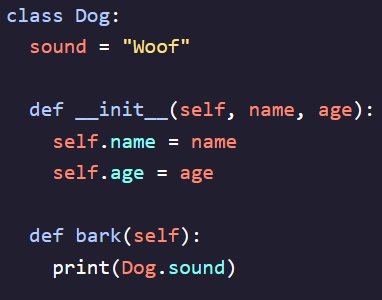
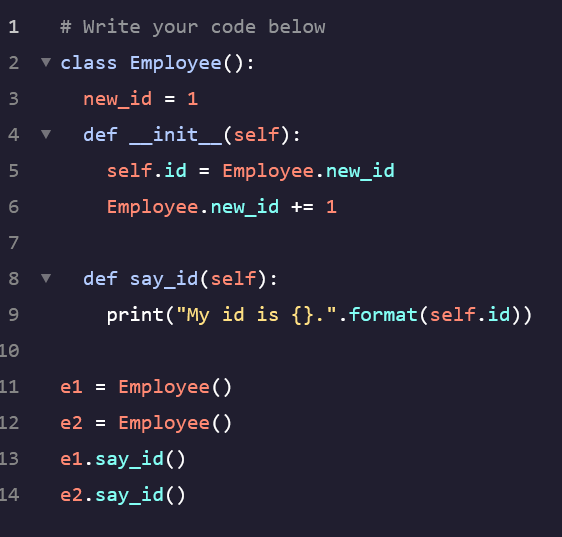
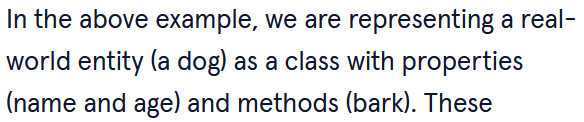
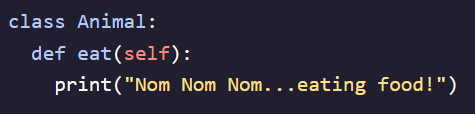
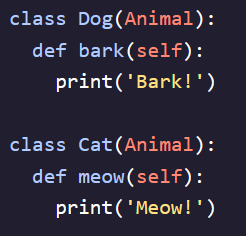
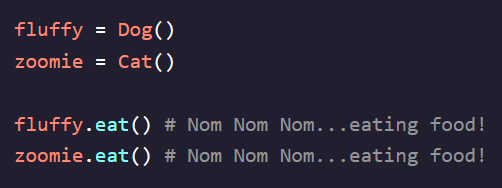
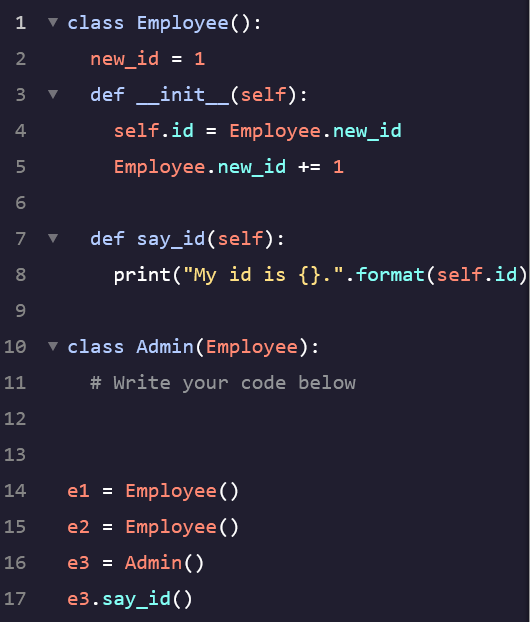
**Programming Paradigm** - a way to classify different programming languages and the unique features that they offer  
- Code may fall into multiple paradigm categories as modern languages offer more than one specific paradigm we can program in   
- In *OOP* there must be the ability to create programs based around classes and objects  
   


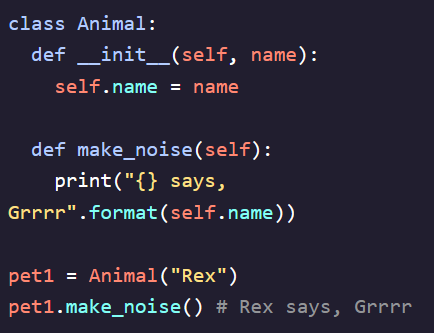
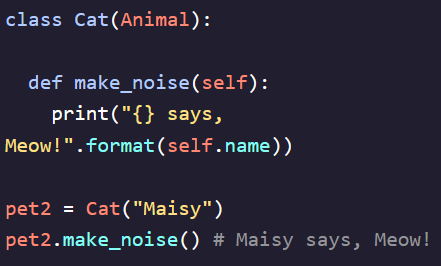


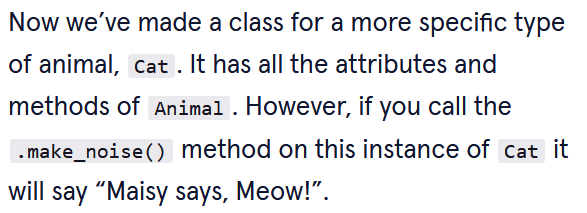
**OOP Pillar: Inheritance:**

- Can create a global class that contains methods that other classes can call and use in their own code  
- This allows us to cut down on code reuse and prevent writing the same code into each method  
    


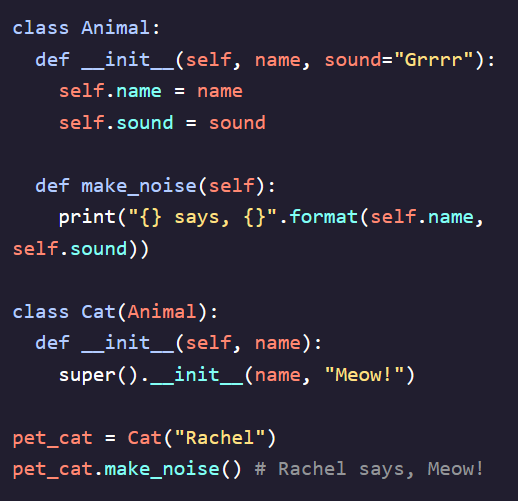


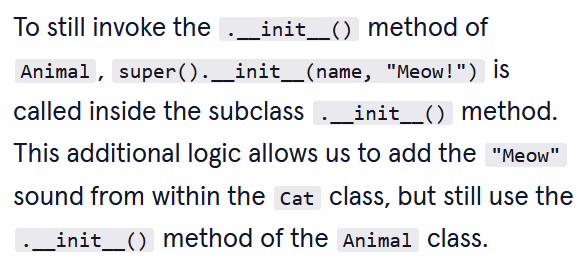
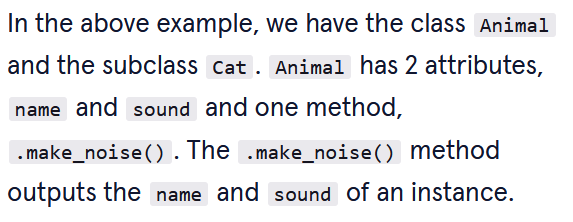
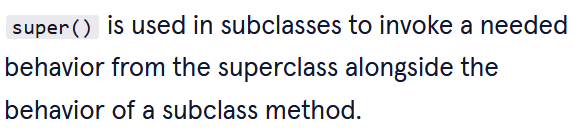
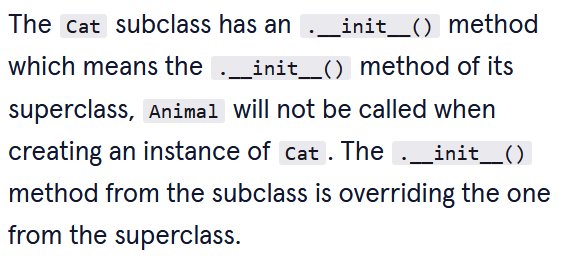
**Overriding Methods:**

- Useful when we want to override the behavior of a child class from a parent class  
- An overriding method in a subclass is one that has the same definition as the parent class but contains different behavior  
 

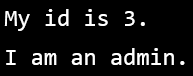


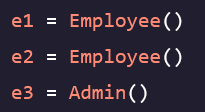
**super():**

- When overriding methods we sometimes want to still access the behavior of the parent method  
- *super()* gives us a *proxy object* that we can use to invoke the method of an objects parent class (aka superclass)  


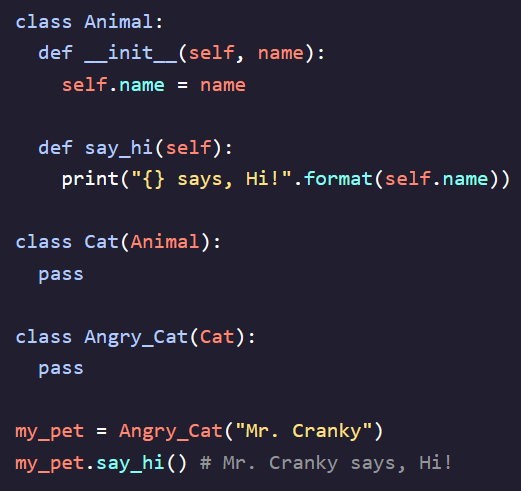
  


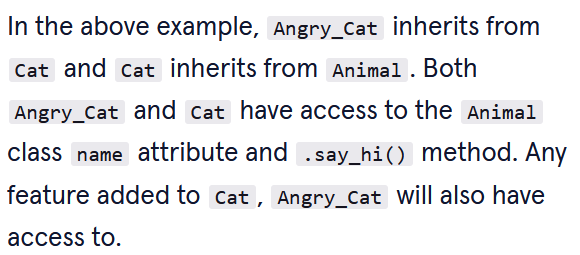


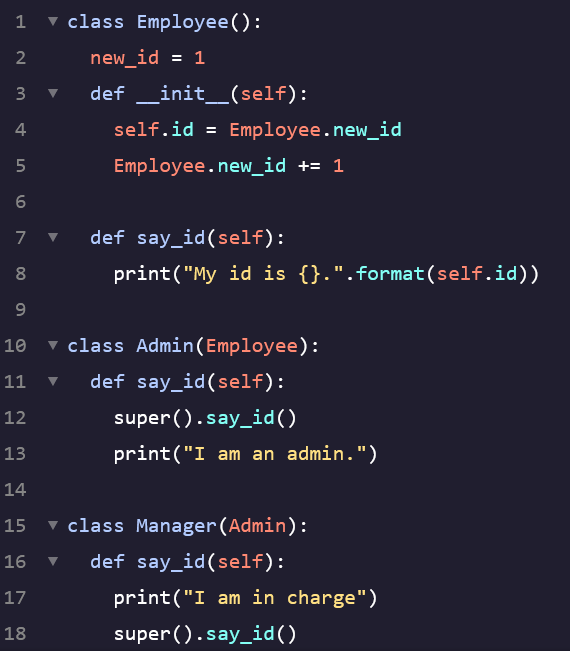
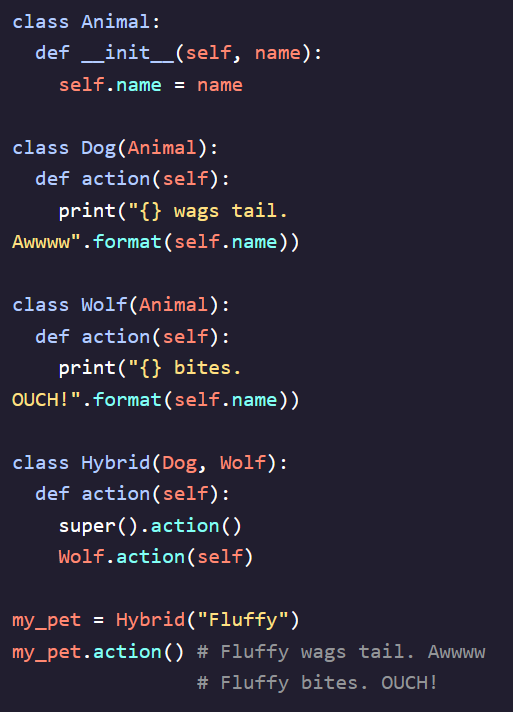


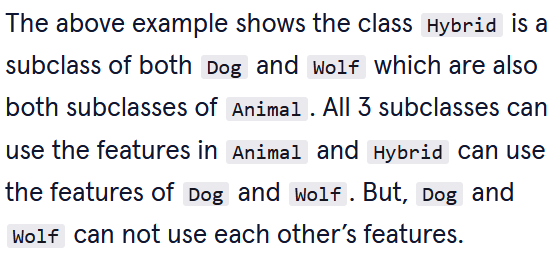


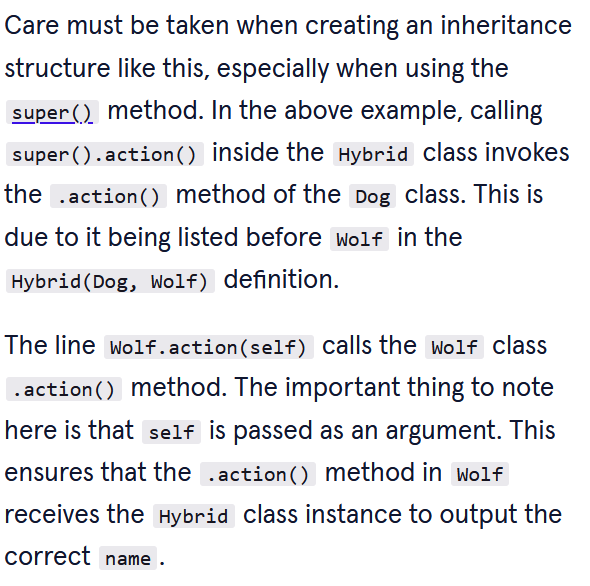
**Multiple Inheritance:**

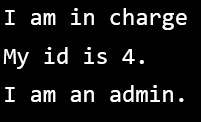
- This is a feature in Python where a subclass inherits from more than one superclass (superclass 🡪 super-superclass 🡪 etc)  


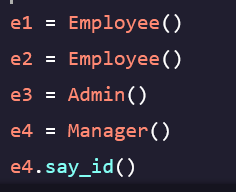


  
  
- Another form of inheritance involves a subclass that inherits directly from two classes and can use the attributes and methods of both  




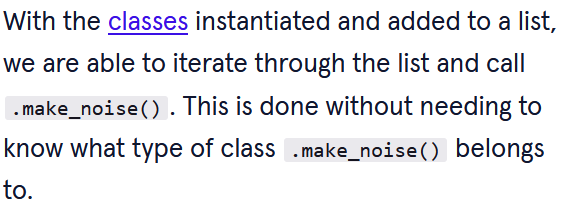


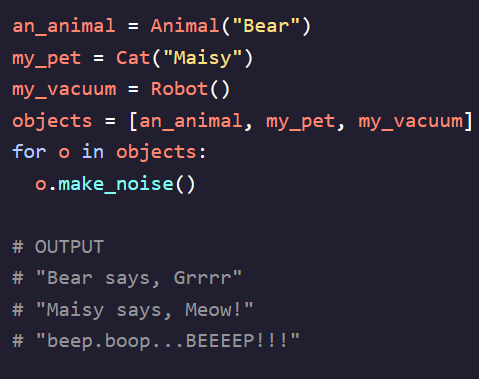


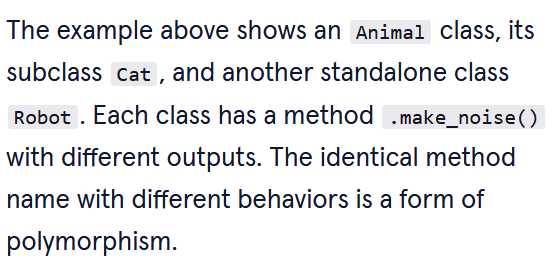


**OOP Pillar: Polymorphism:**

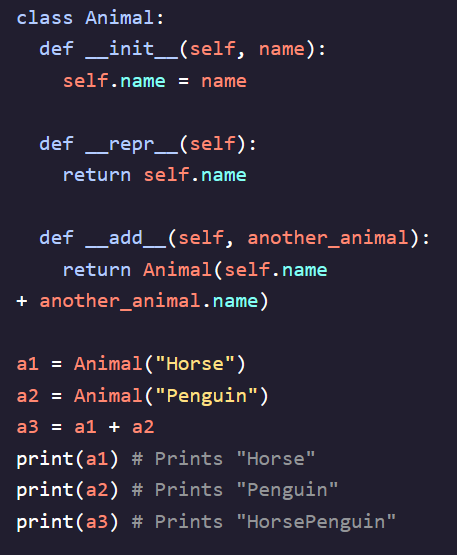
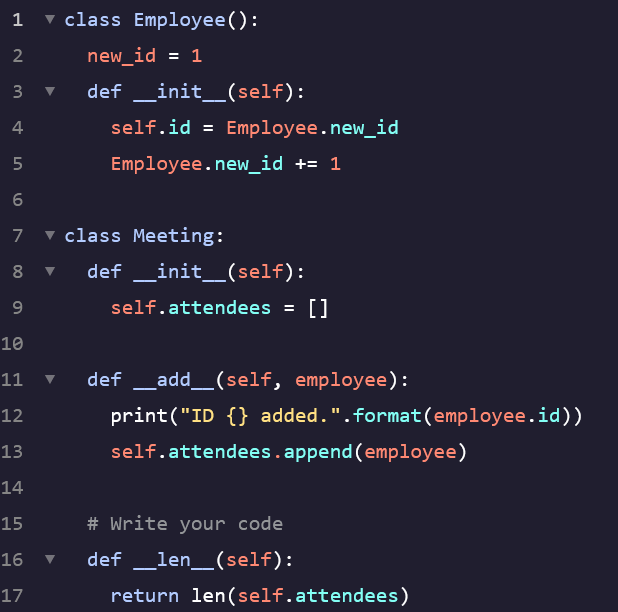
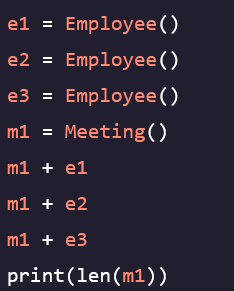
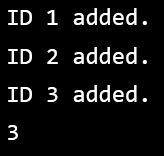
- The ability to apply an identical operation onto different types of objects  
- Useful when an object type may not be known at program runtime  

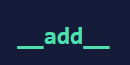



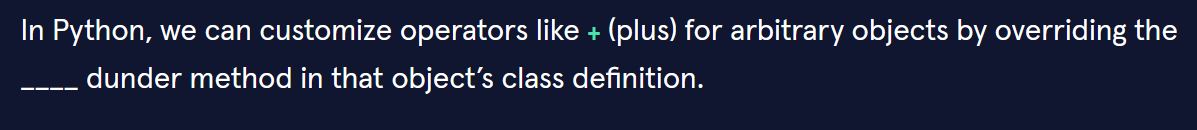


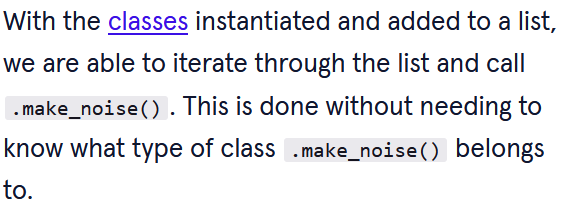
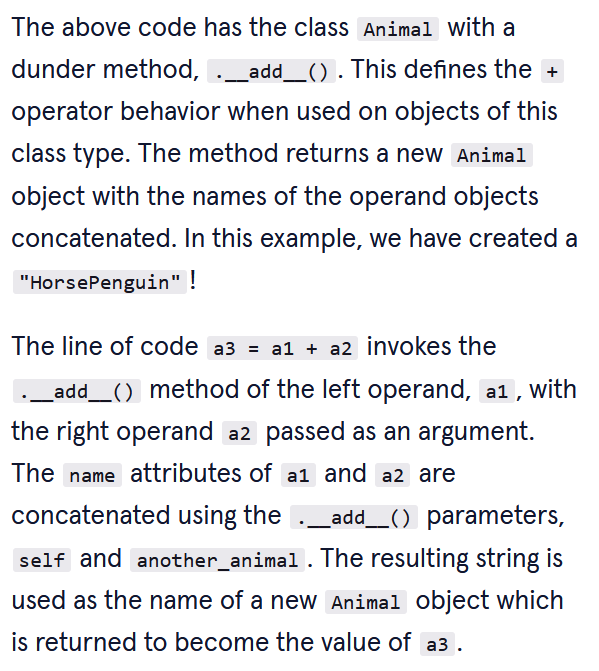


**Operation Overloading:**

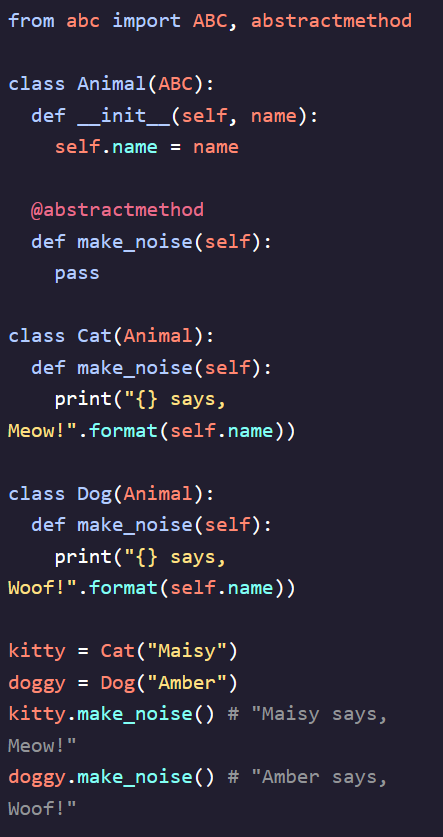
- Performed by defining a classes ***dunder methods*** – stands for double under, use a special syntax to perform class-specific operations in Python  
  
  

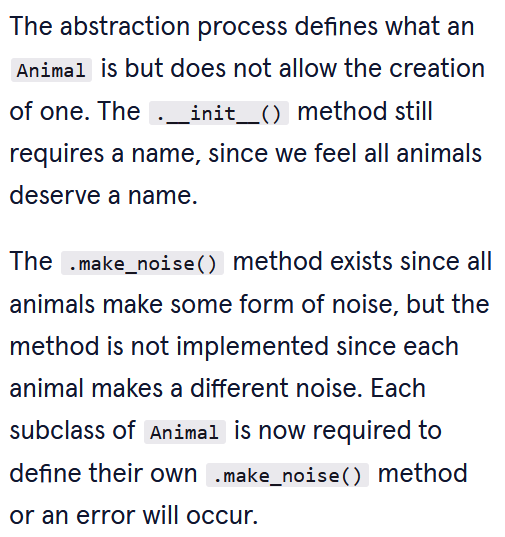
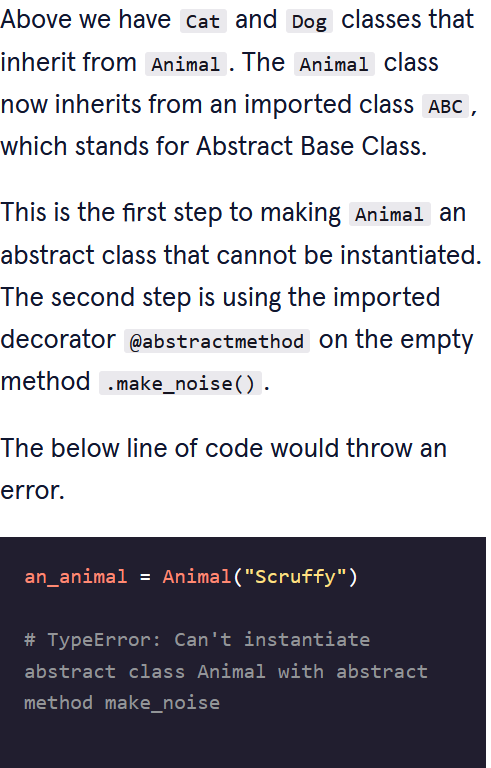


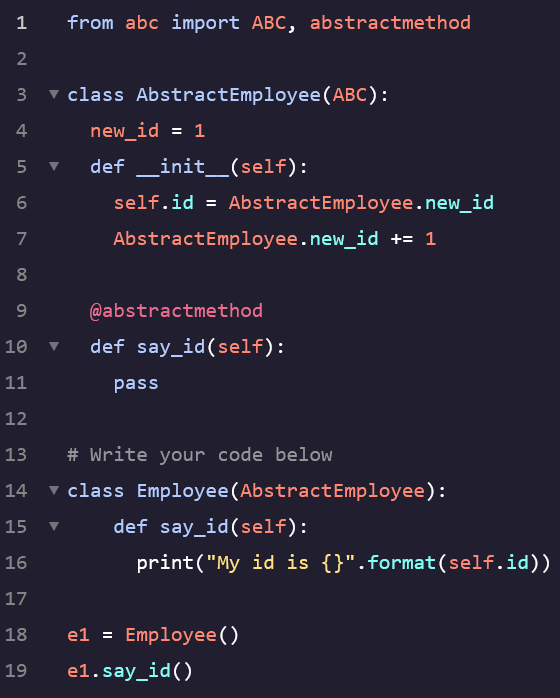


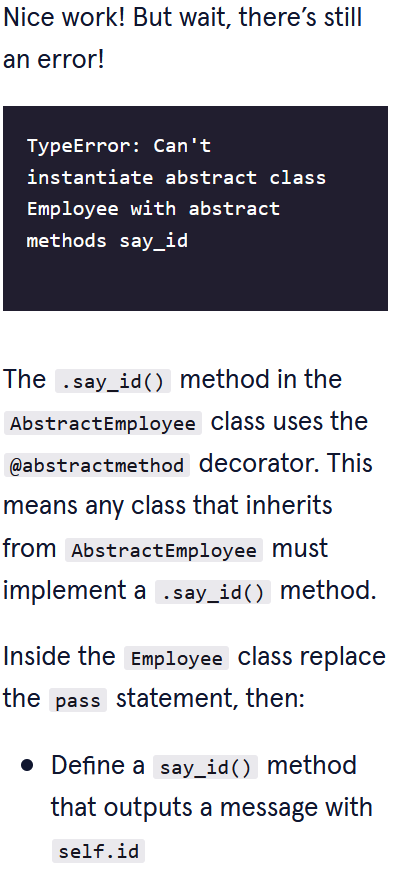


**OOP Pillar: Abstraction:**

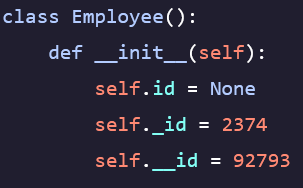
- Helps with code design by defining necessary behaviors to be implemented within a class structure  
- By doing so, it helps to avoid leaving out or overlapping class functionality as class hierarchies get larger  






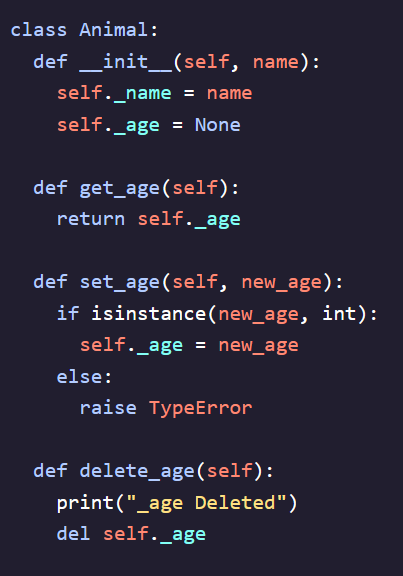


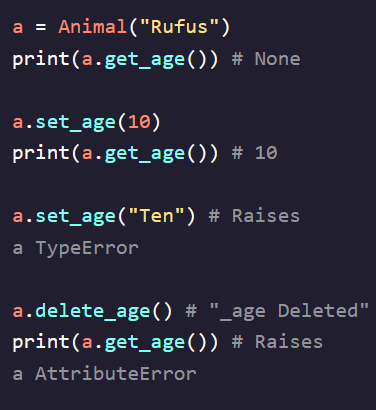
**OOP Pillar: Encapsulation:**

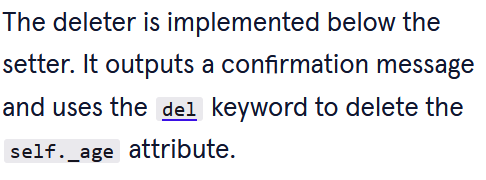


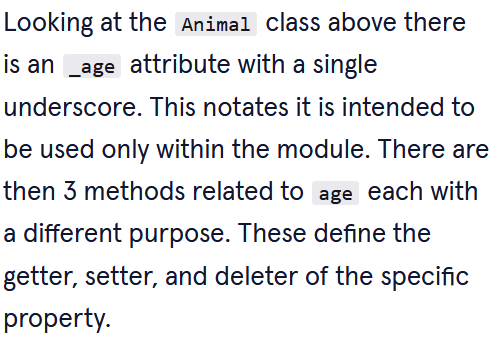
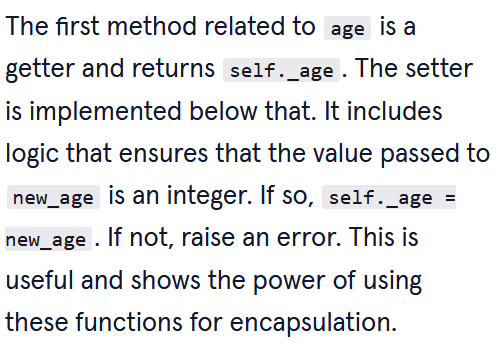
- The process of making methods and data hidden inside the object they relate to  
- Are accomplished by using access modifiers like:  
 - *Public* – can be accessed from anywhere *- Protected* – can only be accessed from code within the same module (*self.\_x)  
 - Private* – can only be accessed from code within the class that these members are defined (*self.\_\_x)*  
- All members are *public* in Python but a common convention is to use *self.\_x* to indicate that a member is protected. Use *self.\_\_x* to indicate that it is private  
- Accessing a protected member outside a module will not cause an error, developers add it to alert others that they should be careful when accessing this member  
- *Name Mangling* – Members that are preceded with two underscores have their names modified in the background to *obj.\_Classname\_x* – can still be publicly accessed but will prevent clashing member names during inheritance by future classes

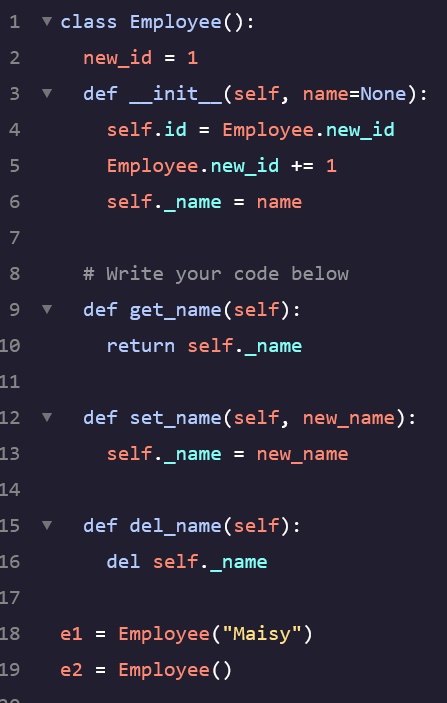
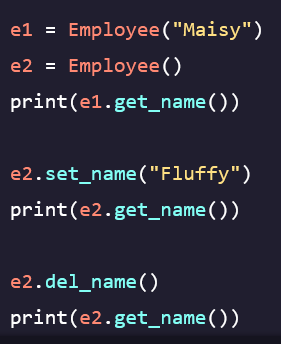
**Getters, Setters, and Deleters:**

- These functions are one way to implement encapsulation where the state of class attributes can be handled within the class  
- Useful in making sure that the data being handled is appropriate for the defined class functionality  






**End of Class Code:**

