 这个特性比较简单，依旧长话多说，只说重点。下面这个demo我们经常使用重载函数来处理问题的场景： Code 1 public void Process(int p1, float p2, string p3) 2 { 3 //todo 4 } 5 6 public void Process(int p1, float p2) 7 { 8 Process(p1, p2, "fanweixiao"). 9 } 10 11 public void Process(int p1) 12 { 13 Process(p1, 0f). 14 } 在C#4.0时代给我们提供了两个新功能，命名参数（Named Parameters）和可选参数（Optional Parameters）。他们是两个完全独立的概念，只是经常一块使用。改写后的这个函数为： Code 1 public int NBProcess(int p1=0, float p2=0f, string p3="fanweixiao") 2 { 3 //todo 4 } 这样我们就可以用NBProcess(10)来调用这个函数，相当于NBProcess(10, 0f, "fanweixiao")的调用。 如果想省去第二个参数我们可以这样调用：NBProcess(10, p3:"FanWeixiao")。麻烦一点也可以写成NBProcess(p1:10,p3:"FanWeixiao")。甚至还可以把参数改变顺序NBProcess(p3:"FanWeixiao",p1:10)这样来调用。 对于构造函数和indexer也可以这么用。 对于有这样的参数的重载，究竟如何判断是调用哪个，逻辑也很简单：最相近原则。(applicable) 从参数是5来看，首先排除了M(string,int)，因为它要求第一个参数是string类型的。M(int, string)是可以的因为string是可选参数，它和M(int)都比M(object)要好，object是“万恶之源”，比起object，5和int可是亲切多了。那么最后显而易见M(int)是最好的。 构造函数不同类型参数的位置和个数是决定其区分的关键，而c#4.0的这个特性把参数的名字

放入规则中了，以后给参数起名也得正经点了:)。其实像python这样的动态语言，上面的功能都支持，还有一个更酷的“ *参数 ”： Code 1 def stepper(what, by=1): 2 what = by 3 return what 4 5 def average(first, *rest): 6 sum = first 7 for value in rest: sum = value 8 result = 1.0 * sum / (len(rest) 1) 9 return result 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com