User-Defined Methods

Python 2.x 2.6–2.7 , Python 3.x 3.0–3.6

Improvements requested:

Examples

Improvements requested:

Creating user-defined method objects

User-defined method objects may be created when getting an attribute of a class (perhaps via an instance of that class), if that attribute is a user-defined function object, an unbound user-defined method object, or a class method object.

```
class A(object):
    # func: A user-defined function object
    #
    # Note that func is a function object when it's defined,
    # and an unbound method object when it's retrieved.
    def func(self):
        pass

    # classMethod: A class method
    @classmethod
    def classMethod(self):
        pass

class B(object):
    # unboundMeth: A unbound user-defined method object
    #
    # Parent.func is an unbound user-defined method object here,
    # because it's retrieved.
    unboundMeth = A.func

a = A()
b = B()

print A.func
# output: <unbound method A.func>
print a.func
# output: <bound method A.func of <__main__.A object at 0x10e9ab910>>
print B.unboundMeth
# output: <unbound method A.func>
print b.unboundMeth
# output: <unbound method A.func>
print A.classMethod
# output: <bound method type.classMethod of <class '__main__.A'>>
print a.classMethod
# output: <bound method type.classMethod of <class '__main__.A'>>
```

When the attribute is a user-defined method object, a new method object is only created if the class from which it is being retrieved is the same as, or a derived class of, the class stored in the original method object; otherwise, the original method object is used as it is.

```
# Parent: The class stored in the original method object
class Parent(object):
    # func: The underlying function of original method object
    def func(self):
        pass
    func2 = func

# Child: A derived class of Parent
class Child(Parent):
    func = Parent.func

# AnotherClass: A different class, neither subclasses nor subclassed
class AnotherClass(object):
    func = Parent.func

print Parent.func is Parent.func                # False, new object created
print Parent.func2 is Parent.func2              # False, new object created
print Child.func is Child.func                  # False, new object created
print AnotherClass.func is AnotherClass.func    # True, original object used
```

Turtle example

The following is an example of using an user-defined function to be called multiple(x) times in a script with

The following is an example of using an user-defined function to be called multiple(∞) times in a script with ease.

```
import turtle, time, random #tell python we need 3 different modules
turtle.speed(0) #set draw speed to the fastest
turtle.colormode(255) #special colormode
turtle.pensize(4) #size of the lines that will be drawn
def triangle(size): #This is our own function, in the parenthesis is a variable we have defined
    turtle.forward(size) #to begin this function we go forward, the amount to go forward by is t
    turtle.right(90) #turn right by 90 degree
    turtle.forward(size) #go forward, again with variable
    turtle.right(135) #turn right again
    turtle.forward(size * 1.5) #close the triangle. thanks to the Pythagorean theorem we know th
while(1): #INFINITE LOOP
    turtle.setpos(random.randint(-200, 200), random.randint(-200, 200)) #set the draw point to a
    turtle.pencolor(random.randint(1, 255), random.randint(1, 255), random.randint(1, 255)) #ran
    triangle(random.randint(5, 55)) #use our function, because it has only one variable we can s
    turtle.pencolor(random.randint(1, 255), random.randint(1, 255), random.randint(1, 255)) #ran
```

Syntax

Parameters

Remarks