## Random Numbers, Files, and Onwards from Here

## **Random Numbers**

- How can we make our program produce random values?
- It is impossible for a computer to algorithmically produce a *truly* random value.
- There are algorithms that will generate *pseudo-random* values.
  - These values are 'good enough' they look random to a person, and are evenly distributed amongst the space of possible values.
  - But they are generated by an algorithm, so the precise sequence of random values could be repeated using the same algorithm.
  - A seed value is used when starting the pseudo-random algorithm for the same seed, the same sequence of pseudo-random values will be produced.
  - If we set a different seed each time we run our program for example, by setting the seed to the current time - we will get a different sequence of values.
- The Python module random allows us to generate random numbers.
  - Remember, being a module, we need to first bring it into our code using import random.
  - The documentation for this module can be found at <a href="https://docs.python.org/3/library/random.html">https://docs.python.org/3/library/random.html</a>.
- When the module starts, it is automatically seeded using the current system time.
- We can set our own seed using random. seed ().
  - o random.seed() will seed the generator with the current system time.
  - o random.seed(x) will seed the generator with the value of x.
- There are many different functions, given in the documentation, to extract different kinds of random values. Some useful ones are:
  - o random.random() Returns a random floating point value between 0.0 (inclusive) and 1.0 (exclusive).
  - o random.randint(a,b) Returns a random integer between a and b inclusive.
  - o random.shuffle(x) Shuffle the values in list x. (This modifies x, rather than returning a new list).
- Exercise:
  - Try generating some random numbers! Test how using the same seed leads to the same output.

## Files

- We can read and write files using Python.
- This is a built-in feature of the language, and does not require a module.
- To open a file, call the open function:
  - o myfile = open(filename, mode)
  - filename should be a string containing the filename.
  - mode should be a string containing one of several possible values, indicating how the file should be treated:
    - r File is read-only.
    - w File is write-only, and will be overwritten completely.
    - **a** File is write-only, but data will be added to the end.

- There are further options; see: https://docs.python.org/3/library/functions.html#open
- The function returns a file object, which we need to store in a variable.
- We will work with the file using the file object.
  - File objects have various methods we can call to perform operations on the file
- To close a file when finished with it, call the close method:
  - o myfile.close()
  - o The file object should not be used any more once it has been closed.
- To write to a (writeable) file, call the write method:
  - o myfile.write(data)
  - o The string data will be written into the file.
  - Certain characters cannot appear in strings, such as newlines or quotes. To add these to your string, you can use escape codes:
    - \n will be replaced with a newline.
    - \t will be replaced with a tab.
    - \' or \" will be replaced with a single or double quote (as normally, the quote would finish the string).
    - \\ will be replaced with a single backslash, as a backslash on its own indicates an escape code!
    - myfile.write("First line.\nSecond line.\n\"Quoted line.\"")
  - Alternatively, use myfile.writelines (data), where data is a list of strings, to write each string as a line in the file.
- To read from a (readable) file, there are many possible methods:
  - o data = myfile.read()
  - o The entire file will be read, and stored in data.
  - o data = myfile.read(x)
  - o The next x bytes of the file will be read, and stored in data.
  - o data = myfile.readline()
  - o The next line of the file will be read, and stored in data.
  - o data = myfile.readline(x)
  - $\circ$  The next x bytes, or the next line of the file whichever is shorter will be read, and stored in data.
  - o data = myfile.readlines()
  - o The entire file will be read, and stored as a list of separate lines in data.
- Exercise:
  - Try reading and writing some data from and to files!

## Where to go from here?

- We have now reached the end of the taught part of this course.
- You have learnt a lot of Python enough to write some quite complex programs.
  - There will still be things we haven't covered if you ever find yourself wondering if something is possible, try searching online!
- The key thing now is practice...

• For the remaining sessions, we will provide a list of longer challenges and exercises, for you to pick and choose what to practice with.

We hope you have enjoyed the course!