

Welcome to GPN

Follow the URL and fill out the
Start of Day Survey!



www.girlsprogramming.network/perth-workshop

What is GPN?



- Girls' Programming Network
- For girls, by girls
- All over Australia



Who are the tutors?

- Name
- What do you do?
- How many times have you been to GPN?
- What do you love the most about GPN?
- What's your favourite dessert?



Who are you?

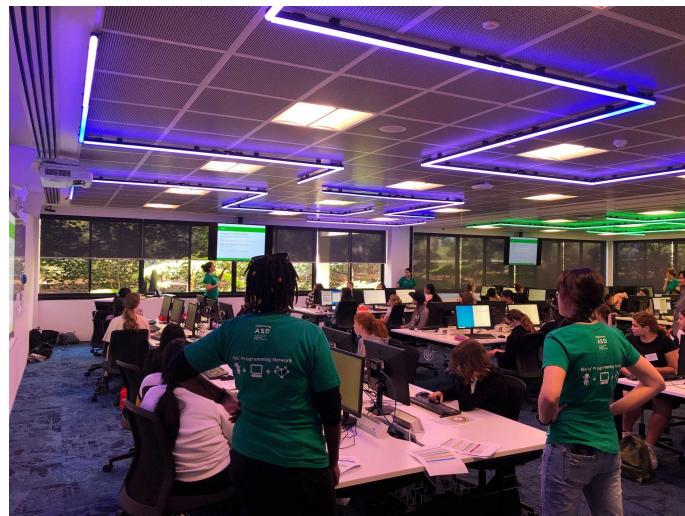
- Name
- What year are you in?
- How many times have you been to GPN?
- What do you love the most about GPN?
- What's your favourite dessert?



Introduce your friends!

Make a group of three or four preferably people you don't know in the room!

You have 3 minutes to learn three cool facts about each of them!



Log on

Log on and jump on the GPN website

 www.girlsprogramming.network/perth-workshop

You can see:

- These **slides** (to take a look back or go on ahead).
- A digital copy of your **workbook**.
- Helpful 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 Form
and fill it in now!



Today's project!

Markov Chains!



What is a Markov Chain?

A Markov chain is a simple Artificial Intelligence!

Let's play a game with some cups to help explain it



Let's play the cups game!

Let's generate some text in the style of
Green Eggs & Ham by Dr Seuss

Do you like green eggs and ham?

I do not like them, Sam-I-am.

I do not like green eggs and ham.

Would you like them here or there?

I would not like them here or there.

I would not like them anywhere.



Let's play the cups game!

- Each cup is labelled with a word from Green Eggs and Ham
- Each cup contains the words that follow the "label" word in Green Eggs and Ham

Let's play the cups game!

Read the outside of your cup!

If you hear someone shout the word on the outside of your cup:

1. Pick a piece of paper from inside your cup
2. Shout out the word on the piece of paper
3. Put the piece of paper back in your cup



Today we'll be making Markov Chains!

Markov chains are exactly what we just did with the cups!

Today we'll make the computer do it too to make some crazy stories!!

Here's one we made from some Shakespeare!

doth stay! All days when I compare thee to unseeing eyes be blessed made By chance, or eyes can see, For all the top of happy show thee in dark directed. Then thou, whose shadow shadows doth stay! All days when I compare thee in your self in inward worth nor outward fair, Can make bright, How would thy shade Through heavy sleep on the eye of life repair, Which this, Time's pencil, or my pupil pen, Neither in the living day, When in eternal lines of that fair from fair thou grow'st, So should the lines to a summer's day?



Imagine if you used one of these to do your homework!!

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.

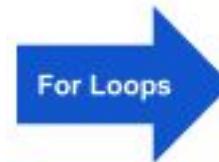


CHECKPOINT



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!

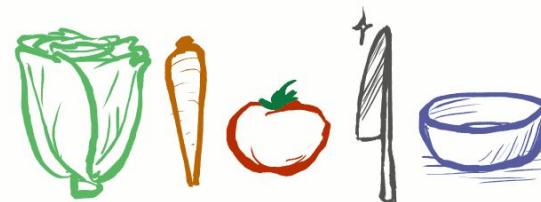
Programming is like a recipe!

Computers do EXACTLY what you say, every time.

Which is great if you give them a good recipe!

SALAD INSTRUCTIONS

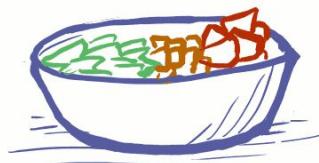
1) GET A LETTUCE HEAD, A CARROT, A TOMATO, A KNIFE, AND A BOWL



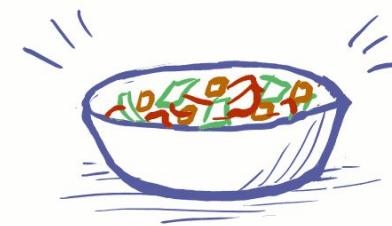
2) USE THE KNIFE TO CUT UP THE LETTUCE HEAD, CARROT, AND TOMATO



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



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

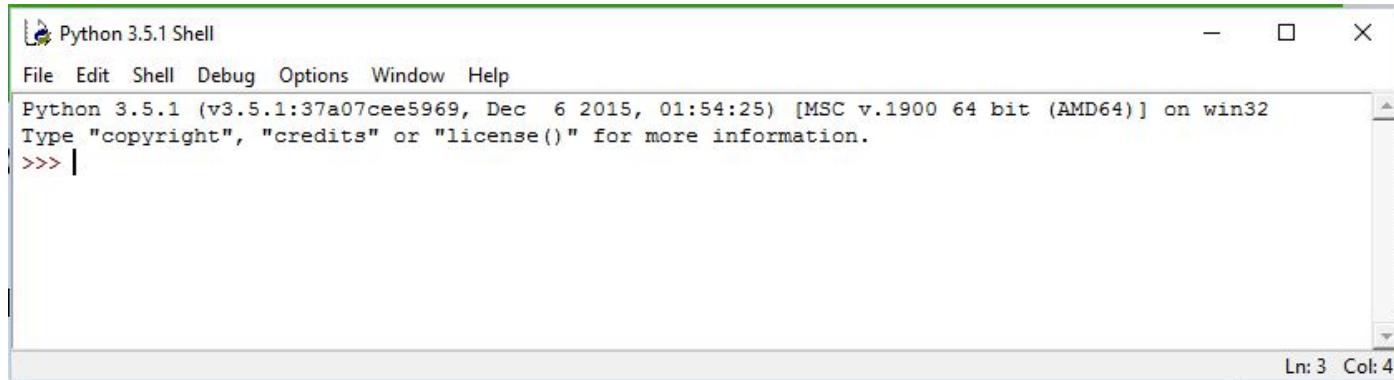
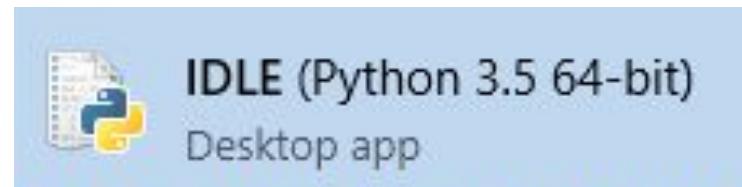
Let's get coding!



Where do we program? In IDLE

Click the start button and type IDLE!

Make sure you click one that says **Python 3.x**



Make a mistake!

Type by **button mashing** the keyboard!

Then press enter!

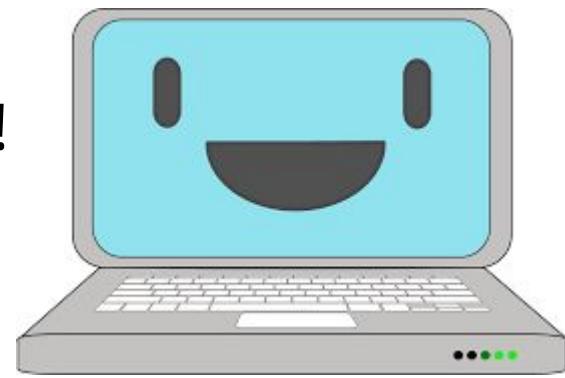
asdf asdj1kj;pa j;k4uroei

Did you get a big red error message?

Mistakes are great!

Good work you made an error!

- 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!



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

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

**KeyError:
'Hairy Potter,'**

**SyntaxError:
Invalid Syntax**

**ImportError:
No module
named humour**

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

Write some code!!

This is the first bit of code we will do. What do you think it does?

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

It prints the words “hello world” onto the screen!



Tell me more!

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

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

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

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

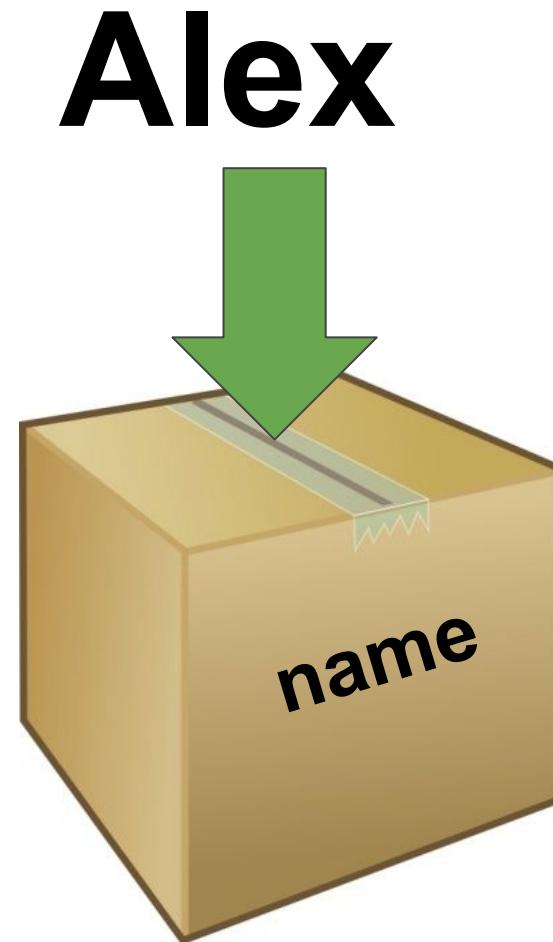


Variables

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

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

You can think of it like putting information in a box and giving it a name



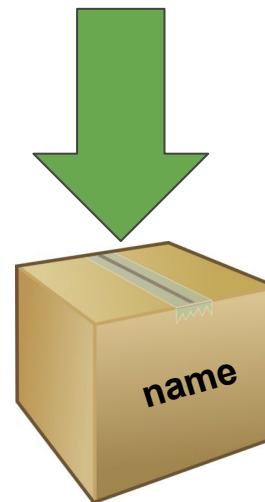
Variables

Instead of writing a name, we can use the name that is inside our variable! Here, we get the name out of the box.

```
print(name)
```

Alex

Alex



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?

My favourite animal is a dog
My favourite animal is a cat
My favourite animal is a catdog



Asking a question!

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

Let's get the computer to ask us a question!

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

What do you think happens?



Asking a question!

Store the answer
in the variable
my_name

Writing input tells
the computer to
wait for a response

This is the question
you want printed to
the screen

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

What do you think happens?

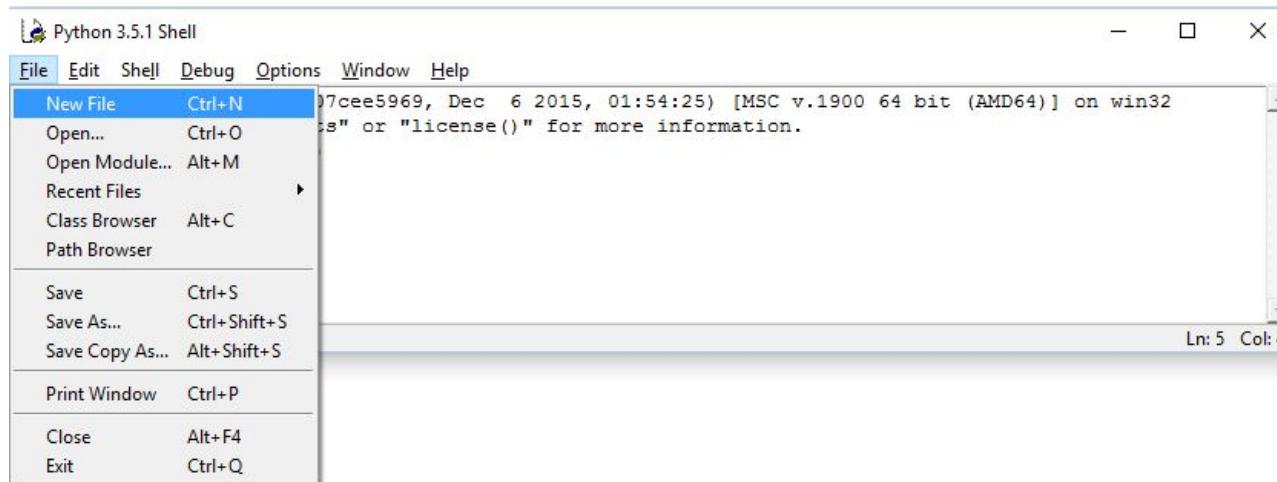
What is your name? Maddie

Hello Maddie

We can use the answer
the user wrote that we
then stored later!

Coding in a file!

Part 0 of your workbook is to create a new file. This is a picture of how to do it in IDLE



Name your file **markov_chains.py**

Adding a comment!

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

Use comments to write a note or explanation of our code
Comments make code easier for humans to understand

```
# This code was written by Sheree
```

We can make code into a comment if we don't want it to run (but don't want to delete it!)

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

Project time!

You now know all about printing, variables and input!

Let's put what we learnt into our project

Try to do Part 0 - Part 2

The tutors will be around to help!



If Statements and Lists



Conditions!

Conditions let us make decision.

First we test if the condition is met!

Then maybe we'll do the thing



If it's raining take an umbrella

Yep it's raining

..... take an umbrella

Booleans (True and False)

computers store whether a condition is met in the form
of
True and False

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

$5 < 10$	True	"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
- 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")
```

What do you think happens?

>>>



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")
```

What do you think happens?

>>> that's a small number



Conditions

How about a different number???

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



Conditions

Find out if it's **True**!

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

Put in the answer to the question

Is it **True** that fave_num is less than 10?

- Well, fave_num is 9000
- And it's not **True** that 9000 is less than 10
- So it is **False**!

Conditions

How about a different number???

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



What do you think happens?

>>>



Conditions

How about a different number???

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

What do you think happens?

>>>



Nothing!

If statements

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

This line ...

... controls this line

If statements

Actually

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

This line ...

... controls anything below it
that is indented like this!

If statements

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

What do you think happens?

>>>

If statements

```
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!")
```

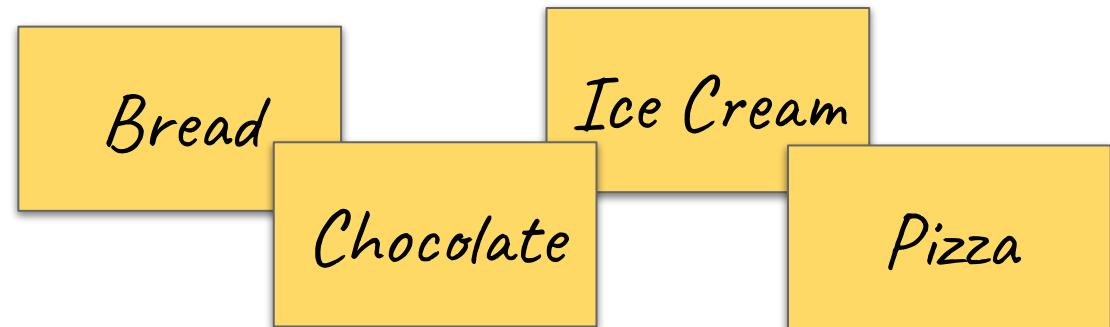
What happens?
>>> GPN is awesome!



Lists

When we go shopping, we write down what we want to buy!

But we don't store it on lots of little pieces of paper!



We put it in one big shopping list!

-
- The diagram shows a single large yellow rectangular box containing a bulleted list. The list items are: '● Bread', '● Chocolate', '● Ice Cream', and '● Pizza'. Each bullet point is preceded by a solid black circle.
- 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"]
```



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



Project Time!

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

See if you can do Part 3

The tutors will be around to help!



Random!

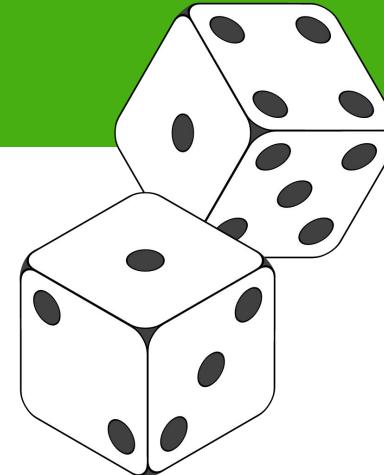


That's so random!

**There's lots of things in life that
are up to chance or random!**



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



**We want the computer to
be random sometimes!**



Using the random module

Let's choose something randomly from a list!

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

Here's an example!

1. Import the random module!

```
>>> import random
```



2. Copy the shopping list into IDLE

```
>>> shopping_list = ["eggs", "bread", "apples", "milk"]
```

3. Choose randomly! Try it a few times!

```
>>> random.choice(shopping_list)
```

Using the random module

You can also assign your random choice to a variable

```
>>> import random  
>>> shopping_list = ["eggs", "bread", "apples", "milk"]  
>>> random_food = random.choice(shopping_list)  
>>> print(random_food)
```



Project Time!

Raaaaaaaaandom! Can you handle that?

Let's try use it in our project!

Try to do Part 4

The tutors will be around to help!



For Loops

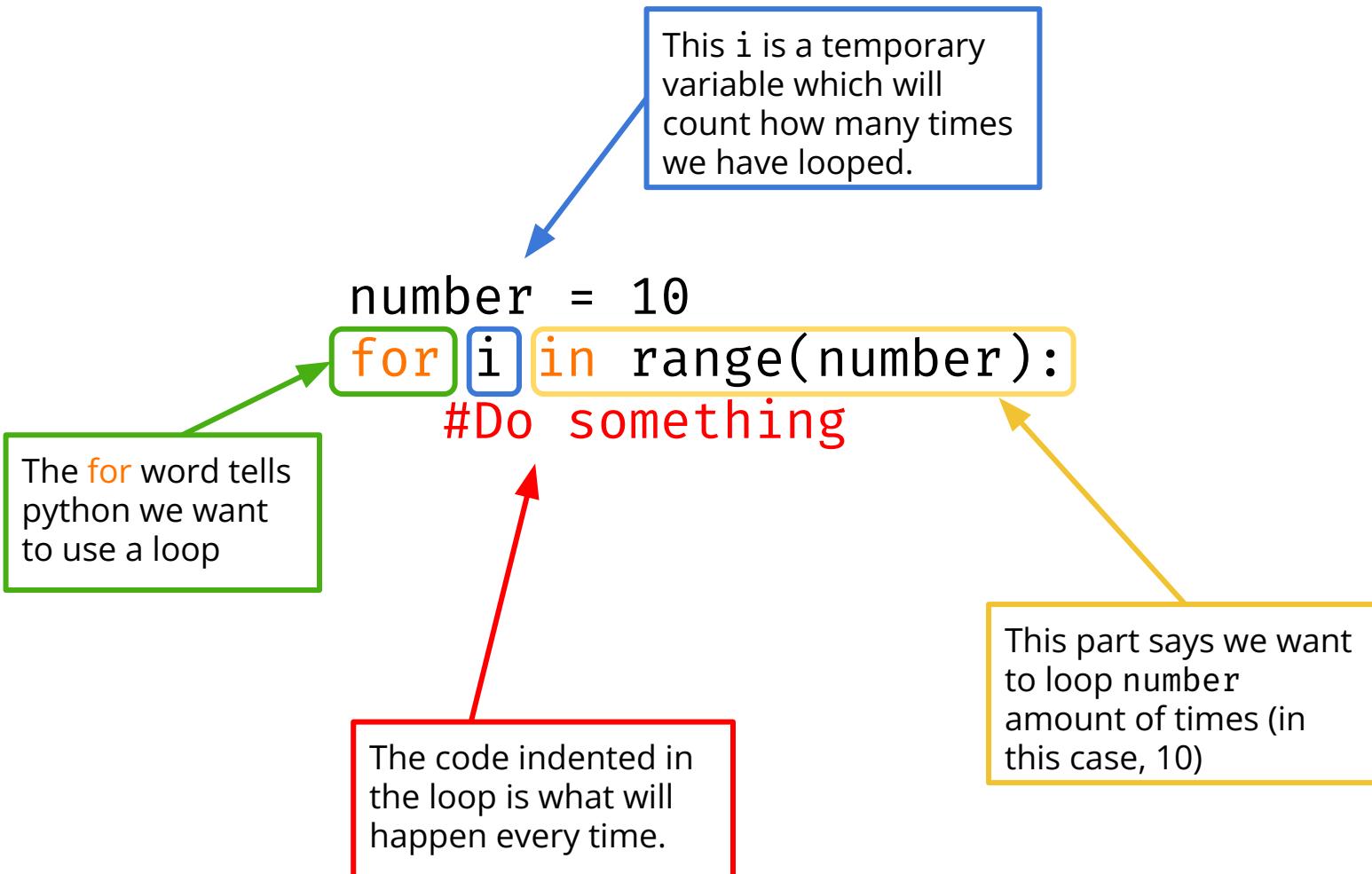


For Loops

For loops allow you to do something a certain number of times.

We use them when we know exactly how many times we want to do something!

For Loops



Looping how many times?

We can loop through a list:

```
friends = 4
for i in range(friends):
    print("Hello friend!")
```

What's going to happen?

Looping how many times?

We can loop through a list:

```
friends = 4  
for i in range(friends):  
    print("Hello friend!")
```

What's going to happen?

```
>>> Hello friend!  
>>> Hello friend!  
>>> Hello friend!  
>>> Hello friend!
```

We do what's in the for loop as many times as what is in the "range"

Project Time!

Now you know how to use a for loop!

Try to do Part 5
...if you are up **for it!**

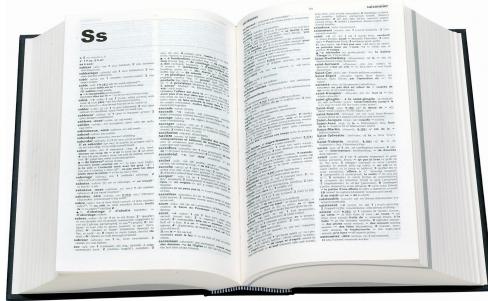
The tutors will be around to help!



Lists and Dictionaries



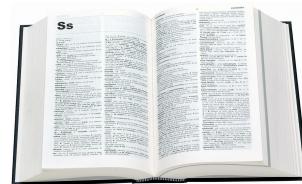
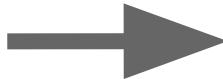
Dictionaries!



You know dictionaries!

They're great at looking up things by a word, not a position in a list!

Look up
Hello



Get back

A greeting (salutation) said when meeting someone or acknowledging someone's arrival or presence.

Looking it up!

There are lots of times we want to look something up!



Phone Book

Name → Phone number



Competition registration

Team Name → List of team members



Vending Machine

Treat Name → Price

Looking it up!



Phone Book

Name → Phone number

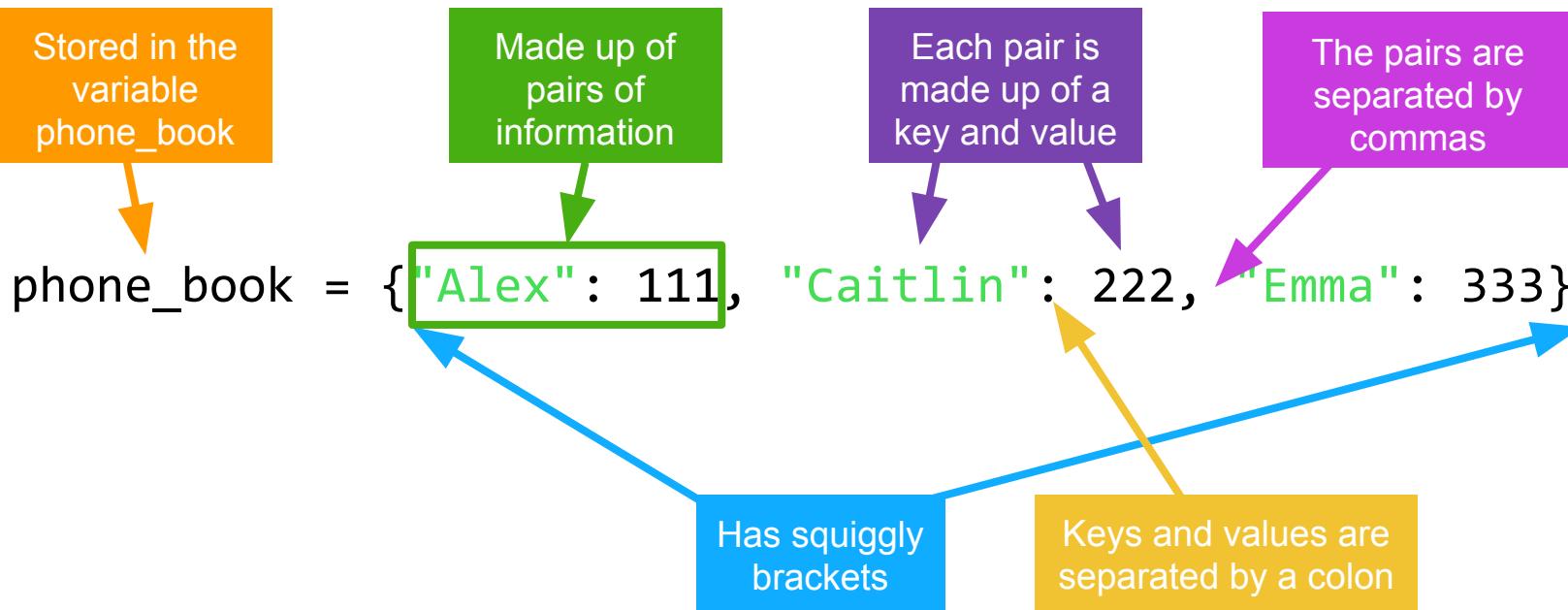
Key

Value

We can use a dictionary for anything with a
key → value pattern!

Dictionaries anatomy!

This is a python dictionary!



This dictionary has Alex, Caitlin and Emma's phone numbers

Playing with dictionaries!

Let's try using the phone book!

- Let's create the phonebook

```
>>> phone_book = {  
    "Alex": 111, "Caitlin": 222, "Emma": 333  
}
```

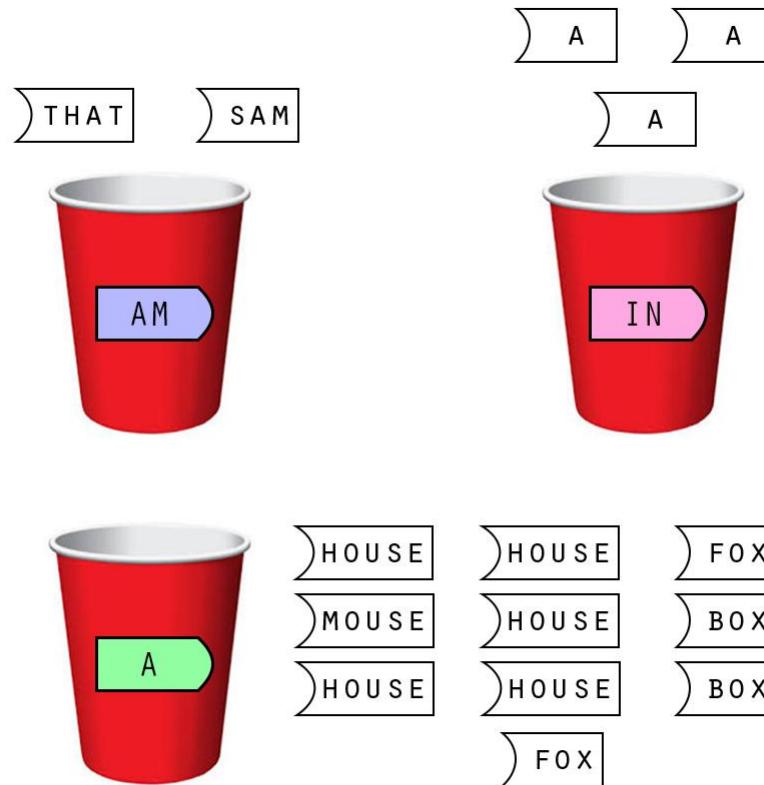
- Let's get Alex's number from the phonebook

```
>>> phone_book["Alex"]  
111
```



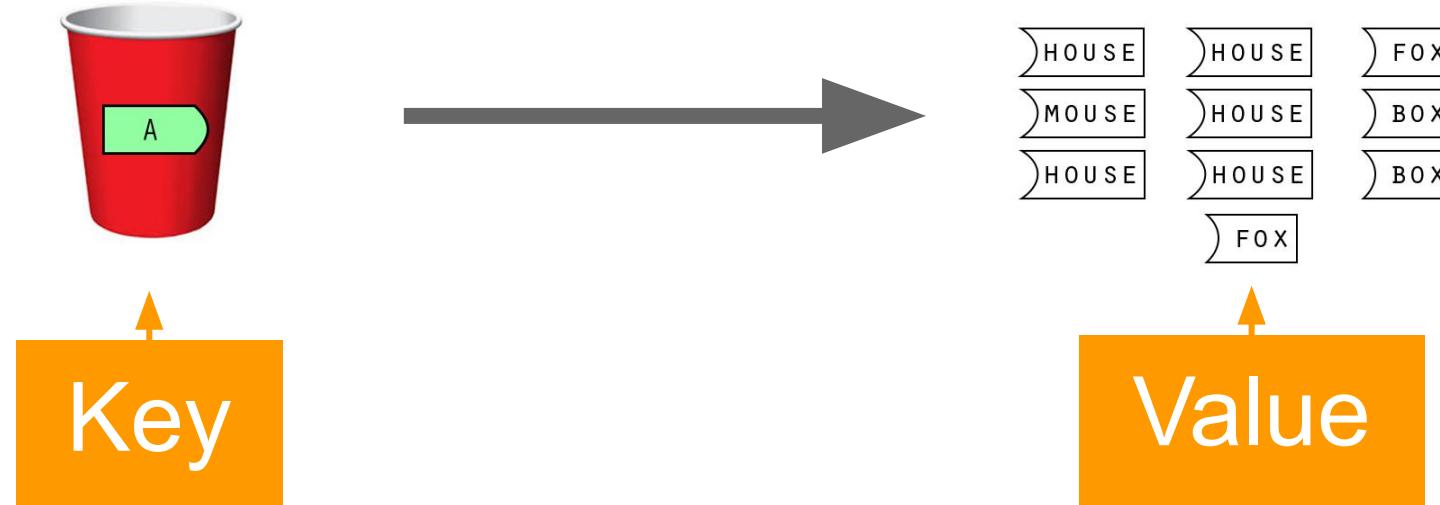
Cups!!

Remember the cups activity from the start of the day?



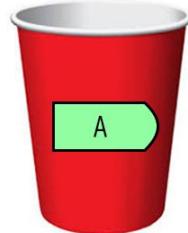
A Single Cup!

The word “A” can be followed by Any of these words



A Single Cup!

The word “A” can be followed by Any of these words



HOUSE	HOUSE	FOX
MOUSE	HOUSE	BOX
HOUSE	HOUSE	BOX
		FOX

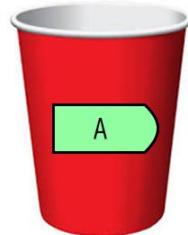
We can store the
slips of paper as
a python list!



```
[ 'house', 'mouse', 'house',
'mouse', 'box', 'fox', 'box',
'fox', 'house', 'mouse' ]
```

A Single Cup!

The word “A” can be followed by Any of these words



HOUSE	HOUSE	FOX
MOUSE	HOUSE	BOX
HOUSE	HOUSE	BOX
		FOX

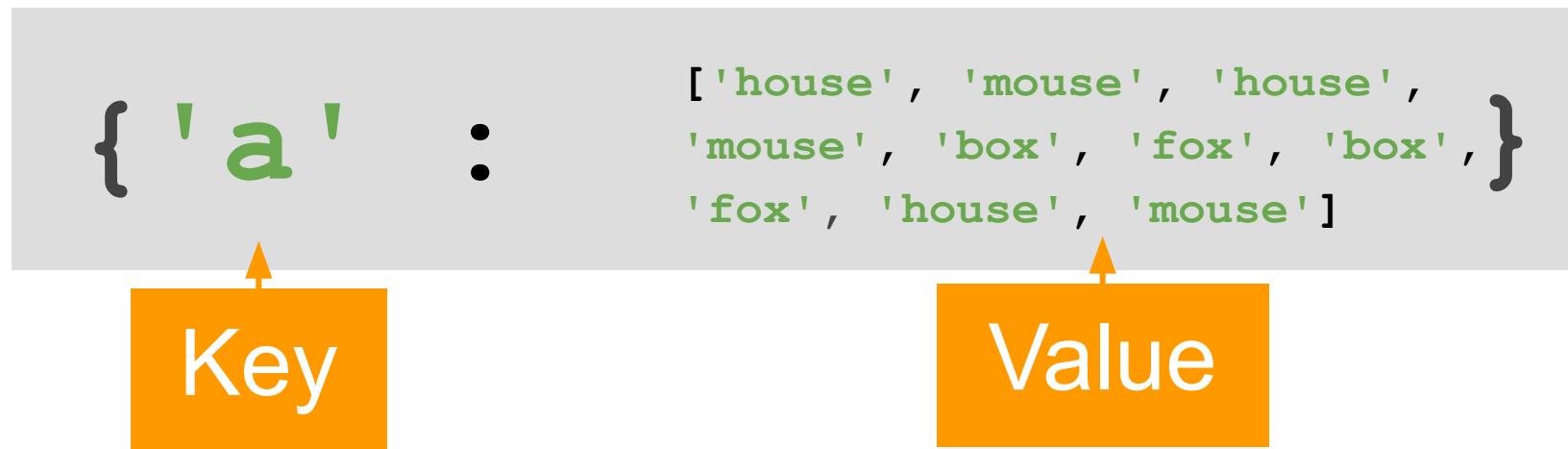
 We want to look up
the word “a” and get
back the list!

{ 'a' :

```
[ 'house', 'mouse', 'house',  
 'mouse', 'box', 'fox', 'box',  
 'fox', 'house', 'mouse' ]
```

A Single Cup!

So we get a Dictionary with a List value!

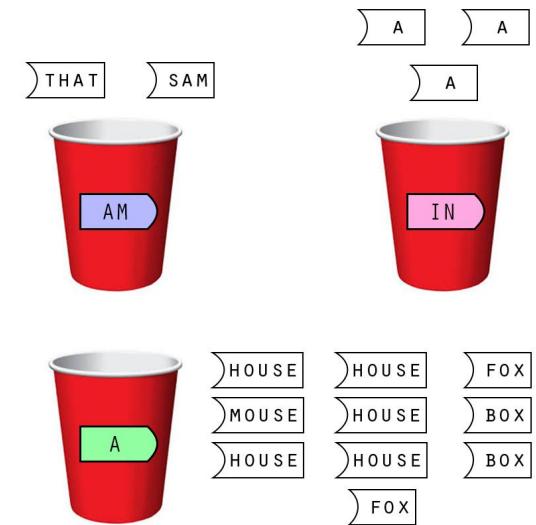


If you look up “A” you get back a list of all the words that can follow “a”

Cups → Dictionary with lists!

Here's what it looks like for a few more cups!

```
cup = { 'am': ['Sam', 'That'],
        'In': ['a', 'a', 'a'],
        'a' : ['house', 'mouse',
               'house', 'mouse',
               'box', 'fox', 'box',
               'fox', 'house',
               'Mouse'],
        .... }
```



You can get the whole cup dictionary from today's website!

Project time!

You now know all about lists and
dictionaries!

Let's put what we learnt into our project
Try to do Part 6

The tutors will be around to help!

