Classes Have Folders Too

Object Folders
- Separate for each instance

Class Folders
- Data common to all instances

Recall: Objects can have Methods
- Method: function tied to object
  - Function call:
    - `<function name>(<arguments>)`
  - Method call:
    - `<object variable>.__method__(<arguments>)`
  - Use of a method is a method call
  - Example: `p.distanceTo(q)`
    - Both `p` and `q` act as arguments
    - Very much like `distanceTo(p, q)`
  - Methods (often) in class folders

Name Resolution for Objects
- `(object)<name>` means:
  - Go the folder for object
  - Find attribute/method name
  - If missing, check class folder
  - If not in either, raise error
  - For most Python objects:
    - Attributes are in object folder
    - Methods are in class folder
  - But rules can be broken...

The Class Definition
- Go inside a module, just like a function definition.
- Use `class`: begins naming of a class definition
- Specification (similar to one for a function)
  - `<function definitions>`
  - `<assignment statements>`
  - `<any other statements>` also allowed
  - Example:
    ```python
    class Example(object):
        """The simplest possible class."""
        pass
    ```
  - Python creates after reading the class definition

Instances and Attributes
- Assignments add object attributes
  - `<object>.<att>` = `<expression>`
  - Example: `e.b = 42`
- Assignments can add class attributes
  - `<class>.<att>` = `<expression>`
  - Example: `Example.a = 29`
- Objects can access class attributes
  - Example: `print e.a`
  - But assigning it creates object attribute
  - Example: `e.a = 10`
- Rule: check object first, then class

The Class Specification
- `class`: begins naming of a class definition
  - `<class-name>` = `<object>`
  - **Class specification**
  - `<function definitions>`
  - `<assignment statements>`
  - `<any other statements>` also allowed
  - Example:
    ```python
    class Worker(object):
        """An instance is a worker in an organization."
        """In an instance is a worker in an organization.
        """Instance has basic worker info, but no salary information.
        """
        """ATTRIBUTES:
        """lname: Worker's last name. [str]
        """ssn: Social security no. [int in 0..999999999]
        """boss: Worker's boss. [Worker, or None]
        """
    ```
  - Short summary
  - More detail
  - Attribute list
  - Description
  - Invariant
  - Interface
  - Instantiation
  - Example
**Recall: Objects can have Methods**

- **Method**: function tied to object
  - Function call: `<function-name>(<arguments>)`
  - Method call: `<object-variable>.<function-call>`
  - Use of a method is a method call

**Example: p.distanceTo(q)**

- Both `p` and `q` act as arguments
- Very much like `distanceTo(p, q)`

---

**Initializing the Fields of an Object (Folder)**

- Creating a new `Worker` is a multi-step process:
  - `w = Worker()`  
  - `w.lname = 'White'`
  - `...`

- Want to use something like

  ```python
  w = Worker('White', 1234, None)
  ```

- Create a new `Worker` and assign fields
  - `lname` to 'White'
  - `ssn` to 1234
  - `boss` to None

- Need a **custom constructor**

---

**Making Arguments Optional**

- We can assign default values to `_init__` arguments
  - Write as assignments to parameters in definition
  - Parameters with default values are optional

**Examples:**

```python
p = Point3()  # (0,0,0)
p = Point3(1,2,3)  # (1,2,3)
p = Point3(1,2,3)  # (1,2,3)
p = Point3(0)  # (0,0,0)
p = Point3(0,0)  # (0,0,0)
```

---

**Methods Calls**

**Example: `a.distanceTo(b)`**

- Both `a` and `b` as arguments

**Special Method: __init__**

```python
def __init__(x, y, z):
    # Constructor creates a Worker
    x, y, z = x, y, z  # Defaults are optional
    self.x = x
    self.y = y
    self.z = z
```

---

**Important!**

**YES**

```python
class Point3(object):
    """Instances are 3d points""
    x = 0.0  # x coord, float
    y = 0.0  # y coord, float
    z = 0.0  # z coord, float
```

- **3.0-Style Classes**
  - Well-Designed

**NO**

```python
class Point3:
    """Instances are 3d points""
    x = 0.0  # x coord, float
    y = 0.0  # y coord, float
    z = 0.0  # z coord, float
```

- **“Old-Style” Classes**
  - Very, Very Bad