Chapter 1

A Very Brief Introduction to Python

1.1 Introduction

This chapter is added just to make you feel comfortable in understanding the syntax used for CFD codes in this book, which are written in Python programming language. If you already know Python, then you may choose to skip this chapter, and you may use Internet resources to understand the specific feature of the language when you come across something new in the CFD codes in upcoming chapters. Python is a computer programming language of choice now a days for the purpose of demonstrations, prototyping and testing applications. Python, due to its clear and readable syntax, is also easy to learn and demonstrate algorithms to a third person. Therefore, it is a very powerful tool for teachers to teach complex algorithms to students. The additional libraries available for Python language (like numpy and matplotlib) are very rich in functionality and are very robust. This makes Python a programming language of choice for scientific analysis and numerical computations. This small tutorial on Python is by no means a complete reference of the programming language, and the reader is advised to refer to more complete books to understand the language in greater detail.

1.2 Motivation for using Python

Some of the attractive features of Python which make it the language of choice for introduction of the basic CFD concepts are mentioned below:

1. **Easy to learn and use:** Python is a "high level language" and therefore learning this language is extremely easy compared to languages like FORTRAN, C, C++ and other "low level languages". In addition, the syntax is similar to MATLAB® therefore if you already know MATLAB® then understanding Python syntax is easy.
2. **Dynamically typed language:** In Python, the variable type is not defined explicitly. The variable type is inferred by Python interpreter at the runtime based on the value passed to it. This feature reduces development time of the code, and makes it easy to read.
3. **Very useful and robust libraries for numerical computations and plotting:** Installation of some additional packages such as numpy and matplotlib enriches the capabilities of the Python language (and makes it faster due to the optimizations done in these libraries) to do complex scientific analysis and 2D and 3D data plotting.

1.3 Introduction to Python syntax

1. **Comment:** Comment is text beginning with a "hash" character and is ignored by the Python interpreter while running the code. Comments are used to add information about the code for the person trying to

understand the code details later. Comments are also useful to make a section of code non executable (mostly done for debugging).

2. **Variables:** Variables are used for storing and operating on values such as integers, real numbers, complex numbers, arrays etc. In Python all variables are interpreted as objects (you might know about objects if you have used object oriented languages, if not then do not bother!). Variable names can be any combination of alphanumeric characters without any spaces and not starting with a number. In addition, underscores can be used to make the variable name readable and meaningful. For example, abc 12 3 is a legal variable name, but, 3abc 12 is not a legal variable name, because it starts with a number. A simple example of Python code showing the syntax for comments and variables is given below.

Listing 1.1: Comments and variables

```
# This line is a comment
# This line is also a comment
i = 2        # 'i' is an integer, as 2 is an integer
j = 5.2      # 'j' is floating point number, as 5.2 has decimal point

# Next line of code will evaluate to a floating point
mult_ij = i * j

l = mult_ij**i     # Two asterisk means power
```

Note that in the code above, the variable types are not defined explicitly. For example, variable 'i' is not defined as integer, and variable 'j' and 'mult ij' are not defined as floating point variables. If you have programmed using strictly typed programming languages in the past, then you will see and appreciate this feature of Python more clearly.

3. **Defining code blocks:** Block of code inside conditional statements, loops, functions etc. is defined by using a colon and indentation. The start of the block is defined by a colon and the block continues till the indentation continues. An example code for conditional block (if else) and loop block (for loop) is given below.

Listing 1.2: Code blocks

```
# Code block demonstration
year = 2013
if year >= 2050:      # Colon defines start of block
    print ("You are in future.")  # 1st statement in if block
    print ("Welcome to CFD.")    # 2nd and last statement in if block
else:
    print ("Year 2050 is yet to come.") # 1st statement in else
    print ("You are welcome to CFD.") # 2nd and last statement in else

for i in range(1, 20):
    j = i * 2           # 1st statement in for block
    print (j)           # 2nd statement in for block
    print (i)           # 3rd and last statement in for block

print ("Out of for block.")
```

4. **Importing libraries:** The import keyword is used to import additional libraries into the code to add functionality to standard Python capabilities. Some of the libraries used by the scientific community are numpy, matplotlib and mayavi. These libraries enhance the power of Python enormously for scientific analysis, 2D and 3D plotting. In this book, these libraries are used extensively and thus the next section provides a brief introduction to some of the functionality from these libraries which is used in this book. Example code for importing libraries is shown below.

Listing 1.3: Importing libraries

```
# Code for demonstration of import
```

```
import numpy as np
import matplotlib.pyplot as plt

# and now we can use the libraries by short name 'np' and 'plt'
x = np.linspace(0, 10, 100)   # array of 100 floating point numbers [0..10]
plt.plot(x, np.sin(x))
plt.show()
```

5. **Working with numpy array:** The numpy library has a load of features designed for numerical computation. The base of most of the computations is the ndarray class. Some of the important operations on an array object include creation, data access and slicing which are shown in the code snippet below.

Listing 1.4: Numpy array

```
import numpy as np
# numpy array can be created in many ways. Here are a few examples
a1 = np.array([5, 4, 8, 9])   # array initialized with a list of numbers
a2 = np.zeros(10)             # array of size 10 filled with 0.0
a3 = np.ones(100)             # array of size 100 filled with 1.0

# numpy array is zero indexed, therefore the first element has index of 0
num = a1[0]          # num is assigned first value of a1. 5 in this case.

# slicing provides a sub view of the array [start : end+1 : stride]
sub = a1[1 : 3]      # sub points to a part of array a1  > [4, 8]
```

In this book the libraries ‘numpy‘ and ‘matplotlib‘ are extensively used. These libraries are very well documented and the community around Python is very helpful and friendly. If you do not understand any of the syntax presented, then a simple search over the Internet should be good enough to get the required help.