Using Inheritance

- Let's build an application that organizes info about people!
  - Person: name, birthday
    - Get last name
    - Sort by last name
    - Get age

Building a class

```
import datetime

class Person(object):
    def __init__(self, name):
        """create a person called name"""
        self.name = name
        self.birthday = None
        self.lastName = name.split(' ')[-1]

    def getLastName(self):
        """return self's last name"""
        return self.lastName

    # other methods

    def __str__(self):
        """return self's name"""
        return self.name
```

Building a class (more)

```
import datetime

class Person(object):
    def __init__(self, name):
        """create a person called name"""
        self.name = name
        self.birthday = None
        self.lastName = name.split(' ')[-1]

    def setBirthday(self,month,day,year):
        """sets self's birthday to birthDate"""
        self.birthday = datetime.date(year,month,day)

    def getAge(self):
        """returns self's current age in days"""
        if self.birthday == None:
            raise ValueError
        return (datetime.date.today() - self.birthday).days

    # other methods
```

How `plist.sort()` works

- Python uses the timsort algorithm for sorting sequences – a highly-optimized combination of merge and insertion sorts that has very good average case performance
- The only knowledge needed about the objects being sorted is the result of a “less than” comparison between two objects
- Python interpreter translates `obj1 < obj2` into a method call on `obj1` → `obj1.__lt__(obj2)`
- To enable sort operations on instances of a class, implement the `__lt__` special method

Building a class (more)

```
import datetime

class Person(object):
    def __init__(self, name):
        """create a person called name"""
        self.name = name
        self.birthday = None
        self.lastName = name.split(' ')[-1]


    def __lt__(self, other):
        """return True if self's ame is lexicographically
           less than other's name, and False otherwise"""
        if self.lastName == other.lastName:
            return self.name < other.name
        return self.lastName < other.lastName

    # other methods

    def __str__(self):
        """return self's name"""
        return self.name
```