



# OOP With Python

## Unit 02 - Classes

{<coding:lab>}

# Quick Review of Prior Sessions

---

## Check Point 20 (Before we start)

- All students must be able to
  - Read from and write to files
  - Use persistent storage with Pickle module

# Object Oriented Programming

Introduction to Object Oriented Programming

## What is Object Oriented Programming

- An approach to programming
- Groups functions and variables together to create classes.
- Each class can be used to create objects
- Objects will share the same variables and functions as the class
- A function in a class is called a method
- A variable that's part of a class is called an attribute.

## Defining a Class

- Create a class by doing the following

```
1  #Define a class name: Rectangle
2  class Rectangle(object):
3      # __init__ is a function (known as initialiser/constructor) to initialise an object
4      # Initialise arguments of the object: width, height
5      def __init__(self, width, height):
6          self.width = width
7          self.height = height
8          self.area = width * height
```

Setting the attributes of the object

## Key Notes

- Good practice to capitalise the class names (to differentiate from functions)
- The `__init__()` method with `self` as argument always needed
- `__init__()` initialises the class – tells Python what to do when you use it for the first time in a program

## Creating an Object Instance

- Create an object by calling the class name with the initial arguments (8)

```
1  #Define a class name: Rectangle
2  class Rectangle (object):
3      def __init__(self, width, height):
4          self.width = width
5          self.height = height
6          self.area = width * height
7
8  #Creating an object named 'smallRect' whose width = 4, height = 6
9  smallRect = Rectangle(4,6)
```



## Updating the Attributes

- Update the value of the attribute like the way you update a variable. For

```
1 #Creating an object named 'smallRect' whose width = 4, height = 6
2 smallRect = Rectangle(4,6)
3
4 #The original width of smallRect is: 4 and the area is: 24" will be printed
5 print("The original width of smallRect is: " + str(smallRect.width) + " and the area is: " + str(smallRect.area))
6
7 #Updating the width attribute of smallRect
8 smallRect.width = 15
9 smallRect.area = smallRect.width * smallRect.height
10
11 #The updated width of smallRect is: 15 and the area is: 90" will be printed
12 print("The updated width of smallRect is: " + str(smallRect.width) + " and the area is: " + str(smallRect.area))
```

## Methods

- Methods are functions associated with the class
- Like functions, they allow for reusability of code
- They can be used (only) by the instances of that class

## Methods - Defining a Method

- Define it the same way you define a function within a class, for example

```
1  #Defined Method
2  def resize(self, factor):
3      self.width = self.width * factor
4      self.height = self.height * factor
5      self.area = self.width * self.height
```

## Methods - Calling a Method

- Call a method for that instance by using the dot notation (8)

```
1 #Creating an object named 'smallRect' whose width = 4, height = 6
2 smallRect = Rectangle (4, 6)
3
4 #The original width of smallRect is: 4 and the area is: 24" will be printed
5 print("The original width of smallRect is: " + str(smallRect.width) + " and the area is: " + str(smallRect.area))
6
7 #Calling the defined method, "resize"
8 smallRect.resize(1.5)
9
10 #The updated width of smallRect is: 6.0 and the area is: 9.0" will be printed
11 print("The updated width of smallRect is: " + str(smallRect.width) + " and the area is: " + str(smallRect.area))
```

## Methods - Returning a Value with method

- Similar to a function, a method can have a return value. For example

```
1 # Method to return parameter length
2 def perimeter(self):
3     | return (self.width + self.height) * 2
4
5 #Calling a method with return value
6 perimeter = smallRect.perimeter()
7 print(perimeter)
```



## Let's Experiment with Creating and Using Class and Objects (Demo/Practice - 21)

- Write a “Rectangle” class which allows the user to create objects representing rectangles. Note that a rectangle is characterized by:
  - Length of its width
  - Length of its height
- Write methods to return:
  - Area of the rectangle
  - Perimeter of the rectangle

## Check Point - 21

- Every student must be able to
  - Define a class with attributes and methods
  - Create an object
  - Access/Update attributes and call the methods
- For students who are waiting, try the following:
  - Continue working on your To-Do List



## Additional Notes - Class Attributes

- Class Attributes
  - A variable which is shared across all object instances in a class
  - For example the shapeName variable is a class attribute

```
1  #Define a class name: Rectangle
2  class Rectangle (object):
3      shapeName = "Rectangle"
4      def __init__(self, width, height):
5          self.width = width
6          self.height = height
7          self.area = width * height
```



## Additional Notes - Public Accessibility of Attributes

- In other OOP languages like Java, there is the concept of the accessibility of an attribute of an object
- For example, in Java, an attribute could be private/protected/public
- In Python, we don't worry too much about this
- All the attributes are public by default
- Possible to change. But we won't be learning
- Visit <https://docs.python.org/3/tutorial/classes.html> to learn more

---

# Advanced To-Do List

Create a To Do list program

## Advanced To Do List Program Specifications (Demo/Practice - 22)

- Improve on your To-Do List program.
- Each to do item will have three attributes:
  - To Do Description
  - Priority
  - Due Date
- Similar to previous to do list program, it allows user to create, display update and delete to do items. The list will be stored on persistent storage
- Make sure your program is robust



## Check Point - 22

- All students must be able to create a to-do list program with object oriented programming
- For Students who have completed and waiting, try to
  - Develop a student records database
  - Develop a class which can handle irrational numbers
    - Holds an irrational by recording its numerator and denominator
    - Methods to compute additions, subtraction, multiplication and division

{<oding:lab}



---

# Reflections

## What is Object Oriented Programming

- An approach to programming
- Groups functions and variables together to create classes.
- Each class can be used to create objects
- Objects will share the same variables and functions as the class
- A function in a class is called a method
- A variable that's part of a class is called an attribute.

## Questions to Ask Yourself

- Why do we use object oriented programming?
  - Code reusability
  - Clear modular structure
  - Easy to maintain and modify existing codes