An Example Python Class

class Account(object):
    def __init__(self, name, balance):
        self.name = name
        self.balance = balance
    def deposit(self, amt):
        self.balance += amt
    def withdraw(self, amt):
        self.balance -= amt
    def inquiry(self):
        return self.balance
The Details

- **object** is the root of the class tree
  - use it as superclass when you are not extending another class

- __init__ implicitly defines instance variables

- self is a parameter to all instance methods
  - but you don’t include it in the parameter list when you call an instance method
  - technically you can use any identifier in place of self (but don’t do it!)

Augmenting the Example:

```python
class Account(object):
    num_accounts = 0
    def __init__(self, name, balance):
        self.name = name
        self.balance = balance
        Account.num_accounts += 1
    def __del__(self):
        Account.num_accounts -= 1
    def deposit(self, amt):
        self.balance += amt
    def withdraw(self, amt):
        self.balance -= amt
```

- `num_accounts` is a class variable
- `del` is a special method used for cleanup, etc.
More Details

- Class variables are like static variables in Java
  - They belong to the class not to an instance
  - All instances share a single copy of a class variable
- \_\_del\_\_ is often absent
  - It is used to do things such as update class variables (as in our example), close network connections, release locks, etc.
  - Calling del on an object does not necessarily invoke \_\_del\_\_ -- del reduces reference count

Creating Instances:

```python
from account import Account

checking = Account('David', 50000)
savings = Account('David', 100000)

# It's payday
checking.deposit(25000)

# Can I buy a 488?
if savings.inquiry() > 500000:
    print("Go shopping. Make sure it's red.")

print('Created: {}'.format(Account.num_accounts))
```

Create two Account instances

Self does not appear here

Access class variable

Prints 2
Exercise

• Exercise: create a Python class called Car
• This class should include:
  • Three attributes
    • make
    • model
    • Year
  • Class variable num_cars that tracks the number of cars created
  • An __init__ method
  • Method print_description that prints the attributes for a Car instance
  • Setter methods for each of the attributes
• Write a main method that creates a couple of Car instances and applies methods to them

Exercise solution

class Car(object):
    num_cars = 0
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        num_cars += 1
Exercise solution

def print_description(self):
    print('Make: {}'.format(self.make))
    print('Model: {}'.format(self.model))
    print('Year: {}'.format(self.year))

def set_make(self, new_make):
    self.make = new_make

def set_model(self, new_model):
    self.model = new_model

def set_year(self, new_year):
    self.year = new_year

Exercise solution

def main():
    f488 = Car('Ferrari', '488', '2019')
    mc40 = Car('MINI', 'Cooper S', '2004')

    f488.print_description()
    mc40.set_model('Cooper S MC40')

    print('Number of cars = {}'.format(Car.num_cars))

if __name__ == '__main__':
    main()