Object Oriented Programming

Class

- Class:
    - Template or definition for a collection of objects (not a single one)
    - These objects have a common set of properties and share the same data attributes and methods
- Object:
    - An instance of a class
    - Has its own values for each fo the data attributes
- Class Methods:
    - Public functions
    - Includes parameter self
    - Accesses class data attributes
- Encapsulation
    - Combining methods and attributes as a single object type
    - Usually public methods created to provide read/write access to a hidden attribute
    - Used in conjunction with data hiding
- Inheritance
    - Ability to create new classes (subclasses) which possess ALL data attributes and methods of the existing parent class (superclass)
- Polymorphism
    - Ability of different classes to respond to same methods in different ways
    - Methods have the same name
    - Subclass method overrides superclass method
- Method overloading
    - Methods / functions of the same name
    - Different number of parameters / different types of parameters

Data Attributes

- Data attributes / Properties:
    - Private data
        - Starts with "__"

        - E.g.self.__*name = name*
    - Prevents external functions from accessing
    - Only accessed by class methods
- Data hiding
    - Achieved by declaring data members to be private
    - Can only be accessed through public methods
- Reusability
    - Software components can be used in many different applications without having to modify code in the component
    - Reliability
        - Already tested and debugged
        - Developed by specialists
    - Saves time in software development
        - Decreases maintenance
- Advantages:
    - Ability to hide data
    - Reusability
    - Methods can be changed without affecting how they are used

Class Methods:

- Initialising class:

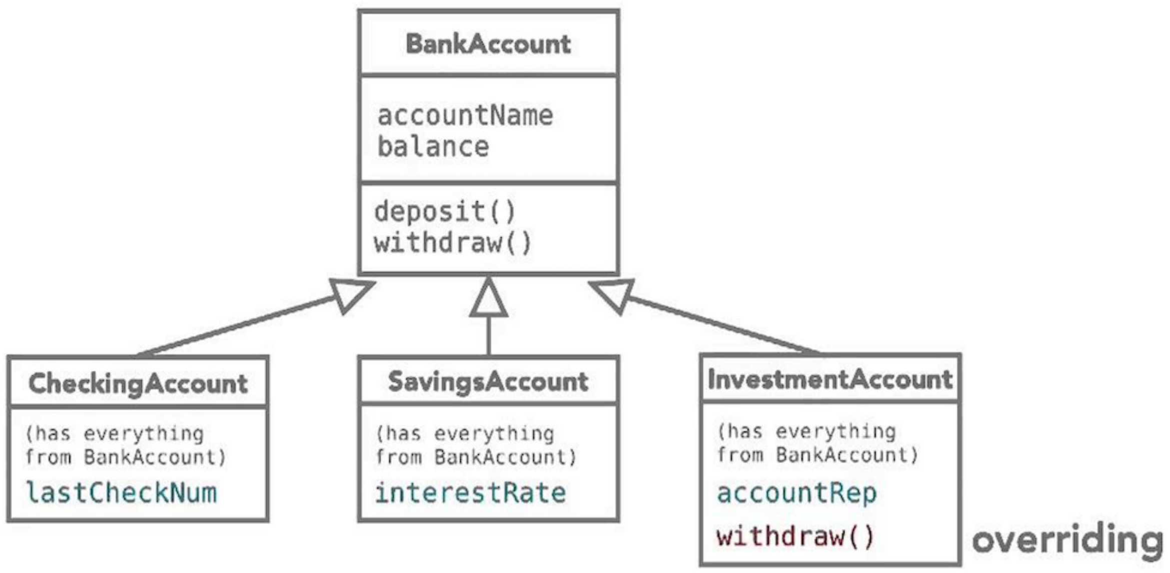
```
class ABC:
  def __init__(self, data1, data2):
     self.data1 = data1
     self.data2 = data2

  def display(self):
     print(self.data1)
     print(self.data2)
```

- Subclass inheritance & polymorphism:

```
class DEF(ABC):
  def __init__(self, data1, data2, data3):

     # initialise superclass
     super().__init__(data1, data2)
```

```
            self.data3 = data3

        # polymorphism (same name as function in
superclass)
        # overrides function superclass
        def display(self):
            print(self.data3)

        # output: data3
```

Inheritance Diagram:



BankAccount
accountName
balance
deposit()
withdraw()
CheckingAccount
(has everything from BankAccount)
lastCheckNum
SavingsAccount
(has everything from BankAccount)
interestRate
InvestmentAccount
(has everything from BankAccount)
accountRep
withdraw()
overriding