**PYTHON WEEK 1 SUMMARY**

**Day 2 – Variables**

Variables are memory storage units that hold data in a program. When a variable's value changes, the program's state changes accordingly. They act as containers for various data items or values and are essential in programming. A solid grasp of variables is crucial for writing code effectively, as they are fundamental components used in all programming languages. Every variable is created with an initial value. A variable can be in three states:

* Variable creation (Declaration)
* Variable assignment (Initialization)
* Variable changed (Execution)

Once the code which created the variable has finished executing, the variable is destroyed. In Python, variables are defined in a standard, using the assignment character (=). This changes the value of variable. Naming conventions specify the way in which variables should be named. This standard is used to make code more readable, and thus easier to understand.

The rules include start and continuation characters. Variable names may contain any upper or lower case letter (A-Z, a-z), a number, or the underscore character. They may not begin with a number or contain spaces. Continuation characters are any characters except whitespace characters like tab and space.

Examples of valid variable names include: c, ref\_number, admin, aVeryLongName

Examples of invalid variable names include: True, $name, 12Graph

In Python identifiers are case sensitive, so for example, firstName, FirstName, FIRSTNAME, and firstname are four different identifiers. A second rule is that variables cannot have the same name as Python’s keywords. We can find out what keywords are in Python, by using the function called dir(). If this function is called with the \_\_builtins\_\_ attribute, it returns a list of Python’s built-in attributes.

The \_\_builtins\_\_ module contains all Python’s built-in attributes, which can be used with the dir()function. The ones that are returned are identified with the following characteristics:

* Python’s built-in exceptions start with a capital letter.
* The rest are either functions or data type names.
* Identifiers that start and end with one or two underscores are special methods.

All variables must be assigned to a data type like a string or an integer. In Python, a variable is assigned automatically to an appropriate data type.

**Casting –** Casting can be done in two ways:

* Implicitly: The compiler automatically casts a value from one data type to another when assured that there will be no data loss.
* Explicitly: A value cannot be automatically cast from one data type to another if it will result in data loss. Extra code has to be written to ensure that the value stays the same and only the data type changes.

**Day 3 – Data Types**

**Integers -** Python has three distinctive numeric types; this means that for every number that you use in your programs there is a suitable data type that you will use to ensure that your programs are truly performance efficient. Integers are numbers. Numbers can have negative and positive values. Monetary numbers can have decimal values.

**Booleans -** Boolean data type has corresponding integer values. There are only two possible values that a Boolean variable can have, True (1) or False (0). When returning Booleans as strings they are seen as “True” and “False”, and never as “1” and “0”. True and False are case-sensitive in Python. Boolean tests whether conditions are valid or not. The three logical operators used to test conditions between two arguments are:

* The and operator
* The or operator
* The not operator

**Floating point numbers -** Floating point numbers are better known as floats. Float is the data type that manages numbers with decimal places with very accurate precision. The float data type can be called as a function with zero or 1 argument of any data type. If no argument is given, then float returns 0.0. If an argument is given, an attempt will then be made to convert the value to a float data type, but this does not mean it is always possible. For example, float("21.765") will be converted to a float, but float("FF909A") will raise an exception. A string value cast to a float must contain only numbers and only one occurrence of the dot (.) character.

**Complex numbers –** Complex numbers are two numbers contained in a single variable. The first part of a complex number is the real part (float), and the second part is the imaginary part (float), assigned in this manner: complex(real, image). Imaginary numbers are real multiples of the imaginary unit, written with a suffix of j (J). The imaginary part is the square root of -1. Python has built-in support for complex numbers. The latter notation is written as follows: 4+8j.

**Strings –** Strings are represented by the immutable (unchangeable) str data type. Strings are a sequence of Unicode characters which form a single manageable string. The str data type can be called to create a string; when there is no argument supplied, it returns an empty string. s = str("") is the same as s = str(), when an argument is passed to the string method that is not a string value, it is passed as a string representation of the type supplied: s = str(17.2354), is the same as s = str("17.2354"). The string function is often used to convert other data types to strings.

**Lambda expressions –** Small anonymous functions can be created with the lambda keyword. This function returns the sum of its two arguments: lambda a, b: a + b. Lambda functions can be used wherever function objects are required. They are syntactically restricted to a single expression. Semantically, they are just syntactic sugar for a normal function definition.

**Day 4 – Operators**

**Using operators –** Expressions can be broken up into two parts: 2+3 and a/b. Operators provide the functionality to an expression and can be represented by symbols such as + or by keywords such as and. Operators require data to operate, and this data is known as an operand.