函数 function

pycharm 相关设置

1. “代码自动完成”时间延时设置
   File -> Settings -> Editor -> General -> Code Completion -> Autopopup in (ms):0
2. 快捷键：
   Ctrl + P　　参数信息（在方法中调用参数）
   Ctrl + Q　　快速查看文档
   Ctrl + Alt + M　　提取方法

定义

1. 用于封装一个特定的功能，表示一个功能或者行为。
2. 函数是可以重复执行的语句块, 可以重复调用。

作用

提高代码的可重用性和可维护性（代码层次结构更清晰）。

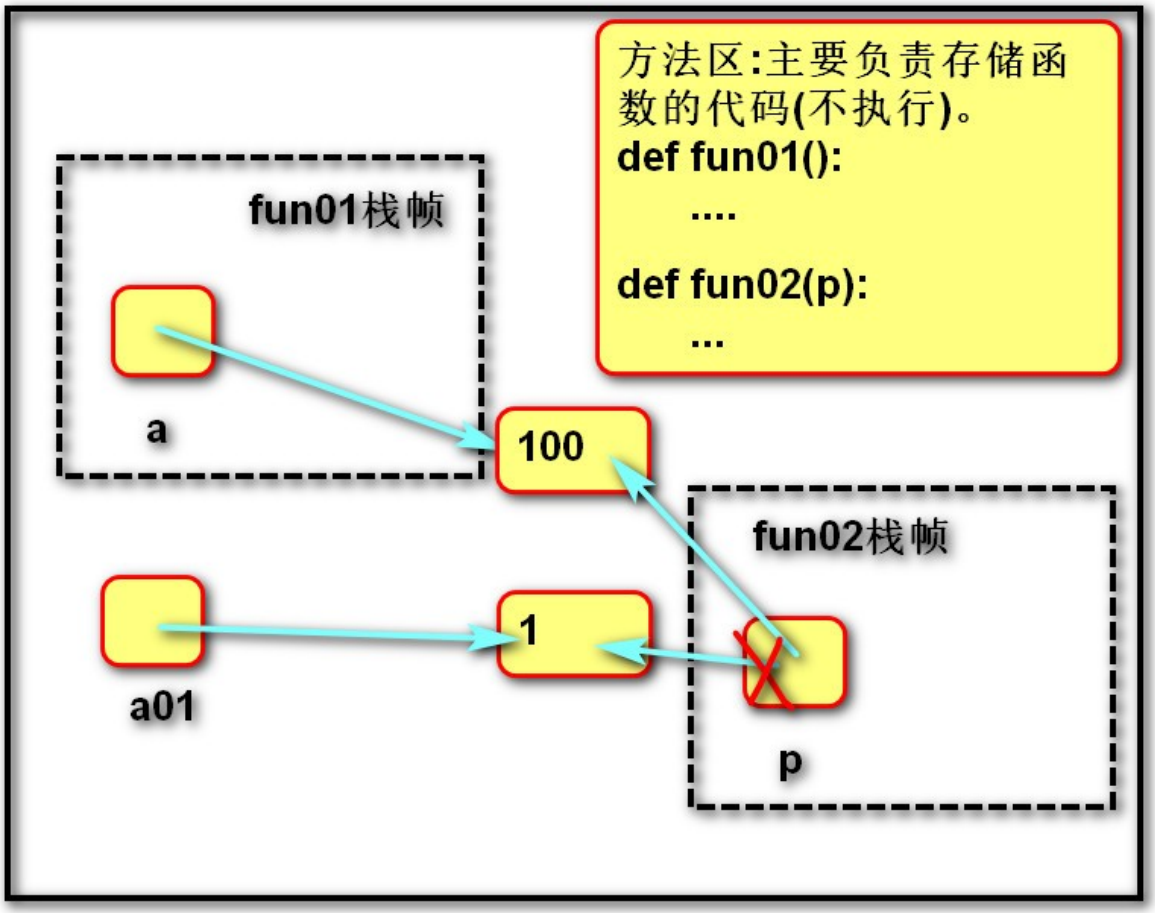
定义函数

1. 语法：

```
def 函数名(形式参数):
    函数体
```

2. 说明：
   def 关键字：全称是 define，意为”定义”。
   函数名：对函数体中语句的描述，规则与变量名相同。
   形式参数：方法定义者要求调用者提供的信息。
   函数体：完成该功能的语句。
3. 函数的第一行语句建议使用文档字符串描述函数的功能与参数。

```
def fun01():
    a = 100
fun01()
def fun02(p):
    p = 100
a01= 1
fun02(a01)
print(a01)# 1
```



调用函数

1. 语法：函数名(实际参数)
2. 说明：根据形参传递内容。

"""

函数内存分配

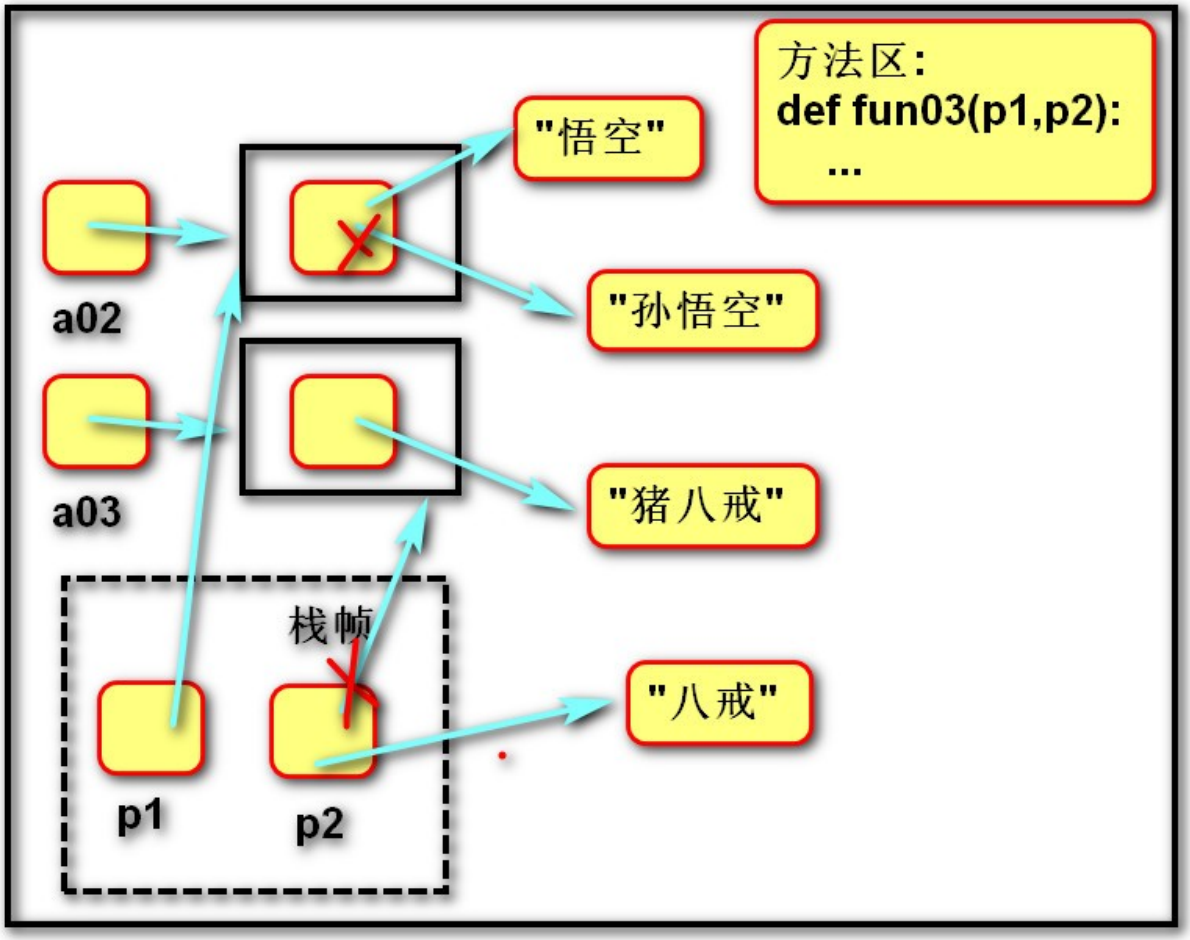
练习：exercise04.py

```
    练习：exercise05.py
    练习：exercise06.py
"""
# 将函数代码存入方法区(不执行函数体)
def fun01():
    a = 100
# 开辟栈帧
fun01()
# 栈帧销毁
def fun02(p):
    # 修改 fun02 栈帧中的变量
    p = 100
a01 = 1
fun02(a01)
print(a01)# 1
def fun03(p1, p2):
    # 修改列表的元素
    p1[0] = "悟空"
    # 修改栈帧中的变量
    p2 = "八戒"
a02 = ["孙悟空"]
a03 = ["猪八戒"]
fun03(a02, a03)
print(a02)
print(a03)
def fun04(p1, p2):
    # 修改列表的元素
    p1[:] = ["悟空"]# 将["悟空"]遍历后，存入 p1 指向的列表中
    temp = p2[:]# 浅拷贝 p2 指向的列表
    temp[:] = ["八戒"]# 修改拷贝后的列表元素

def fun01(p1, p2):
    # 修改栈帧中的变量
    p1 = "悟空"
    # 修改列表的元素
    p2[0] = "八戒"
a = 100
b = [200]
fun01(a, b)
print(a)  # ?100
print(b)  # ['八戒']
```

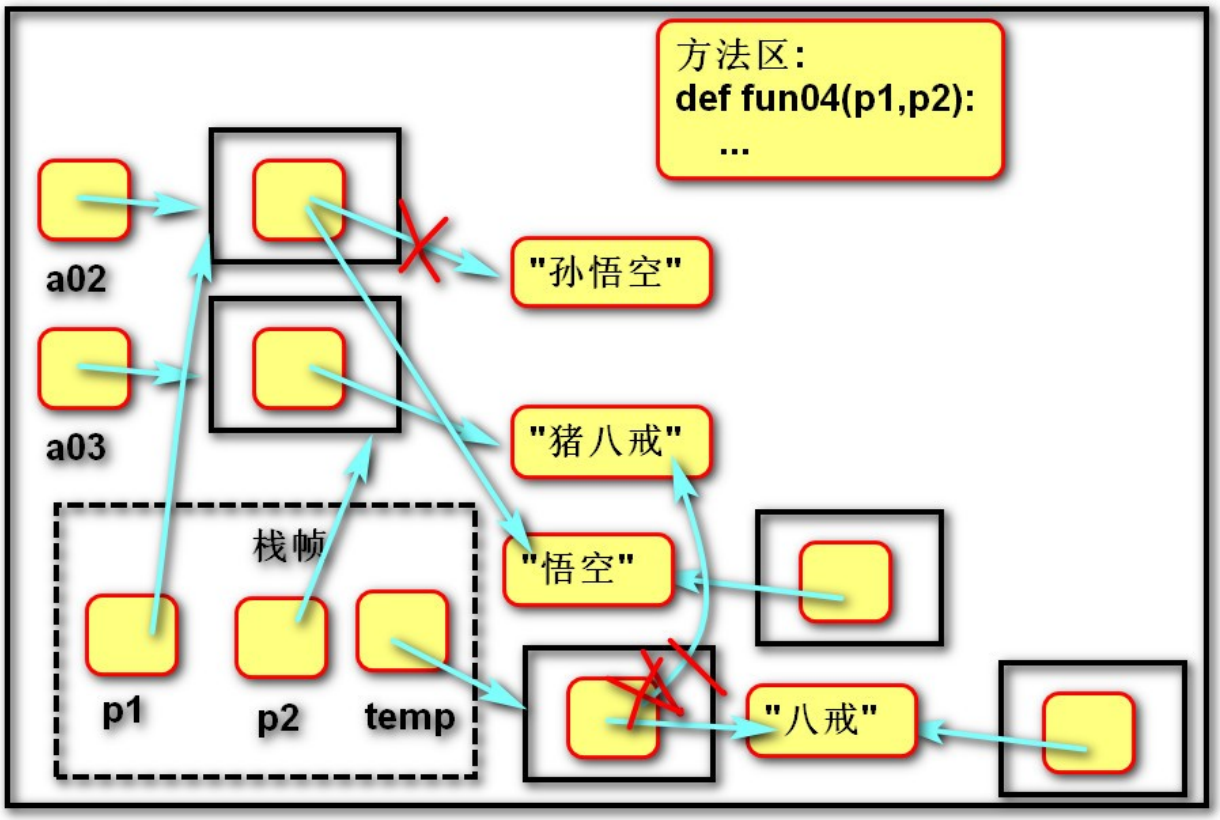
```python
#总结：
# 1. 向函数传递可变类型对象(列表)
# 2. 函数内部修改可变对象
# 3. 函数执行过后，不用通过返回值，也能拿到修改后的结果。
# 练习1:定义矩阵转置函数
# day07/exercise07.py
def matrix_transpose(list_matrix):
    result = []
    for c in range(len(list_matrix[0])):
        result.append([])
        for r in range(len(list_matrix)):
            result[c].append(list_matrix[r][c])
    return result
list01 = [
    [1, 2, 3],
    [5, 6, 7],
    [9, 10, 11],
]
re = matrix_transpose(list01)
print(re)
```

3.

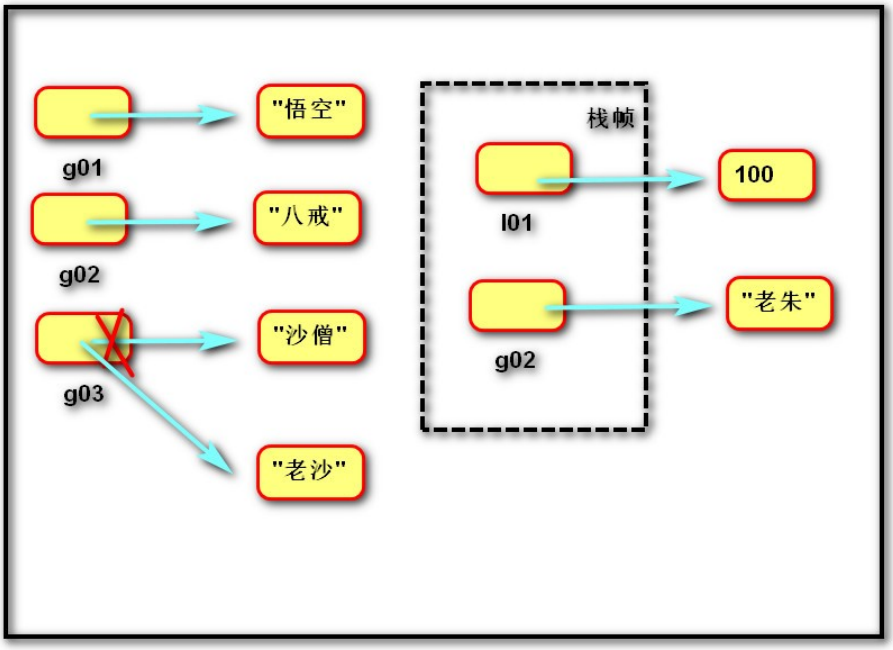
```
def fun03(p1,p2):
    p1[0] = "悟空"
    p2 = "八戒"

a02 = ["孙悟空"]
a03 = ["猪八戒"]
fun03(a02,a03)
print(a02)
print(a03)
```



```
def fun04(p1,p2):
    p1[:] = ["悟空"]
    temp = p2[:]
    temp[:] = ["八戒"]

a02 = ["孙悟空"]
a03 = ["猪八戒"]
fun04(a02,a03)
print(a02)
print(a03)
```



```
g01 = "悟空"
g02 = "八戒"
g03 = "沙僧"
def fun01():
    l01 = 100
    print(g02)
    g02 = "老朱"
    global g03
    g03 = "老沙"
fun01()
```



"""

```
    变量作用域
        在局部(函数内部)作用域中定义的变量，就是局部变量.
        在全局(.py文件中)作用域中定义的变量，就是全局变量.
    练习：exercise07.py
"""
# 全局变量
g01 = "悟空"
g02 = "八戒"
g03 = "沙僧"
def fun01():
    l01 = 100
    print(l01)# 只能在函数内部使用局部变量
    print(g01)# 在函数内部可以访问全局变量
    # 创建了局部变量g02，覆盖了全局变量g02
    g02 = "老朱"
    print("fun01---"+g02)# 老朱
    # 声明全局变量g03
    global g03
    g03 = "老沙"
fun01()
print("全局---"+g02)# 八戒
print(g03)
```

返回值

1. 定义：
   方法定义者告诉调用者的结果。
2. 语法：
   return 数据
3. 说明：
   return 后没有语句，相当于返回 None。
   函数体没有 return，相当于返回 None。

函数

```
定义：
    def 函数名称(形式参数):
        函数体
        return 数据
调用：
    函数名称(实际参数)
参数：函数调用者传递给函数定义者的信息
返回值:函数定义者告诉函数调用者的结果
    return 返回结果    退出方法
```

可变／不可变类型在传参时的区别

1. 不可变类型参数有:
   数值型(整数，浮点数,复数)
   布尔值 bool
   None 空值
   字符串 str
   元组 tuple
   固定集合 frozenset
2. 可变类型参数有:
   列表 list
   字典 dict
   集合 set
3. 传参说明：
   不可变类型的数据传参时，函数内部不会改变原数据的值。
   可变类型的数据传参时，函数内部可以改变原数据。

函数参数

实参传递方式 argument

位置传参

定义：实参与形参的位置依次对应。

序列传参

定义：实参用*将序列拆解后与形参的位置依次对应。

关键字传参

定义：实参根据形参的名字进行对应。

字典关键字传参

1. 定义：实参用**将字典拆解后与形参的名字进行对应。
2. 作用：配合形参的缺省参数，可以使调用者随意传参。

```
"""
    位置实参
        实际参数
"""
def fun01(a,b,c):
    print(a)
    print(b)
    print(c)
# 1. 位置实参: 实参根据位置与形参进行对应
fun01(1,2,3)
# 2.序列实参: 使用星号将序列中的元素拆开,与形参进行对应. 拆
# 2. 序列: 字符串, 列表, 元组
list01 = ["a","b","c"]
fun01(*list01)
str01 = "abd"
fun01(*str01)
```

```
# 3. 关键字实参:实参根据名称与形参进行对应
# 备注,作用7.10讲.
fun01(b = 2,a = 1, c = 3)
# 4. 字典实参:使用双星号将字典中的元素拆开,根据键形参进行对应,传递值.
dict01 = {"c":3,"a":1,"b":2}
fun01(**dict01)
dict01 = {"c":3,"a":1,"b":2} # c b a
fun01(*dict01)  # 只使用字典的键
dict01 = {"c":3,"a":1,"b":2,"c":33} #修改"c"
fun01(**dict01)
```