Lists

What are lists?

A list is an example of a ‘data structure’. This is the name for a number of different programming techniques used to organize and manipulate data in programs.

So far we have used variables to store data in our programs. These hold one piece of data and use an identifier (variable name) to represent it in the code.

Lists work in the same way as variables, but they store multiple items of data using one identifier.

Lists are created in Python like this:

```
countries = [“UK”, “USA”, “Chad”, “Australia”, “Thailand”]
```

```
prime_numbers = [1, 3, 5, 7, 11, 13]
```

Notice that the list containing string (text) data has each item in quotation marks. The list containing integer (whole number) data doesn’t need them. Note that different data types can be mixed in a list. The list below contains a float, and integer, a string and a boolean value.

```
[3.14, 72, "cat", True]
```

To identify individual items in a list, the data is indexed (given a number) - just like we did with strings. Indexing starts at 0, meaning that the first item in a list will be number 0, the second number 1 and so on.

Lists - Key Concepts/Misconceptions

You may hear lists referred to as ‘arrays’ in other programming languages. There are small differences but at this stage we will treat them as essentially the same thing.
A list can store multiple items of data.

- Items in a list are surrounded by square brackets. Each item is separated by a comma.
- Lists are indexed - each item is given a numerical position in the list. Indexing starts at 0. Students often forget this as they are in the habit of starting to count with 1.
- Lists are named in the same way as variables - the programmer decides what they are called. The same rules apply: choose descriptive names, use an underscore to separate words, etc.
- A list is *mutable* - each individual item can be edited, added or removed. Extra items can be added to the list.
  A list allows duplicate items.

Output From Lists

The print command is used to output data from lists, just like it is to output from a variable.

Output One Item From A List

To output one item, use print followed by square brackets with the item’s index number.

```
prime_numbers = [1, 3, 5, 7, 11, 13]
print(prime_numbers[0])
```

This would print `1` from the `prime_numbers` list, as that is the data at index 0.

```
countries = [“UK”, “USA”, “Chad”, “Australia”, “Thailand”]
print(countries[2])
```

This would print `Chad` from the countries list, as that is the data at index 2.

Output Every Item In A List

Python makes it simple to output a whole list like this:

```
print(prime_numbers)
```

This code actually uses a loop to go through each item in the list one by one printing them to the console, but this complexity is hidden behind the scenes.

Let The User Choose Which Item To Output

To do this, we get the user to input a number and store it in a variable.

In the print statement, replace the number in square brackets with this variable.

```
num1 = int(input("Enter the index to be displayed: "))
print(countries[num1])
```

Tasks and instructions

```
fruit = ["Apple", "Banana", "Grape", "Strawberry", "Melon", "Orange"]
```

Task 1

1. Add comments to predict what the following lines of code will do.
2. Alter the fourth print command so that it outputs a valid item from the list that hasn't yet been used.

```
print(fruit[3])
```

```
print(fruit[5])
```

```
print(fruit[0] + " " + fruit[2])
```

```
print(fruit[6])
```

Task 2

Write code to output the whole list - you should be able to do this with one line of code. (You're still working with the `fruit` list.)

Task 3

Ask the user to input a number between 0 and 5. Output the item in the list that matches the number they have input.

Assigning To Lists

Data in lists can be edited in the same way that data in a variable can.
Let's use the `countries` list from earlier in this document as an example:

```
countries = [“UK”, “USA”, “Chad”, “Australia”, “Thailand”]
```

The `=` symbol is used to assign new data. For example, this code replaces `Australia` in the `countries` list with `Mexico`:

```
countries[3] = “Mexico”
```

This code concatenates (joins) 'ingdom' to the end of 'UK'. `countries[0]` will now contain `UKingdom` (not a great example I know, but it works).

```
countries[0] = countries[0] + “ingdom”
```

Now let's use the prime_numbers list to see some math examples. This list contains integers so we can do math with them.

```
prime_numbers = [1, 3, 5, 7, 11, 13]
```

This code adds one to the first item in the list. `prime_numbers[1]` will now contain `2`.

```
prime_numbers[0] += 1
```

This code multiplies the second & third items in the list and stores the result in the `total` variable. `total` will now contain `15`.

```
total = prime_numbers[1] * prime_numbers[2]
```

This code subtracts the fourth from the fifth item in the list and stores the result in the first position. `prime_numbers[0]` will now contain `2`.

```
prime_numbers[0] = prime_numbers[4] - prime_numbers[3]
```

Tasks

Task 1

```
countries = ["UK", "USA", "Chad", "Australia", "Thailand"]
```

1. Add comments to the code to explain what the following lines do.

```
countries[3] = "Mexico"
```

```
countries[0] = "Iceland"
```

```
countries[1] = countries[4]
```

2. Add comments to predict what the list looks like now.

3. Add a line of code to print the whole list and check your prediction

Task 2

```
square_numbers = [1, 4, 9, 16, 25, 36]
```

1. Add comments to explain what the following lines of code do.

```
square_numbers[5] = 49
```

```
square_numbers[0] += 1
```

```
total = square_numbers[3] - square_numbers[1]
```

2. Add comments to predict what the list looks like now.

3. Add a line of code to print the whole list and check your prediction

Assigning to Lists - Independent Challenge

Task

1. Create a list called `names` that stores five names in it (you choose the names).
2. Ask the user what their name is, store their input in a variable.
3. Ask the user to enter a number between 0 and 4. Store their input in a variable.
4. Replace the data at the position that matches the number entered by the user in the `names` list with their name.

Adding To & Removing From Lists

These activities look at how we add and remove items from a list. There are two different methods of doing each of these tasks, and they work in slightly different ways. The code for them is:

Adding Items

`list_name.append(item)` - adds an item to the end of the list.

`list_name.insert(index, item)` - inserts the item at the position given in the brackets. The index number comes first, then a comma, then the item.

Removing Items

`list_name.remove(item)` - searches through the list and removes the item in brackets.

`list_name.pop()` - removes the last item in the list.

Checking To See If An Item Is In The List

Python makes it very easy to check if an item is in a list. This is where we can start to combine what we've previously learned about selection with lists. The code for this is:

```
if item in list_name:
    Run this code if condition is True
```

Tasks

Task 1

```
food = ["bacon", "cheese", "pasta", "beans"]
```

1. Add comments to explain what the following lines of code do.

```
food.append("tomatoes")
```

```
food.insert(1, "ice cream")
```

```
food.remove("cheese")
```

```
food.pop()
```

2. Add a comment to predict what the list looks like open

3. Write code to print the whole list. Was your prediction correct?

Task 2

```
video_games = ["Mario", "Sonic", "Joust", "Zelda"]
```

1. Write code to perform the following tasks.

- Add 'Minecraft' to the start of the list.
- Ask the user to input a number between 0 and 4 and store it in a variable. Output the item at this position in the list.
- Ask the user to input the name of a video game and store it in a variable. If this video game is in the list then remove it from the list. If it isn't in the list then add it to the end.

Task - Beat The Zombie

This is the most complex task yet. It requires several variables and two instances of selection. Encourage students to have some fun with the dialogue.

Create a list of possible weapons.
In a variable called `zombie_weakness` store the name of one of the weapons from the list.
Output messages telling the user that they have encountered a zombie and should prepare to fight.
Output the list of weapons to the user. Ask if they want to type 1 to use one from the list or 2 to pick their own. If they type 1 then they should input the weapon name - store it to a new variable. If they type 2 they should input the weapon name - add it to the list and save it to a new variable.
If the weapon picked matches the value of `zombie_weakness`, output a message telling the user that they have won the fight.
Otherwise output a message saying that they have lost.