

## 第6讲 函数2

王肖燕





# 内容介绍

- 一. 函数返回值
- 二. 函数作用域
- 三. 函数闭包
- 四. 装饰器



# enumerate

- `L1=[3,2,1,6,5,9]`
- For index, i in `enumerate(L1)`:
  - `print(index, i)`

## ➡ 函数返回值

- 没有返回值的函数，返回为None
- return语句
  - ① 不加返回对象，表示结束函数，返回为None
  - ② 可同时返回多个值
    - return a,b,c
    - 调用函数返回(a,b,c)

## ➡ 举例

- ① 编写没有参数的函数calScore1，实现成绩输入，返回最高分、最低分、平均成绩的功能。
- ② 编写确定参数个数的函数calScore2，实现返回最高分、最低分、平均成绩的功能。
- ③ 编写不确定参数个数的函数calScore3，实现返回最高分、最低分、平均成绩的功能；函数调用时传入不确定个数的分数。



# 函数作用域

- **l**ocal
  - 局部变量，相对全局而言
- **e**nclosing
  - 有函数嵌套时
  - nonlocal
- **g**lobal
  - 全局变量，定义在函数外面
- **b**uiltins
  - 所有py文件执行的时候都会加载的内置模块

# ➡ local

```
1 def func():
2     print(number)
3     for i in range(number):
4         print(i)
5
6 number = 10
7 func()
```

如果函数内没有定义与全局同名的变量，而函数内只是使用了此变量，没有问题

```
1 def func():
2     print(number)
3     for i in range(number):
4         print(i)
5         number+=3
6
7 number = 10
8 func()
```

如果函数内没有定义与全局同名的变量，而函数内想改变此变量，则报错：  
UnboundLocalError: local variable 'number' referenced before assignment

# ➡ global

```
1 def func():  
2     global number  
3     print(number)  
4     for i in range(number):  
5         print(i)  
6     number+=3  
7  
8 number = 10  
9 func()
```

函数里可以使用全局变量，需要使用global  
关键字对全局变量在函数内进行声明



## ➡ 函数内修改全局

- 可变类型的全局变量
  - 在函数内可被修改

```
11 L=['a','b','c']
12
13 def func():
14     L=[]
15     L.append("d")
16     print(L)
17
18 func()
19 print(L)
```

```
11 L=['a','b','c']
12
13 def func():
14     #L=[]
15     L.append("d")
16     print(L)
17
18 func()
19 print(L)
```



# 函数内定义函数

```
1 a=10
2 def func():
3     a=20
4     b="abc"
5     c=['f','m','n']
6     def inner_func():
7         a=30
8         print("内部函数a=",a)
9     inner_func()
10    return a,b,c,inner_func
11
12 r=func()
13 print(r)
```

内部函数的调用  
内部函数可以被返回

```
1 a=10
2 b=15
3 def func(b):
4     a=100
5     def inner_func(c):
6         global b
7         c=a+b+c
8         b=c
9         print("a=",a)
10        print("b=",b)
11        print("c=",c)
12    inner_func(8)
13 func(6)
```

内部函数使用全局变量

```
1 a=10
2 b=15
3 def func(b):
4     a=100
5     def inner_func(c):
6         nonlocal b
7         c=a+b+c
8         b=c
9         print("a=",a)
10        print("b=",b)
11        print("c=",c)
12    inner_func(8)
13 func(6)
```

内部函数使用外部函数  
中的局部变量

# ➡ enclosing

```
1 a=10
2 def func():
3     a=20
4     b="abc"
5     c=['f','m','n']
6     def inner_func():
7         a=30
8         print("内部函数a=",a)
9     inner_func()
10    return a,b,c,inner_func
11
12 r=func()
13 print(r)
```

```
1 a=10
2 def func():
3     a=20
4     b="abc"
5     c=['f','m','n']
6     def inner_func():
7         #a=30
8         print("内部函数a=",a)
9     inner_func()
10    return a,b,c,inner_func
11
12 r=func()
13 print(r)
```

如果把func函数中的a=20也注释了， inner\_func()中会打印什么？

## ➡ 变量使用规则

1. 先找内部函数自身的变量
2. 如果内部函数没有自身的变量，则找外部函数的变量
3. 如果外部函数也没此变量，则找全局变量
4. 如果全局也没有此变量，则找builtins
5. 如果内部模块也没有，则报错

# ➔ global & nonlocal

```
1 a=10
2 b=15
3 def func(b):
4     a=100
5     def inner_func(c):
6         global b
7         c=a+b+c
8         b=c
9         print("a=",a)
10        print("b=",b)
11        print("c=",c)
12    inner_func(8)
13 func(6)
```

内部函数使用全局变量

```
1 a=10
2 b=15
3 def func(b):
4     a=100
5     def inner_func(c):
6         nonlocal b
7         c=a+b+c
8         b=c
9         print("a=",a)
10        print("b=",b)
11        print("c=",c)
12    inner_func(8)
13 func(6)
```

内部函数使用外部函数的变量

## ➡ 传递的参数—局部变量

```
16 set1={1,2,3,4}
17 def func(s):
18     s.add(5)
19     print(id(s))
20 func(set1)
21 print(set1)
22 print(id(set1))
```

```
2070992113000
{1, 2, 3, 4, 5}
2070992113000
```

```
16 set1={1,2,3,4}
17 def func(s):
18     s={7,8,9}
19     print(id(s))
20 func(set1)
21 print(set1)
22 print(id(set1))
```

```
2070992113000
{1, 2, 3, 4}
2070992112776
```

## ➡ 闭包

1. 在一个函数中定义了另一个函数
2. 内层函数使用了外层函数的变量
3. 返回值是内层函数

```
26 def outter_func(x):  
27     y=5  
28     def inner_func():  
29         n=x+y  
30         print("n={0}".format(n))  
31     return inner_func  
32 r=outter_func(1)  
33 r()
```

# ➡ 函数作参数

- 参数类型
  - int, str, float, bool, list, tuple, set, dict
- 函数A作为函数B参数
  - 传递的是函数A的地址
  - 函数也会往内存加载，把它当成对象来用
  - 当成特殊类型



# Thank You!

