PYTHON BASICS
函数 - 普通和匿名

2020-04-21

STEVEN WANG

上节总结

流程正常

流程异常

内容	语法	用处
条件语句	if	按条件执行
循环语句	while, for	重复执行
错误类型	*Error	要处理先了解
异常处理	raise/assert try-except-else-finally	先预防 后处理

流程控制

函数（上）

- 函数 101
- 普通函数
- 匿名函数

什么是函数

数学

$$\begin{tikzcd} \boxed{\text{输入} \; x} \arrow[r, "\text{映射}", "f"'] & \boxed{\text{输出} \; y} \end{tikzcd}$$

$$y = f(x)$$

$$y = f(x) = 2 \times x$$

$$y = 2 \times 5 = 10$$

Python

$$\begin{tikzcd} \boxed{\text{输入} \; \texttt{x}} \arrow[r, "\text{映射}", "\texttt{fun}"'] & \boxed{\text{输出} \; \texttt{y}} \end{tikzcd}$$

```
def fun(x):
    pass
```

```
def double(x):
    return 2*x
```

```
y = double(5)
```

为什么用函数

重复使用

```
print('Hello World!')
print('Hello World!')
print('Hello World!')
print('Hello World!')
print('Hello World!')
print('Hello World!')
```

WET

Write Every Time ✗

```
def hello():
    print('Hello World!')

hello()
hello()
hello()
hello()
hello()
hello()
```

DRY

Don’t Repeat Yourself ✓

为什么用函数

```
# Main program

# Code to read file in
<statement>
<statement>
<statement>

# Code to process file
<statement>
<statement>
<statement>
<statement>

# Code to write file out
<statement>
<statement>
```

模块化

```
def read():
    # Code to read file in
    <statement>
    <statement>
    <statement>

def process():
    # Code to process file
    <statement>
    <statement>
    <statement>

def write():
    # Code to write file out
    <statement>
    <statement>

# Main program
read()
process()
write()
```

VIPCLASS.SG

为什么用函数

命名空间

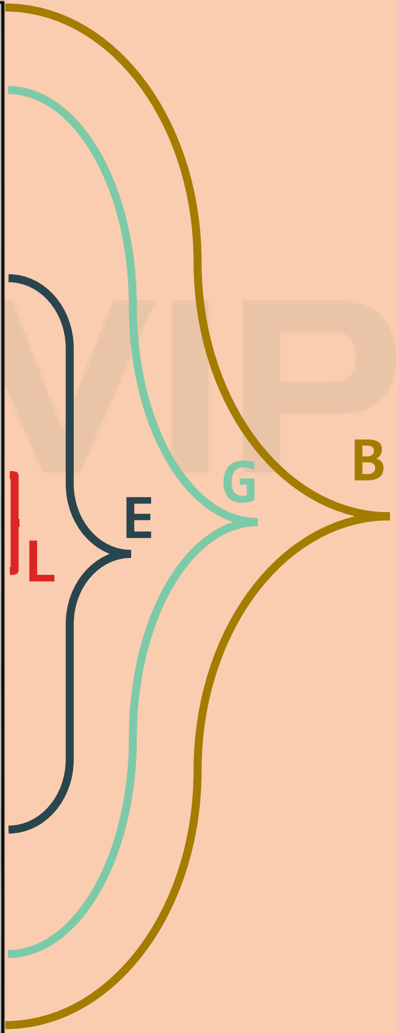
```python
x = 'global x'

def outer():
    x = 'enclosed x'

    def inner():
        x = 'local x'
        print(x)

    inner()
    print(x)

outer()
print(x)
```



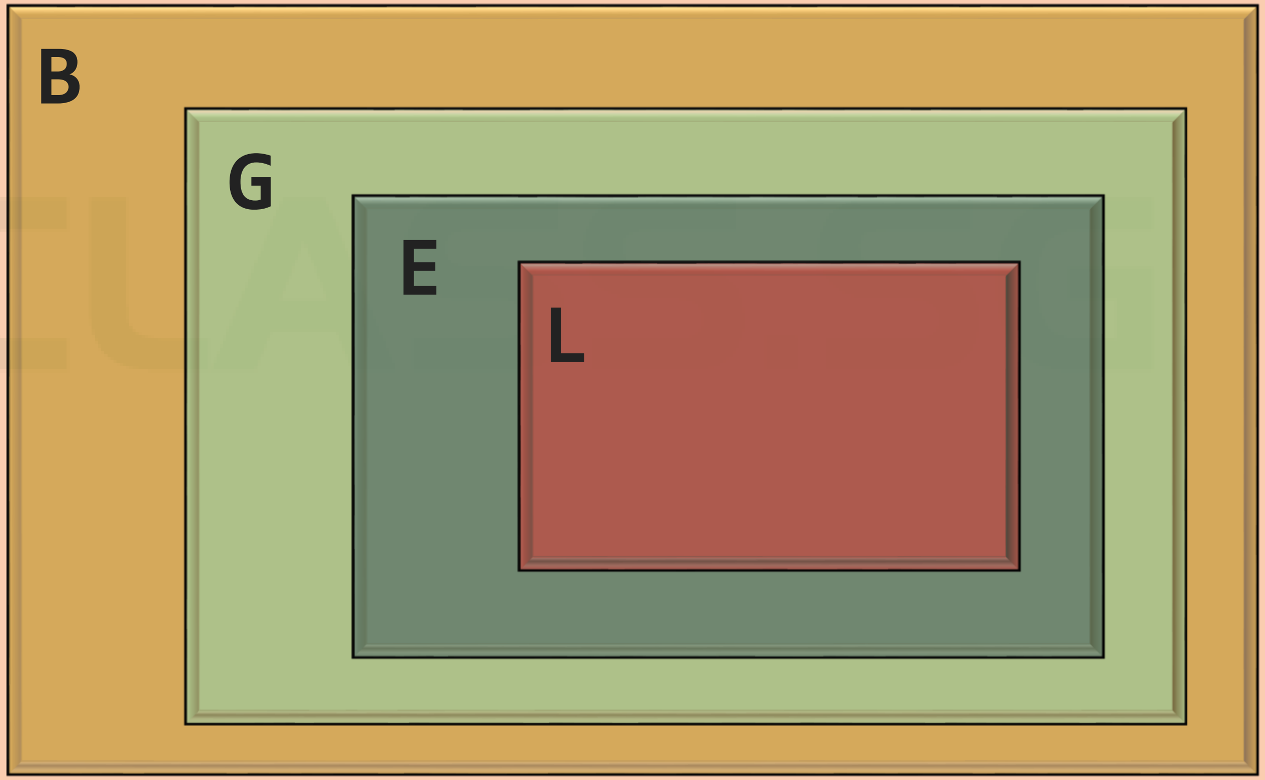
Local: 本地

Global: 全局

LEGB

Enclosed: 内嵌

Built-in: 自带



BGEL = 倍感饿了

实参和形参

函数定义

函数调用

$$\texttt{def fun(}\underbrace{\texttt{item, price}}\texttt{):} \longleftarrow \texttt{fun(}\underbrace{\texttt{'apple', 2.5}}\texttt{)}$$

形参
Parameters
Formal Parameters

实参
Arguments
Actual Parameters

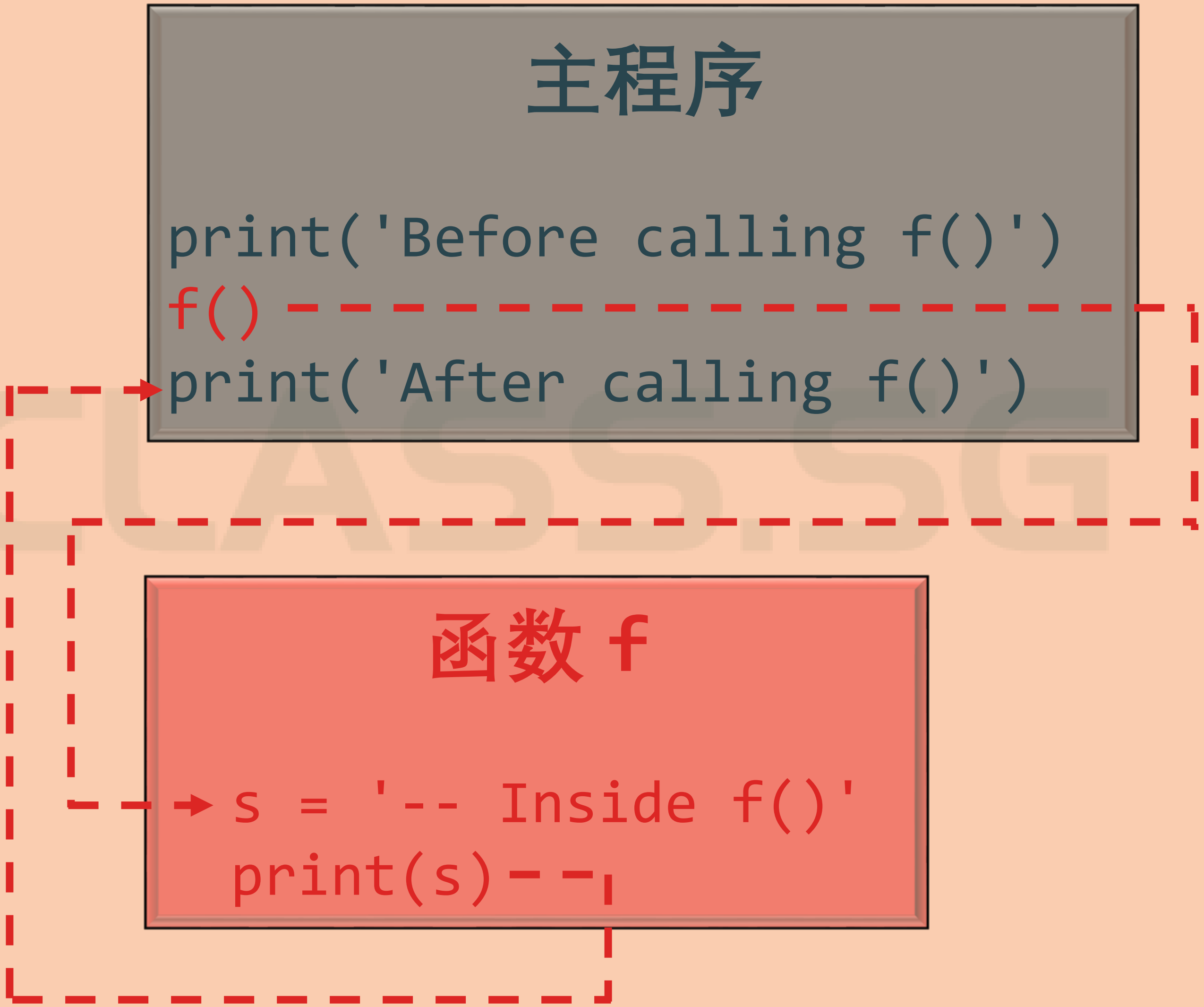
形参		实参
`item`	←	`'apple'`
`price`	←	`2.5`

VIPCLASS.SG

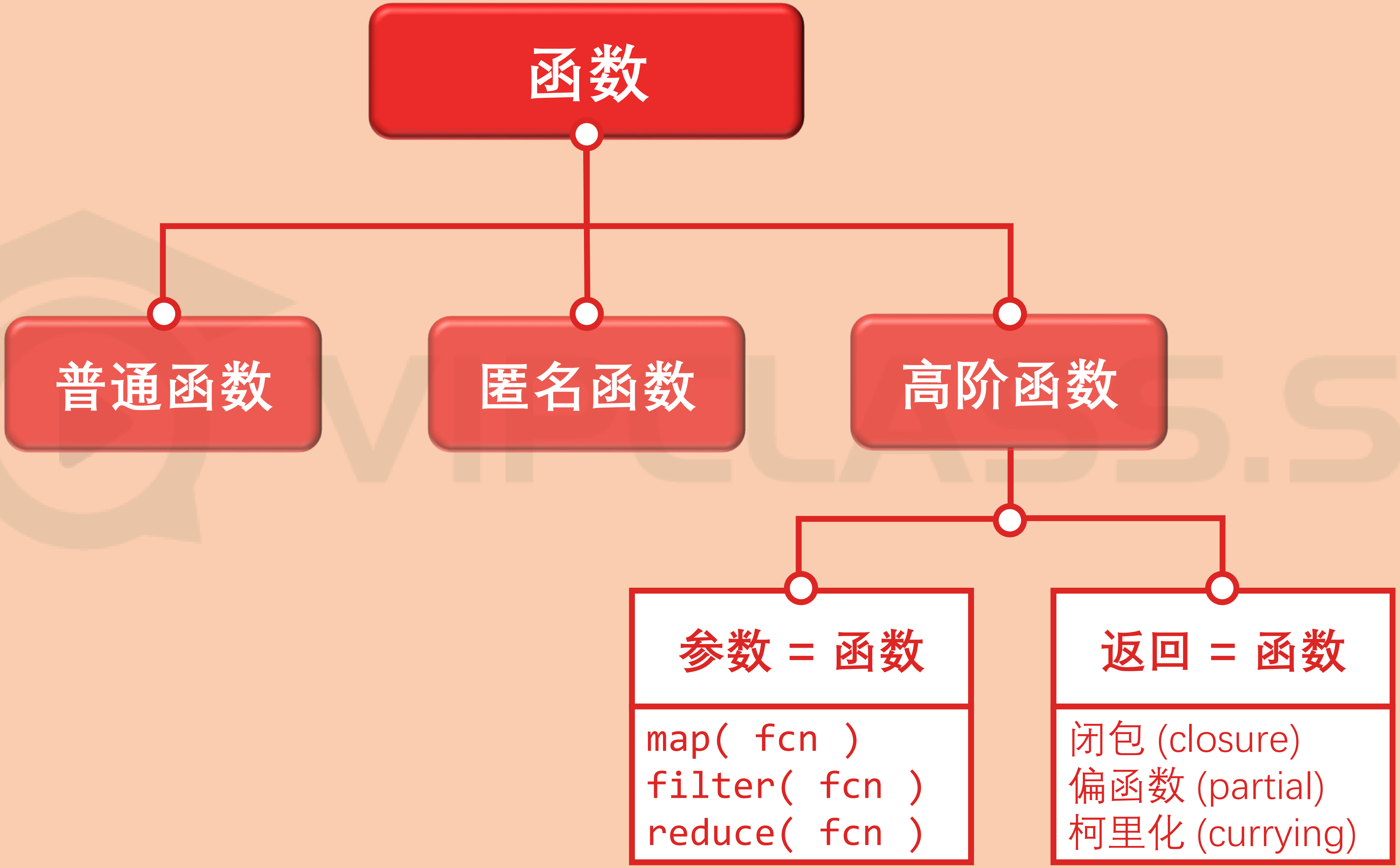
程序流程

```
def f():
    s = '-- Inside f()'
    print(s)

print('Before calling f()')
f()
print('After calling f()')
```



函数分类



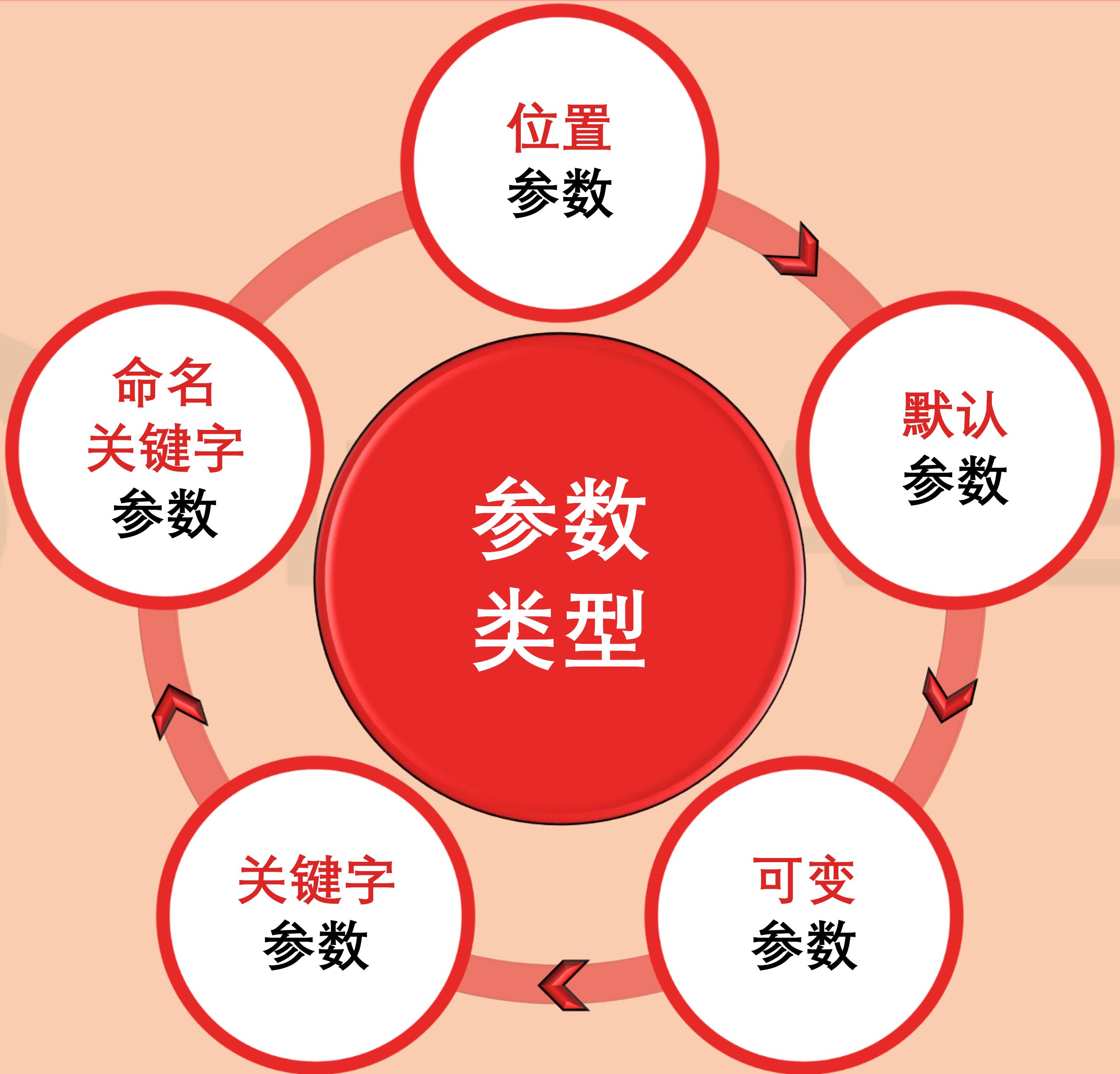
```
map( fcn )
filter( fcn )
reduce( fcn )
```

闭包 (closure)
偏函数 (partial)
柯里化 (currying)

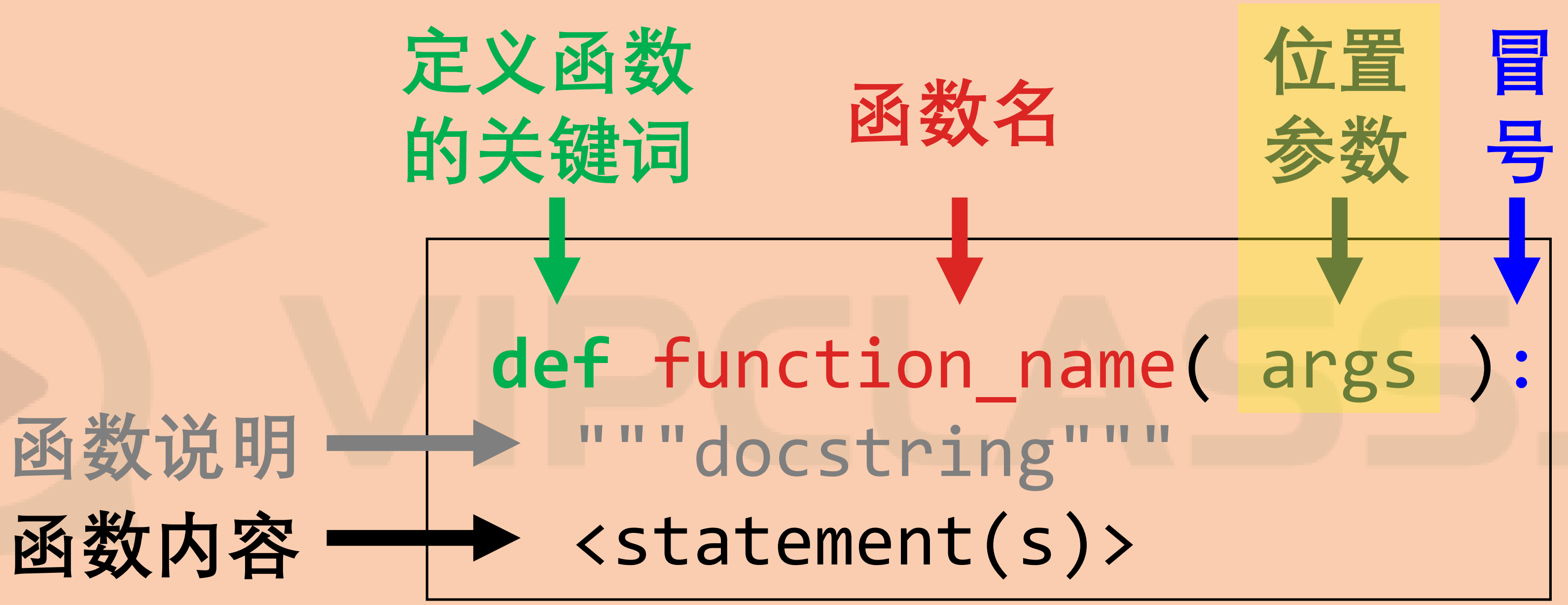
函数（上）

- 函数 101
- 普通函数
- 匿名函数

五大参数类型



位置参数 (Positional Argument)



```
def function_name( args ):
    """docstring"""
    <statement(s)>
```

位置参数例子

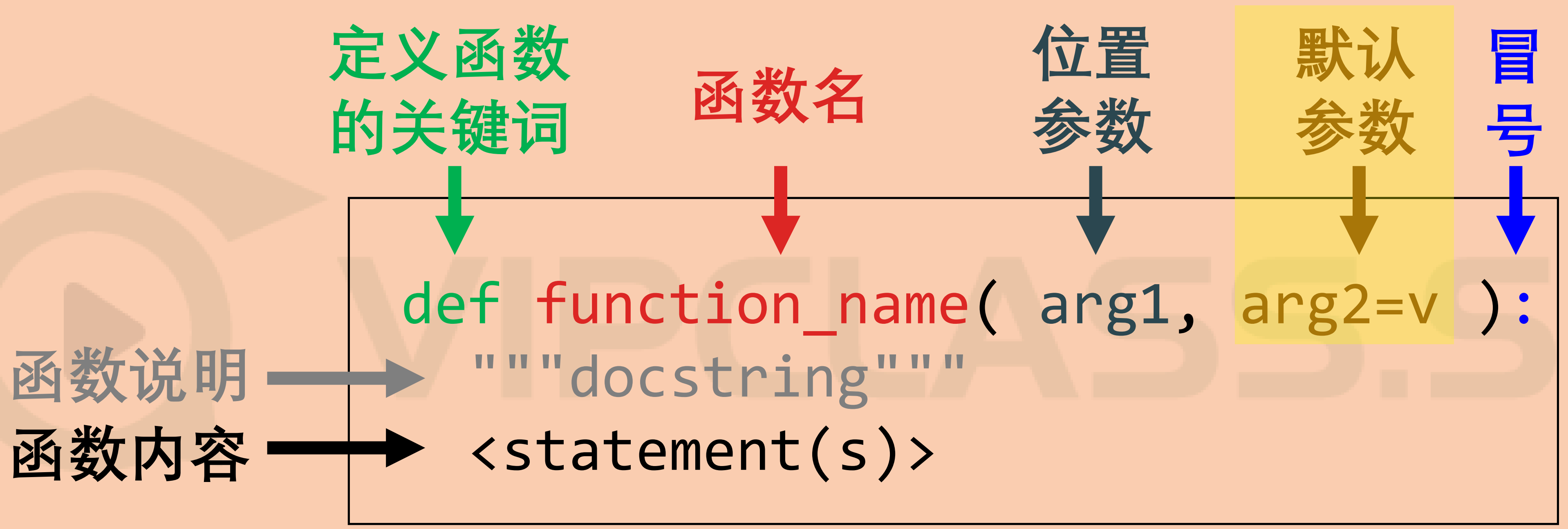
```
def function_name( args ):
   """docstring"""
   <statement(s)>
```

```
def inst( id, ntl ):
    print( 'id:', id )
    print( 'notional:', ntl )
```

```
inst( 'MM1001', 100 )
```

```
id: MM1001
notional: 100
```

默认参数



```
def function_name( arg1, arg2=v ):
    """docstring"""
    <statement(s)>
```

默认参数例子

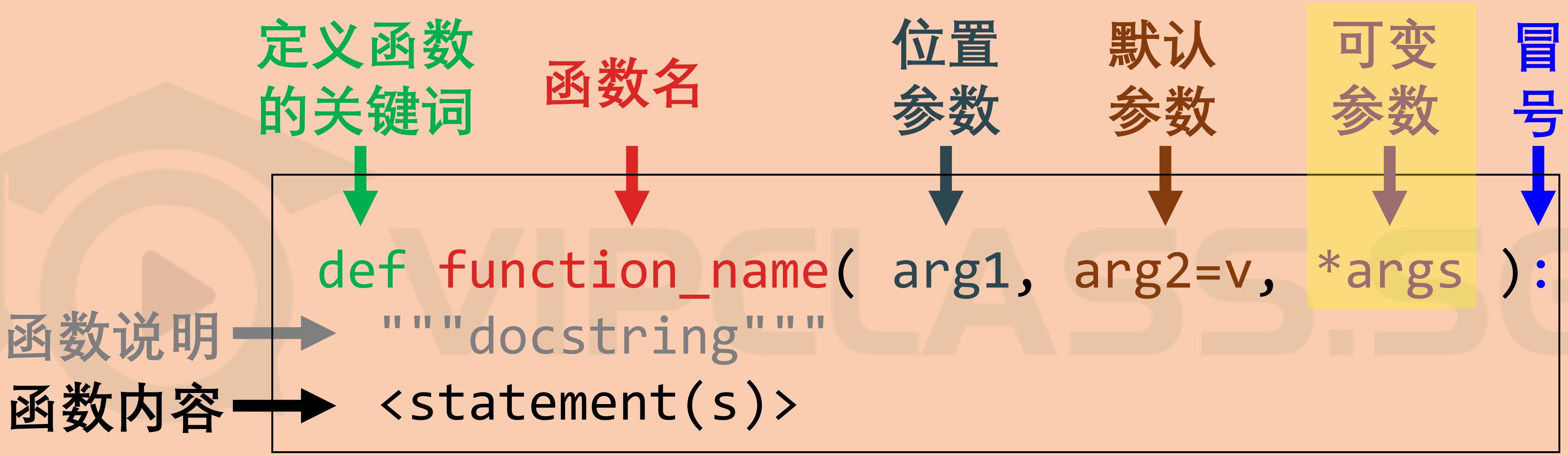
```
def function_name( arg1, arg2=v ):
   """docstring"""
   <statement(s)>
```

```
def inst( id, ntl=1, curR='CNY' ):
    print( 'id:', id )
    print( 'notional:', ntl )
    print( 'reporting currency:', curR )
```

```
inst( 'MM1001', 100 )
```

```
id: MM1001
notional: 100
reporting currency: CNY
```

可变参数



```
def function_name( arg1, arg2=v, *args ):
    """docstring"""
    <statement(s)>
```

可变参数例子

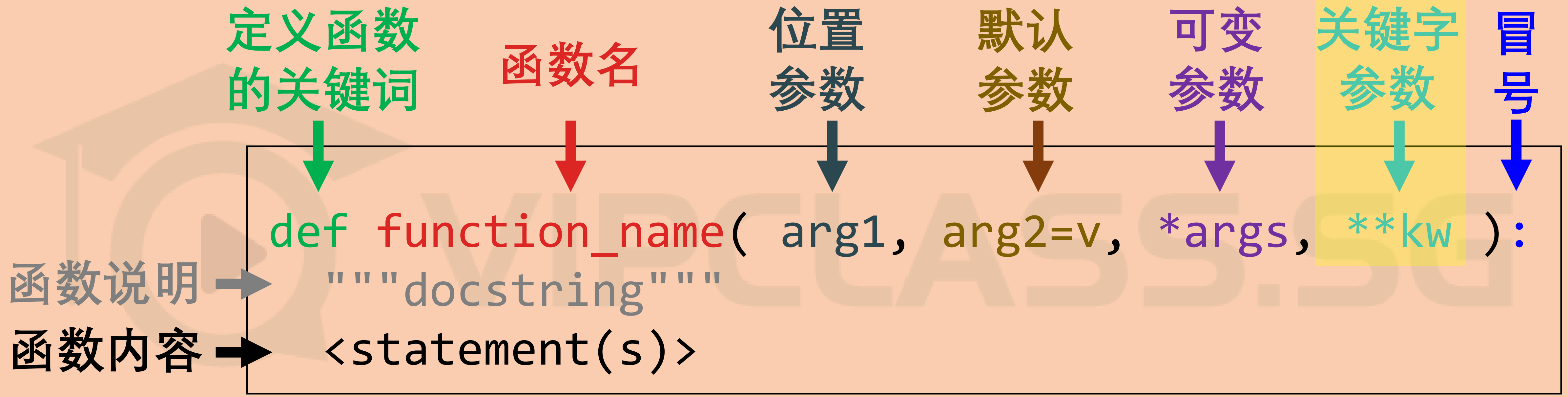
```
def function_name( arg1, arg2=v, *args ):
  """docstring"""
  <statement(s)>
```

```
def inst( id, ntl=1, curR='CNY', *args ):
    PV = 0
    for n in args: PV = PV + n
    print( 'id:', id )
    print( 'notional:', ntl )
    print( 'reporting currency:', curR )
    print( 'present value:', PV*ntl )
```

```
inst('MM1001', 100, 'EUR', 1,2,3)
```

```
id: MM1001
notional: 100
reporting currency: EUR
present value: 600
```

关键字参数



```
def function_name( arg1, arg2=v, *args, **kw ):
    """docstring"""
    <statement(s)>
```

关键字参数例子

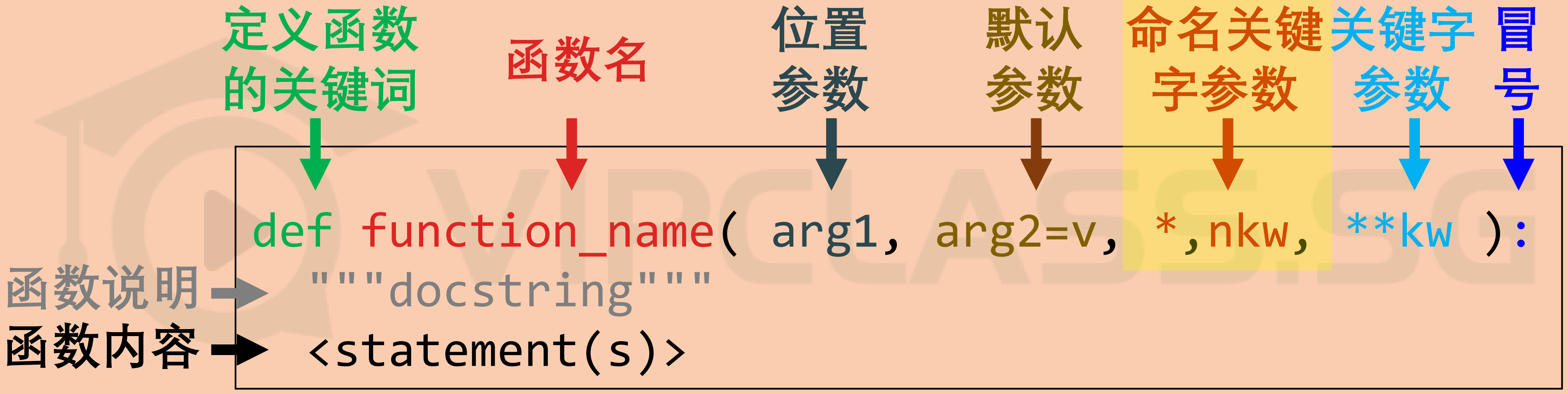
```
def function_name( arg1, arg2=v, *args, **kw ):
   """docstring"""
   <statement(s)>
```

```
def inst( id, ntl=1, curR='CNY', *args, **kw ):
    PV = 0
    for n in args: PV = PV + n
    print( 'id:', id )
    print( 'notional:', ntl )
    print( 'reporting currency:', curR )
    print( 'present value:', PV*ntl )
    print( 'keyword:', kw )
```

```
inst('MM1001', 1,2,3, ctp='GS')
```

```
id: MM1001
notional: 100
reporting currency: EUR
present value: 6
keyword: {'ctp': 'GS'}
```

命名关键字参数



```
def function_name( arg1, arg2=v, *,nkw, **kw ):
    """docstring"""
    <statement(s)>
```

命名关键字参数例子

```
def function_name( arg1, arg2=v, *,nkw, **kw ):
    """docstring"""
    <statement(s)>
```

```
def inst( id, ntl=1, curR='CNY', *,ctp, **kw ):
    print( 'id:', id )
    print( 'notional:', ntl )
    print( 'reporting currency:', curR )
    print( ‘counterparty:', ctp )
    print( ‘keyword:', kw )
```

```
inst('MM1001', 10, ctp='GS',
asset='FX')
```

```
id: MM1001
notional: 10
reporting currency: CNY
counterparty: GS
keyword: {'asset': 'FX'}
```

参数组合

对于位置参数、默认参数、可变参数、命名关键字参数和关键字参数 5 个参数，可按以下两组顺序使用

1. 位置参数 ⇨ 默认参数 ⇨ 可变参数 ⇨ 关键字参数
2. 位置参数 ⇨ 默认参数 ⇨ 命名关键字参数 ⇨ 关键字参数

可变参数：将元组或列表传递给 *args

关键字参数：将字典传递给 **kw

命名关键字参数：在分隔符 * 后面

为使代码易读，尽量少用太多的参数组合。

函数（上）

- 函数 101
- 普通函数
- 匿名函数

匿名函数

匿名函数的关键词 函数参数 冒号 表达式

```
lambda argument_list : expression
```

位置参数

```
func = lambda x, y: x*y
func(2, 3)
```

```
6
```

默认参数

```
func = lambda x, y=5: x*y
func(2)
```

```
10
```

可变参数

```
func = lambda *args: sum(args)
func(1, 2, 3, 4, 5)
```

```
15
```

关键字参数

```
func = lambda **kw: 1
func( name='Steven', age='36' )
```

```
1
```

误用 (misuse) 和过用 (overuse)

误用

```
lbd_sqr = lambda x: x ** 2
def sqr(x): return x ** 2

print( lbd_sqr )
print( sqr )
```

```
<function <lambda> at 0x232A855FAE8>
<function sqr at 0x232A855F268>
```

如果用 lambda 函数只是为了赋值给一个变量，用 def 来定义普通函数

过用

```
product = ["asian Option", "Barrier
Option", "Forward", "swap", "Cap",
"Swaption", "Accumulator"]
```

```
sorted( product, key=lambda p:
        (p.casefold(), len(p)) )
```

```
def alphabetical_and_length(str):
    return (str.casefold(), len(str))

sorted( product,
        key=alphabetical_and_length )
```

如果一个函数很重要，
它需要一个正规名字

总结

类型	定义

普通函数	`def fun_name(args):`
	`    statement`

匿名函数	`lambda args : expr`

args
- 位置参数
- 默认参数
- 可变参数
- 命名关键字参数
- 关键字参数

下节预告：高阶函数

VIPCLASS.SG
small class, world class

终身学习
快乐学习

王圣元　　　Steven Wang
微信公众号：王的机器