Password Cracker

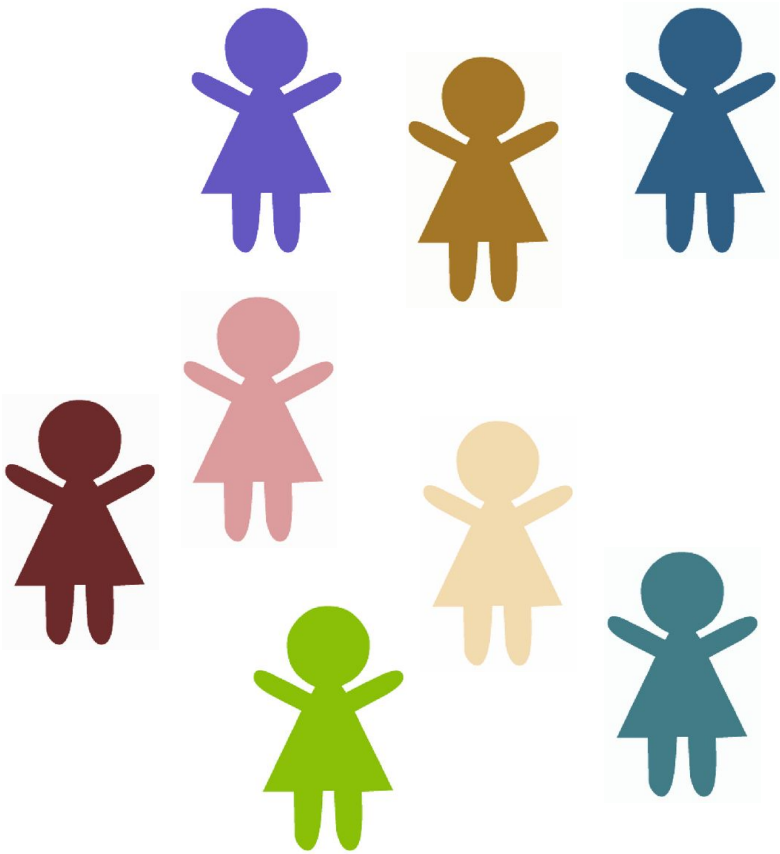
Welcome to the labs!

Who are the tutors?

Who are you?

People Bingo

Rules: 1) Read each square. 2) Find a new friend who can complete any of the squares. 3) One friend can fill in **one square only**. 4) The first person to fill in **every square** wins!				
Can fluently speak more than one language 🗣️	Has been to GPN before 💚	Has visited another country ✈️	Can play chess ♟️	Has more than 1 pet 🐶
Favourite food is Pizza 🍕	Knows what a Kumquat is ❓	Has made their own pasta 🍝	Has eaten Dragon fruit 🐲	Was born in another country 🌎
Knows how to Juggle 🤹‍♀️	Likes to play Tetris 🎮	Is an only child 👧	Likes to paint 🎨	Knows a magic trick 🪄
Plays an instrument 🎶	Can ride a skateboard 🛹	Likes to read 📖	Has the same favourite colour as you 🌈	Has the same eye colour as you 👀
Has a part time or casual job 💰	Is in the same school year as you 🚌	Knows the second verse of the national anthem 🇳🇿	Has created their own coding project 💻	Plays sport on the weekend ⚽



Tell us you're here!

Click on the

Start of Day Survey

and fill it in now!

Password Cracker!

Today’s project!

Using the workbook!

The workbooks will help you put your project together!

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

Tasks - The parts of your project

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

Hints - Helpers for your tasks!

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

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

Task 6.2: Add a blah to your code!

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

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

Task 6.1: Make the thing do blah!

Make your project do blah ….

Hint

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

```
print('This example is not part of the project' )
```

Using the workbook!

The workbooks will help you put your project together!

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

Checklist - Am I done yet?

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

Lecture Markers
This tells you you'll find out how to do things for this section during the names lecture.

Bonus Activities
Stuck waiting at a lecture marker?
Try a purple bonus. They add extra functionality to your project along the way.

CHECKPOINT

If you can tick all of these off you're ready to move the next part!

- ☐ Your program does blah
- ☐ Your program does blob

★ BONUS 4.3: Do some extra!

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

Intro to Programming

What is programming?

Programming is not a bunch of crazy numbers!

It's giving computers a set of instructions!

A Special Language

A language to talk to dogs!

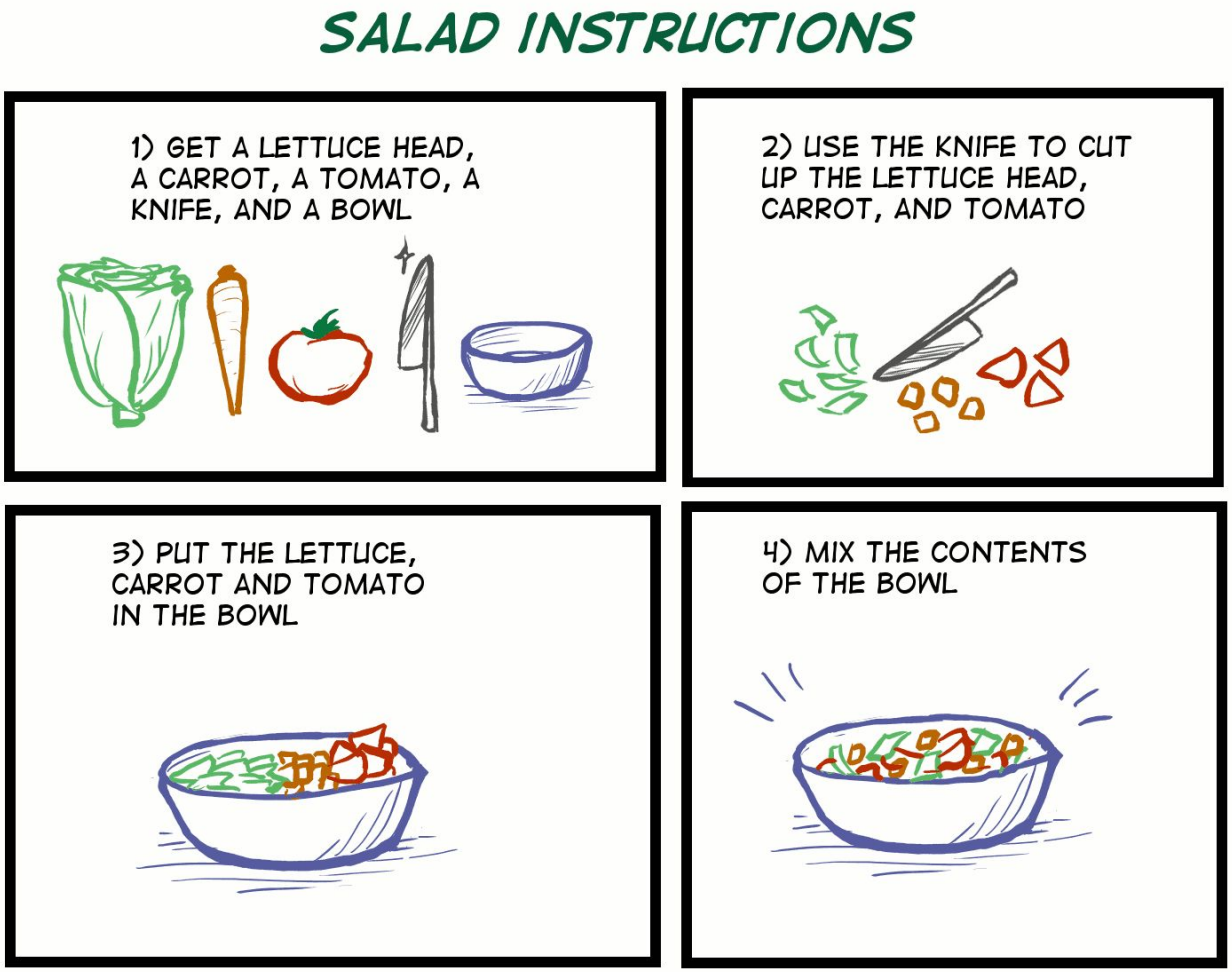
Programming is a language to talk to computers

People are smart! Computers are dumb!

Programming is like a recipe!

Computers do EXACTLY what you say, every time.

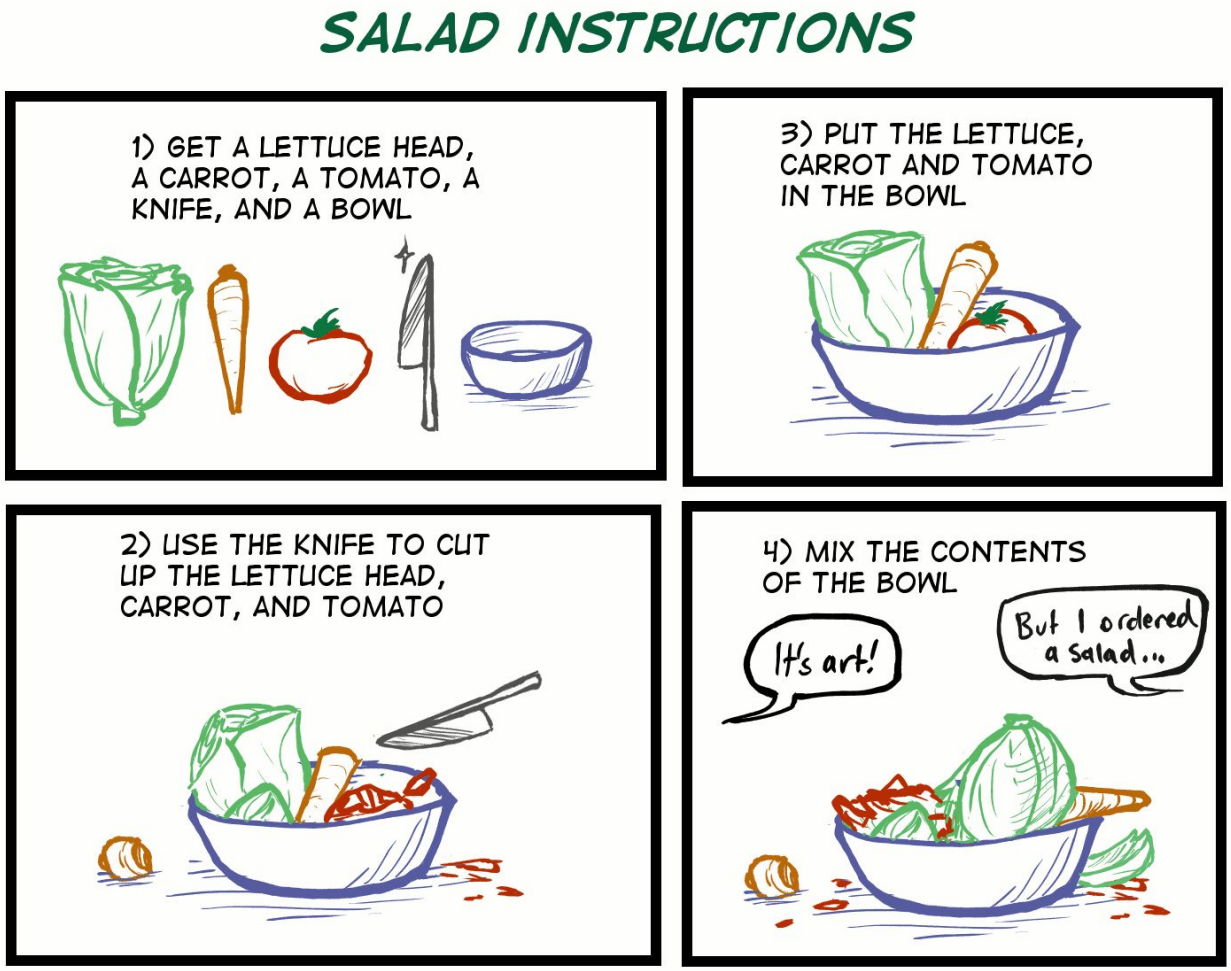
Which is great if you give them a good recipe!



People are smart! Computers are dumb!

But if you get it out of order….

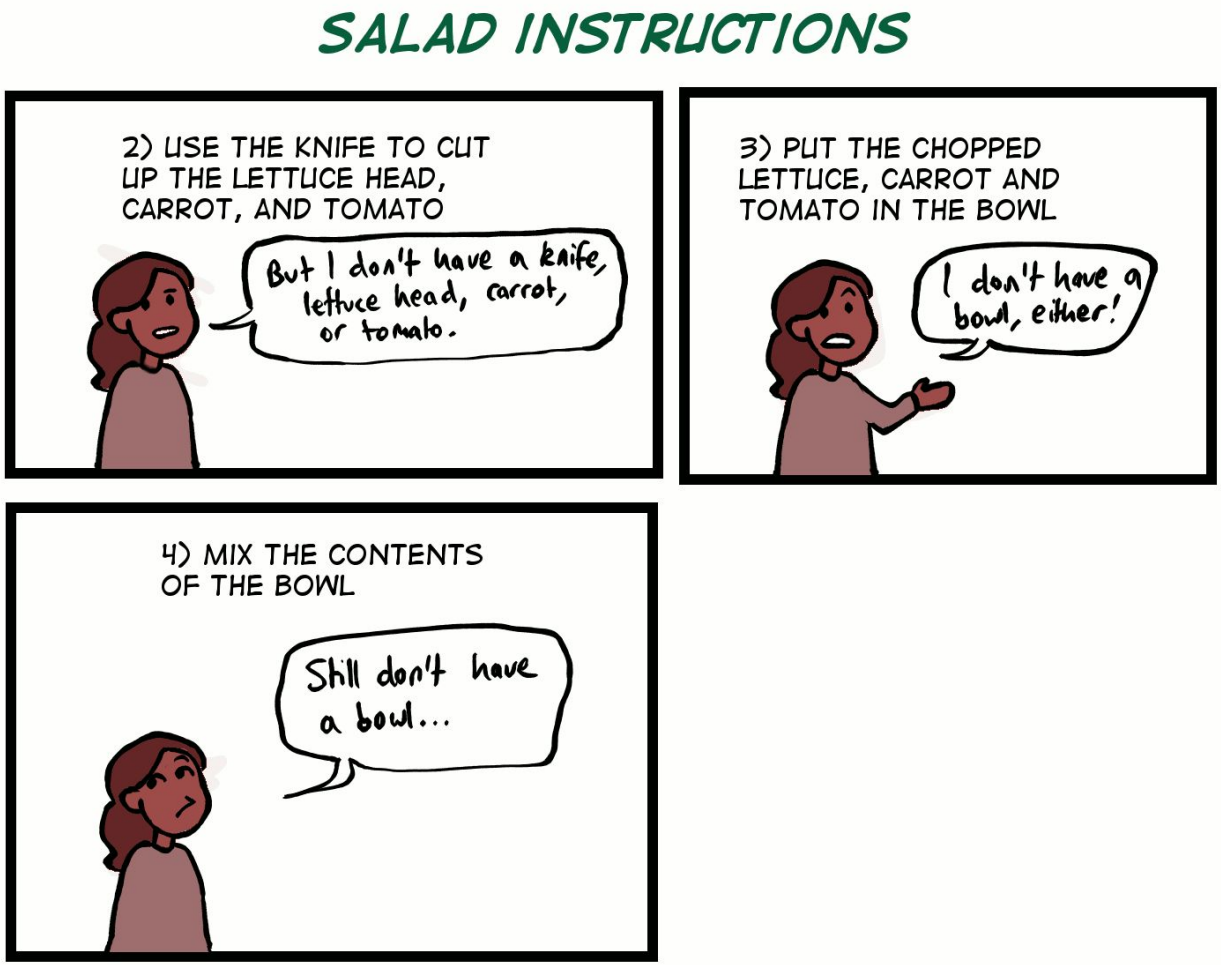
A computer wouldn't know this recipe was wrong!



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!



Everyone/thing has strengths!

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

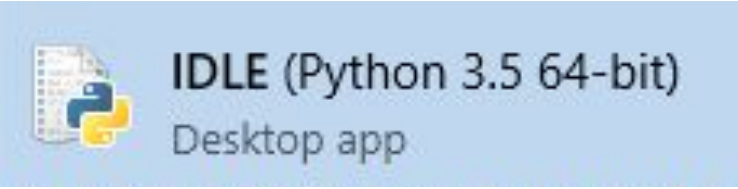
- Does exactly what you tell it
- Does it the same every time
- Doesn't need to sleep!
- Will work for hours on end!
- Get smarter when you tell them how

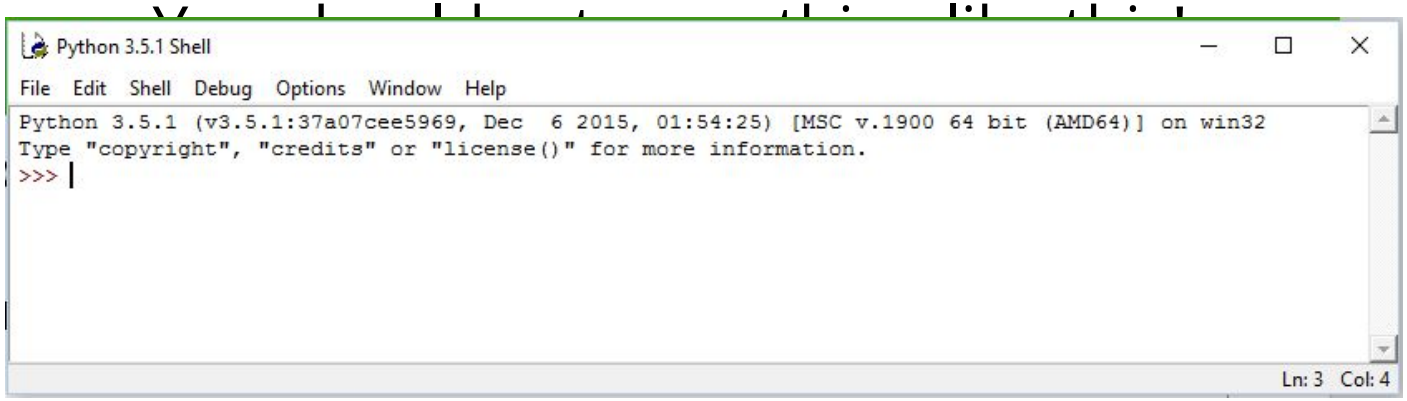
Intro to Python

Let’s get coding!

Where do we program? In IDLE

Click the start button and type IDLE!





Make a mistake!

Type by **button mashing** the keyboard!

Then press enter!

```
asdf asdjlkj;pa j;k4uroei
```

Did you get a big red error message?

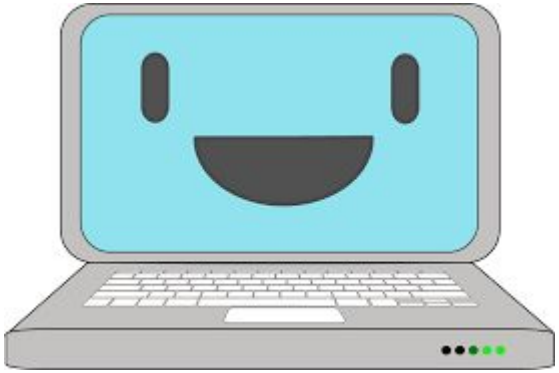
Mistakes are great!

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

Write some code!!

Type this into the window

Then press enter!

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

Did it print:

```
hello world
```

???

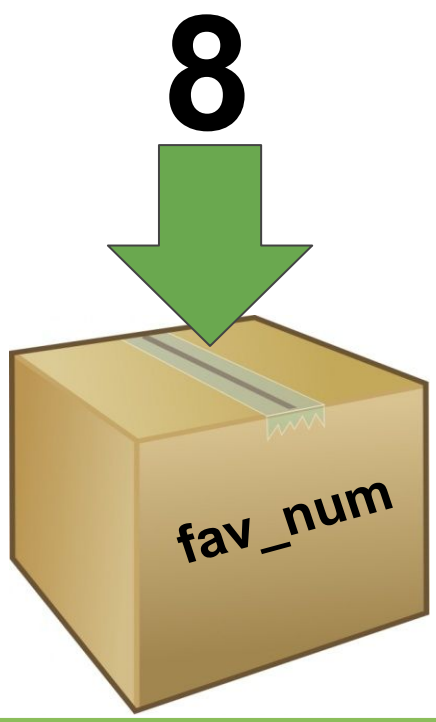
No Storing is Boring!

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!



Variables

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


```
fav_num - 6
    => _
```

```
fav_num + 21
    => __
```

```
fav_num * 2
    => __
```

```
fav_num / 2
    => _
```

Variables

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


```
fav_num - 6
    => 2
```

```
fav_num + 21
    => __
```

```
fav_num * 2
    => __
```

```
fav_num / 2
    => _
```

Variables

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


```
fav_num - 6
    => 2
```

```
fav_num + 21
    => 29
```

```
fav_num * 2
    => __
```

```
fav_num / 2
    => _
```

Variables

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


```
fav_num - 6
```

=> 2

```
fav_num + 21
```

=> 29

```
fav_num * 2
```

=> 16

```
fav_num / 2
```

=> _

Variables

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


```
fav_num - 6
```

=> 2

```
fav_num + 21
```

=> 29

```
fav_num * 2
```

=> 16

```
fav_num / 2
```

=> 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

2

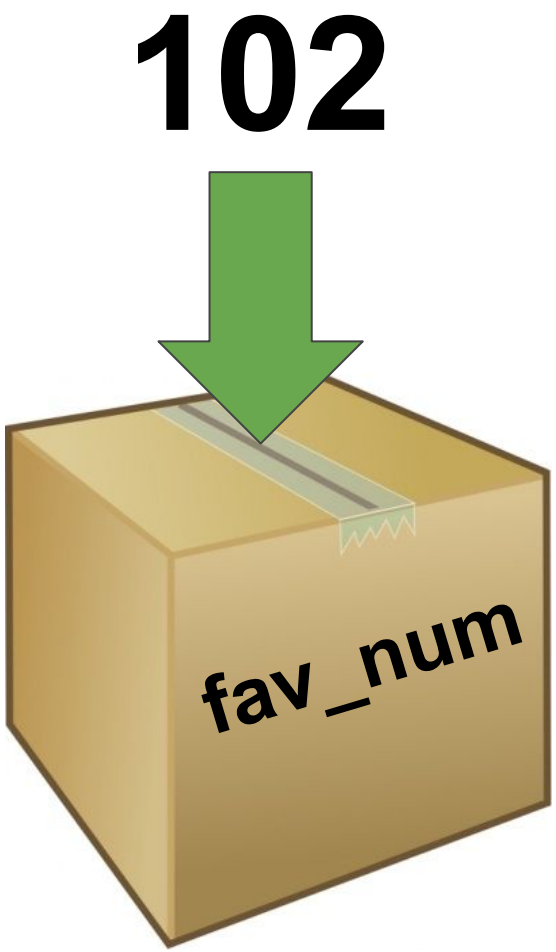
But writing 8 is much shorter than writing `fav_num`???

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
=> __
```

```
fav_num + 21
=> ___
```

```
fav_num * 2?
=> ___
```

```
fav_num / 2?
=> __
```

Variables

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


```
fav_num - 6
=> 96
```

```
fav_num + 21
=> ___
```

```
fav_num * 2?
=> ___
```

```
fav_num / 2?
=> __
```

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?
=> ___
```

```
fav_num / 2?
=> __
```

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?
=> __
```

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

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

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"**
3 + 2 == 5	**"D" in "Dog"**
5 != 5	**"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

What do you think these are? True or False?		
5 < 10	True	**"Dog" == "dog"**
3 + 2 == 5		**"D" in "Dog"**
5 != 5		**"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

What do you think these are? True or False?		
5 < 10	True	"Dog" == "dog"
3 + 2 == 5	True	"D" in "Dog"
5 != 5		"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

What do you think these are? True or False?		
5 < 10	True	"Dog" == "dog"
3 + 2 == 5	True	"D" in "Dog"
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

What do you think these are? True or False?			
5 < 10	True	"Dog" == "dog"	False
3 + 2 == 5	True	"D" in "Dog"	
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

What do you think these are? True or False?			
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

What do you think these are? True or False?			
5 < 10	True	"Dog" == "dog"	False
3 + 2 == 5	True	"D" in "Dog"	True
5 != 5	False	"Q" not in "Cat"	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 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**
- **<u>So it is `True`!</u>**

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")
```

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

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")
```

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

```
>>> 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

It's False!

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

Put in the answer to the question

Conditions

It's False!

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

```
What do you think happens?
>>>
```

Conditions

```
fave_num = 9000
if False:
    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

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 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!
```

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

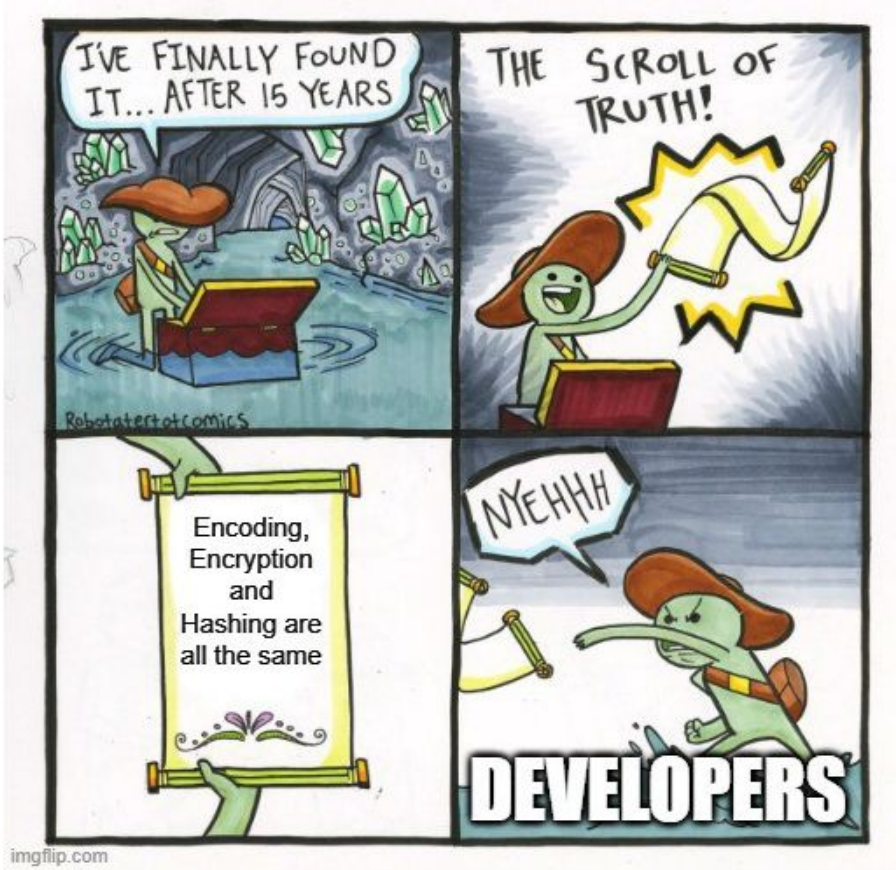
The tutors will be around to help!

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?



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~lK\x01\x0e\x063\x10\xec'

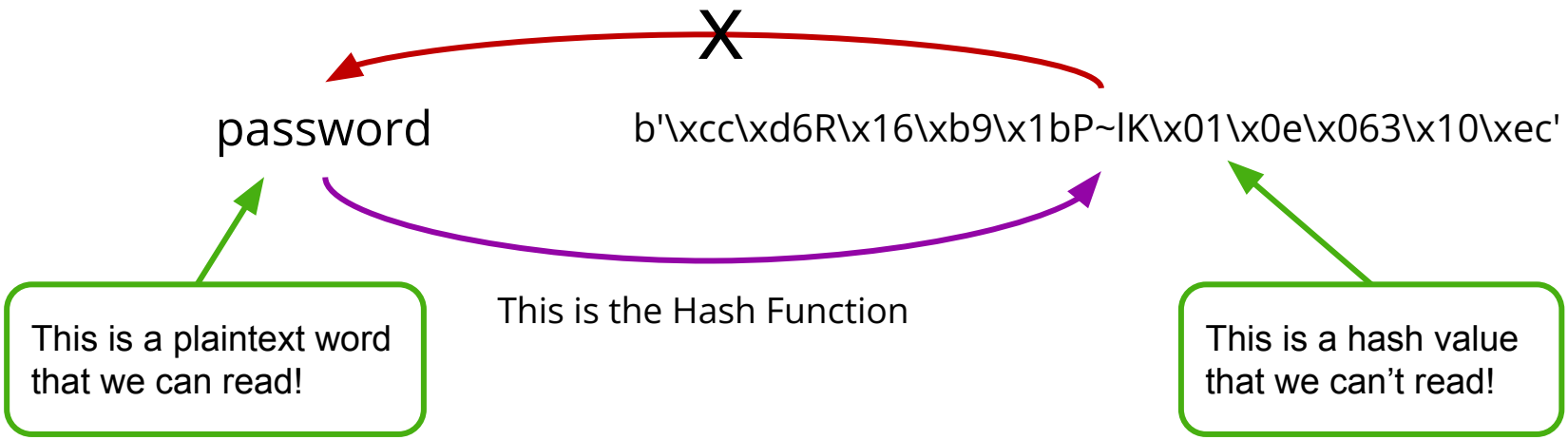
This is a plaintext word that we can read!

This is the Hash Function

This is a hash value that we can't read!

And we use a "Hash function" to turn it into something we can't read!

What is Hashing?



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:

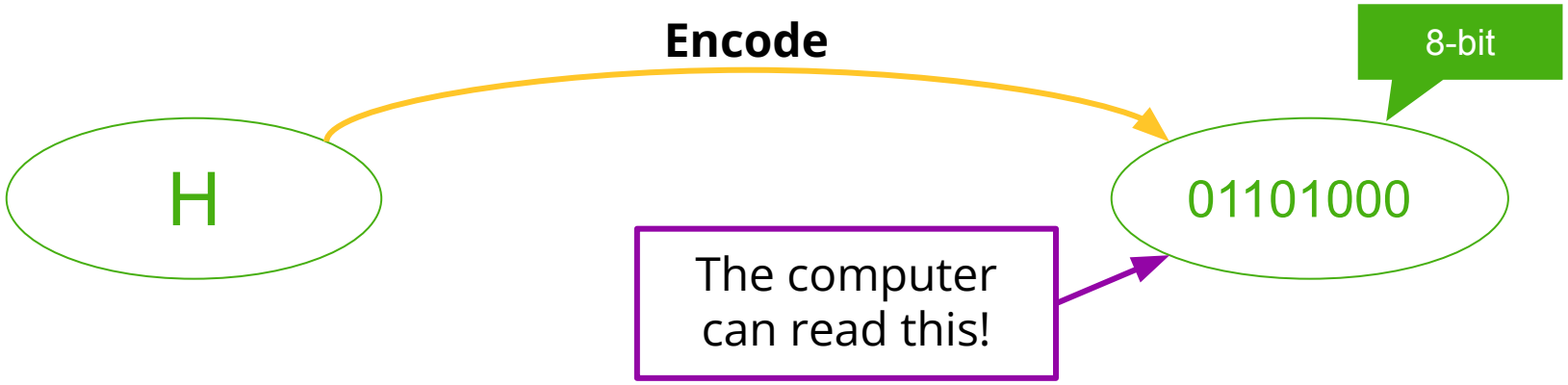
```
import hashlib
```

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!