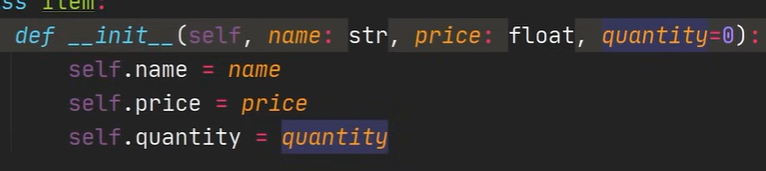
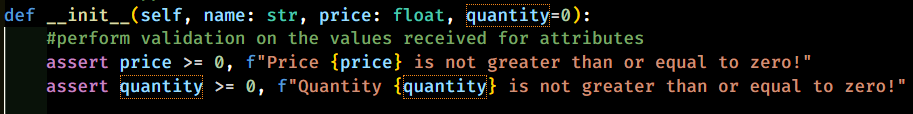
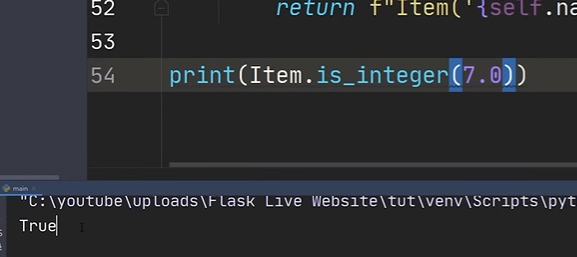
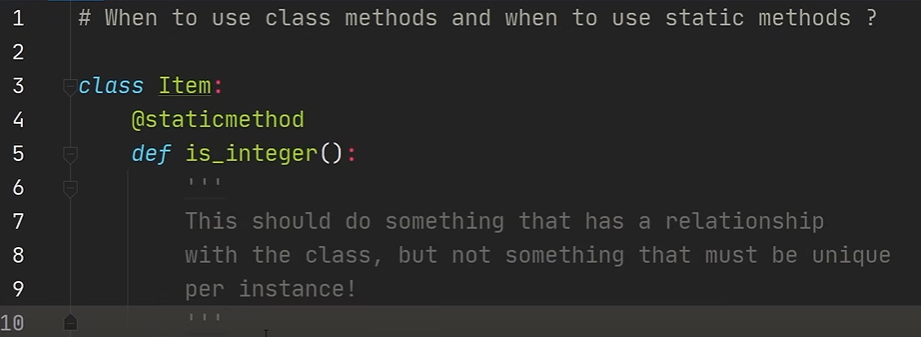
PYTHON OOPs PROJECT NOTES

# General Concepts

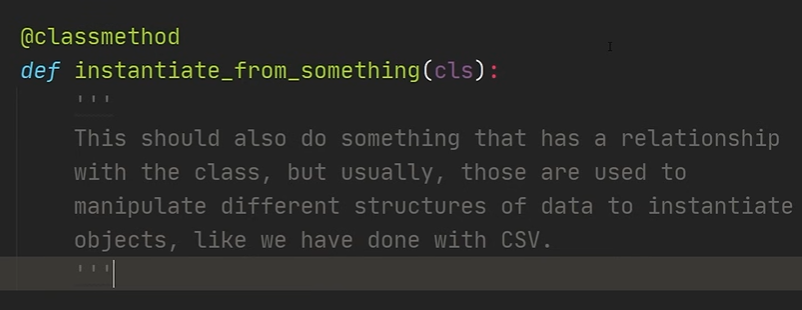
* functions in python that have `\_\_` at the start and end (Like `\_\_init\_\_()` ) are known as magic methods.
* \_\_init\_\_()` is also known as constructor. Runs automatically when an instance of class is created.
* We can add default values for parameters inside constructors.   
  Eg. \_\_init\_\_(self, name=’noname’)
* Note that constructor should only be used to add general attributes. If you want to add specific attributes to your instances, you can always do that separately.
* Specifying types that a class can accept:  
    
  Here, we specify types as attribute: type. Notice that we do not specify type of attribute ‘’quantity’’ as it’s default value is set to the type we want, that is integer.
* We can validate arguments received using the assert statement as following:  
  

# Class methods and static methods

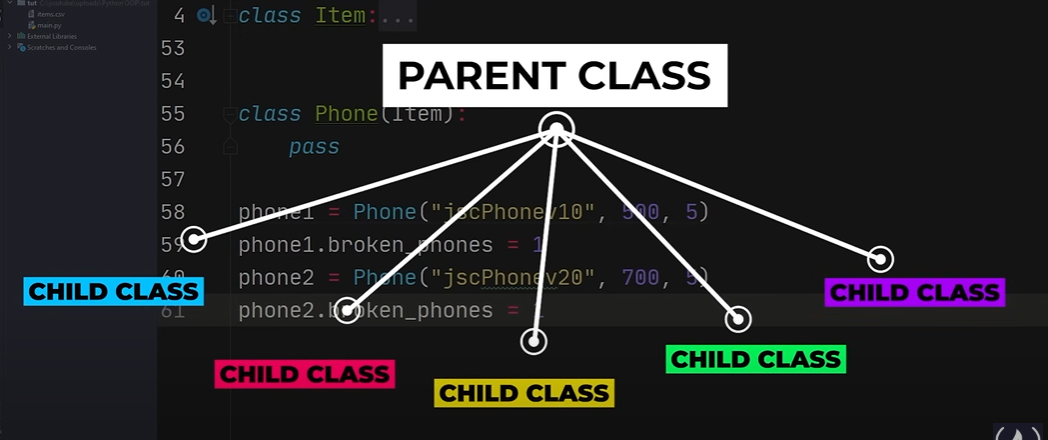
## Static methods

* + Static method is a method that is logically connected with the class and does some work from you. We define it with it’s decorator: ‘@staticmethod’
  + Static methods never take instance ‘self’ as the first argument.
  + We can simply access the static methods outside the class by using classname.static\_method\_name() without even instantiating the class, and as long as the arguments are satisfied, you don’t need to put arguments only associated with the class. (i.e. it is not unique to an instance but it is related to the class.)
  + 
  + sasd

## Class methods

* Example:  
  
  + One application of class method is for instantiating from a CSV file or YAML file, where we want to make instances\* from an external file containing structured data that you own, like a csv or yaml file. It is not a helper function like static method, but instead a function that creates instances of the class. This is the recommended way to instantiate data (best practice way).
* The main difference between a class method and a static method is that we do not have to pass a mandatory class reference (cls) in static method.
* Class and static methods can be called from instance level as well, but there are not a lot of good reason to call them from instance level. You can call both of them by just mentioning class name and calling them. (eg. Classname.method())

# INHERITANCE



It is not a good idea to manually assign attributes and methods once we create an instance of the class. The better way to do that is by creating a child class and adding methods to it. (Because methods may be different for different types of child classes. Like Phone class can have attribute broken phones that is specific to phone, but broken phones will not apply to Laptop. But since both are part of the main class, we create a separate child class called Phone and assign broken\_phones to it.)

* **Super Attribute:** The super attribute allows us to have access to all the attributes of the parent class. This makes sure that we don’t have to keep copying the init, error checks and assigning to attribute code from the parent class to every child class we create.

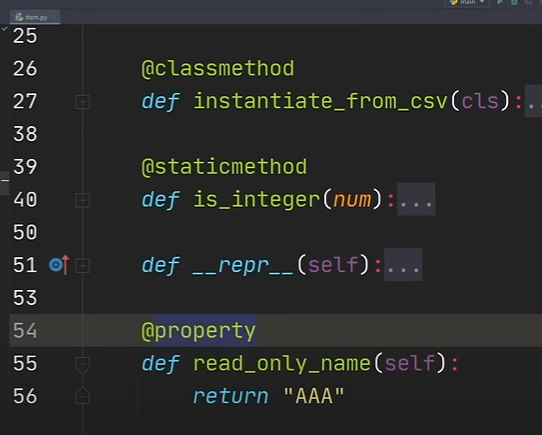
# Reorganization of Code:

* To improve readability, we reorganize our code. We create two separate files for the item class and phone class, and use the main class only for calling their instances and writing action code.

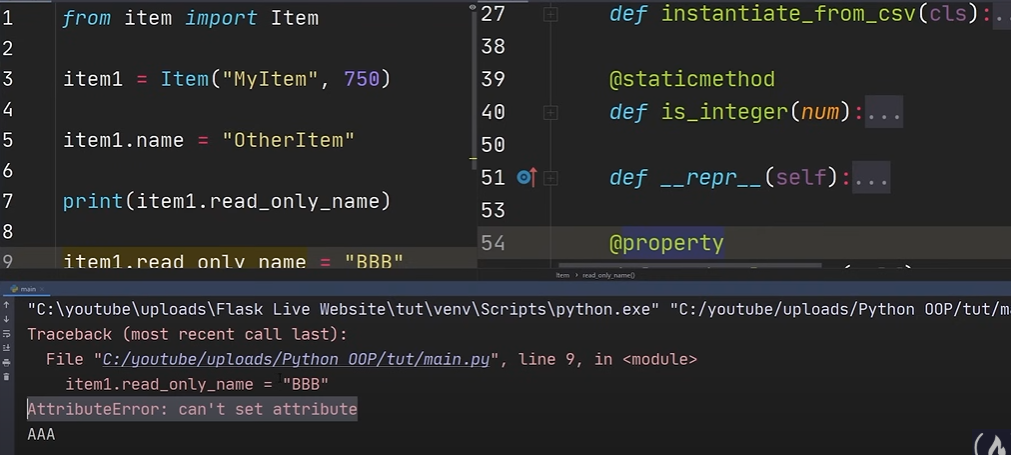
# Getters and Setters

Property Decorator

In item class code we set property decorator for function read\_only\_name

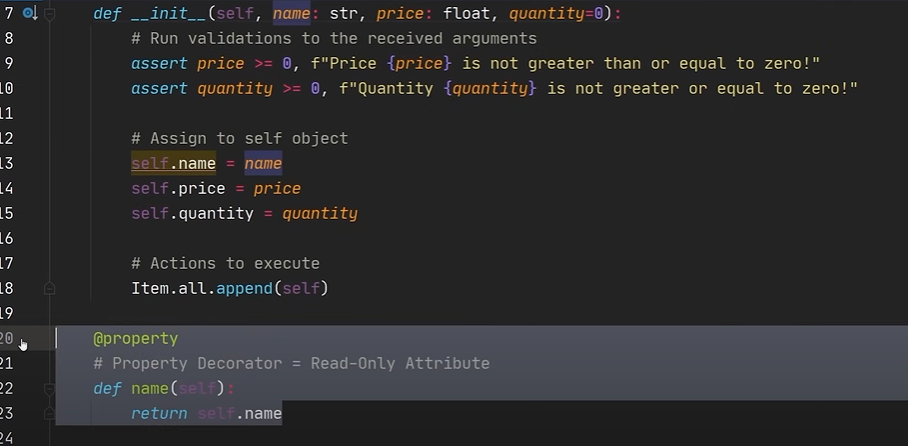


Output if we try to set value for read\_only\_name:

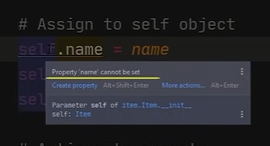


Scenario: If we wanted to set name as a read only property.

We might think that we can use property decorator like above.



However, since we cannot change the value of name, self.name = name will be an illegal assignment, since name cannot change it’s value. So this doesn’t work.



To work around this, python has access modifiers: private and protected.

* Protected: Add one underscore before your variable name (\_variable). Protected variables are only visible in the class that they belong to and the subclasses of that class.
* Private: Add two underscores before your variable name (\_\_variable). Private variables are only visible in the class that they belong to.

We can still print variable from an instance of the class, but we cannot modify it’s value. Meanwhile for protected variable, the code editor will show an error when we mention it in the first place, because the user will be led into believing that it doesn’t exist when called. Outside the class, it doesn’t exist, so it cannot be used or called on it’s own.

Getters: Get value of access protected members. Use: with the ‘@property’ decorator.

Setters: Set value of access protected members. Use: ‘@variable\_name.setter’ decorator.

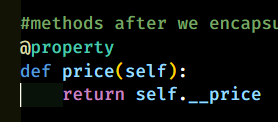
We can also use validations in the getter/setter functions.

# OOP Principles

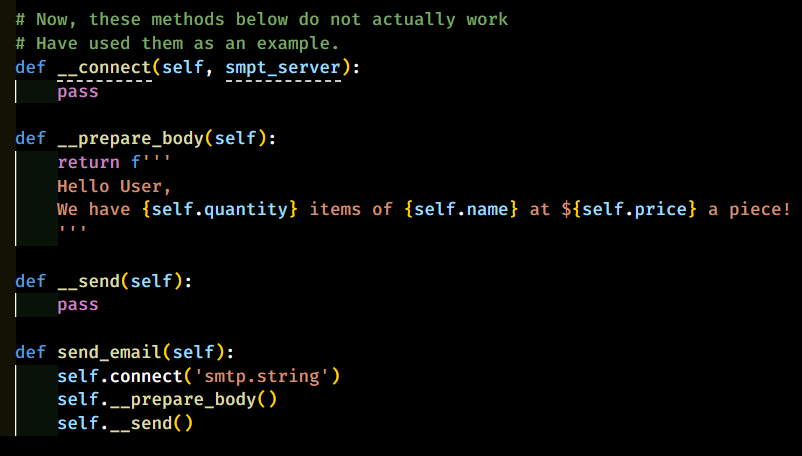
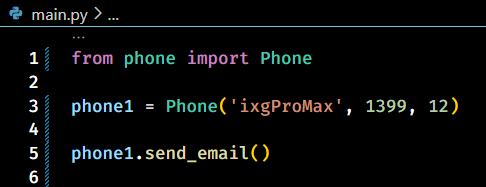
Object Oriented Programming has 4 major principles that we must keep in mind while designing large programs.

* Encapsulation
* Abstraction
* Inheritance
* Polymorphism

## Encapsulation

* Most things explained in code comments
* If we do not specify '--price' here (and just type it as self.price), it will result in recursion calls exceeded error. This is because the function will call the method below, which will in turn call the function since private attribute price can't be accessed, and it will go on.

## Abstraction:

* The process of hiding unnecessary details from the user. Hide (or abstract) unnecessary information as much as possible from the instances.
* For example, in the scenario where we want to send the email, it makes sense to hide the functions that constitute the process of connection with smtp server, preparing mail body, sending the mail etc… from the end user.
* 
* In main file, just call like this:  
  

## Polymorphism

A single function knows how to handle different kinds of objects as expected.

* In our project, the method ‘apply\_discount()’ knows how to work properly both for the Phone class and the Item class. (Since ‘Phone’ inherits the method from the ‘Item’ class.)
* Overriding in child class is legal, so you can always change attribute and method values/behavior after inheritance.
* Method Overloading vs Method Overriding: <https://www.geeksforgeeks.org/difference-between-method-overloading-and-method-overriding-in-python/>