A Brief Introduction to *Python*

Python is an interpreted scripting language. This contrasts with the Fortran 90 taught in the module in both the 'interpreted' and 'scripting' regards. An 'Interpreted' language is one which is executed (essentially) one line at a time. The line is read in, then compiled *and* run before moving on to the next line. This is different to 'compiled' languages, where the entirety of the source code file is compiled in one go, producing a new, separate, executable file which can be run at a later time. 'Interpreted' languages mean that you can usually have the interpreter run as you type, getting result with each line of commands as you go, updating and correcting manually as you go. Its disadvantage is that it is almost always substantially slower than a pre-compiled executable to run, as it is having to also translate the language written as it goes, rather than having had it all done in advance with a compiler.

The definition of 'scripting' language is less formal. Whether a language falls into the category of 'scripting' or 'programming' is generally more dependent on what one is doing with the language. Scripting languages tend to be used for more miscellaneous, ad-hoc, or low-computational intensity tasks. Things like data analysis, file organisation and processing, plotting and similar. Programming languages are for tasks of greater computational complexity, more demanding computations or wherever a fixed, immutable file facilitating the execution of a program is desired.

Python is an extremely versatile language, especially given that it has numerous 'modules' for performing a wide variety of tasks. These modules permit great variety in its application, from plotting, to GUIs, to data processing, to interacting with file systems, and a great deal besides.

Differences with Fortran 90

There are a huge number of differences between the Fortran 90 which is part of this course and Python, here are only a few of the more major differences.

- **Dynamic variable allocation and typing**
  Variables in Python do not need to be described in advance of their use in the same way you require a line of the form '`INTEGER :: i`' or similar in Fortran. In Python, one needs only assign a quantity to a new variable name, and a variable of that name is automatically created, and decides its type based on what type of data is given to it, so '`a = 4`' would automatically be an integer, '`a = 'test'`' would automatically be a string. In addition, if ever you assign a different datatype quantity to that same variable, it will automatically re-designate itself to support the new type. This is both useful, and potentially hazardous. It is convenient, but can easily lead to numerous unintended problems if one is not careful.

- **'Iterator' objects**
  Loops in Python are different to those in Fortran. In Fortran, it is very explicitly a numerical sequence which is stepped through. Python, in contrast, operates by selecting sequential items from 'list' structures, effectively Python's default equivalent of arrays in Fortran. A more complete description will not be elaborated upon here, though multiple examples can be found online and in the versions of the assignment solutions also downloadable. To show the basic format, the following codes are equivalent, first in Fortran, and then two in Python (note that the example with `range` displays '`i+1`', python indices and number ranges are almost always from zero up to the total - 1).

```
DO i = 1, 10, 1
```

```
  WRITE(*,*) i
END DO
```

```
for i in range(10):
  print i+1
```

```
for i in [1,2,3,4,5,6,7,8,9,10]:
  print i
```

In relation to the first point about dynamic typing, lists are not allocated a specific data type throughout, the following example is a valid Python code snippet (note the mixture of strings, integers and floats within the same list; the variable '`i`' will change type and value for each item of the list as required):

```
x = 10
for i in [7, 'one', 'eight', x+1, x-3, 1.4352]:
  print i
```

- **'White space' sensitive**
  In Fortran, so long as key words are not interrupted by spaces or tabs, gaps between words are effectively completely ignored. Indentation is optional, and serves no purpose other than a guide-to-the-eye for any readers of the code. In Python, indentation is how structures are defined, and is vital to the operation of the program. While in Fortran, an `IF` statement is delimited by `IF () THEN` and `END IF`, in Python, only the opening '`if` is used, everything within that `if` statement is then *required* to be *exactly* one additional indentation level from the if statement itself; as soon as the indentation returns to its previous level, the `if` statement is automatically closed. This is unusual amongst programming languages, most languages do not make use of this form of indentation-dependent programming, and it can be quite difficult to get used to.

Teaching yourself Python

As Python's popularity has increased, a huge number of resources have become available to help people learn the language. First among these is arguably the Python reference documentation. The official Python website, https://www.python.org/, hosts all of the key definitions and explanations of all of the processes and structures natively included in Python, as well as examples and 'Getting Started' guides.

Search the web for additional sources related to Python content, and you will find lots of information, but the official Python site would be my suggestion as the best starting point.

In addition to external sources, in the same section that this document was in on Moodle, I have included worked examples of most of the assignment material, constructed in Python rather than Fortran. Some of these deviate from the actual assignment material, partially due to fundamental differences between the languages, and partly in order to better illustrate some of the features of Python.

We wish you the best of luck in any work you do in Python, feel free to ask questions about it, though keep in mind that our time is prioritised towards teaching the Fortran.