

## 2.4 自定义函数

Markdown by 黄艳艳

### 2.4.1 lambda函数

lambda 函数用于定义简单计算式，语法规则如下：

函数名 = lambda 输入参数 : 输入参数组成的计算式 函数名：遵循标识符命名法则

输入参数：Python中的各类对象，以逗号隔开

计算式：可以是多种形式，返回计算结果。可以引用已定义的变量和已导入的模块，当输入参数名与已定义的标识符相同时，已定义的标识符在lambda函数内失效。

调用形式："函数名 (参数)"

```
In [1]: import math
fnc=lambda x,y: math.sqrt(x**2+y**2)
fnc(3,4)
```

Out[1]: 5.0

```
In [2]: import math
x=2
fnc=lambda x,y: math.sqrt(x**2+y**2)
fnc(3,4)
```

Out[2]: 5.0

```
In [3]: import math
x=3
fnc=lambda y: math.sqrt(x**2+y**2)
fnc(4)
```

Out[3]: 5.0

### 2.4.2 自定义函数 def 语句

自定义函数是 Python 模块的重要组成部分，语法结构如下：

def 函数名(位置参数, 关键字参数):

    语句

    return 返回值

位置参数必须在关键字参数前，两者的位置不能混淆。位置参数：输入参数的位置决定  
关键字参数：可与位置参数无关，但需提供默认值。

return可以不写，也可以返回一个或者多个返回值，返回值可以是任意对象，例如函数，列表，字符串等

同样，在函数中可以随意引用主函数中导入的模块和已定义的变量，仅当输入参数名与已定义标识符重名时，已定义标识符在函数内失效。

调用方法："函数名(参数)"

由于 Python 代码是逐行执行的，因此，子函数必须写在调用之前，完成解释后机器才能

知道该函数的存在。

```
In [4]: from math import sqrt
def Pythagoras(a,b):
    c=sqrt(a**2+b**2)
    return c
if __name__=='__main__':
    x = Pythagoras(3,4)
    print(x)
```

5.0

在上段代码中没有设置关键字参数，下面来感受一下关键字参数的作用。

```
In [5]: from math import sqrt
def Pythagoras(a,b,s=None,n=None):
    c=s(a**n+b**n)
    return c
if __name__=='__main__':
    x = Pythagoras(3,4,n=2,s=sqrt)
    print(x)
```

5.0

上面的子函数中，前面的a,b为位置参数，必须按顺序给出，后面的s,n为关键字参数，调用是设置s为开方函数，b为乘方指数，默认值为"None"。调用时，由于关键字的存在，可以不按照顺序给出，需注意关键字参数必须在位置参数之后。调用时若不提供关键字，也可以使用"Pythagoras(3, 4, sqrt, 2)"的方式，但要严格按照位置提供各参数。

## 2.4.3 map 和 filter 函数

当定义了 lambda 函数或子函数时，可使用内置"map()"和"filter()"函数批量执行。"map()"的功能可看作是为函数逐一提供参数。

```
In [6]: fnc=lambda x: x**2
list(map(fnc,range(10)))
```

Out[6]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

"filter()"函数也有类似功能。但是，它是以函数返回值为依据进行筛选，返回符合条件的输入参数值。

```
In [7]: fnc= lambda x: x%2==0
list(filter(fnc,range(10)))
```

Out[7]: [0, 2, 4, 6, 8]

## 2.4.4 命名空间

命名空间即作用域。Python 中的命名空间是名称到对象的映射，它以字典的形式保存程序中的各种对象，包括全局变量和局部变量。可使用"globals().keys()"和"locals().keys()"分别进行查看。

```
In [8]: from math import sqrt
def Pythagoras(a,b,s=None,n=None):
    c = s(a**n+b**n)
    print('子函数中的全局变量')
    print(globals().keys())
    print('子函数中的局部变量')
    print(locals().keys())
    return c
if __name__=='__main__':
    x = Pythagoras(3,4,n=2,s=sqrt)
    print('主函数中的全局变量')
    print(globals().keys())
    print('主函数中的局部变量')
    print(locals().keys())
```

子函数中的全局变量

```
dict_keys(['__name__', '__doc__', '__package__', '__loader__', '__spec__', '__builtin__', '__builtins__', '_ih', '_oh', '_dh', 'In', 'Out', 'get_ipython', 'exit', 'quit', 'open', '_', '__', '__session__', '_i', '_ii', '_iii', '_i1', 'math', 'fnc', '_1', '_i2', 'x', '_2', '_i3', '_3', '_i4', 'sqrt', 'Pythagoras', '_i5', '_i6', '_6', '_i7', '_7', '_i8'])
```

子函数中的局部变量

```
dict_keys(['a', 'b', 's', 'n', 'c'])
```

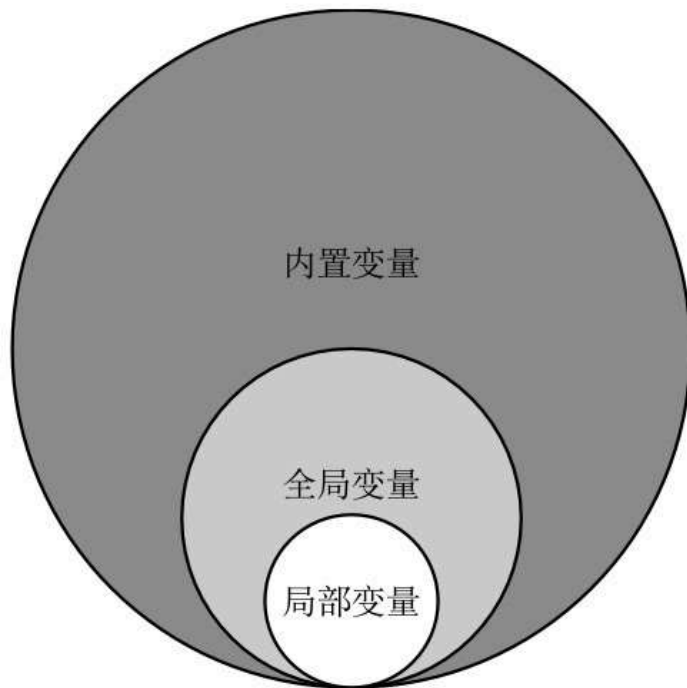
主函数中的全局变量

```
dict_keys(['__name__', '__doc__', '__package__', '__loader__', '__spec__', '__builtin__', '__builtins__', '_ih', '_oh', '_dh', 'In', 'Out', 'get_ipython', 'exit', 'quit', 'open', '_', '__', '__session__', '_i', '_ii', '_iii', '_i1', 'math', 'fnc', '_1', '_i2', 'x', '_2', '_i3', '_3', '_i4', 'sqrt', 'Pythagoras', '_i5', '_i6', '_6', '_i7', '_7', '_i8'])
```

主函数中的局部变量

```
dict_keys(['__name__', '__doc__', '__package__', '__loader__', '__spec__', '__builtin__', '__builtins__', '_ih', '_oh', '_dh', 'In', 'Out', 'get_ipython', 'exit', 'quit', 'open', '_', '__', '__session__', '_i', '_ii', '_iii', '_i1', 'math', 'fnc', '_1', '_i2', 'x', '_2', '_i3', '_3', '_i4', 'sqrt', 'Pythagoras', '_i5', '_i6', '_6', '_i7', '_7', '_i8'])
```

从输出的结果可以看出，在主函数中全局变量与局部变量相同。在子函数中，全局变量是主函数中的变量，局部变量是在子函数中定义的变量。因此，当主函数和子函数(包括 lambda 函数)中命名的标识符相同时，所属变量的优先级是先局部变量、后全局变量、再内置变量。如下图所示：



使用 `global` 和 `nonlocal` 语句可以打破命名空间的基本规则，建议感兴趣的读者通过 Python 在线说明文档进一步了解。

## 2.5 本章小结

**本章介绍了Python的语法特点、基本对象、基本语句、自定义函数。**

想要深入学习和了解 Python 语法基础的读者，建议通过 Python 在线说明 文档进行深入学习，有时网络的各类论坛也是个好帮手，或者系统性地阅读相关书籍。推荐阅读《Python 学习手册》，[美] 马克·卢茨(Mark Lutz)著，秦鹤和林明译。

提升Python学习兴趣，很多Python内置的模块可以边学边练，例如：

Python 内置的 turtle 绘图模块边学边练

(<https://docs.python.org/3/library/turtle.html>) 儿童Python教学的常用模块之一

Microbit(<https://python.microbit.org/>) 用于机器人编程的模块