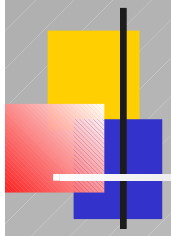
OOP and Scripting in Python

Part 2 - OOP Features

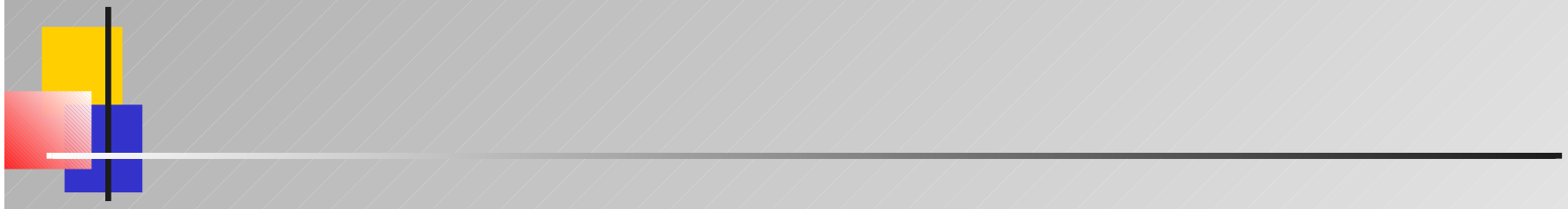
Giuliano Armano - DIEE Univ. di Cagliari



Part 2 - OOP Features

Python: OOP Features

- Classes, Methods, and Instances
- Methods Dispatching and Binding
- Inheritance
- Polymorphism
- Operators Handling
- Exception handling



Classes, Methods, and Instances

Part 2 – OOP Features: Classes, Methods, and Instances

Classes, Methods, and Instances

- Encapsulation (= class construct) YES
- Information hiding ~NO

Classes, Methods and Instances

Encapsulation: a sample

```
>>> from math import *
>>> class Point:
...   def __init__(self,x=0,y=0):
...     self.x, self.y = x,y
...   def distance(self,p):
...     d2 = (self.x-p.x)**2 + (self.y-p.y)**2
...     return sqrt(d2)
...

>>> p1 = Point()
>>> print p1.x, p1.y
0 0
>>> p1.distance(Point(1,1))
```

a class name

a method name

a class instance

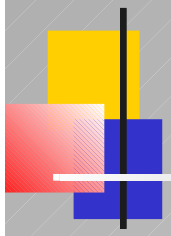
Classes, Methods and Instances

Information hiding: private and public slots

```
>>> class Blob:
...   def __init__(self):
...     self.public = 'I am public'
...     self.__private = 'I am private'
...

>>> b = Blob()
>>> b.public
'I am public'
>>> b.__private
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in -toplevel- b.__private
AttributeError: Blob instance has no attribute '__private'
>>>
```

Seemingly, one may define "private" slots; actually, they may be accessed from outside. See the Python reference manual for more information.



Methods Dispatching and Binding

Part 2 - OOP Features: Methods

Method Dispatching and Binding

- Method dispatching (single vs. multiple) SINGLE
- Method binding (static vs. dynamic) DYNAMIC

Method Dispatching

```
>>> class Point:
...   def __init__(self,x=0,y=0):
...     self.x = x
...     self.y = y
...   def distance(self,p):
...     return sqrt( (self.x-p.x)**2 + (self.y-p.y)**2 )
...

>>> p1 = Point(1,2)
>>> p2 = Point(10,20)
>>> p1.distance(p2)
20.124611797498108
>>> Point.distance(p1,p2)
20.124611797498108
>>>
```

Method Binding

```
>>> class Point:
...   def __init__(self,x=0,y=0):
...     self.x, self.y = x,y
...   def distance(self,p):
...     return sqrt((self.x-p.x)**2+(self.y-p.y)**2)
...

>>> class CPoint(Point):
...   def __init__(self,x=0,y=0,color=0):
...     Point.__init__(self,x,y)
...     self.color = color
...
```

Method Binding

```
>>> from math import *
>>> p1 = CPoint()
>>> p2 = Cpoint(2,2)
>>>
>>> print p1.distance(p2)
2.82842712475
>>>
>>> CPoint.distance(p1,p2)
2.82842712475
>>>
>>> Point.distance(p1,p2)
2.82842712475
```

Method Binding

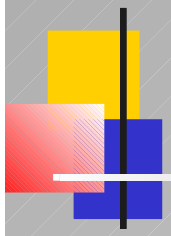
```
>>> class Blob:
...   def foo(self):
...     print 'This is Blob'
...

>>> class Blob1:
...   def foo(self):
...     print 'This is Blob-one'
...
```

Method Binding

```
>>> def foo(x):
...    x.foo()
...

>>> a = Blob()
>>> b = Blob1()
>>>
>>> foo(a)
This is Blob
>>>
>>> foo(b)
This is Blob-one
>>>
```



Inheritance

Part 2 – OOP Features: Inheritance

Inheritance

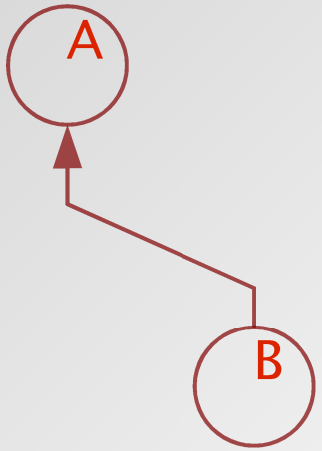
- Interfaces NO
- Constructors inheritance NO
- Multiple inheritance YES

Constructors' Inheritance

```
>>> class A:
...     def __init_(self):
...         self.x = 0
...

>>> class B(A):
...     def __init__(self):
...         A.__init__(self)        # no automatic activation !
...         self.y = 0
...

>>> b = B()
>>> print b.x, b.y
0 0
```



Single Inheritance

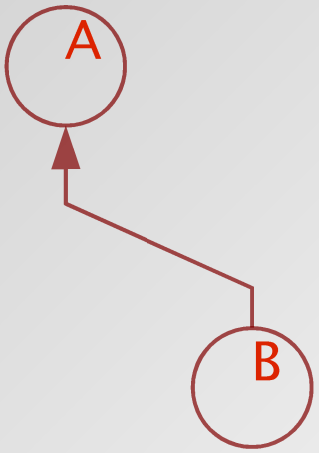
- Method retrieval:

BOTTOM-UP

```
>>> class A:
...   def method1(self):
...     print 'method1 of A'
...

>>> class B(A):
...   pass
...

>>> b = B()
>>> b.method1()
method1 of A
```



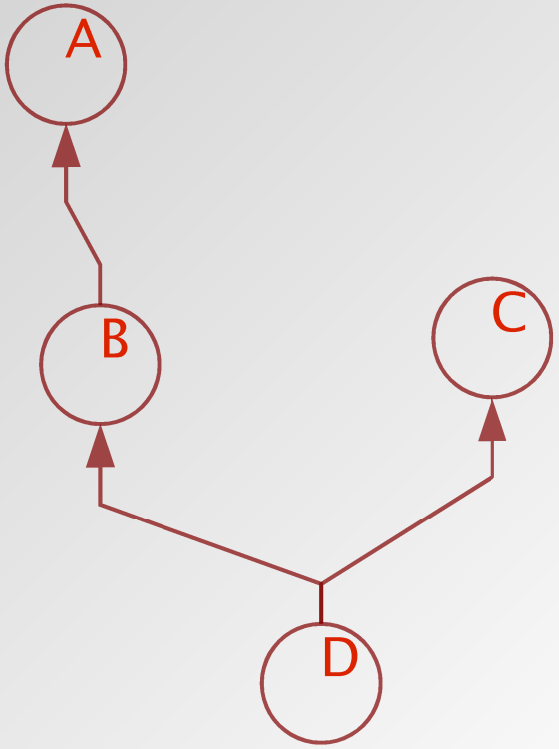
Multiple Inheritance

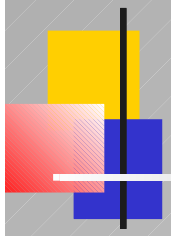
- Method retrieval: BOTTOM-UP + DEPTH-FIRST + LEFT-TO-RIGHT

```
>>> class C:
...   def method1(self):
...     print 'method1 of C'
...

>>> class D(B,C):
...   pass
...

>>> d = D()
>>> d.method1()
method1 of A
```





Polymorphism

Part 2 – OOP Features: Polymorphism

Polymorphism

- Universal
  - Parametric Class — NO
  - By Inclusion — YES
- Ad-Hoc
  - Overloading — NO
  - Coercion — ~YES

Inclusion Polymorphism

```
>>> class B:
...   def method1(self):
...     print 'method1 of B'
...

>>> class D(B):
...   def method1(self):
...     print 'method1 of D'
...

>>> d = D()
>>> d.method1()
method1 of D
```

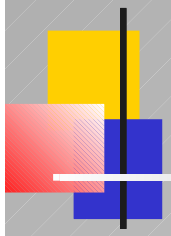
Coercion/Conversion

- Conversion:

```
>>> a = 10
>>> b = float(a)
>>> b
10.0
```

- Coercion:

```
>>> x = 1
>>> y = 2.3
>>> print x+y
3.3
>>>
```



Operators Handling

Part 2 – OOP Features: Exceptions Handling

Comparison Operators

```
__lt__(a, b)      # a < b
__le__(a, b)       # a = b
__eq__(a, b)       # a == b
__ne__(a, b)       # a != b
__ge__(a, b)       # a = b
__gt__(a, b)       # a > b
```

Logical Operators

```
__and__(a, b)     # a and b
__or__(a, b)       # a or b
__xor__(a, b)      # a xor b
__not__(a, b)      # not a
```

Arithmetic Operators

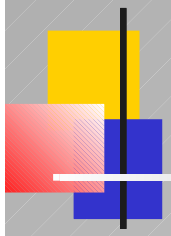
```
__add__(a, b)     # a + b
__sub__(a, b)      # a - b
__mul__(a, b)      # a * b
__div__(a, b)      # a / b
__abs__(a)         # abs(a)
__mod__(a, b)      # a % b
```

Operators Redefinition (an example)

- All operators can be redefined like C++ does ...

```
>>> class Blob:
...   def __init__(self,x=0):
...     self.x = x
...   def __add__(self,y):
...     return self.x + y
...

>>> a = Blob()
>>> print a.__add__(1)
1
>>> print a+1
1
```



Exception Handling

Part 2 – OOP Features: Exceptions Handling

Exception Handling: Structure

```
try:
    do_something()
    do_something_else()
except anExceptionClass, anInstance:
    handle_exception(anInstance)
except anotherExceptionClass:     # instance is optional
    handle_another_exception()
else:                             # catch-all is optional
    do_else()
```

For more information see the Python reference manual.

Exception Handling: An Example

```
>>> class Blob:
...   def __init__(self,value=0):
...     self.value = value
...   def divide(self,y):
...     if (y == 0): raise BlobException
...     else: return self.value / y
...
```

Exception Handling: An Example

```
>>> try:
...     b = Blob(1)
...     print b.divide(10)
...     print b.divide(-10)
...     print b.divide(0)
... except BlobException:
...     print "A Blob Exception has been raised"
...

0
-1
A Blob Exception has been raised
>>>
```