**INF 1511 – Revision notes – Chapter 2**

1. **Performing Arithmetic Operations**

The operations that can be performed on a variable in Python depends on the data type of the variable. The different data types are described on page 29 of the textbook. Numerical data types can be operated on by arithmetic operators. Review the numerical data types listed in the textbook.

In addition to the normal arithmetic operators, +; -; x and /, the following operators are also used in Python

// the truncation or floor division operator which ignores the fraction in the result.

\*\* the exponentiation operator, 23 is written as 2\*\* 3 in Python

% the modulus operator gives the remainder of one number divided by another, e.g. 5%2 gives 1.

+x or –x the unary operators increase or decrease x by one respectively, e.g. if x = 10 then –x gives 9 and +x gives 11.

1. **Multiple Assignment Statement**

Python allows the assignment of multiple variables to multiple values in one statement, i.e.

x, y,z = 1,2,3 will assign 1 to x, 2 to y and 3 to z. This eliminates the need to write 3 assignment statements in the code.

1. **Escape or nonprintable characters**

Familiarise yourself with the escape characters used throughout the exercises, assignments and examples in the book. These are listed on page 42 of the textbook.

1. **Other operations**

Type(x) gives the data type of the variable x.

0o expresses a number in base 8 (octal).

0x expresses a number in base 6 (hexal).

variable=input ('Message') prompts a user of your program to input data.

Int(x), float(x), str(x) all converts a variable x into an integer, float or string respectively.

X << y equivalent to x multiplied by 2\*\*y

X >> y equivalent to x divided by 2\*\*y

X & Y compares corresponding bits of x and y if both are 1, returns 1 otherwise returns 0.

X | Y compares corresponding bits of x and y if both are 0, returns 0 otherwise returns 1.

x ^ y compares corresponding bits of x and y if either is 1, returns 1 otherwise returns 0.

~ x converts binary 1 to 0 and vice versa (inversion).

Addition of Complex numbers – complex numbers are made up of a real and imaginary component, when adding complex numbers, add real components and imaginary components separately.

1. **Making Decisions**

The **if … else** statement is used to control the flow of logic in a program

if (logical expression):

statement(s)

else:

statement(s)

**Comparison operators** are used in the logical expression. Comparison operators are listed on page 55 of the textbook.

The **if-elif-else** statement is used to execute code based on several conditions that are related, e.g if the marks in a class are compared and different statements are to be executed based on the range in which a particular mark falls.

**Logical operators** – and, or, not are used to combine if statements, e.g. if the test mark is < 50 and the assignment mark is < than 50 then execute statements 1 and 2.

Comparison operators can also be chained to reduce if statements, e.g.

if 1<= m <=10 which means if m is between 1 and 10, including the numbers 1 and 10.

1. **Loops**

Loops are used to execute the same statements more than once.

The **while loop**

while expression :

statement1

statement2

statement3

expression represents a condition that must be checked every time the statements in the loop are executed. Once the expression is true, the loop will end. The comparison operators discussed above are used in the expression. Alternatively, the **break statement** can be used in an if … else statement in the loop to end the loop.

The **continue statement** is used to skip part of the code in the loop. If the continue statement is executed, the remained of the statements below it are skipped.

The **pass statement** is used to represent an empty block of code. It tells the program to do nothing for the condition in which the pass statement is.

The range () function is used with the loop for returning values of integers.

**The for loop**

for iterating\_var in sequence:

statement1

statement2

statement3

used to process sequences, e.g. a tuple containing many records.

The for loop can be used with the **membership operators**, **in** and **not in.**

The **choice ()** function is used with the for loop and picks a random item from the sequence.