OOP and Classes

Info 206

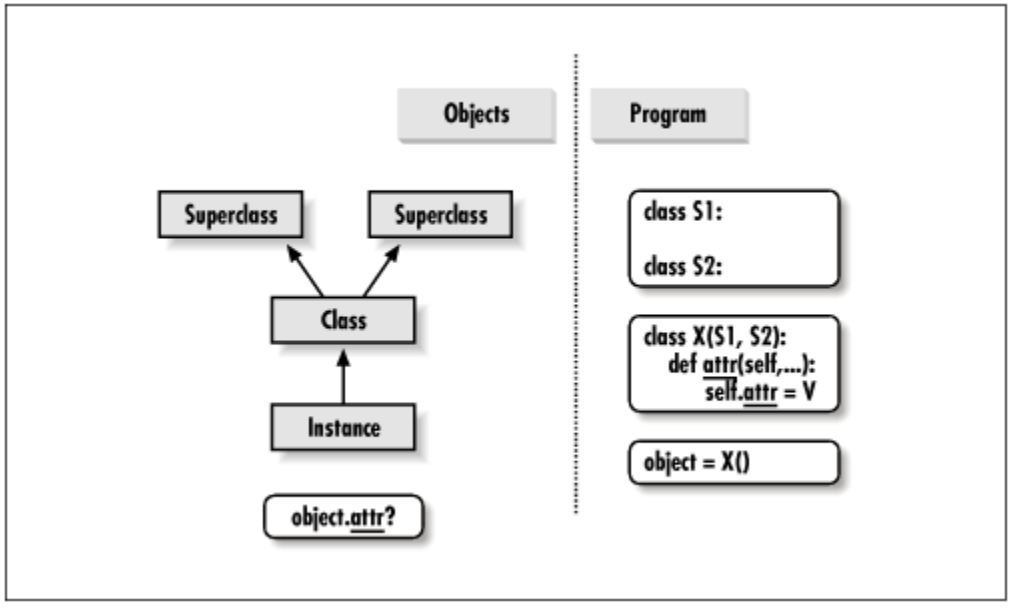
Niall Keleher

03 October 2017

Today's Outline

1. Class Inheritance
2. Exercise

Class Inheritance



Class Attribute Tree

Inheritance & Namespace

```
class Super:
  def method(self):
      print('in Super.method')          # Default behavior
  def delegate(self):
      self.action()                     # Define some action

class Inheritor(Super):                 # Inherit method verbatim
  pass

class Replacer(Super):                  # Replace method completely
  def method(self):
      print('in Replacer.method')

class Extender(Super):                  # Extend method behavior
  def method(self):
      print('starting Extender.method')
      Super.method(self)
      print('ending Extender.method')
```

Caution in using super()

- Allows for multiple inheritance
- Method Resolution Order (MRO) - algorithm for solving clashes in multiple inheritance
- Be careful in using the super() method, adds more complexity and confusion for reading code, if multiple inheritance is not needed
- Use `MyClass.__init__(self, ...)` instead to construct a suclass
- See Minecraft example

An aside on Metaclasses

"Metaclasses are deeper magic than 99% of users should ever worry about. If you wonder whether you need them, you don't (the people who actually need them know with certainty that they need them, and don't need an explanation about why)."

~ Tim Peters

Metaclasses

- Metaclasses are used when you want to control how classes are created
- `__new__` contructor established a super `type`
- See Lutz, Chapter 40 for more details

Exercise

- Mortgages group exercise
- Due at the end of the day on Friday Oct 6

End of Meeting #12

For next meeting

- Videos:
  1. Measuring Execution Time (8 mins)
  2. Big O Notation (10 mins)
  3. Common Growth Functions 1 (6 mins)
  4. Common Growth Functions 2 (8 mins)
  5. Insertion Sort (6 mins)
  6. Merge Sort (7 mins)
  7. A Complexity Bound for Sorting (6 mins)
  8. NP-Hard Problems: Conversation with Benjamin Johnson (17 mins) [optional]

- Readings:
  - Lee and Hubbard - Chapter 2: Computational Complexity