Lecture #13: Objects and Classes

Data Abstraction vs. Function Abstraction

• Functions perform computations; their specifications abstract from possible implementations of a particular computation.
• In the old days, programs tended to be organized around functions or modules comprising related functions. The data were just the operands.
• Now we tend to organize instead around the data—around objects or types (classes) of objects.
• Objects have state, which is accessed and manipulated by means of attributes.
• The set of attributes and their behavior is analogous to the syntactic and semantic specification of a function.
• In previous lectures, we've seen standard Python objects and ways to get (in effect) new kinds of objects using functions and non-local variables. We've defined data types using them by defining a set of functions to be used to construct, query, and modify them.
• Python also provides a standard way to extend Python with new kinds of data:
  * Just as def defines functions and allows us to extend Python with new operations.
  * In particular, an operation for initializing the state of an object.
  * A means of creating new objects.

Simple Classes: Bank Account

```python
# type name
class Account:
    # constructor method
    def __init__(self, initial_balance):
        self.balance = initial_balance

    # instance method
    def balance(self):
        # instance variable:
        return self.balance

    def deposit(self, amount):
        if amount < 0:
            raise ValueError("negative deposit")
        self.balance += amount

    def withdraw(self, amount):
        if 0 <= amount <= self.balance:
            self.balance -= amount
        else:
            raise ValueError("bad withdrawal")

>>> mine = Account(1000)
>>> mine.deposit(100)
>>> mine.balance()
1100
>>> mine.withdraw(200)
>>> mine.balance()
900
```

Class Machinery

• The account type illustrated how we get each of these.
  * A way of defining new types of data.
  * A means of defining and accessing state for these objects.
  * A means of defining operations specific to these objects.
  * A means of creating and accessing state.

Class Concepts

• Just as def defines functions and allows us to extend Python with new operations, classes define types and allow us to extend Python with new objects.
  * What do we want out of a class?

Extending the Mutable Objects: Classes

• In languages such as Python, Java, and C++, an object is an instance of a class; the class is called the object's type.
• The Python class statement defines new classes or types, creating new, vaguely dictionary-like varieties of object.

Class Attributes

• Attributes of new types of data.

Object Concepts

• Objects have state, which is accessed and manipulated by means of methods (functions of objects).
In general, the notation $X.Y$ means "The value named $Y$ in the object pointed to by $X". Unlike C++ or Java, Python takes a very dynamic approach. Classes and class instances behave rather like environment frames. Given a pointer to some object, $-\ obj.x = value$ looks for a definition of $x$ in the object referenced by $obj$, creating one if it doesn't exist, and assigning $value$ to it. When not being assigned to, $obj.x$ returns the definition of $x$ in the object referenced by $obj$, if any. If not defined in the object, it returns the value defined for $x$ in the class itself, if any.

### Modeling Attributes in Python

```python
class Account:
    total_deposits = 0
    def __init__(self, ...):
        self.balance = ...
        Account.total_deposits = ...

acct1 = Account(1000)
acct2 = Account(10000)
acct1.deposit(300)
```

### Assigning to Attributes

Assigning to an attribute of an object (including a class) is like assigning to a local variable: it creates a new binding for that attribute in the object selected from (i.e., referenced by the expression on the left of the dot).

```python
class Value:
    ... value = 0
...
val1 = Value()
val2 = Value()
val2.value = 3
val1.value
Value.value
val2.value
```

### Attributes of Classes

In Python, classes themselves are objects. (You might well ask "What is the type of a class?" Answer: a builtin class called `type`, whose type is itself.) Therefore, classes themselves have attributes. Assignments and defs immediately inside a class define class attributes. Since $obj.x$ looks for $x$ in the class of $obj$ if it doesn't find it in $obj$ itself, the attributes defined in a class provide default values for attributes of the object that are instances of the class.

### Methods

Consider:

```python
class Foo:
    ...
def set(self, x):
        self.value = x

aFoo = Foo(10)
set = aFoo.set
set(13)
aFoo.value
```

A bound method is an ordinary function that has its first parameter "pre-bound" to a particular value—in this case to $aFoo$.

```python
set(aFoo, 13)
```

### Class Attributes in Python

Sometimes, a quantity applies to a type as a whole, not a specific instance of the type. The attributes of the object that are instances of the class itself, the attributes defined in a class provide default values for attributes of the object. Assignments and defs immediately inside a class define class attributes. Since `$obj.x$ looks for $x$ in the class of $obj$ if it doesn't find it in $obj$ itself.

### Attributes of Classes

```python
... Look for $y$ starting at $x$
```

### Modeling Attributes in Python

```python
... Look for $y$ starting at $x$
```
class Account:
    total_deposits = 0

    def __init__(self, initial_balance):
        self.balance = initial_balance
        Account.total_deposits += initial_balance

    def deposit(self, amount):
        self.balance += amount
        Account.total_deposits += amount

    def total_deposits():
        # Define a class method.
        return Account.total_deposits

>>> acct1 = Account(1000)
>>> acct2 = Account(10000)
>>> acct1.deposit(300)

>>> Account.total_deposits()
11300

Classes and Operators

- Many standard operators defined in Python are essentially syntax.
- Functions are defined in the class body.
- Function-values attributes of classes are called methods.
- Classes beget objects called instances, created by "calling" the class.
- By convention, we call the first argument of a method self to indicate that it is the object from which we got the method.
- A method called __del__ (default) is essentially the same as a method that can access using object.attribute notation.
- Each such access of object initially shows the attributes of its class.
- Class methods, attributes of classes are called methods.
- Class methods created by assignment statements and del

Class Machinery: Summary

- Classes have attributes, created by assignment statements and del.
- Function-values attributes of classes are called methods.
- Classes beget objects called instances, created by calling the class.
- By convention, we call the first argument of a method self to indicate that it is the object from which we got the method.