例：程序assign\_test/assign\_test1.py

pam, ham = 'yum', 'YUM' # 元组赋值，按位置从左到右匹配  
print(spam, ham) # yum YUM  
[spam1, ham1] = ['yum', 'YUM'] # 列表赋值，按位置从左到右匹配  
print(spam1, ham1) # yum YUM  
  
# 序列赋值，从左到右匹配  
# 包含了元组赋值和列表赋值  
# 字符串也是序列  
# 如果无法匹配就会报错  
a, b, c, d = 'spam'  
print(a, b, c, d) # s p a m  
# a, b, c = 'spam' # Error  
  
# 即使左右两边类型不一致，  
# 也支持序列赋值  
[a2, b2, c2] = (1, 2, 3)  
print(a2, b2, c2) # 1 2 3  
  
# 扩展的序列解包  
# 第一个字符s匹配a1  
# 剩下的赋值给b1  
a1, \*b1 = 'spam'  
print(a1, b1) # s ['p', 'a', 'm']  
  
# 增强赋值，在原处修改，  
# 对于不变对象，  
# 与spams = spams + 42没有区别  
# 不变对象不支持原处修改  
# 对于可变对象，在原处修改，会影响  
# 到其他相关的变量，相当于list1.append  
spams = 0  
hams = spams  
spams += 42  
print(spams) # 42  
print(hams) # 0  
print()  
list1 = [1, 2, 3]  
list2 = list1  
list1 += [4, 5, 6]  
print(list1) # [1, 2, 3, 4, 5, 6]  
print(list2) # [1, 2, 3, 4, 5, 6]

例：程序assign\_test/assign\_test2.py

string = 'spam'  
# a, b, c = string # Error  
a, b, c = string[0], string[1], string[2:]  
  
# a = s, b = p, c = am  
print('a = {0}, b = {1}, c = {2}'.format(a, b, c))  
  
# a1 = s, b1 = p, c1 = am  
a1, b1, c1 = list(string[:2]) + [string[2:]]  
print('a1 = {0}, b1 = {1}, c1 = {2}'.format(a1, b1, c1))  
  
# a2 = s, b2 = p, c2 = am  
(a2, b2), c2 = string[:2], string[2:]  
print('a2 = {0}, b2 = {1}, c2 = {2}'.format(a2, b2, c2))  
  
# a3 = S, b3 = P, c3 = AM  
((a3, b3), c3) = ('SP', 'AM')  
print('a3 = {0}, b3 = {1}, c3 = {2}'.format(a3, b3, c3))  
  
l1 = [3]  
front = l1[0]  
# last = l1[1] # IndexError  
last = l1[1:] # OK，空列表[]  
print(front, last) # 3 []

例：程序assign\_test/assign\_test3.py

L1 = [1, 2, 3, 4]  
print('original L1', L1) # original L1 [1, 2, 3, 4]  
# 元组不能如此赋值，不支持原处修改  
L1[1:3] = [8, 9]  
print('after assign', L1) # after assign [1, 8, 9, 4]  
print()  
  
# 扩展序列解包  
# 不仅支持列表，也支持其他序列类型  
seq = [1, 2, 3, 4]  
  
# a = 1, b = [2, 3, 4]  
a, \*b = seq  
print('a = {0}, b = {1}'.format(a, b))  
  
# a1 = [1, 2, 3], b1 = 4  
\*a1, b1 = seq  
print('a1 = {0}, b1 = {1}'.format(a1, b1))  
  
# a2 = 1, b2 = [2, 3], c2 = 4  
a2, \*b2, c2 = seq  
print('a2 = {0}, b2 = {1}, c2 = {2}'.format(a2, b2, c2))  
  
# 边界情况  
# a3 = 1, b3 = 2, c3 = 3, d3 = [4]  
# d3是一个列表  
a3, b3, c3, \*d3 = seq  
print('a3 = {0}, b3 = {1}, c3 = {2}, d3 = {3}'.format(a3, b3, c3, d3))  
  
# a4 = 1, b4 = 2, c4 = 3, d4 = 4, e4 = []  
# e4是一个空列表  
a4, b4, c4, d4, \*e4 = seq  
print('a4 = {0}, b4 = {1}, c4 = {2}, d4 = {3}, e4 = {4}'  
.format(a4, b4, c4, d4, e4))  
  
# a5 = 1, b5 = 2, c5 = 3, d5 = 4, e5 = []  
a5, b5, \*e5, c5, d5 = seq  
print('a5 = {0}, b5 = {1}, c5 = {2}, d5 = {3}, e5 = {4}'  
.format(a5, b5, c5, d5, e5))  
# a5, b5, \*e5, c5, \*d5 = seq # Error，\*只能有1个  
# \*a5 = seq #Error \*必须在列表或元组中  
\*a6, = seq  
print('a6 = {0}'.format(a6)) # a6 = [1, 2, 3, 4]

例：程序assign\_test/assign\_test4.py

# Python中的增强赋值种类  
# += &= -= |= \*= ^= /= >>= %= <<= \*\*= //=  
  
# 增强赋值后 L的id不变，列表支持原处修改  
L = [1, 2, 3]  
print("L id: ", id(L)) # 830456954632  
L += [4, 5] # 增强赋值，在原处修改  
print("after increase L id: ", id(L)) # 830456954632  
print()  
  
# 合并会产生新的对象，合并后的L1的id与之前的不同  
L1 = [1, 2, 3]  
print("L1 id: ", id(L1)) # 830456954568  
L1 = L1 + [4, 5] # 合并，产生新的对象  
print("after merge L1 id: ", id(L1)) # 830456030472  
print()  
  
# 不变对象不支持原处修改，增强赋值  
# 会产生新的对象  
a = 1  
print("a id: ", id(a)) # a id: 1894149600  
a += 2  
print("after increase a id: ", id(a)) # after increase a id: 1894149664  
  
X = Y = [1, 2, 3]  
print("X = ", X, " Y = ", Y) # X = [1, 2, 3] Y = [1, 2, 3]  
X += [4, 5]  
print("X = ", X, " Y = ", Y) # X = [1, 2, 3, 4, 5] Y = [1, 2, 3, 4, 5]