Random Numbers, Files, and Onwards

Random Numbers

Computers cannot produce *truly* random numbers.

We make do with *pseudo-random* numbers, which are good enough.

The pseudo-random generation algorithm is given a *seed* value.

The same seed will always produce the same sequence of pseudo-random values.

Random Numbers

The `random` module provides random number generation.

```
import random
```

To set the seed to the current system time:

```
random.seed()
```

To set the seed to a specific value:

```
random.seed(x)
```

Random Numbers

To generate a random floating point value in the range [0.0, 1.0):

```
random.random()
```

To generate a random integer in the range [a, b]:

```
random.randint(a, b)
```

To shuffle a list of values (in-place):

```
random.shuffle(x)
```

Exercise: Random Numbers

Try generating some random numbers!

Test how the same seed leads to the same output.

Files

File access is built in to Python - no module is required.

Opening a File

To open a file: `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.
  - https://docs.python.org/3/library/functions.html#open
- The function returns a file object, which we need to store in a variable.

Closing a File

We should close the file when we are finished with it.

```
myfile.close()
```

Once closed, the file object should not be used further.

Writing to a File

To write to a file: `myfile.write(data)`

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
- `\\` will be replaced with a single backslash

```
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.

Reading from a File

`data = myfile.read()`	Read entire file.
`data = myfile.read(x)`	Read `x` bytes of file.
`data = myfile.readline()`	Read next line of file.
`data = myfile.readline(x)`	Read next `x` bytes or the next line of the file - whichever is shorter.
`data = myfile.readlines()`	Read entire file, as list of lines.

Exercise: Files

Try reading and writing 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.