[Upbeat music]

[Isla] Hi. I'm Isla and you’re watching coding@home, a regular educational program that you can catch right here on 10 Peach during Term 2, and online.

Our mission is to show you just how fun and awesome coding can be.

In the last episode, we showed you how to start writing code using Python in IDLE.

We showed you how to receive a User Input and save it in a Variable.

We also learned the print() and the input()  statements, and to write a simple program that would print a personalised welcome  message when a User entered their name.

Then we got a bit fancy and used concatenation to merge words with variables to create an interactive message.

Sasha showed us how to use IF, ELIF and ELSE statements to create branches in our code.

We're going to use these again in this episode so here's a bit of a recap.

Remember we used a Variable to store the name we entered.

We think of Variables as a box we can store things in, and the label or name should tell us what's inside so we don’t get confused when our program gets bigger.

Printing out the message using our ’name' Variable was pretty easy too.

We used concatenation to combine the words, 'Hello', the Variable 'name’ and our message to the User.

All of this was a really good start in making an awesome chatbot that remembered our names, but to make it even better we're going to have to dig a little deeper into some of the techniques in Python that we can use to make our code easier to write, read, and maintain.

The first of these is called Repetition.

Imagine you're throwing a ball towards a target, and you've promised yourself that you'll hit the target at least once before you can stop.

We can think of this as a loop.

Until we've hit the target once, we have to keep throwing.

Sometimes our programs need to repeat the same thing over and over again until we reach a correct, or a valid answer.

This is where we can use Repetition.

A good example of Repetition in Python is the WHILE loop.

We can write the program as if we're throwing the ball.

WHILE target not hit, throw ball.

In this we can see a similar structure as our IF statements from before.

We have three major components of our WHILE loop.

The WHILE keyword - to tell the program we want a WHILE loop.

A Condition - in this case, 'targetNotHit'.

This will tell the program when it should stop looping - which is once the target has been hit.

The Body - what we actually want to repeat.

In this case, 'throwing a ball'.

Just like our conditional statements, this section needs to be indented one level.

Remember we need to be precise, or our code will create a syntax error.

Let's write some code to show how this works.

I want to write an awesome program that’ll check if I'm the User.

I'll need some User Input and because I want it to keep checking that it's me, I'll need to use a WHILE loop.

My WHILE loop will need a Condition and an Action.

The Condition to check for will be, 'any user that's not me'.

To do this I'm going to need a new sign.

Earlier we used two equal signs '=='  to check if two things were equal.

To check if two things are 'not equal' we use '!='.

The Action will be to print, "Access denied" and ask again who the User is.

Let's write the code together.

First we need to collect some User Input and save it in the Variable 'user'.

Now for the WHILE loop - I’m being careful to check my syntax, and making sure that I use a ':’ after the condition in my WHILE loop.

So this reads, "WHILE the User Input does not equal 'Isla', print 'Access Denied!',

ask for the User's name again, and repeat the check for 'Isla' in the User Input".

Once 'Isla' is entered, the loop should stop, and the last line will print.

Let's give it a try.

Now to try my name.

Hang on... Why did the program not welcome me when I typed my name?

When we find something wrong in our code it's called a 'bug'.

To remove that bug we need to go through line by line to see what went wrong.

Fixing our code is called 'debugging'.

The first line is working because the 'user' Variable is set.

What about the WHILE loop?

Oh look at the Condition - it’s expecting my name with a 'I'.

Computers are very particular about that sort of thing - I'll try again.

Awesome! Now I have a program that'll only greet people who type in my name correctly.

Hmm...I'm feeling a bit hungry right now.

It'd be great if I had a sandwich-making robot.

Unfortunately I don't have one of those yet, but

how would I go about describing how to make a sandwich to a robot?

It'd need to know all the little steps like - 'Get two slices of bread', 'Butter one side of each slice', 'Add filling between the slices, and 'Then put the two slices together around the filling'.

What if we wanted the robot to make lots of sandwiches?

It would take a long time to type all the code in, even if we copied and pasted it.

And it would be difficult to change how we made sandwiches, because the same code would be in so many places.

This is where Functions come in.

A Function is a block of code that we can write that contains all the parts we need to complete a larger task.

In fact we've been using some Functions this whole time.

Some statements in Python are Functions - like the print() statement and the input() statement.

Let's write a Function of our own for making a sandwich.

First we start with the 'def' keyword to show Python that we're going to define a new Function.

This is followed by the name of the Function.

Functions can be named anything, but just like with Variables from our last episode, they must start with a letter, and the convention in Python is that they start with a lowercase letter.

Then we indent one level, and write the body of our Function.

In this case, each line of code in the body is a step in making a sandwich.

Let's write a Function called, 'makeSandwich'.

Now we have written the Function, but it won’t do anything we can see, until we call it.

Calling a function is when a function is run in the code.

When we type 'print()' we are calling the print()  function so I'll add 'makeSandwich' to my code.

Now let's run it - I'm getting pretty hungry.

That's pretty cool!

I wonder...if I put 'makeSandwich' inside a WHILE loop...would I make infinite sandwiches?

Now you know how to use WHILE loops and Functions to handle all of the repetitive tasks you'll ever need to do.

Make sure you pay careful attention to the syntax as you write.

It's important to have the brackets and colons match, as well as making sure the body section is indented so the computer knows the structure of your code.

Loops are a great tool for repetitive problems,

and Functions give you a really neat way to reuse parts of code all through your program.

Have a go at using some of these techniques in your chatbot.

I know I will once I've finished eating all these sandwiches.

[Isla] Welcome back to coding@home.

Before the break we learnt more tips that we can use in our Python code.

A little later we'll find out how we can make our code more robust, but first let's catch up with some big thinkers who’ll give us their vision of the future.

[Stacey] Have you ever wondered how you distinguish globular clusters from the cosmic environment, or how the brightness of a supernova can reveal how much dark matter is in the universe?

Well wonder no more - because today we’re at the Sir Thomas Brisbane Planetarium to find out how artificial intelligence is being used to find the answers to these questions.

[Stacey] Hey Sam, you're an astrophysicist -it's such an impressive title- but what do you actually do?

[Sam] So an astrophysicist is someone who studies space.

If it's in space - I care about it.

If it's dark energy, dark matter, the fate of the universe, or just galaxies, stars, and planets - I want to know about it.

[Stacey] What are some of the interesting projects that you've worked on?

[Sam] So the project that I'm working on right now is with the Dark Energy Survey.

So we're trying to constrain the nature of dark energy.

Dark energy is this mysterious anti-gravity force in the universe that's causing the universe to expand, and not just expand but get faster and faster in its expansion, and to do that we use supernovae - we observe supernovae.

Supernovae are just exploding stars [but] there's an issue, stars can explode in a whole bunch of different ways, but we only care about a single way, because there's one way when stars explode -they explode with the same brightness - and that's what we care about if you have two things that are the same brightness, but one of them looks 4x dimmer to us than the other, then it must be twice as far away.

So you can use that to measure distances in the universe.

So we have to take all the data we have from the Dark Energy Survey and train these machine learning classifiers to say,

"Okay this is a Type 1A this is a Type 2A, Type 2P, a Type 2N, this is a quasar.

There's so much stuff in the universe and we need to get rid of everything except for the specific sort of object that we care about.

[Stacey] Now you mentioned machine learning - is that one of the AI tools that you use in your project?

[Sam] So machine learning is essentially when you ask a computer to go away and train itself and then come back and give you answers.

So you've probably seen many examples of this in your life.

If you have any video streaming services, you probably give things 'thumbs-up',  'thumbs-down', 'five-star', 'one-star'.

And so what you've done is you've given data to them - what you like what you don't.

Their algorithms have learned -hopefully your preferences - and then they can make recommendations.

So it's very similar to what I was doing, where I say well, I've looked at 100 objects

myself and I know that this one's a galaxy, this is a globular cluster, this is a star.

The algorithm learns from that and it says, based off that - here are all of the globular clusters that we think exist.

[Stacey] Now we know how artificial intelligence is advancing our knowledge of outer space, let's discover what the future of AI looks like back on Earth.

[Stacey] Now Marek, you're a Professor at QUT, how are you using this cutting-edge technology?

[Marek] Stace, there are two ways we’re using those technologies.

One of those ways is at work, helping us with doing what we need to be doing at the university.

The other way is when working with students, so basically teaching our students.

We have social media listening centres, where very sophisticated algorithms are trying to understand the conversation on Twitter, or on other social media platforms, and we teach students to work with those algorithms, so that when they're back at work, or where they get employed, they actually have those skills.

[Stacey] What are some examples of how AI is already influencing us every day?

[Marek] Think about ride sharing platforms.

There are algorithms that decide which driver gets the job, but also if a driver doesn't have -you know a good enough rating- and the rating falls under a certain level of stars or -whatever the rating is- that algorithm can fire the driver.

So there are algorithms that are like our managers, we have to deal with them, and they decide whether we work or not.

[Stacey] Can you explain how it's all evolving?

[Marek] Imagine being able to tell your dishwasher, "Hey buy a dishwashing liquid that's good for the environment, and it's not too expensive."

And then your dishwasher will shop around and buy the one that you need.

So this is a second stage of evolution, the first one was what I call 'hard-coded'.

They just do the same thing all the time.

The second stage is what I call 'preference-based'.

You tell the dishwasher what your preference is, and it just does it.

There's a third stage, when you have a smart fridge.

A hard-coded smart fridge will just replenish what you have inside.

So if you remove milk, you're out of milk, it will order more milk.

A preference-based fridge will understand that you want healthy milk and not maybe the cheapest milk, or you know something along those lines.

The third stage will be an 'inspirational' fridge.

That's a fridge that will try to understand your needs, what you expect, and then it will try to surprise you.

Maybe it will buy you a drink that you've never tried before, maybe it'll get you an almond milk, just to see whether you like it.

That's a pretty advanced level - we're not seeing such algorithms out there in the world just yet.

[Stacey] So if artificial intelligence is influencing how we work, think, and live - how will it influence our jobs in the future?

[Marek] In future we'll have 'codies’, they will be a bit like 'tradies',

but they will be really, really good at working with code.

This means that for instance, imagine you will buy a robotic lawnmower, and install it in your backyard.

Most likely it's a bit too tricky for you or me to configure it, so that it works, you know the best, you will just pick up your phone and ask for a 'codie'.

To come in and they will set everything up, configure everything.

So 'codies' will be the 'tradies' of the future.

[Stacey] From supernovas to smart fridges, when it comes to artificial intelligence it sounds like the future is already here.

[Isla] Welcome back to coding@home. We're really making strides with our coding in Python, but now it's time to take a big leap!

Let's level up our coding by using functions and repetition to save us time and space.

[Daniel] Hi there my name is Daniel.

I'm a designer and a programmer at Micromelon Robotics.

We build robots and software that teach programming for schools, and sometimes even universities.

Robots are a great way to learn programming, while seeing how your code can respond and affect the world around it.

In a lot of ways, programming a robot is very similar to writing a chatbot.

The robot has to respond to what its sensors see in the world, just as a chatbot has to respond to the messages that it's receiving.

Last week we used the input() statement  and branching, to print different messages depending on who was using our chatbot, and what commands they wanted to use.

Now that we have a simple greeting message, we need to make the chatbot a bit more interactive.

Let's add a simple line that tells our user what our chatbot does.

Hi there, I'm Chatty the cyber security chatbot.

One of the easiest ways to add more interaction to our chat bot is to add more IF statements that ask our User questions.

In this case we're going to be using cybersecurity questions.

How about we ask our User if they know what a phishing email is?

I'm going to add an input()  statement asking the question, and then record the user's input in a Variable called 'answer'.

I will also tell my User how to answer the question, by selecting 'A' or 'B'.

My Condition will be, 'IF the User enters 'B', the Action will be to print an affirmation that the User chose the correct answer.

IF the User inputs anything else, they will get the message telling them that they were incorrect, and also telling them the correct information.

This is looking pretty good, but I think we need to give it a quick test.

Alright, "What is your name?"

I know this one, 'Daniel'.

"What is phishing?"

I like catching fish, so I'm going to go with 'A', and I got it wrong.

Oh well.

That's looking pretty good though. We have quite a bit of code now the smarter we make our chatbot, the more code we're going to have to add.

This is going to make our code a bit cluttered, and hard to read.

And what happens if we need to change our greeting message?

Well we'd have to go back to each line with a greeting message on it and change it manually.

Wouldn't it be easy if we only had to make one edit, and that would change all of them?

One way we can reduce complexity in our code is by using Functions.

Functions are a brilliant way we can reduce the amount of code we need for our chatbot to work.

Let's make a greeting Function that takes our User's name as an 'argument'.

Our first step is to create our Function with 'def'.

Next we give our Function a name that'll describe what the Function does.

Let's call it 'greet'.

Then we add brackets.

Inside the brackets we define what 'arguments' will be used by the function.

We know we want our greeting to change depending on what the user's name is, so we will make an 'argument' called 'user'.

The last step at this stage is to end the line with a ':'.

Now we need to write the 'body' of the Function - which is what the Function will do.

Every time we run the 'greet' Function, we want to print a message to our User.

Let's put a print() in the body of our Function  and type in, '"Hello there " + user + "!"'.

You can see that the body of the Function is indented.

That's our complete Function - it's going to sit above the rest of the code so that Python knows what we're talking about when we call the 'greet' later on.

Now we can go back through our code and replace all our print() statements  with the 'greet' Function.

That looks much better.

And now if we want to change our greeting message we only have to change it in one place.

Now I want my chatbot to be a bit of a cowboy so I'm going to make it say, "Howdy".

Howdy there - so now no matter what name our User enters in our chatbot, it's going to respond with a friendly, "Howdy".

Hmm... I've noticed something with my chatbot.

Can you see a pattern here?

We have a lot of branches and they all call the same Function, because we are calling 'greet' every time in our IF, ELIF, ELSE statement.

The same Action is being run for every Condition.

I don't think we need the IF, ELIF, ELSE statement at all anymore.

Now that we have the 'greet' Function, we can simplify our code by replacing the IF, ELIF, ELSE statement by calling 'greet' with user as the argument.

And if we ever need our chatbot to greet our user again, all we have to do is call the 'greet function'.

Our question and answer part of the code is okay, but there's one small problem.

Anytime the User enters anything besides a 'B', they're going to get the error message.

We need to help our user input a correct response, if they enter an invalid response. We need to keep asking them until they enter a correct response.

This sounds like the job for a WHILE loop and a Condition.

To keep our WHILE loop running until the User enters a valid response, we will need our WHILE Condition to read, 'WHILE the User's input != "A" OR "B"'.

We need to check that the answer does not equal 'A' or 'B'.

This is what it's going to look like.

Our condition is that the loop will run WHILE the answer is not 'B' or WHILE the answer is not 'A'.

Note that the Variable 'answer' is used each time we check it.

All our code running inside the loop will need to be indented once more.

I will also need to add an ELSE to ask the input again.

Let's test it out.

I'm going to start by entering in some invalid responses first.

Awesome! It worked.

Now for a valid response.

Uh-Oh -that's not good- looks like Python's entered an infinite loop.

Good thing we can just close the shell.

Let's go back to our code and start debugging it.

It all seemed to work fine until we entered a valid response.

Once I selected 'B' the value of the answer was set as 'B', this means my loop Condition is met and my loop goes on forever.

I can fix this with a command called 'break'.

We can add 'break' anywhere inside the loop, and the loop will stop as soon as it's run.

Let's add 'break' after each valid response in our loop.

Let's give this a test and see how it runs.

An infinite loop usually signals that there's a problem in our code.

We avoid that by using 'break', but really we should look for a mistake in our Condition.

See if you can rewrite the WHILE loop with a better Condition.

Great - now we can add even more interactivity to our chatbot, and we can keep trying, even if we make a mistake.

Today we've covered two really powerful tools in programming.

First, Functions can simplify your code and make it easier to change by writing a reusable chunk of code once.

And second, WHILE loops allow you to repeat a section of code until a Condition is met.

With a few small changes our chatbot now works even if our User makes a mistake.

See if you can use loops and Functions to make your chat bots even chattier.

Good luck!

[Isla] Well, talk about levelling up!

We really are on our way to becoming coding experts and we nearly have all the tools we need for our coding projects.

Remember if you want to catch up on any part of the coding process you can see previous episodes online at 10play.com.au.

We'll catch you next week with some final tips, plus some advice on evaluating and presenting your project.

Happy coding!

[Upbeat music]

Authorised by the Queensland Government, Brisbane.