

OOP With Python

Unit 02 - Classes

{<oding:lab}</pre>



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

1111111

Create a class by doing the following

```
#Define a class name: Rectangle
class Rectangle (object):

# __init__ is a function (known as initialiser/contsurctor) to initialise an object

# Initialise arguments of the object: width, height

def __init__(self, width, height):

self.width = width

self.height = height
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

1111111

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

```
#Define a class name: Rectangle
class Rectangle (object):
def __init__(self, width, height):
self.width = width
self.height = height
self.area = width * height

#Creating an object named 'smallRect' whose width = 4, height = 6
smallRect = Rectangle(4,6)
```



Updating the Attributes

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

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

a



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 <u>within a class</u>, for example

```
    #Defined Method
    def resize(self, factor):
    self.width = self.width * factor
    self.height = self.height * factor
    self.area = self.width * self.height
```



Methods - Calling a Method

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

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

```
# Method to return parameter length
def perimeter(self):
    return (self.width + self.height) * 2

#Calling a method with return value
perimeter = smallRect.perimeter()
print(perimeter)
```

- Let's Experiment with Creating and Using Class and Objects AGUE (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





- 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

```
#Define a class name: Rectangle
class Rectangle (object):
  shapeName = "Rectangle"
  def init (self, width, height):
     self.width = width
     self.height = height
     self.area = width * height
```



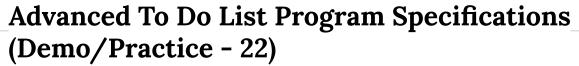
Additional Notes - Public Accessibility of Attributes

- 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





- 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



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.





- Why do we use object oriented programming?
 - Code reusability

- Clear modular structure
- Easy to maintain and modify existing codes