# Object-Oriented Programming

| | |
|---|---|
| Chapter No. | 13 |
| Status | Completed |

- Classes & Objects
    - A class is a blueprint that defines the properties and methods of a group of similar objects
        - A class contains properties and methods
            - Properties are the defining features of a class in terms of data
            - Methods are lines of code designed to perform particular tasks on the data
        - A Unified Modelling Language (UML) diagram or class diagram is typically used to illustrate OOP concepts
            - Private properties and methods are indicated by a '-' sign
            - Public properties and methods are indicated by a '+' sign
        - 4 Types of Methods of a Class
            - Constructor
                - The __init__ function allocates storage when an object of a class is created
            - Accessor/Getter
                - The 'get' functions access the data stored in an object
            - Mutator/Setter
                - The 'set' functions modify the data stored in an object
            - Utility
                - These methods extend the functionality of a class, but are not inherently accessor or mutator methods
        - Defining a Class in Python

```
class <name>(<optional superclass>):

  def __init__(self, <optional parameters>):
    <constructor body>

  def <method name>(self, <optional parameters>):
    <method body>
  ...
```

- An object is a specific instance of a class that has the same properties and methods as the class from which it is built

▼ Encapsulation

- Encapsulation refers to the concept of bundling properties and methods together as a package

▼ OOP classes utilise encapsulation to hide the internal representation of an object from the outside in a process known as information hiding

  - Keeping the properties of objects private protects them from being accidentally or intentionally modified by unauthorised parties
  - However, since the users need a way to access them, a set of methods are made public

▼ Encapsulation also allows for implementation independence, as the developer can code classes and their associated methods in any way he wishes to, as long as it fulfils the requirements of the class

  - The end user only needs to know how to use the methods of the class
  - The end user does not need to know how the methods perform their function

▼ Inheritance

- Inheritance refers to the concept of properties and methods in one class being shared with its subclass
- In OOP, the subclass inherits all the properties and methods of the superclass, but the subclass may have additional functionality from the

superclass with the addition of certain properties or methods

- Inheritance promotes software reuse/code reuse as it allows developers to create subclasses that reuse code declared already in a superclass
  - Software reuse/code reuse refers to the use of existing software, or software knowledge, to build new software
  - Software reuse/code reuse saves time and money
- Defining an Inherited Subclass in Python

```
class <name>(<superclass>):

  def <additional method name>(self, <optional parameters>):
    <method body>
  ...
```

- Polymorphism
  - Polymorphism refers to the concept of an object being able to take on multiple forms, where inherited subclass methods can be used in different ways
  - Polymorphism promotes extensibility as it is implemented through method overriding, which refers to the redefining of the implementation of a method provided by the superclass
  - Polymorphism enables code generalisation as a method with a specific name can behave slightly differently depending on the subclass that the method is acting on, leading to more concise code and allowing for easier maintainability
  - Method Overriding in Python

```
class <name>(<superclass>):

  def <overiding method name>(self, <optional parameters>):
    <method body>
  ...
```

- Built-In Modules in Python
    - random
        - random( )
            - A random float in the range [0.0, 1.0)
        - randint(a, b)
            - A random integer in the range [a, b]
        - randrange(stop)
            - A random integer in the range [0, stop)
        - randrange(start, stop)
            - A random integer in the range [start, stop)
        - randrange(start, stop, step)
            - A random integer in the range [start, stop) in intervals of step
        - shuffle(lst)
            - Shuffles the element in the list lst
    - math
        - trunc(n)
            - Returns the integer that is the truncated integer part of n
        - floor(n)
            - Returns the integer that is n rounded down to the nearest integer
        - ceil(n)
            - Returns the integer that is n rounded up to the nearest integer
        - pow(n, x)
            - Returns the float when n is raised to the power of x
        - exp(n)
            - Returns the float when e is raised to the power of n
        - log(n)
            - Returns the float that is the logarithm to the base of e of n

- ▼ sqrt(n)
    - Returns the float that is the square root of n
- ▼ datetime
  - ▼ datetime.now( )
    - Returns a datetime object representing the current date and time
  - ▼ datetime(year, month, day, {hour, {minute, {second, {microsecond}}}})
    - Returns a datetime object representing the specified date and time
  - ▼ datetime.strptime(str, format)
    - Returns a datetime object from a given string str of a given format
  - ▼ <datetime>.strftime(format)
    - Returns a string of a given format representing the datetime object
  - ▼ <datetime>.isoformat( )
    - Returns a string representing the date and time in ISO 8601 format (YYYY-MM-DDTHH:MM:SS)
  - ▼ <datetime>.year( )
    - Returns an integer representing the current year
  - ▼ <datetime>.month( )
    - Returns an integer representing the current month
  - ▼ <datetime>.day( )
    - Returns an integer representing the current day
  - ▼ <datetime>.hour( )
    - Returns an integer representing the current hour
  - ▼ <datetime>.minute( )
    - Returns an integer representing the current minute

- ▼ <datetime>.second( )
    - Returns an integer representing the current second
- ▼ <timedelta>.days( )
    - Returns an integer representing timedelta in terms of a number of days
- ▼ <timedelta>.seconds( )
    - Returns an integer representing timedelta in terms of a number of seconds