sandeepsuryaprasad / python_tutorials Private

Code Issues Pull requests Actions Projects Security Ins

master python_tutorials / 10_oops / 4_inheritance.py / <> Jump to Go to file ...

Sandeep Suryaprasad clean up Latest commit 9edadc5 on 30 Mar History

1 contributor

109 lines (90 sloc) | 3.06 KB Raw Blame

```python
"""
1. Inheritance is a mechanism for creating a new class that specializes or mo
   the behavior of an existing class.
2. The original class is called a base class, superclass, or parent class.
3. The new class is called a derived class, child class, subclass, or subtype
4. When a class is created via inheritance, it inherits the attributes define
   However, a derived class may redefine any of these attributes and add new
"""
# ---------------------------------------------------------------------------
class Parent:
    def __init__(self, value):
        self.value = value

    def apple(self):
        print('Parent.Apple', self.value)

    def google(self):
        print('Parent.Google')
        self.apple()

# Completely Independent Method
class Child1(Parent):
    def yahoo(self):
        print('Child1.Yahoo')


# Overriding Parent class Method
```

```python
class Child2(Parent):
    def apple(self):
        print('Child2.Apple', self.value)

# Overriding Parent class Method but reusing the original method in Parent
class Child3(Parent):
    def apple(self):
        print('Child2.Apple')
        super().apple()

# Adding a new Attribute
class Child4(Parent):
    def __init__(self, value, extra):
        self.extra_value = extra
        super().__init__(value)

class Parent2:
    def __init__(self, some_value):
        self.some_value = some_value

    def facebook(self):
        print('Parent2.Facebook')

# Child Inheriting from more than one parent
class Child5(Parent, Parent2):
    # Since, Parent and Parent2 has constructors, it is "Child5" resposibilit
    # Initialize the constructors of both the parents
    def __init__(self, value, some_value):
        Parent.__init__(self, value)
        Parent2.__init__(self, some_value)

# Method Resolution Order - MRO
c = Child5(10)
print(c.__class__.__mro__)
# ----------------------------------------------------------------------------
# Multi-level inheritance
class A:
    def demo(self):
        print('A')

class B(A):
    def demo(self):
        print('B')
        super().demo()

```

```python
class C(B):
    def demo(self):
        print('C')
        super().demo()

# ---------------------------------------------------------------------------
# Advanced Inheritance
class Parent:
    def spam(self):
        print('Parent Spam')

class Child1(Parent):
    def spam(self):
        print('Child1.Spam')
        super().spam()

class Child2(Parent):
    def spam(self):
        print('Child2.Spam')
        super().spam()

# Multiple Inheritance
class Child3(Child1, Child2):
    pass

# ---------------------------------------------------------------------------
# Overriding class variables
class Spam:
    a = 10
    def apple(self):
        print('apple', self.__class__.a)

class Apple(Spam):
    a = 20  # Overrides the value of class variable "a" in parent class
    def google(self):
        print('google')
# ---------------------------------------------------------------------------
```