Classes in Python

Python contains a class creation mechanism that's fairly similar to what's found in C++ or Java.

There are significant differences though:

- · All class members are public.
- Instance fields aren't declared.
 Rather, you just create fields as
 needed by assignment (often in
 constructors).
- There are class fields (shared by all class instances), but there are no class methods. That is, all methods are instance methods.

 All instance methods (including constructors) must explicitly provide an initial parameter that represents the object instance. This parameter is typically called self. It's roughly the equivalent of this in C++ or Java.

C5 538 Spring 2008 556 C5 538 Spring 2008

Defining Classes

You define a class by executing a class definition of the form

class name:

statement(s)

A class definition creates a class object from which class instances may be created (just like in Java). The statements within a class definition may be data members (to be shared among all class instances) as well as function definitions (prefixed by a def command). Each function must take (at least) an initial parameter that represents the class instance within which the function (instance method) will operate. For example,

```
class Example:
  cnt=1
  def msg(self):
    print "Bo"+"o"*Example.cnt+
     "!"*self.n
>>> Example.cnt
>>> Example.msg
<unbound method Example.msg>
Example.msg is unbound because
we haven't created any instances
of the Example class yet.
We create class instances by using
the class name as a function:
>>> e=Example()
>>> e.msg()
AttributeError: n
```

We get the AttributeError message regarding n because we haven't defined n yet! One way to do this is to just assign to it, using the usual field notation:

```
>>> e.n=1
>>> e.msg()
Boo!
>>> e.n=2;Example.cnt=2
>>> e.msg()
Booo!!
```

We can also call an instance method by making the class object an explicit parameter:

```
>>> Example.msg(e)
```

B000!!

It's nice to have data members initialized when an object is created. This is usually done with a constructor, and Python allows this too.

CS 538 Spring 2008th 560

You can also define the equivalent of Java's tostring method by defining a member function named __str__(self).

For example, if we add

```
def __str__(self):
    return "<%d>"%self.n
```

to Example,

then we can include **Example** objects in **print** statements:

```
>>> e=Example(2)
>>> print e
<2>
```

A special method named __init__ is called whenever an object is created. This method takes self as its first parameter; other parameters (possibly made optional) are allowed.

We can therefore extend our **Example** class with a constructor:

CS 538 Spring 2008[®] 56

INHERITANCE

Like any language that supports classes, Python allows inheritance from a parent (or base) class. In fact, Python allows *multiple inheritance* in which a class inherits definitions from more than one parent.

When defining a class you specify parents classes as follows:

```
class name(parent classes):
    statement(s)
```

The subclass has access to its own definitions as well as those available to its parents. All methods are virtual, so the most recent definition of a method is always used.

```
class C:
    def DoIt(self):
        self.PrintIt()
    def PrintIt(self):
        print "C rules!"

class D(C):
    def PrintIt(self):
        print "D rules!"
    def TestIt(self):
        self.DoIt()

    dvar = D()
    dvar.TestIt()
```

If you specify more than one parent for a class, lookup is depth-first, left to right, in the list of parents provided. For example, given class A(B,C): ... we first look for a non-local definition in B (and its parents), then in C (and its parents).

CS 538 Spring 2008[®]

564

CS 538 Spring 2008

Operator Overloading

You can overload definitions of all of Python's operators to apply to newly defined classes. Each operator has a corresponding method name assigned to it. For example, + uses __add__, - uses __sub__, etc.

```
Given
class Triple:
 def __init__(self,A=0,B=0,C=0):
   self.a=A
   self.b=B
   self.c=C
 def __str__(self):
   return "(%d,%d,%d)"%
    (self.a, self.b, self.c)
 def __add__(self,other):
  return Triple(self.a+other.a,
    self.b+other.b,
    self.c+other.c)
the following code
t1=Triple(1,2,3)
t2=Triple(4,5,6)
print t1+t2
produces
(5,7,9)
```

Exceptions

Python provides an exception mechanism that's quite similar to the one used by Java.

You "throw" an exception by using a raise statement:

```
raise exceptionValue
```

There are numerous predefined exceptions, including OverflowError (arithmetic overflow), EOFError (when endof-file is hit), NameError (when an undeclared identifier is referenced), etc.

CS 538 Spring 2008 568 CS 538 Spring 2008 5

except clause. If the raised exception is not matched by any except clause, the next enclosing try is considered, or the exception is reraised at the point of call.

For example, using our **badValue** exception class,

You may define your own exceptions as subclasses of the predefined class **Exception**:

```
class badValue(Exception):
    def __init__(self,val):
        self.value=val
```

You catch exceptions in Python's version of a **try** statement:

```
try:
    statement(s)
except exceptionName<sub>1</sub>, id<sub>1</sub>:
    statement(s)
...
except exceptionName<sub>n</sub>, id<sub>n</sub>:
    statement(s)
```

As was the case in Java, an exception raised within the try body is handled by an except clause if the raised exception matches the class named in the

Modules

Python contains a module feature that allows you to access Python code stored in files or libraries. If you have a source file mydefs.py the command

```
import mydefs
```

will read in all the definitions stored in the file. What's read in can be seen by executing

dir(mydefs)

To access an imported definition, you qualify it with the name of the module. For example,

mvdefs.fct

accesses **fct** which is defined in module **mydefs**.

To avoid explicit qualification you can use the command

```
from modulename import id_1, id_2, ...
```

This makes ia_1 , ia_2 , ... available without qualification. For example,

```
>>> from test import sqrt
>>> sqrt(123)
(11.0905365064+0j)
```

You can use the command from modulename import * to import (without qualification) all the definitions in modulename.

THE Python Library

One of the great strengths of Python is that it contains a vast number of modules (at least several hundred) known collectively as the *Python Library*. What makes Python really useful is the range of prewritten modules you can access. Included are network access modules, multimedia utilities, data base access, and much more.

See

www.python.org/doc/lib
for an up-to-date listing of what's
available.

CS 538 Spring 2008

572

CS 538 Spring 2008

573

JAVA 1.5/1.6 (Tiger JAVA)

Java has been extended to include a variety of improvements, many drawn from functional languages.

Added features include:

· Parametric polymorphism.

Classes and interfaces may be parameterized using a type parameter.

```
class List<T> {
    T head;
    List<T> tail;
}
Interfaces may also be
parameterized.
```

· Enhanced loop iterators.

```
for (v : myArray) {
  // each element of myArray
  // appears as a value of v }
```

 Automatic boxing and unboxing of wrapper classes.

Conversion from int to Integer or Integer to int is now automatic.

- Typesafe enumerations.
 public enum Color {RED, BLUE, GREEN};
- Static imports.

You may import all static members of a class and use them without qualification. Thus you may now write out.println rather than

System.out.println.

- · Variable argument methods.
- Formatted output using printf: out.printf("Ans = %3d",a+b);

Reading Assignment

 Pizza Tutorial (linked from class web page)

CS 538 Spring 2008

CS 538 Spring 2008[®]

576

- Classes have single inheritance.
- C# generates code for a virtual machine to support cross-platform execution.
- Interfaces are provided to capture functionality common to many classes.
- Exceptions are very similar in form to Java's.
- Instance and static data within an object must be initialized at point of creation.

C#

C# is Microsoft's answer to Java. In most ways it is very similar to Java, with some C++ concepts reintroduced and some useful new features.

Similarities to Java include:

- C# is object-based, with all objected descended from class Object.
- Objects are created from classes using new. All objects are heapallocated and garbage collection is provided.
- All code is placed within methods which must be defined within classes.
- Almost all Java reserved words have C# equivalents (many are identical).

C# Improves Upon Some Java Features

 Operators as well as methods can be overloaded:

- Switch statements may be indexed by string literals.
- In a switch, fall-throughs to the next case are disallowed (if nonempty).
- · Goto's are allowed.
- Virtual methods must be marked.

· Persistent objects (that may be stored across executions) are available.

C# Adds Useful Features

- Events and delegates are included to handle asynchronous actions (like keyboard or mouse actions).
- Properties allow user-defined read and write actions for fields. You can add get and set methods to the definition of a field. For example,

```
class Customer {
  private string name;
  public string Name {
  get { return name; }}
Customer c; ...
string s = c.Name;
```

CS 538 Spring 2008

CS 538 Spring 2008

• Indexers allow objects other than arrays to be indexed. The [] operator is overloadable. This allows you to define the meaning of

obj[123] Or obj["abc"] within any class definition.

· Collection classes may be directly enumerated:

foreach (int i in array) ...

· Fields, methods and constructors may be defined within a struct as well as a class. Structs are allocated within the stack instead of the heap, and are passed by value. For example:

```
struct Point {
  int x,y;
  void reset () {
    x=0; y=0; }
}
```

· When an object is needed, a primitive (int, char, etc.) or a struct will be automatically boxed or *unboxed* without explicit use of a wrapper class (like **Integer** or Character). Thus if method **List.add** expects an object, you may write

List.add(123); and 123 will be boxed into an **Integer** object automatically.

- Enumerations are provided: enum Color {Red, Blue, Green};
- Rectangular arrays are provided: int [,] multi = new int[5,5];
- · Reference, out and variable-length parameter lists are allowed.
- Pointers may be used in methods marked unsafe.

CS 538 Spring 2008 CS 538 Spring 2008