2.6 Inheritance: Thoughts on Design

THINGS THAT CAN BE DONE WITH AN INHERITED METHOD:

1. Simply inherit an implemented method
   - what the subclasses tend to do :use methods from the superclass
   - **Keep in mind**: *abstract methods from the superclass have to be overiden by the subclasses!*
2. Override an abstract method to implement it
3. Override an implemented method to replace it
   - *does the subclass need another behaviour?*
4. Override an implemented method to extend it
   - Use the behaviours from the superclass but add some more!
   - Any inherited method can be extended, not just the initializer

```
class SalariedEmployee(Employee):
    def pay(self, pay_date: date) -> None:
        Employee.pay(self, pay_date)  # Call the superclass method as a helper.
        print('Payment accepted! Have a nice day. :)')

>>> fred = SalariedEmployee()
>>> fred.pay(date(2017, 9, 30))
An employee was paid 3200 on September 30, 2017.
Payment accepted! Have a nice day. :)
```

- The client can write code to an interface defined once in the abstract class that will work for **any** of its subclasses!!!!!
- Polymorphic: Base classes that have multiple subclasses (taking many forms)

INHERITANCE VS COMPOSITION:

INHERITANCE:
- "Is a " relationship
- Any change in the superclass affects all its subclasses

COMPOSITION:
- "Has a" relationship