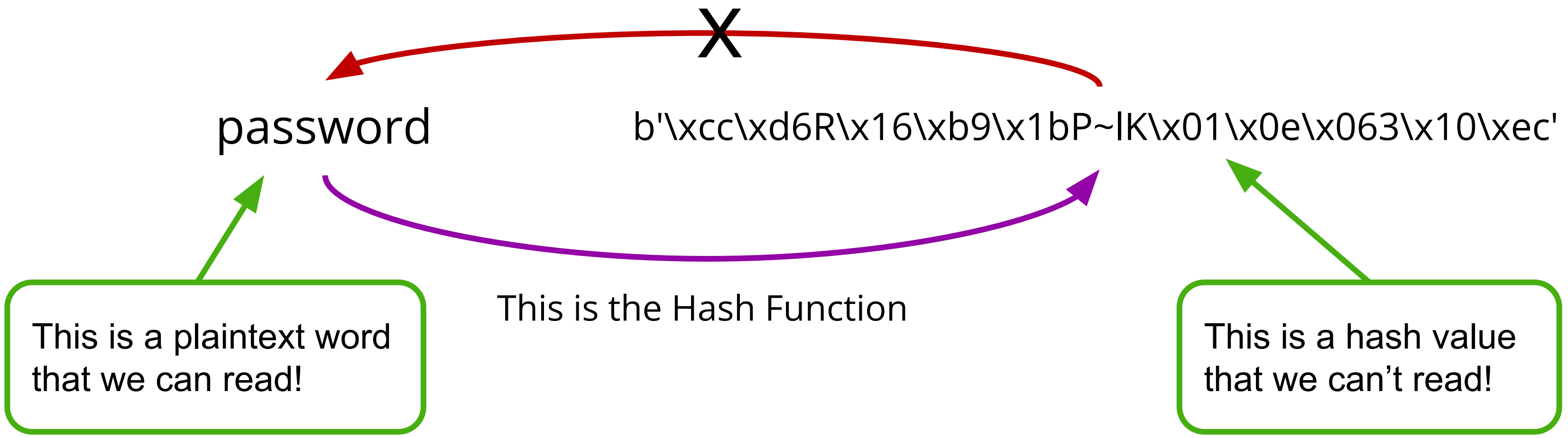
How does it work?

We take a readable word or phrase (this is called plaintext) like this:


password
b'\xcc\xd6R\x16\xb9\x1bP~IK\x01\x0e\x063\x10\xec'
This is a plaintext word that we can read!
This is the Hash Function
This is a hash value that we can't read!

And we use a “Hash function” to turn it into something we can’t read!

What is Hashing?


X
password
b'\xcc\xd6R\x16\xb9\x1bP~lK\x01\x0e\x063\x10\xec'
This is a plaintext word that we can read!
This is the Hash Function
This is a hash value that we can't read!

The coolest thing about a Hash function is that you can only go **one way**, so you can't work out what the plaintext word was if you only have the hash value - this makes it secure!

Hashing in Python

Here's all the code we need to hash some text in Python

```
import hashlib
my_string = "hello"
my_string_encoded = my_string.encode()
my_string_hashed = hashlib.md5(my_string_encoded)
```

Now let's go through each line and see what it does.

Hashing in Python

Firstly to use the Python code we need to import the hashing library!

We can do this by writing:

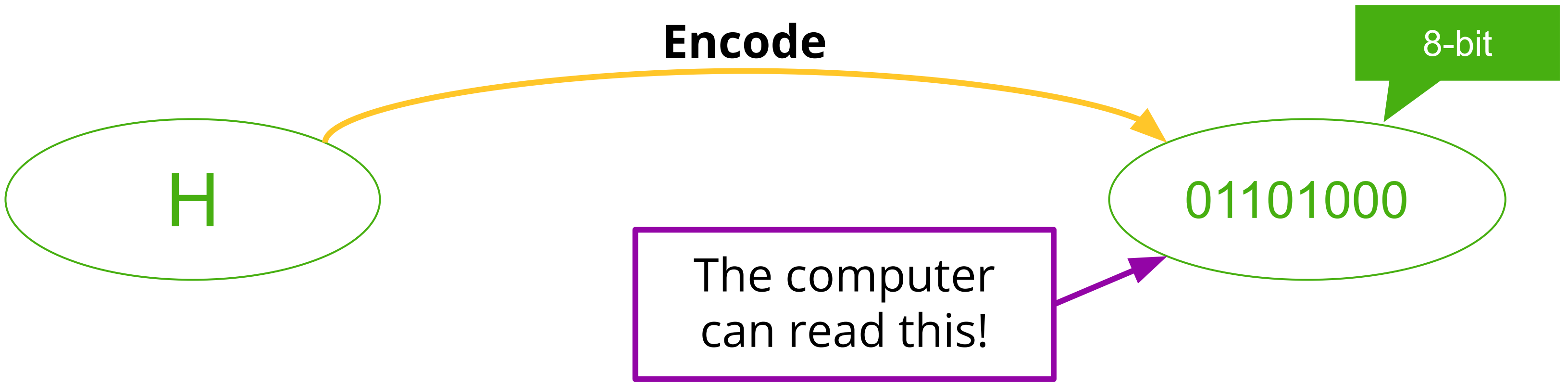
```
import hashlib
```

at the top of our code!

Encoding

After we have imported our library we can start hashing by first encoding our variables using the .encode() method!

```
my_string = “hello”
my_string_encoded = my_string.encode()
```



Hashing!

Now we can actually hash our value!

To hash a value we can use the .md5() function like this:

This is the encoded string from the last slide!

```
my_string_hashed = hashlib.md5(my_string_encoded)
```

MD5 is the name of the hash function that we are using!

Digest!

After hashing our variable we want to turn it into a value we can use, so we use the .digest() method, written:

```
my_string_hashed = hashlib.md5(my_string_encoded).digest()
```

This turns the hash into something that we can use!

Digest!

After hashing our variable we want to turn it into a value we can use, so we use the .digest() method, written:

```
my_string_hashed = hashlib.md5(my_string_encoded).digest()
```

Result:

```
b']A@*\xbcK*v\xb9q\x9d\x91\x10\x17\xc5\x92'
```

Project Time!

Hashing!

Let’s put what we learnt into our project

Try to do Parts 3 - 5!

The tutors will be around to help!

Extension: Meme Generator

Show me the memes!

We have some accounts for you to try and crack into! They are some accounts for our secret website, the GPN Meme Exchange!

Once you've cracked the passwords, head over there and try them out!

https://girls-programming-network.github.io/meme-exchange/

The link is also on the website from the start of the day!

Tell us what you think!

Click on the

End of Day Form

and fill it in now!