Overriding base methods

A method can be declared in a derived class to override a matching method in the base class – if both method declarations have the same name and the same number of listed arguments. This effectively hides the base class method as it becomes inaccessible unless it is called explicitly, using the base class name for identification.

Where a method in a base class supplies a default argument value this can be used in an explicit call to the base method or alternative values can be supplied by overriding methods.

Person.py

1 Create a new Python script that declares a base class with an initializer method to set an instance variable and a second method to display that variable value

```
class Person :

        '''A base class to define Person properties.'''

        def __init__( self , name ) :
                self.name = name

        def speak( self , msg  = '(Calling The Base Class)' ) :
                print( self.name , msg )
```

Man.py

2 Next, create a script that declares a derived class with a method that overrides the second base class method

```
from Person import *

'''A derived class to define Man properties.'''

class Man( Person ) :
        def speak( self , msg ) :
                print( self.name , ':\n\tHello!' , msg )
```

Hombre.py

3 Now, create another script that also declares a derived class with a method that once again overrides the same method in the base class

```
from Person import *

'''A derived class to define Hombre properties.'''

class Hombre( Person ) :
        def speak( self , msg ) :
                print( self.name , ':\n\tHola!' , msg )
```