INTRODUCTION TO PYTHON – SUMMARY MATERIALS

For further information on everything that I will cover here, there is a huge amount of online material and google is your best friend. In terms of tiers of information, I would recommend

- **The Python website** (e.g. https://docs.python.org/3/tutorial/) is always up-to-date and has all the info you should need in great detail. If you have a question about anything we cover, it will be answered in the tutorial. The tutorial assumes a larger degree of knowledge than I am assuming – but hopefully by the end of this video it should be usable.
- **Me!** Post in the Slack with any question (large or small) and I will answer, or bring to the drop-in sessions/email me. These videos are not comprehensive
- **StackOverflow** (https://stackoverflow.com/) – this can be a somewhat less trustworthy source of information, but is a reliable place to get code snippets and look for questions. Almost every question has been answered here – but sometimes the answers are out-of-date or don't form great recommendations.

The best way to learn any of these concepts is not just by watching the video, but by getting stuck in. I would suggest following along with the video as much as possible, and focusing on the exercises for this section.

BASIC CONCEPTS: VARIABLES, TYPES, OBJECTS, OPERATORS

Doing any sort of work in Python will usually involve variable creation and assignment. Each variable has an **associated type** and **value**. Whilst Python will do the work in determining what type you want usually, occasionally it will get it wrong (usually in converting between strings and numbers or date values).

There are subtle distinctions, but in general, everything in Python can be thought of as an **object**. This includes so-called *primitive types*, which covers the built-in type (e.g. int, float, str, bool) and any other more complex types such as lists, dictionaries etc.

An **object** is essentially a collection of zero or more bits of data together with zero or more functions (usually called **methods** if they are associated with an object). For example, the *list* type is an object, and a method on lists is the *append* method (as detailed in the video on data collections).

Understanding this in great detail is not incredibly important, but it's good to have a sense of what's going on under the hood and what terminology you might need when describing issues or looking for help.

We haven't introduced how to read in data from a file yet – that will be coming in the next set of videos.

You can find more documentation on this here: https://docs.python.org/3/library/stdtypes.html

DATA STRUCTURES: LISTS AND DICTIONARIES

Lists and dictionaries are examples of **data collections** or **data structures** – basically, ways of structuring your data so that you can easily access the things you want. Lists are sequential, so great for sequence data (such as lists of child IDs). Dictionaries are lookup based, so great for single data records, or grouping related bits of data where you want each piece of data to have an associated name.

There are plenty of other built in types too and plenty of ways to create them.

You can find more documentation on this here (again): https://docs.python.org/3/library/stdtypes.html

CONTROL FLOW STATEMENTS: LOOPS, CONDITIONALS, FUNCTIONS

In the video I give a very quick introduction to control flow – the best resource for more detail on this is the aforementioned Python tutorial.

This covers *for* loops, *if* statements, and defining functions – as well as some other details not covered in this video. This should be a helpful resource in solving the exercises for this week.

You can find this here - https://docs.python.org/3/tutorial/controlflow.html

EXERCISES

The exercises for this week are all programming based – and some may require more bits than was covered in the video – so I'd encourage you to get used to looking around for answers, either in the resources I've flagged or online.

Submission for this will be via the Replit Teams platform – you should see the project IntroToPython in the team assigned to you. Write all your code for every step in there and submit it when you feel you have done all the tasks, or if you have any questions, let me know.

I can review your code at any time and when you submit I can annotate any issues and send back to you.

INTRODUCTION TO PYTHON – MINI PROJECT

In the training materials, I have attached an excel worksheet with 903 data on 5 children. This is the dataset you will be using for this task. Feel free to use any external resources you wish if necessary – there are no restrictions on how you solve this and I actively encourage googling answers. The workflow should be similar to the video.

Step 1: Translate the data

- Open up header.csv Excel – similar to the video, we can look at translating this into Python variables.
- Manually (by hand) copy in data on child ID, sex, ethnicity and UPN. (Yes, this is a pain – but in the next set of videos we'll cover reading in files).
- Choose an appropriate way to encode this (e.g. dictionary of lists, list of dictionaries, any other option you want to try).

Step 2: Write the following functions

- Write a function which returns True if a child id is divisible by 3.
- Write a function which returns True if the ethnicity starts with W.
- Stretch goal: Write a function which checks that the UPN is the correct format – that is
    - Is it 13 characters long?
    - Are the last 12 characters numbers?
        - Hint: you might need a list of all the numbers you are checking
        - Hint: you may need to think about types (integer vs. string)
        - You might also need to use the "in" operator
    - Stretch: Is the first character an uppercase letter?
        - Hint: for this you will need a list of all uppercase letters – you can write this out by hand.

Step 3: Write code that will apply these checks to the appropriate columns on the data, using these functions.

- You will need to use loops over the data structure you've created, and if statements to check the appropriate columns.

Step 4: Design **print** statements to show your output

- For each row in the data, print out:
    - A summary of the data
    - Whether each check passes or fails