Welcome to the labs!

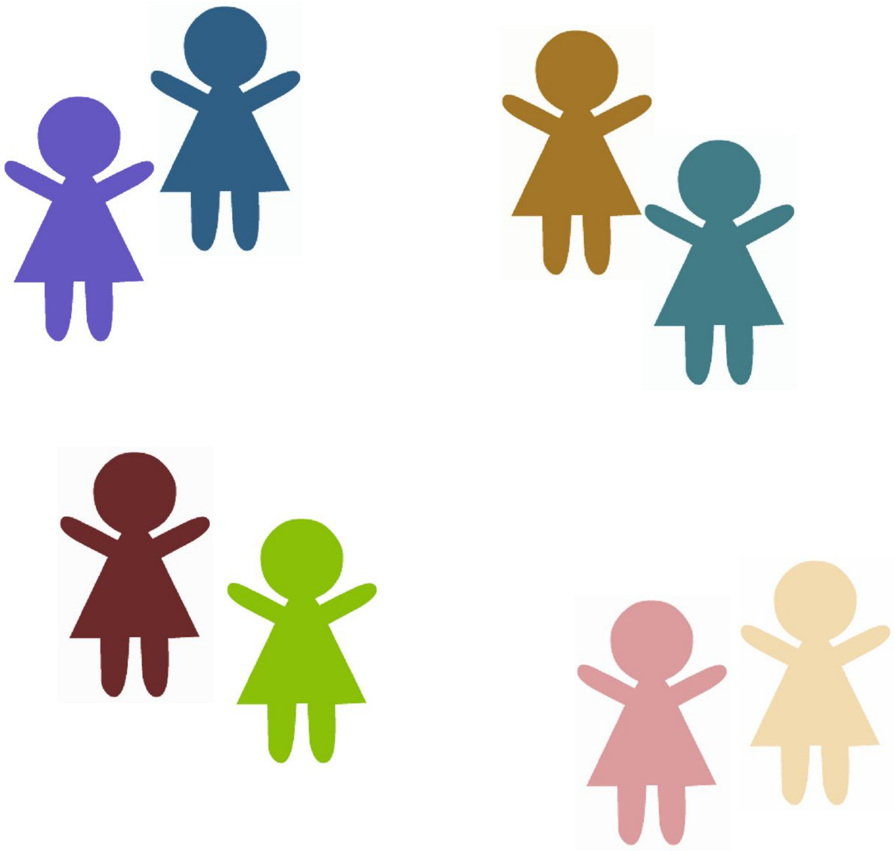
Cryptography

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

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!

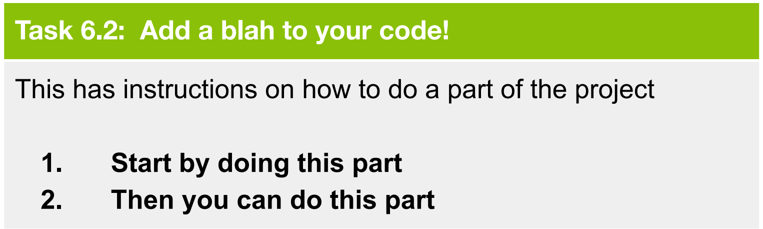
Tasks - The parts of your project

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**!

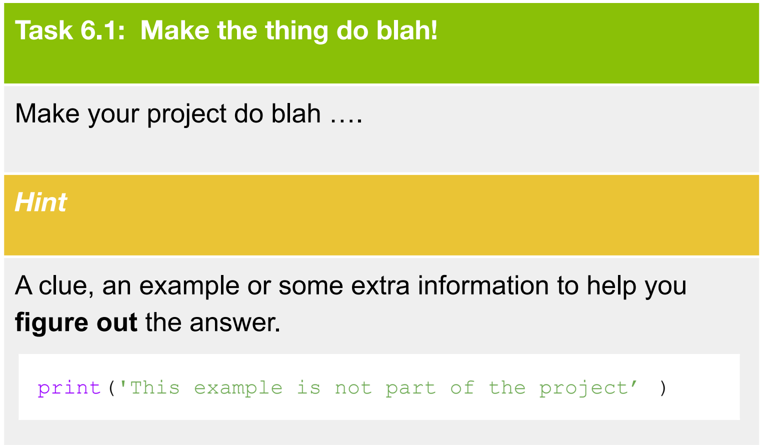
The hints have **unrelated** examples, or tips. **Don't copy and paste** in the code, you'll end up with something **CRAZY**!



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.

☑ **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 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

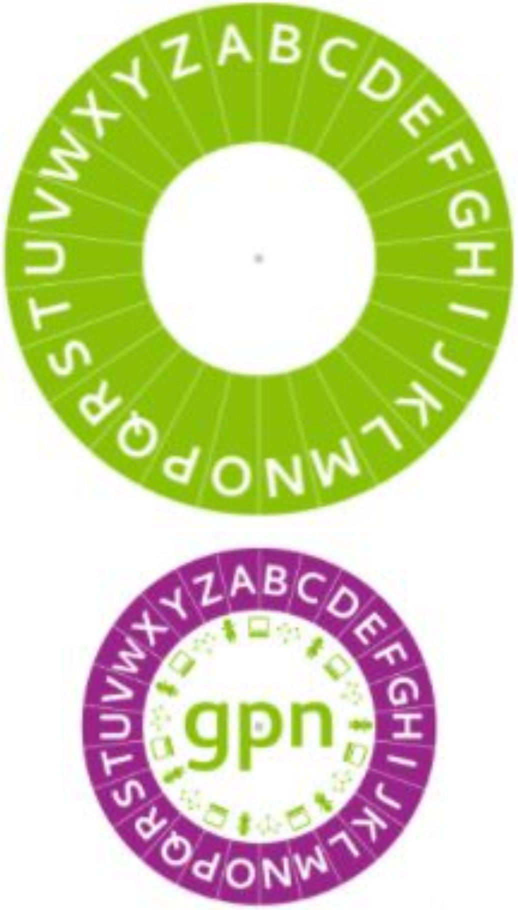
So what's a Caesar Cipher?

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

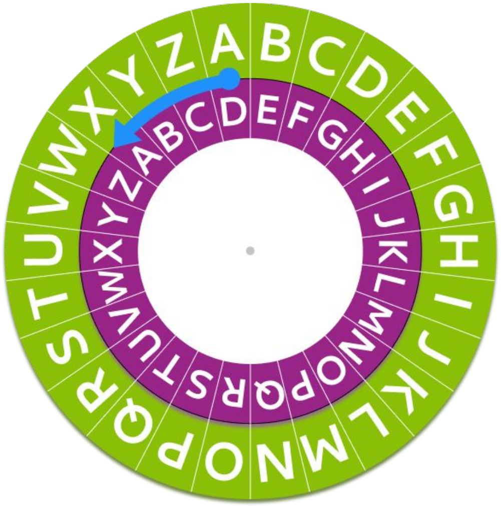
Now, let’s encrypt **I love coding** using the wheel

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 it’s matching letter on the* ***inside*** *wheel* = the letter ‘l’

Encrypting

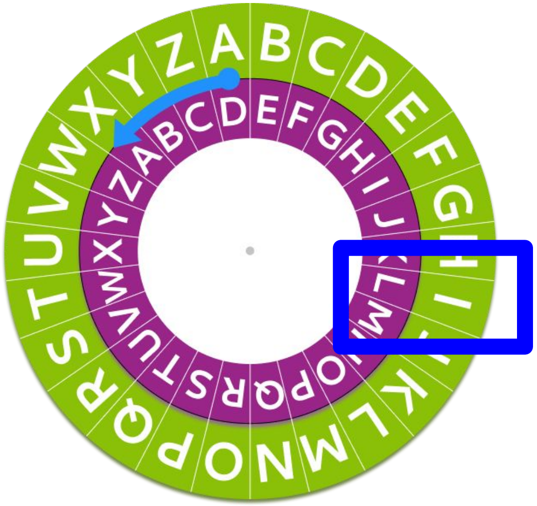
Now, let’s encrypt **I love coding** using the wheel

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?


Z 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
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

>>> *Find letter i on the **outside** wheel and replace it with it’s matching letter on the **inside** wheel* = the letter ‘l’

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	
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

l	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	l
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 = Idrrhbz)

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

Let’s check it works!

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

Let's check it works!

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

Let’s check it works!

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

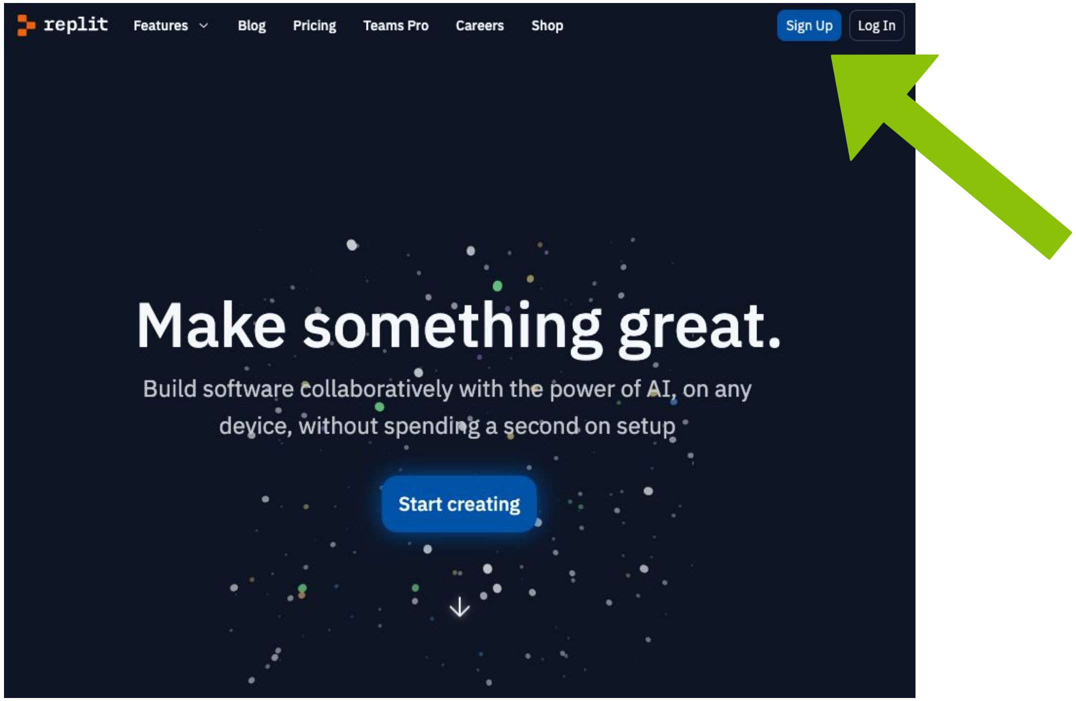
Your tutors are here to help you if you get stuck

Intro to Python

Let's get coding!

Where do we program?

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



Make something great.

Build software collaboratively with the power of AI, on any device, without spending a second on setup

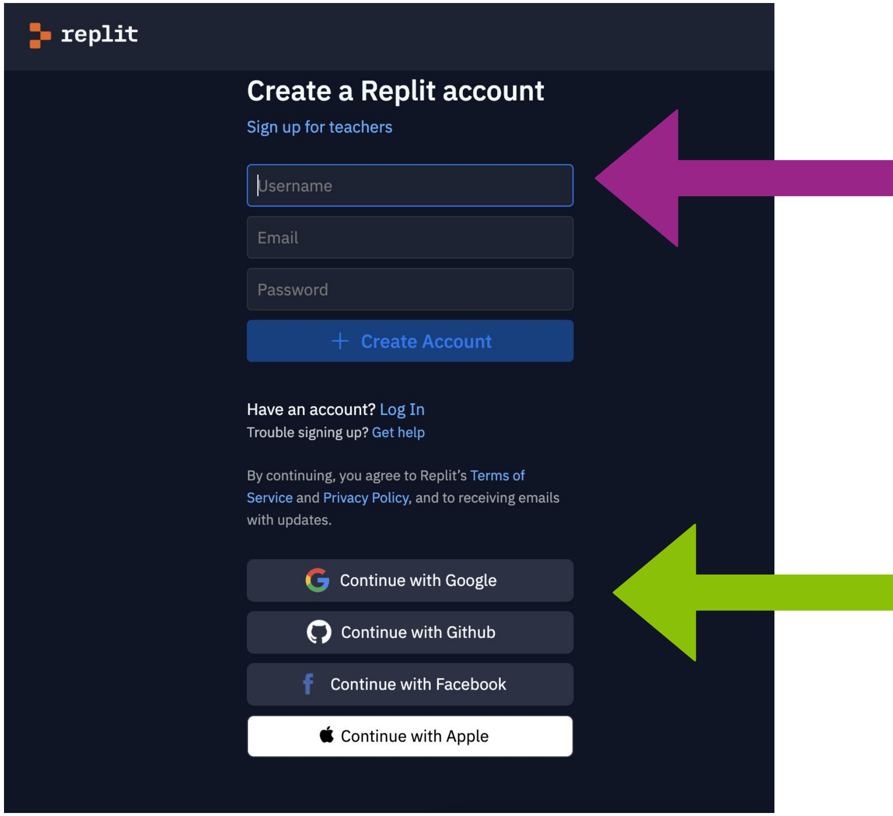
Go to replit.com in your web browser

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.



Create a Replit account

Sign up for teachers

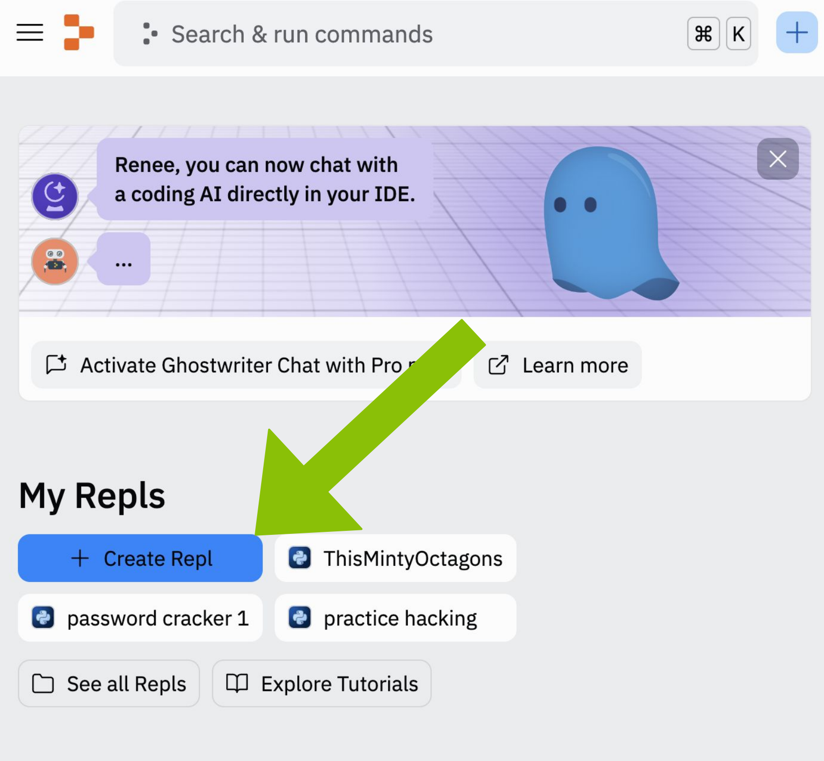
Have an account? Log In

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



Search & run commands ⌘ K +

Renee, you can now chat with a coding AI directly in your IDE.

...

Activate Ghostwriter Chat with Pro r

Learn more

My Repls

+ Create Repl

ThisMintyOctagons

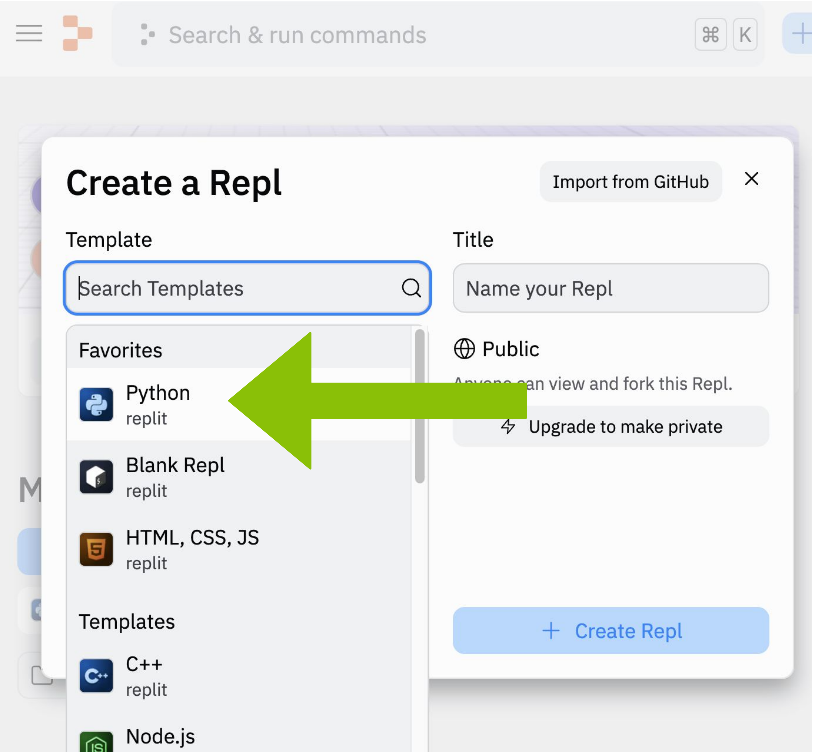
password cracker 1

practice hacking

See all Repls

Explore Tutorials

Select Python for the project template



Search & run commands ⌘ K

Create a Repl

Import from GitHub ×

Template

Search Templates

Title

Name your Repl

Favorites

Python
replit

Blank Repl
replit

HTML, CSS, JS
replit

Public

Anyone can view and fork this Repl.

Upgrade to make private

Templates

C++
replit

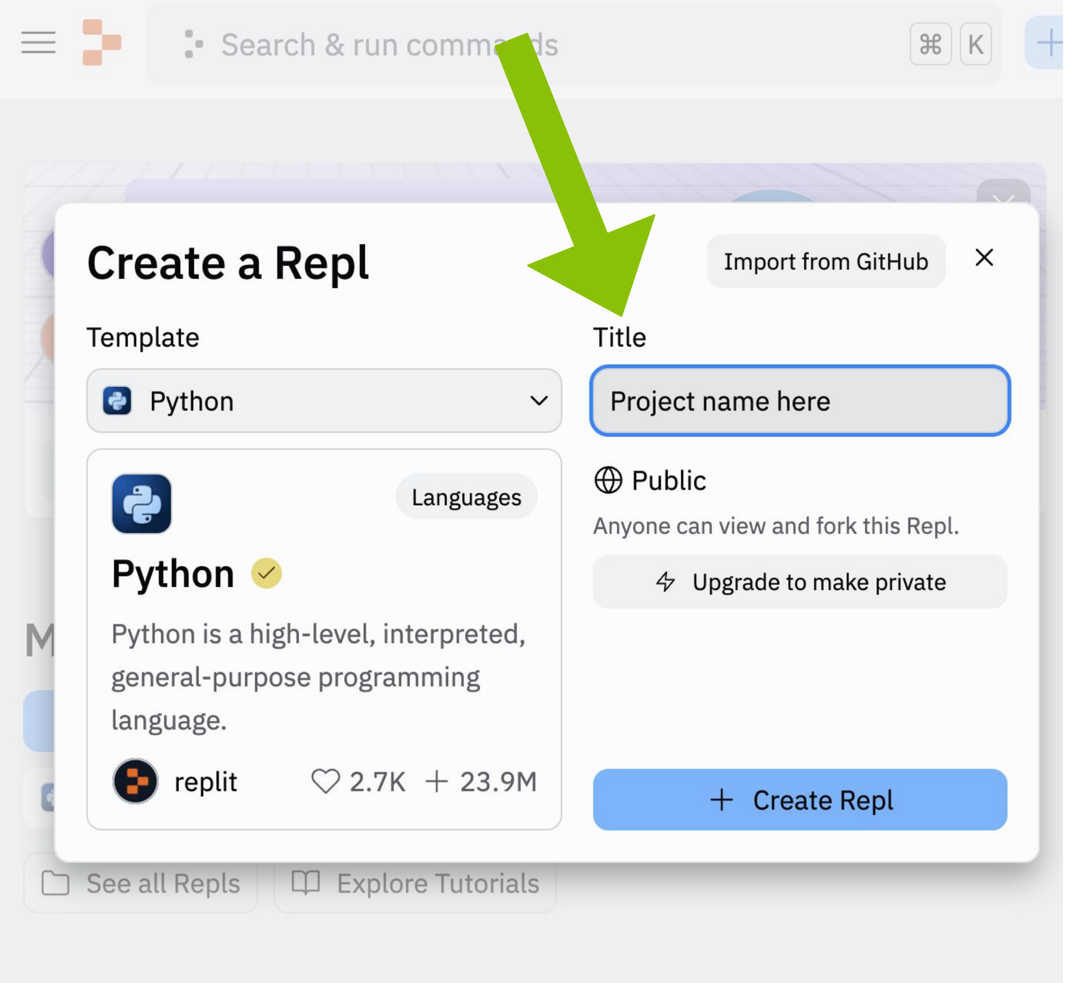
Node.js

+ Create Repl

Creating our **Repl It Project**

Don't forget to give your project a name!

Name it after today's project!



Create a Repl

Template: Python

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

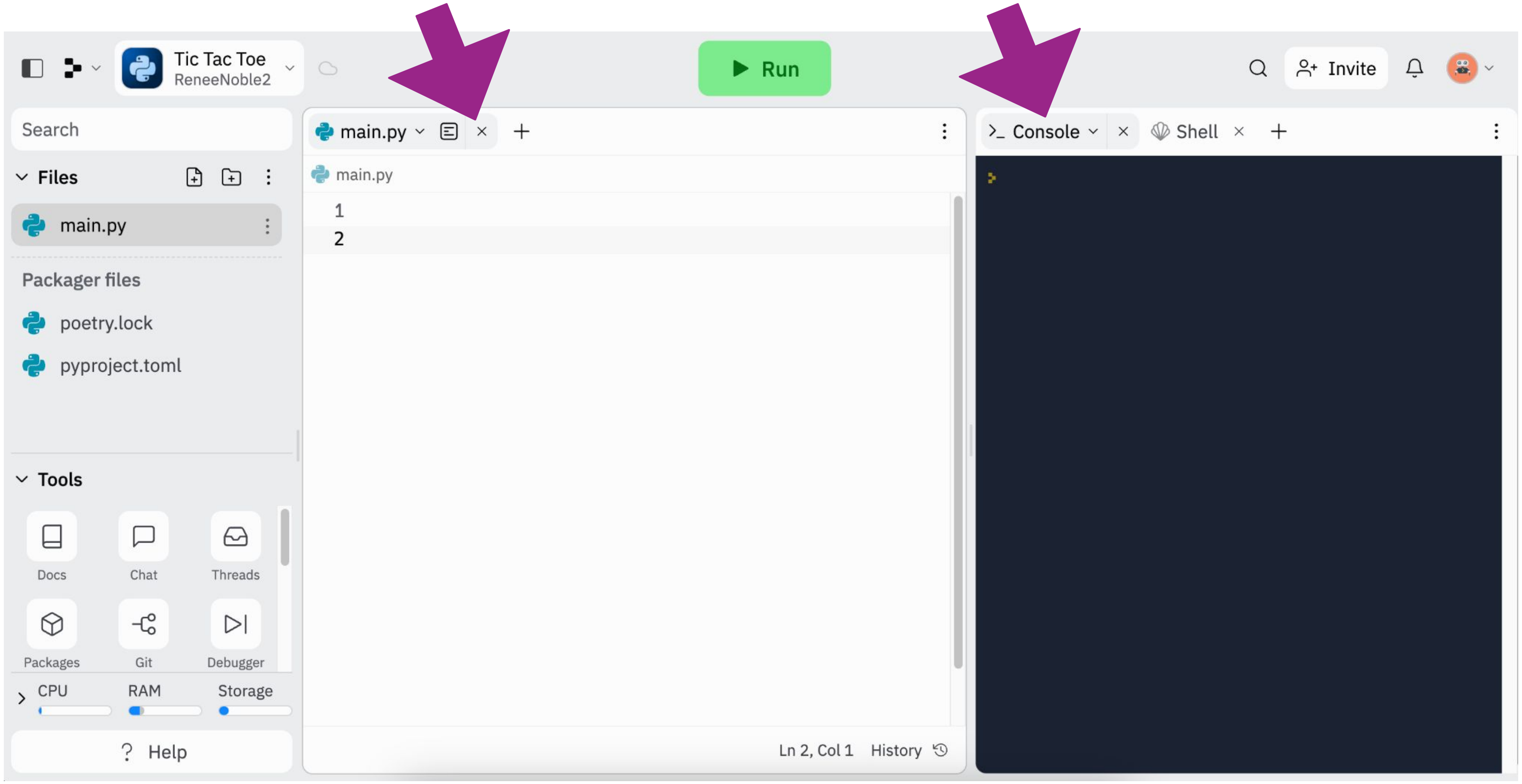
Title: Project name here

Public — Anyone can view and fork this Repl.

We’re ready to code!

We’ll write our project here in main.py

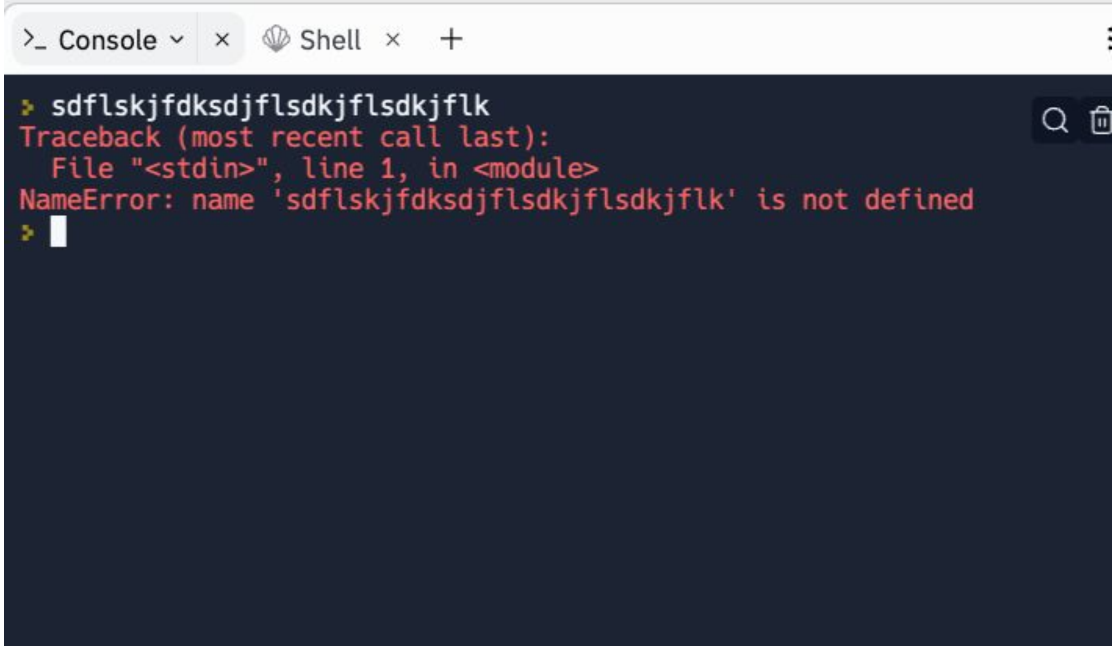
You can test out Python code in the console



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

Type by **button mashing** the keyboard!

Then press enter!



```
> 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?

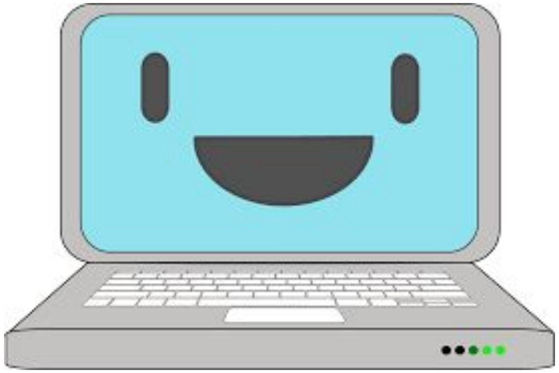
Mistakes are great!

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'

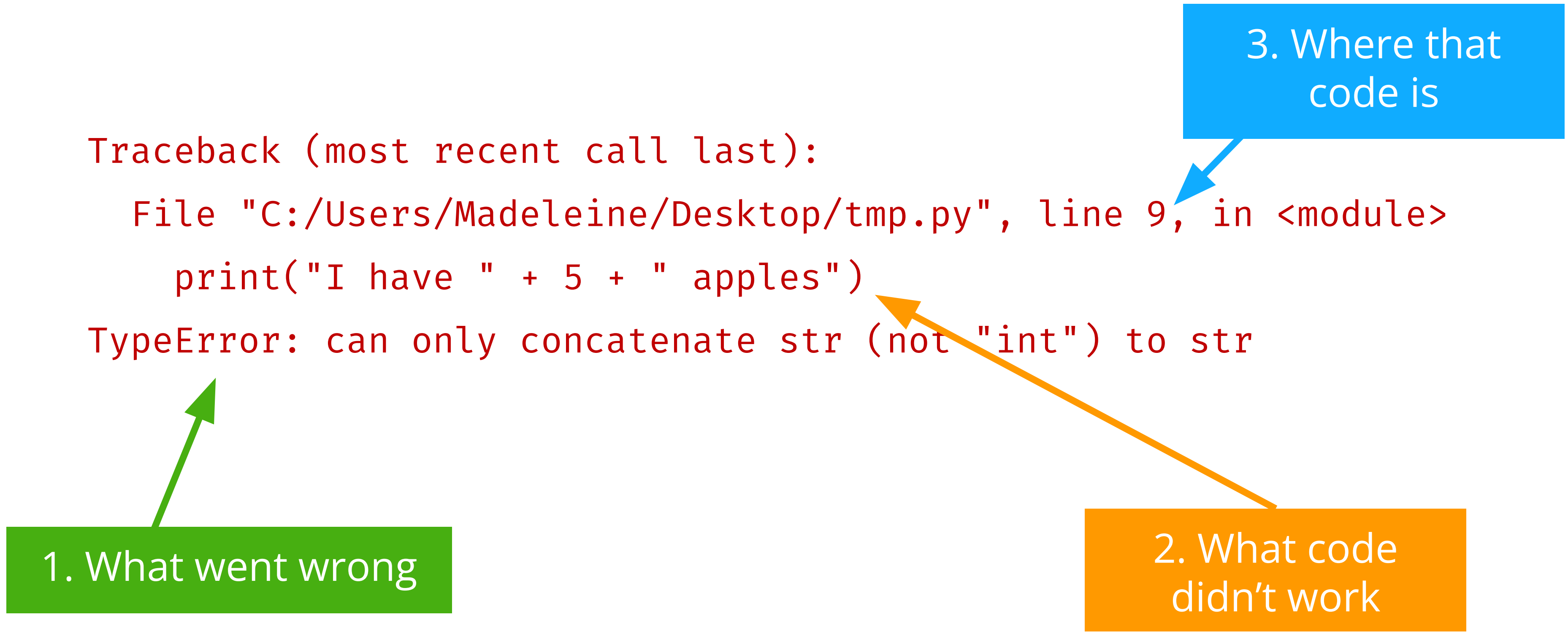
AttributeError:
'NoneType' object
has no attribute
'foo'

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



3. Where that code is

```
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

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?

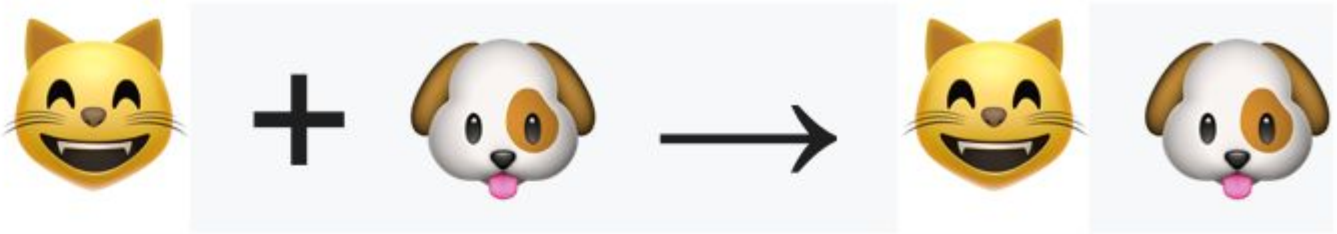
```
>>> "cat" + "dog"
catdog

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

Strings!

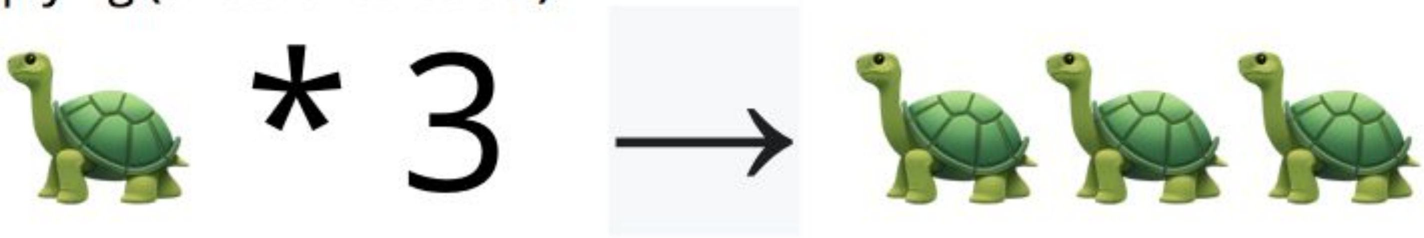
Strings are things with `"quotes"`

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()

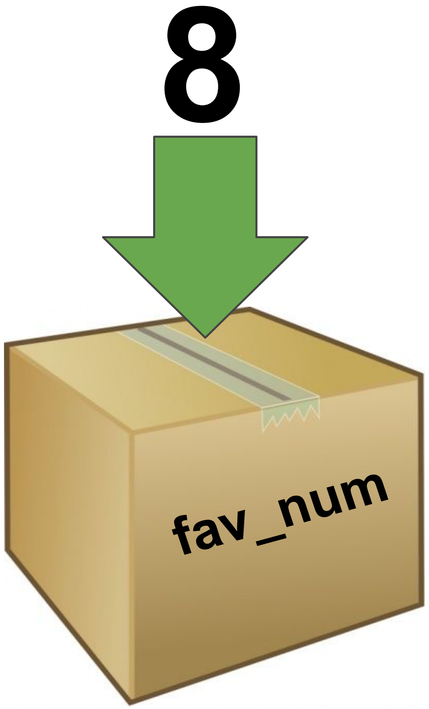
```
>>> str(5) + "5"
'55'
```

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.

```
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
```

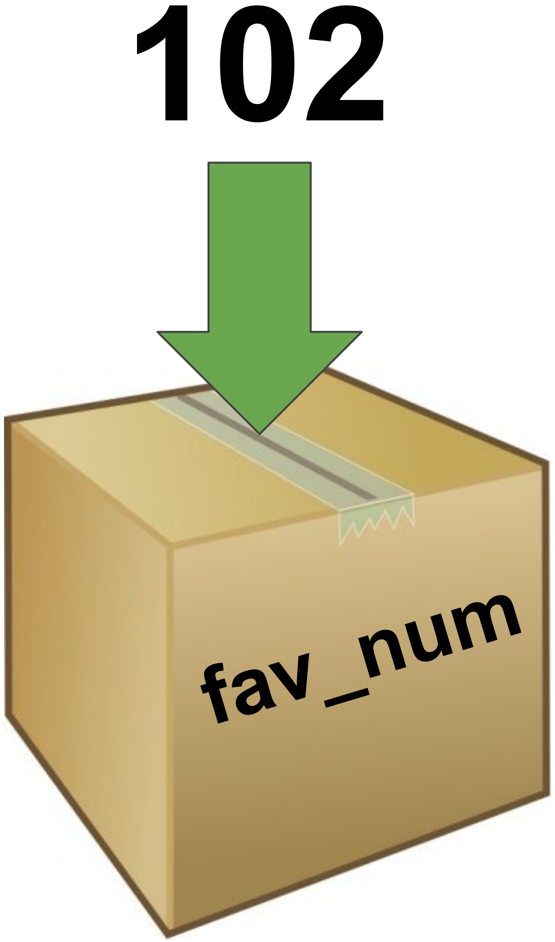
But writing 8 is much shorter than writing `fav_num`???

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
    => 96
```

```
fav_num + 21
    => 123
```

```
fav_num * 2?
    => 204
```

```
fav_num / 2?
    => 51
```

No variables VS using variables

4
Changes

1
Change

8 - 6

8 * 2

8 + 21

8 / 2

→

102 - 6

102 * 2

102 + 21

102 / 2

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:

```python
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
```

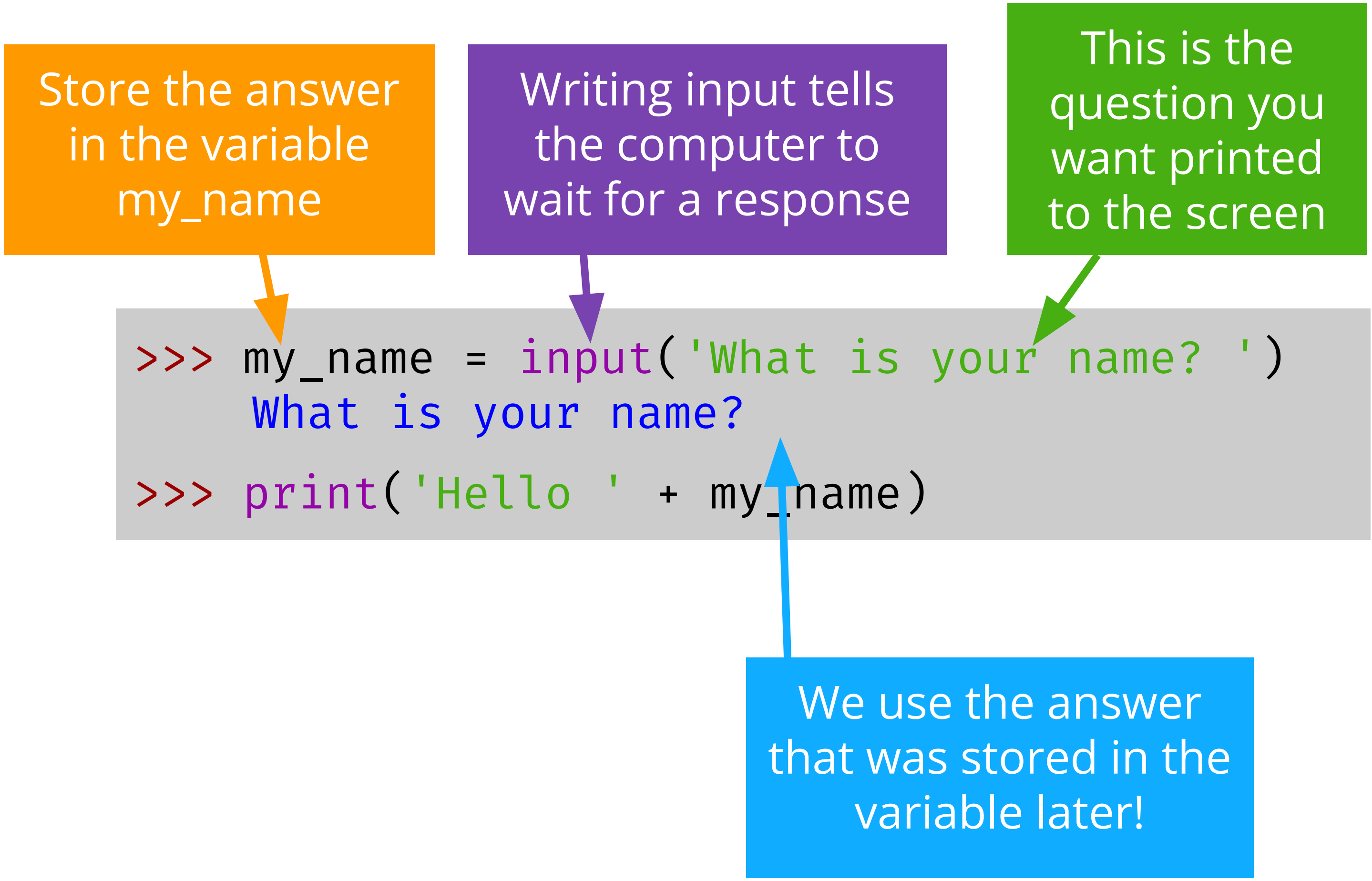
Asking a question!

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? ')
>>> 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'
>>> yum[5]


>>> yum[-1]


>>> yum[500]
```

Computers start counting from 0, not 1!

Strings

We can get individual letters from a string using indexes.

```
>>> yum = "chocolate"
>>> yum[0]
'c'
>>> yum[5]
'l'
>>> yum[-1]


>>> yum[500]
```

Computers start counting from 0, not 1!

Strings

We can get individual letters from a string using indexes.

```
>>> yum = "chocolate"
>>> yum[0]
'c'
>>> yum[5]
'l'
>>> yum[-1]
'e'
>>> yum[500]
```

Computers start counting from 0, not 1!

Strings

We can get individual letters from a string using indexes.

```
>>> yum = "chocolate"
>>> yum[0]
'c'
>>> yum[5]
'l'
>>> yum[-1]
'e'
>>> yum[500]
IndexError: string index out of range
```

Computers start counting from 0, not 1!

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

Only the index of the first ‘o’ is returned!

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
>>> yum.index('z')
ValueError: substring not found
```

Only the index of the first ‘o’ is returned!

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?

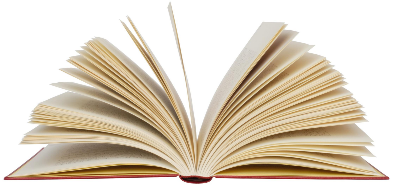
```
word = ‘cars’


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)
```

```
>>> c
```

Let’s set i to to the first character in the string!

`i` is now `’c’`

`print(i)`

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 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

Girls' Programming Network
Perth

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

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"
```

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

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")
```

<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

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

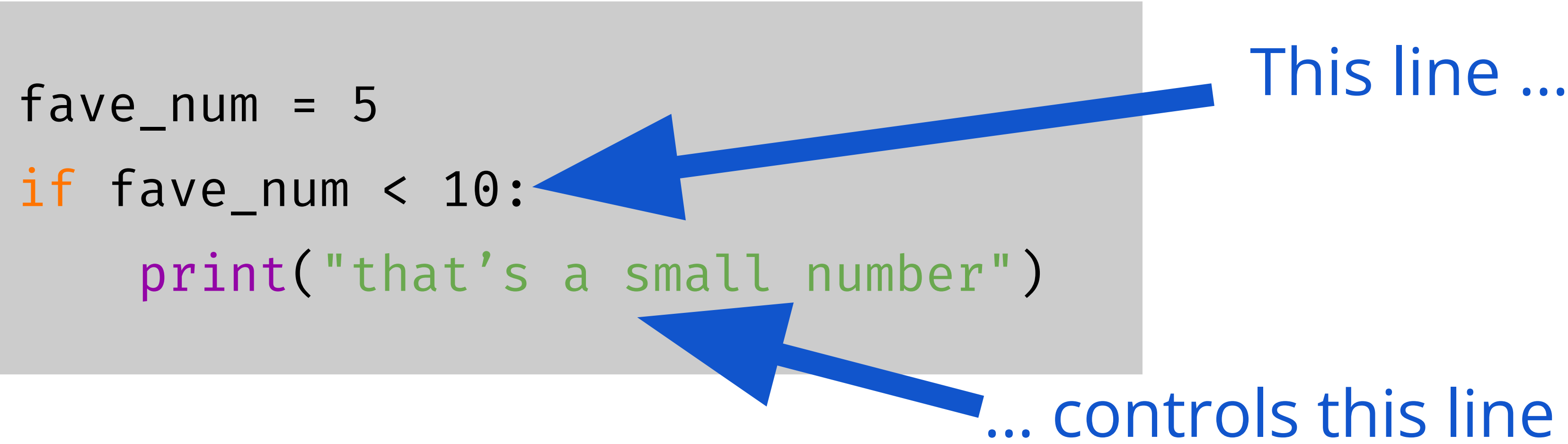
```
fave_num = 9000
if fave_num < 10:
    print("that’s a small number")
```

What do you think happens?

```
>>>
```

Nothing!

If statements



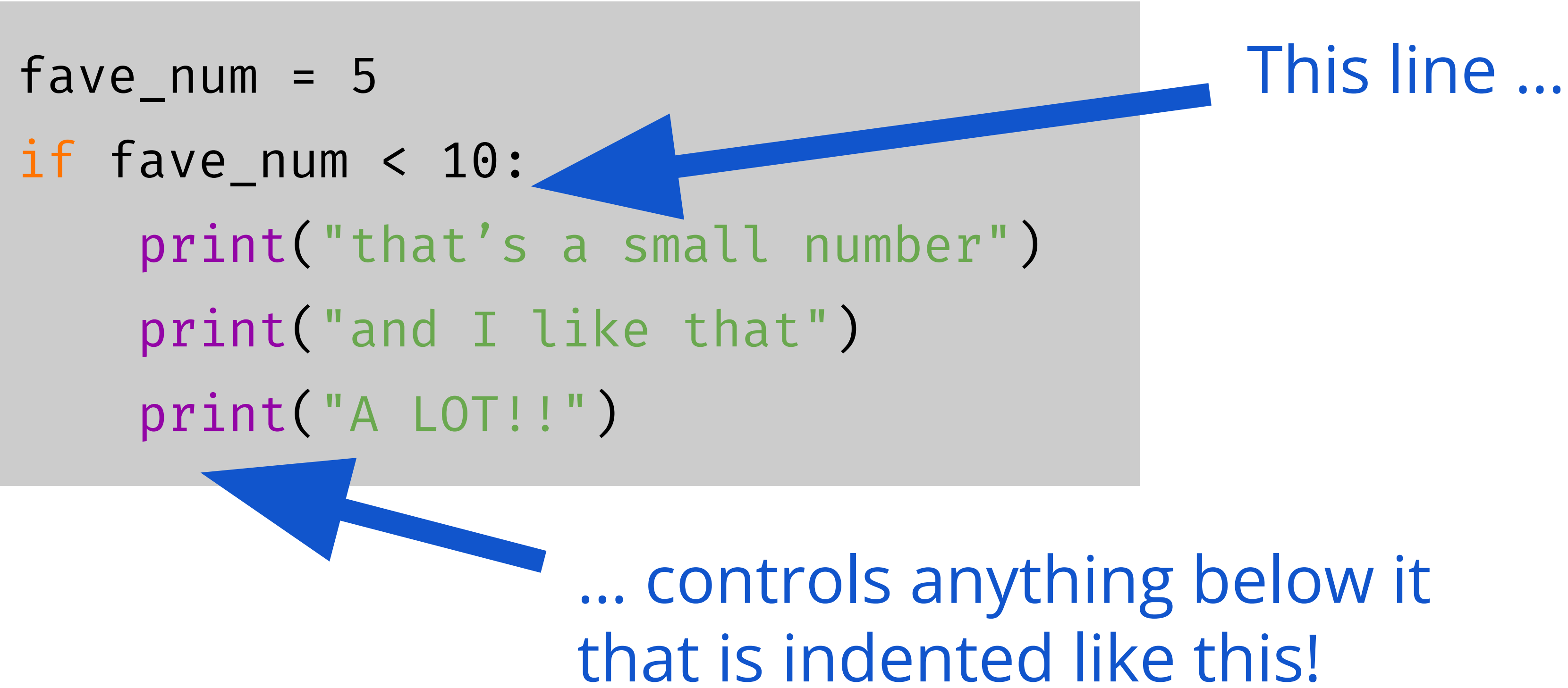
```
fave_num = 5
if fave_num < 10:
    print("that’s a small number")
```

This line ...

... controls this line

If statements

Actually .....



```python
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!!")
```

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

```
>>>
```

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 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 :(
```

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!

Intro to Vigenere Ciphers

Caesar Cipher

So now you know what a Caesar Cipher is, let's look at a more complicated cipher!

A Caesar Cipher uses just 1 key to encrypt and decrypt the message, a Vigenere cypher uses a whole word as the key!

The keyword

Let’s see how it uses a whole word by doing an example together!

Let’s use the keyword

pizza

Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number

(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys by replacing each letter with its number in the alphabet

p	i	z	z	a

Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number

(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys by replacing each letter with its number in the alphabet

p	**i**	**z**	**z**	**a**
15				

Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number

(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys by replacing each letter with its number in the alphabet

p	**i**	**z**	**z**	**a**
15	8			

Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number

(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys by replacing each letter with its number in the alphabet

p	**i**	**z**	**z**	**a**
15	8	25		

Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number

(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys by replacing each letter with its number in the alphabet

p	**i**	**z**	**z**	**a**
15	8	25	25	

Splitting it into keys

Now we take the keyword and we split it into a bunch of keys!

Each letter of the alphabet equals a different number

(a=0, b=1, c=2 etc.)

Now we change our keyword into a bunch of different keys by replacing each letter with its number in the alphabet

p	**i**	**z**	**z**	**a**
15	8	25	25	0

Loop the word

Let’s try encrypting a message with our keyword using a Vigenere cipher now!

I love coding

Each letter in our message will line up with a letter in our keyword and we will keep looping the keyword like this:

i	l	o	v	e	c	o	d	i	n	g
p	i	z	z	a	p	i	z	z	a	p

Using the numbers

Now we replace each letter of our keyword with the numbers that we worked out before:

i	l	o	v	e	c	o	d	i	n	g
15	8	25	25	0	15	8	25	25	0	15

Next we just shift each letter in our message like we do with a Caesar Cipher but with the key that it lines up with.

What key does the letter C use?

Using the numbers

Now we replace each letter of our keyword with the numbers that we worked out before:

i	l	o	v	e	c	o	d	i	n	g
15	8	25	25	0	15	8	25	25	0	15

Next we just shift each letter in our message like we do with a Caesar Cipher but with the key that it lines up with.

What key does the letter C use? 15

Making the secret message

i	Using key: **15**	Is replaced with
l	Using key: **8**	Is replaced with
o	Using key: **25**	Is replaced with
v	Using key: **25**	Is replaced with
e	Using key: **0**	Is replaced with
c	Using key: **15**	Is replaced with
o	Using key: **8**	Is replaced with
d	Using key: **25**	Is replaced with
i	Using key: **25**	Is replaced with
n	Using key: **0**	Is replaced with
g	Using key: **15**	Is replaced with

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	
o	Using key: **25**	Is replaced with	
v	Using key: **25**	Is replaced with	
e	Using key: **0**	Is replaced with	
c	Using key: **15**	Is replaced with	
o	Using key: **8**	Is replaced with	
d	Using key: **25**	Is replaced with	
i	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	
v	Using key: **25**	Is replaced with	
e	Using key: **0**	Is replaced with	
c	Using key: **15**	Is replaced with	
o	Using key: **8**	Is replaced with	
d	Using key: **25**	Is replaced with	
i	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	
e	Using key: **0**	Is replaced with	
c	Using key: **15**	Is replaced with	
o	Using key: **8**	Is replaced with	
d	Using key: **25**	Is replaced with	
i	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	u
e	Using key: **0**	Is replaced with	
c	Using key: **15**	Is replaced with	
o	Using key: **8**	Is replaced with	
d	Using key: **25**	Is replaced with	
i	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	u
e	Using key: **0**	Is replaced with	e
c	Using key: **15**	Is replaced with	
o	Using key: **8**	Is replaced with	
d	Using key: **25**	Is replaced with	
i	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	u
e	Using key: **0**	Is replaced with	e
c	Using key: **15**	Is replaced with	r
o	Using key: **8**	Is replaced with	
d	Using key: **25**	Is replaced with	
i	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	u
e	Using key: **0**	Is replaced with	e
c	Using key: **15**	Is replaced with	r
o	Using key: **8**	Is replaced with	w
d	Using key: **25**	Is replaced with	
i	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	u
e	Using key: **0**	Is replaced with	e
c	Using key: **15**	Is replaced with	r
o	Using key: **8**	Is replaced with	w
d	Using key: **25**	Is replaced with	c
i	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	u
e	Using key: **0**	Is replaced with	e
c	Using key: **15**	Is replaced with	r
o	Using key: **8**	Is replaced with	w
d	Using key: **25**	Is replaced with	c
i	Using key: **25**	Is replaced with	h
n	Using key: **0**	Is replaced with	
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	u
e	Using key: **0**	Is replaced with	e
c	Using key: **15**	Is replaced with	r
o	Using key: **8**	Is replaced with	w
d	Using key: **25**	Is replaced with	c
i	Using key: **25**	Is replaced with	h
n	Using key: **0**	Is replaced with	n
g	Using key: **15**	Is replaced with	

Making the secret message

i	Using key: **15**	Is replaced with	x
l	Using key: **8**	Is replaced with	t
o	Using key: **25**	Is replaced with	n
v	Using key: **25**	Is replaced with	u
e	Using key: **0**	Is replaced with	e
c	Using key: **15**	Is replaced with	r
o	Using key: **8**	Is replaced with	w
d	Using key: **25**	Is replaced with	c
i	Using key: **25**	Is replaced with	h
n	Using key: **0**	Is replaced with	n
g	Using key: **15**	Is replaced with	v

Secret Message

So our secret encrypted message is **x tnue rwchnv**

To decrypt it you do the same thing with each letter and key that you did to decrypt in the Caesar cipher

- change the key value to become the negative of the encryption key value
- turn the wheel backwards (clockwise) to undo the encryption and get the secret message
- this shifts the alphabet the opposite way to what we did to encrypt the message

Turn it back!

x	Using key: **15**	Is replaced with
t	Using key: **8**	Is replaced with
n	Using key: **25**	Is replaced with
u	Using key: **25**	Is replaced with
e	Using key: **0**	Is replaced with
r	Using key: **15**	Is replaced with
w	Using key: **8**	Is replaced with
c	Using key: **25**	Is replaced with
h	Using key: **25**	Is replaced with
n	Using key: **0**	Is replaced with
v	Using key: **15**	Is replaced with

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	
n	Using key: **25**	Is replaced with	
u	Using key: **25**	Is replaced with	
e	Using key: **0**	Is replaced with	
r	Using key: **15**	Is replaced with	
w	Using key: **8**	Is replaced with	
c	Using key: **25**	Is replaced with	
h	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	
u	Using key: **25**	Is replaced with	
e	Using key: **0**	Is replaced with	
r	Using key: **15**	Is replaced with	
w	Using key: **8**	Is replaced with	
c	Using key: **25**	Is replaced with	
h	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	
e	Using key: **0**	Is replaced with	
r	Using key: **15**	Is replaced with	
w	Using key: **8**	Is replaced with	
c	Using key: **25**	Is replaced with	
h	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	v
e	Using key: **0**	Is replaced with	
r	Using key: **15**	Is replaced with	
w	Using key: **8**	Is replaced with	
c	Using key: **25**	Is replaced with	
h	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	v
e	Using key: **0**	Is replaced with	e
r	Using key: **15**	Is replaced with	
w	Using key: **8**	Is replaced with	
c	Using key: **25**	Is replaced with	
h	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	v
e	Using key: **0**	Is replaced with	e
r	Using key: **15**	Is replaced with	c
w	Using key: **8**	Is replaced with	
c	Using key: **25**	Is replaced with	
h	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	v
e	Using key: **0**	Is replaced with	e
r	Using key: **15**	Is replaced with	c
w	Using key: **8**	Is replaced with	o
c	Using key: **25**	Is replaced with	
h	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	v
e	Using key: **0**	Is replaced with	e
r	Using key: **15**	Is replaced with	c
w	Using key: **8**	Is replaced with	o
c	Using key: **25**	Is replaced with	d
h	Using key: **25**	Is replaced with	
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	v
e	Using key: **0**	Is replaced with	e
r	Using key: **15**	Is replaced with	c
w	Using key: **8**	Is replaced with	o
c	Using key: **25**	Is replaced with	d
h	Using key: **25**	Is replaced with	i
n	Using key: **0**	Is replaced with	
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	v
e	Using key: **0**	Is replaced with	e
r	Using key: **15**	Is replaced with	c
w	Using key: **8**	Is replaced with	o
c	Using key: **25**	Is replaced with	d
h	Using key: **25**	Is replaced with	i
n	Using key: **0**	Is replaced with	n
v	Using key: **15**	Is replaced with	

Turn it back!

x	Using key: **15**	Is replaced with	i
t	Using key: **8**	Is replaced with	l
n	Using key: **25**	Is replaced with	o
u	Using key: **25**	Is replaced with	v
e	Using key: **0**	Is replaced with	e
r	Using key: **15**	Is replaced with	c
w	Using key: **8**	Is replaced with	o
c	Using key: **25**	Is replaced with	d
h	Using key: **25**	Is replaced with	i
n	Using key: **0**	Is replaced with	n
v	Using key: **15**	Is replaced with	g

Your Turn!

Now you try on your own!

Try doing Part 0 - Part 1 of the second workbook!

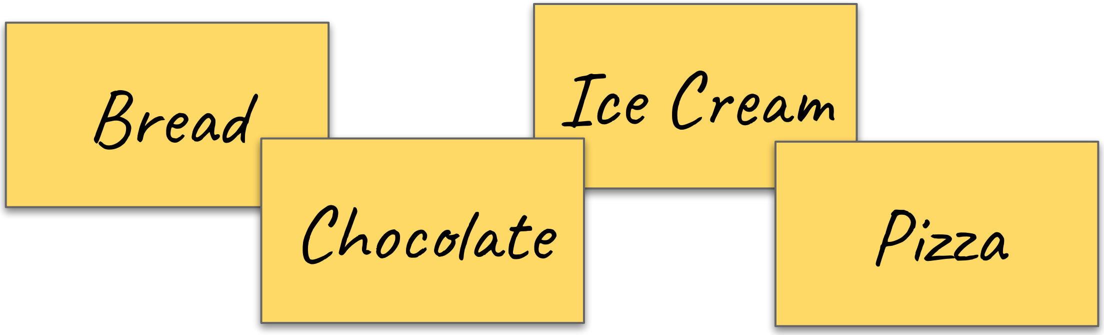
Your tutors are here to help you if you get stuck

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!



Bread

Ice Cream

Chocolate

Pizza

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

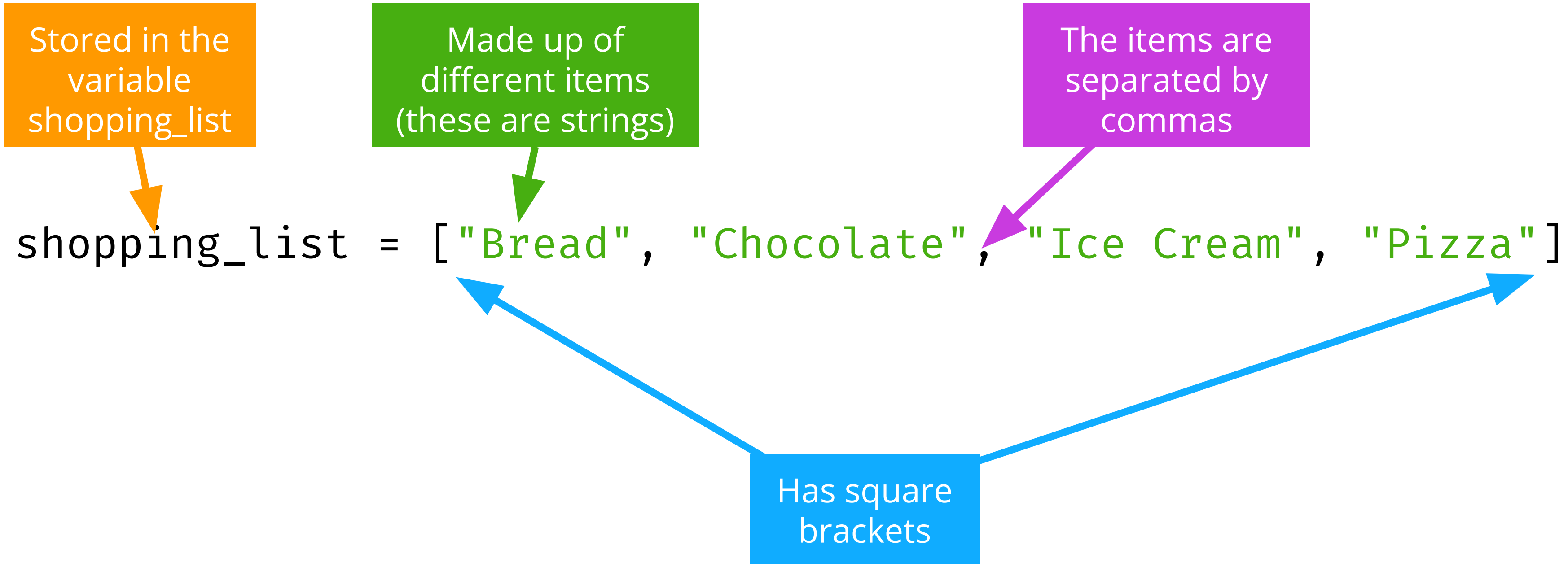
  ```
  >>> primes = [1, 2, 3, 5, 11]
  ```

- You can have a list of strings

  ```
  >>> mixture = ["one", "two", "three"]
  ```

- Every element of a list should be the same (eg integer, string). You should be able to treat every element of the list the same way.

List anatomy



Stored in the variable shopping_list

Made up of different items (these are strings)

The items are separated by commas

```
shopping_list = ["Bread", "Chocolate", "Ice Cream", "Pizza"]
```

Has square brackets

Accessing Lists!

Make a list of your favourite things

```
faves = ['books', 'butterfly', 'chocolate', 'skateboard']
```

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?

Going Negative

Negative indices count backwards from the end of the `list`

```
>>> faves = ['books', 'butterfly', 'chocolate',
'skateboard']
>>> faves[-1]
'skateboard'
```

What would `faves[-3]` return?

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
```

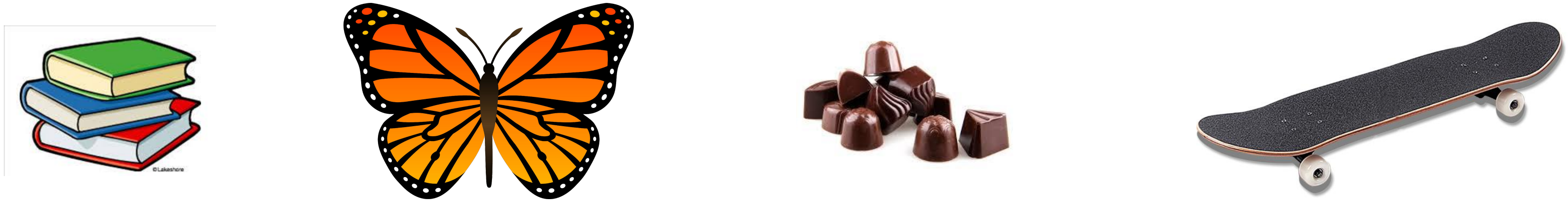
Updating items!

We can also update things in a list:

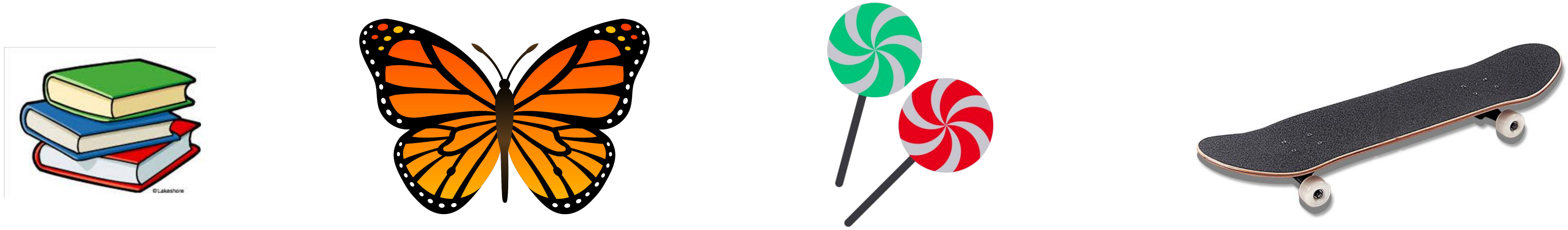
```
>>> faves = ['books', 'butterfly',
'chocolate', 'skateboard']
>>> faves[1]
'butterfly'
>>> faves[1] = 'kittens'
>>> faves[1]
'kittens'
```

Updating items

What if we decided that we didn't like chocolate anymore, but loved lollipops?



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.remove('butterfly')
```

```
What does this list look like now?
```

Adding items!

We can also add new items to the list!

What if we decided that we also liked programming?

```
>>> faves.append('programming')
```

What does this list look like now?

What can you do with a list?

- Define an empty list to add to in your code

```
>>> songs = []
```

- Loop through a list

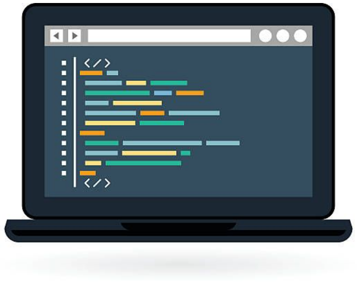
```
>>> odd_numbers = [1, 3, 5, 7]
>>> for i in odd_numbers:
        print(i)
```

Looping through a list

We can use a for statement to loop through a list

What if we wanted to print out all our favourites?

```
>>> for object in faves:
        print('I like ' + object)
'books'
'lollipops'
'skateboard'
'programming'
```



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]
>>> first_pair
["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]
>>> first_player
"Alex"
```

Project time!

You now know all about lists!

Let's put what we learnt into our project.

Try to do Part 2 of the second workbook!

The tutors will be around to help!

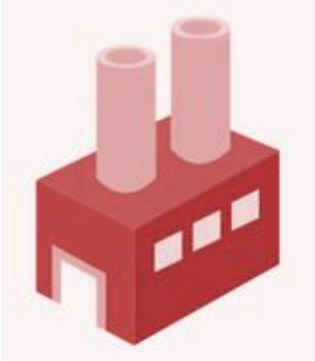
Functions!

Simpler, less repetition, easier to read code!

How functions fit together!

Functions are like factories!

Metal Worker



Your main factory!

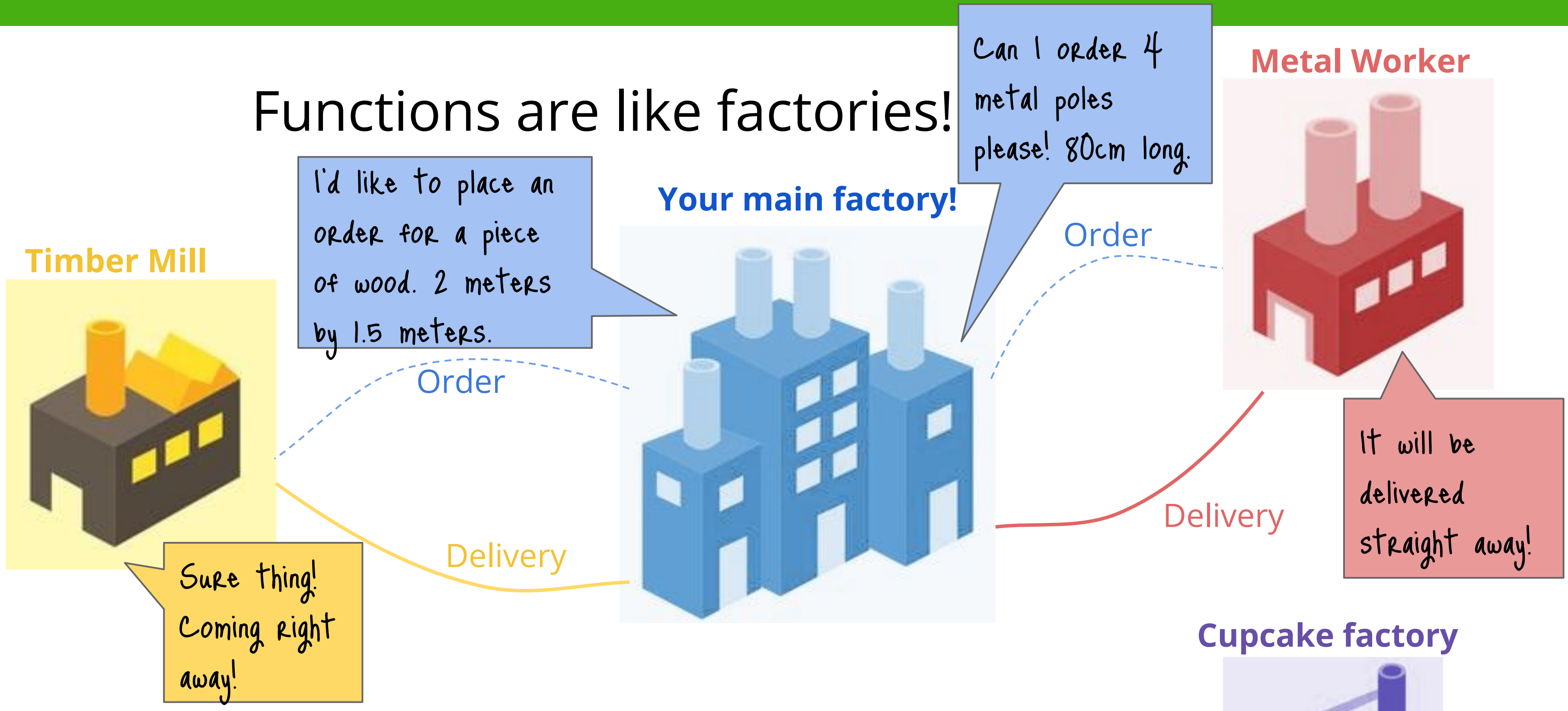
Timber Mill

Cupcake factory

Running a factory doesn’t mean doing all the work yourself, you can get other factories to help you out!

How functions fit together!

Functions are like factories!



Timber Mill

Your main factory!

Metal Worker

Order

Delivery

Order

Delivery

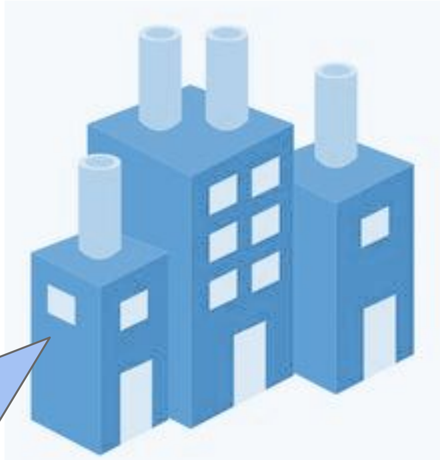
I'd like to place an order for a piece of wood. 2 meters by 1.5 meters.
Sure thing! Coming right away!
Can I order 4 metal poles please! 80cm long.
It will be delivered straight away!

Cupcake factory

Asking other factories to do some work for you makes your main task simper. You can focus on the assembly!

How functions fit together!

Functions are like factories!



Look at this beautiful table I made!

Outsourcing made it simple!

How functions fit together!

Your main code!

You can write a bunch of helpful functions to **simplify** your **main goal**!

You can **write** these **once** and then **use** them **lots** of times! They can be **anything** you like!

Helps with printing nicely

Uses stats to make decisions

Does calculations

Don’t reinvent the wheel

We’re already familiar with some python in built functions like print and input!

There’s lots of functions python gives us to save us reinventing the wheel!

For instance we can use len to get the length of a string, rather than having to write code to count every letter!

```
>>> len("Hello world")
11
```

Try these:

```
>>> name = "Renee"
>>> len(name)
5

>>> int("6")
6

>>> str(6)
"6"
```

Defining your own functions

Built in functions are great! But sometimes we want custom functions!

Defining our own functions means:

- We cut down on repeated code
- Nice function names makes our code clear and easy to read
- We can move bulky code out of the way

Defining your own functions

Then you can use your function by calling it!

```
def cat_print():
    print("""
                      #
                       #
                       #
               ^..^ #####
               =TT=      ;
                #########
                # #   # #
                M M   M M """)


cat_print()
cat_print()
```

Which will do this!

```
       #
        #
        #
^..^ #####
=TT=      ;
 #########
 # #   # #
 M M   M M
       #
        #
        #
^..^ #####
=TT=      ;
 #########
 # #   # #
 M M   M M
```

Defining your own functions

Then you can use your function by calling it!

```
def cat_print():
    print("""
                      #
                       #
                       #
             ^..^ #####
             =TT=      ;
              #########
              # #   # #
              M M   M M """)



cat_print()
cat_print()
```

Which will do this!

```
         #
          #
          #
^..^ #####
=TT=      ;
 #########
 # #   # #
 M M   M M
         #
          #
          #
^..^ #####
=TT=      ;
 #########
 # #   # #
 M M   M M
```

When using a function in a **script** make sure you define the function first.

It doesn’t matter if you call it from inside another function though!

Pretty Word Printer

Create a new file and make a pretty word printer! It can print any word you like.

1. Define a function called pretty_word_print

2. Set a variable called word

3. Have the function print out some decorative marks as long as the word above and below the word like these examples:

```
~~~
GPN
~~~
```

```
***********
Hello World
***********
```

4. Call your function in your file as many times as you like!

Functions often need extra information

Functions are more useful if we can change what they do

We can do this by giving them arguments (aka parameters)

```
>>> def hello(person):
      print('Hello, ' + person + ', how are you?')
>>> hello('Alex')
Hello, Alex, how are you?
```

Here, we give the hello() function a name

Any string will work

```
>>> hello('abcd')
Hello, abcd, how are you?
```

Functions can take multiple arguments

Often we want to work with multiple pieces of information.

You can actually have as many parameters as you like!

This function takes two numbers, adds them together and prints the result.

```
>>> def add(x, y):
        print(x + y)
>>> add(3, 4)
7
```

Arguments stay inside the function

The arguments are not able to be accessed outside of the function declaration.

```
>>> def hello(person):
        print('Hello, ' + person + '!')
>>> print(person)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'person' is not defined
```

Variables stay inside the function

Neither are variables declared inside the function. They are **local variables**.

```
>>> def add(x, y):
      z = x + y
      print(z)
>>> add(3, 4)
7
>>> z
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'z' is not defined
```

Global variables are not affected

Changing a variable in a function **only changes it *inside* the function.**

```
>>> z = 1
>>> def add(x, y):
        z = x + y
        print(z)
>>> add(3, 4)
7
```

Global variables are not affected

Changing a variable in a function **only changes it *inside* the function.**

```
>>> z = 1
>>> def add(x, y):
        z = x + y
        print(z)
>>> add(3, 4)
7
```

What's the value of z now?

```
>>> print(z)
```

Global variables are not affected

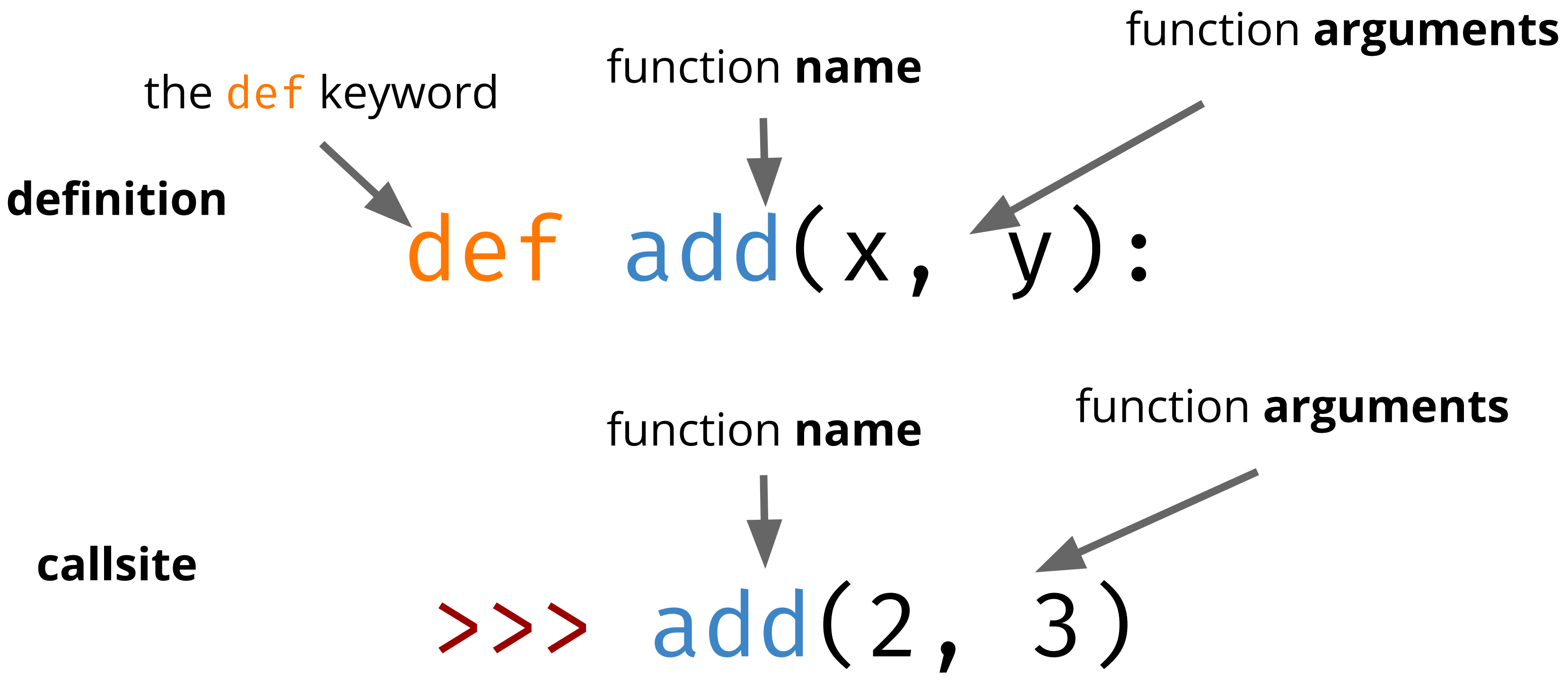
Changing a variable in a function **only changes it *inside* the function.**

```
>>> z = 1
>>> def add(x, y):
        z = x + y
        print(z)
>>> add(3, 4)
7
```

What's the value of z now?

```
>>> print(z)
1
```

Recap: A function signature



the def keyword

function **name**

function **arguments**

definition

```
def add(x, y):
```

function **name**

function **arguments**

callsite

```
>>> add(2, 3)
```

Pretty Word Printer

At the moment our pretty word printer always prints the same word. Let’s fix that!

Edit your pretty word printer function:

1. Change your function so it takes in an argument called word
2. Remove the line where you set word as a variable, now we are passing in word
3. Change the places where you called your pretty word printer, so now you pass in a word as an argument (make sure you pass in a string).
4. Try calling your function multiple times, but with different words

Calling your function with these arguments might look like this:

```
pretty_word_print("Hi everyone")
pretty_word_print("Coding is cool")
```

```
***********
Hi everyone
***********
**************
Coding is cool
**************
```

Giving something back

At the moment our function just does a thing, but it‘s not able to give anything back to the main program.

Currently, we can’t use the result of add()

```
>>> def add(x, y):
        print(x + y)
>>> sum = add(1, 3)
4
>>> sum
```

sum has no value!

Giving something back

Using `return` in a function immediately returns a result.

```
>>> def add(x, y):
...     z = x + y
...     return z

>>> sum = add(1, 3)
>>> sum
4
```

Giving something back

When a function returns something, the *control* is passed back to the main program, so no code after the `return` statement is run.

```
>>> def add(x, y):
        print('before the return')
        z = x + y
        return z
        print('after the return')
>>> sum = add(1, 3)
before the return
>>> sum
4
```

Here, the `print` statement after the `return` never gets run.

Project time!

Now you know how to build function!

Now try to do Part 3 - Part 6 of the second workbook!

The tutors will be around to help!