ICS 104 – Introduction to Programming in Python and C

Objects and Classes

Background

- Chapter 9: Sections 1 - 7

Learning Outcomes

- To understand the concepts of classes, objects and encapsulation.
- To understand how to define classes and instance variables.
- To be able to design, implement and test your own classes.

Object-Oriented Programming

- Procedural paradigm has developed over programming to decomposing tasks into functions.
- Functions are verbs.
- Writing a suitable machine to monitor a system.

- We will now study Objects and Classes
- Objects are the building blocks of modern programs
- To model objects we use the concept of classes

Classes

- A "class" describes objects with the same behavior.
- Every example of a class defines an object-oriented interface that hides a certain capacity and chaos.

Objects and Programs

- The object-oriented paradigm decomposes your program by decomposing tasks into functions.
- Functions are verbs, objects are nouns.
- It is difficult to understand and update a program that consists of a large collection of functions.

- To encourage the development of reusable code, we have a new **object-oriented programming**, a programming style in which tasks are solved by collaborating objects.
- Each object has its own set of data, together with a set of operations that work on the data.

- The frame already implemented the programming tasks when you used strings, lists, and the objects.
- Strings provided the ability to perform tasks.
- For example, you can ask the frame if it knows methods to operate on the objects.

Python Classes

- A class describes a set of objects with the same behavior.
- Every class has a set of methods that can be applied to its objects.
  - This says specifies how a string behaves in response to method calls, such as upper, find and replace methods on strings.
- For example, when you have a str object, you can invoke the upper method:

```
"Hello, World".upper()
```

String object | Method of class String

- In contrast, the list class describes the behavior of objects that can be used to store a collection of values.
- This class has a different set of methods.

- For example, the following can never be legal - the list class has no upper method:

```
[1, 2, 3].upper()
```

- However, let's say a string method, and the following call is legal:

```
"Hello, World".find("World")
```

```
>>> "Hello, World".find("World")
>>> "Hello, World".find("World")
>>> "Hello, World".find("World")
```

- A constructor initializes the object's instance variables
- The constructor is called automatically when an object is created
- This constructor is defined using the special method name `__init__()`
- Default arguments can be used with a constructor to provide different ways of creating an object