**Python Basics**

**Commenting**

A **single line** comment in Python is done with a **#**

***#*** *this is a single line comment*

There is no set **multiline comment** command in Python however people use triple quotation marks to create a message that the interpreter will ignore and acts just like a comment

**" " "**

*This is a string that is*

*not assigned to a variable*

*therefore, it’s a comment*

**" " "**

**Indenting**

In other programming languages indenting is used to make the code easier to read. It is not required of the language and is just a good habit. In Python indenting is essential as it defines blocks of code. Not indenting when forming a new code block will throw an error.

**if 10 < 20:**

**print("This works")**

**if 1 < 2:**

***print("Error")* #** *This line should be indented as it belongs to the second if code block*

The number of spaces you use to indent does not matter however remain consistent. I recommend 4 spaces or one proper tab as it defines the blocks of code well.

**Semicolons**

In other programming languages like JavaScript you use a semicolon **;** to denote the end of a statement. In Python you don’t use them. A new line denotes the end of a statement.

**Useful Input and Output Functions**

With JavaScript we used some inbuilt methods to either message the console or create an alert box. We also used **confirm()** to ask the user for a boolean and **prompt()** to ask them to input a string. Here are the Python equivalents.

**console.log("Hello World");**  **print("Hello World")**

**const age =** **prompt("How old?");**  **age = input("How old: ")**

Note that by default Python doesn’t create a GUI and output the message like JavaScript. It will ask for the input in the command line interface.

**Importing Modules and Packages**

Importing scripts into Python projects is a common thing to do. These scripts are called modules and contain Python code which extend the capabilities of what you can do with Python. These modules might contain functions that allow you to do more advanced math or give you the capability to produce Graphical User Interfaces. They may also be your own code that you’ve written but want to use in another project. By importing them into your .py file you can use their code. It is standard to write the import code at the start of your file.

**import moduleName**

There are 3 different types of modules that you can import into your file:

1. **In-Built Python modules** – these are modules that come standard with Python. All you need to do is import them at the top of your file and you can use their code. <https://docs.python.org/3/py-modindex.html>

**import math** <https://docs.python.org/3/library/math.html#module-math>

**import time** <https://docs.python.org/3/library/time.html#module-time>

**import os** <https://docs.python.org/3/library/os.html#module-os>

1. **Your Own Python Files** – if you have another .py file in the same directory you can import it. Just use import and the file name without the .py extension. If this file is located in another folder you will need to specify this in the import statement.

**import myLocalFileName *#*** *file in same directory*

**import subfolder.myFileName *#*** *file in folder called subfolder*

1. **Non In-Built Python Package Modules** – there are many external Python modules that you can use but since they are not in-built into Python you will need to install them first. You need to use PIP (a Python package manager) to download and install the package first and then you can import the module as before.

Text, logo

Description automatically generated

**from pynput.keyboard import Key, Controller** <https://pypi.org/project/pynput/>

***#*** *here we are importing only certain aspects from the pynput module*

**Dealing with Modules**

It’s always a good idea to read up on the modules that you are importing into your project hence why the links are provided above. You can use **dir()** to print out all the commands that are available to that module.

**import math**

**print( dir( math ) )**

When you have imported the module, in order to use its code, you must reference the name before calling its functions or variables like so. It’s as if all the code belonging to the module is stored inside an object named after the module.

**math.pi *#*** *return 3.14159265359*

**time.sleep( 3 ) *#*** *delays executing file by 3 seconds*

When importing the module you can assign it an alias which you can use in your code instead of its name.

**import myLocalFileName as mycode**

**mycode.callFunction()**

**Exception Handling try…except**

Python will output an error when the interpreter hits one. You can wrap parts of your code in **try** blocks to effectively deal with an error if it arises. The code in the **try** block will be ignored if an error is triggered and so the interpreter will move on to the **except** block without halting and producing the error. Every **try** must have an **except**.

**try:**

**pRInT( "This will not run" ) *#*** *pRInT() is not a function*

**except:**

**print( "This will run instead" )**

Exception handling can be very useful when dealing with user input as you can’t always expect the user to do what they are supposed to.