**INF 1511 – Revision notes – Chapter 5**

1. **The class statement**

A class is a template or blueprint for data and operations. It consists of the following:

1. Data members – consisting of class variables (outside of any method of the class) and instance variables which are inside a method.
2. Member functions – operations that can be applied to the data members.

A class is created using the following syntax:

class classname (base-classes):

statement(s)

The **base-class** above is the class from which the current class inherits from, it is also called a **super class**.

The inheriting class is called a **derived class** or a **subclass**.

The statements are called the body of the class.

The **attributes of a class** are objects that are contained in the body of the class (as defined in the statements) or are **built-in attributes**. Refer to page 132 of the textbook for built-in class attributes.

1. **Defining functions in a class**

Functions defined in a class are known as methods and they always start with the parameter named self. These methods are called instance methods

Instance methods are defined as follows:

class classname(base-classes):

class variable(s)

def method 1(

statement(s) self):

instance variable(s)

1. **Accessing Class Variables in Instance Methods**

To access class variables, the methods defined in a class body must use a fully qualified name; the class object must be fully prefixed with the class variables.

1. **Instances**

An instance is a variable that acts as a replica of a class and an unlimited number of instances can be created and each gets a separate copy of the methods and attributes defined in the class.

To create an instance of a class called classname, you call the class object as if it were without parameters as shown below:

r=classname()

**The \_ \_init\_ \_()** method is used to initiate the class.

Python allows for the assigning of one instance to another.

Inst1 = inst2

This will create the instance if it does not exist and all the variables in inst1 will be set to the values in inst2.

Arguments can be passed to the \_\_init\_\_ using the following statement def \_\_init\_\_ (self, x,y)

1. **Class Methods**

A class method has no self argument and receives a class (called cls) as its first argument.

The class method is defined as follows:

@classmethod

Def f (cls, parm1, parm2, ….):

body of method

where warm1, parm2 … are parameters of the object.

1. **Static Methods**

Static methods are ordinary functions that bind their results to a class attribute. It does not have a self or cls parameter and is automatically inherited by any child classes. The static method is immutable via inheritance.

The static method is defined as follows:

@staticmethod

Def name (parm …):

body of the method

1. **Garbage Collection**

Garbage collection is a procedure for freeing up the memory that is used by the variables or instances that are no longer required.

Python uses a reference count, any object whose reference count is zero is garbage collected.

1. **Inheritance**

Inheritance is a technique of copying the data members and member functions of an existing class into another class.

The members in a class can be defined as:

**Public –** accessed from within or outside of the class.

**Private –** can be accessed from outside of the class.

Member functions in the inheriting class that have the same name as the base class’ function will override the base class’ function, i.e. the function in the sub class will be executed. This is called **method overriding.**

The methods of a base class can be called from the sub class by using fully qualified method names, i.e. base\_classname.methodname.

1. Single inheritance – one class is derived from another single class, implemented as follows:

Class sub-class-name (base-class-name)

1. Multilevel inheritance – when a class inherits a class that in turn is inherited by some other class.
2. Multiple Inheritance – one class inherits from more than one base class. If two classes with the same function name are inherited from, the function from the first base class will be executed.
3. **Operator overloading**

Overloading is when arithmetic operators are used to perform operations on classes.

The comparison operator == can be used to determine if two instances have instance variables with the same value.

**Polymorphism** allows different methods with the same name in different classes to perform different tasks.

**Properties** are used to manage attributes withget **and** set **methods.**

1. **Descriptors**

Descriptors are used to manage instance attributes. There are two types of descriptors;

1. **Non-data descriptors – implements the \_get\_ method.**
2. **Data descriptor –** implements the\_delete\_ , \_set\_ and \_get\_methods.

**The \_\_setattr\_\_ method** is called whenever you assign a value to an instance variable

**The \_\_getattr\_\_method** fetches an attribute of an instance using a string object and is called when attribute look up fails, that is, when you access an undefined attribute.