東南大學 SOUTHEAST UNIVERSITY 人工智能学院

Python编程

授课教师：王洪松

邮箱：hongsongwang@seu.edu.cn

期中摸底考察成绩统计

区间	人数	积累人数	比例（%）	累计比例（%）		
90-100	1	1	1	1		优秀比例 26%
80-90	4	5	6	8		
75-80	12	17	19	27	合格比例 82%	
70-75	7	24	11	38		
65-70	9	33	14	53		
60-65	7	40	11	64		
55-60	7	47	11	75		
50-55	6	53	9	85		
45-50	4	57	6	91		须努力比例 12%
40-45	1	58	1	93		
30-40	3	61	4	98		
20-30	1	62	1	100		

Python函数设计

快速排序（Quick Sort）

- 快速排序也是一种分治算法，它选择一个基准元素（pivot），然后重新排列数组，所有小于基准的元素都放在基准前面，所有大于基准的元素都放在基准后面。然后递归地对这两个子数组进行排序。

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr

    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]

    return quick_sort(left) + middle +
quick_sort(right)
```

Python函数设计：基础

- **为什么要设计函数?**

Python函数设计：基础

- 函数的基本格式

def 函数名([参数列表]):

'''注释'''

函数体

斐波那契数列

```
def fib(n):
    '''accept an integer n.
       return the numbers less than n in Fibonacci sequence.'''
    a, b = 1, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
```

Python函数设计：基础

- 函数的基本格式

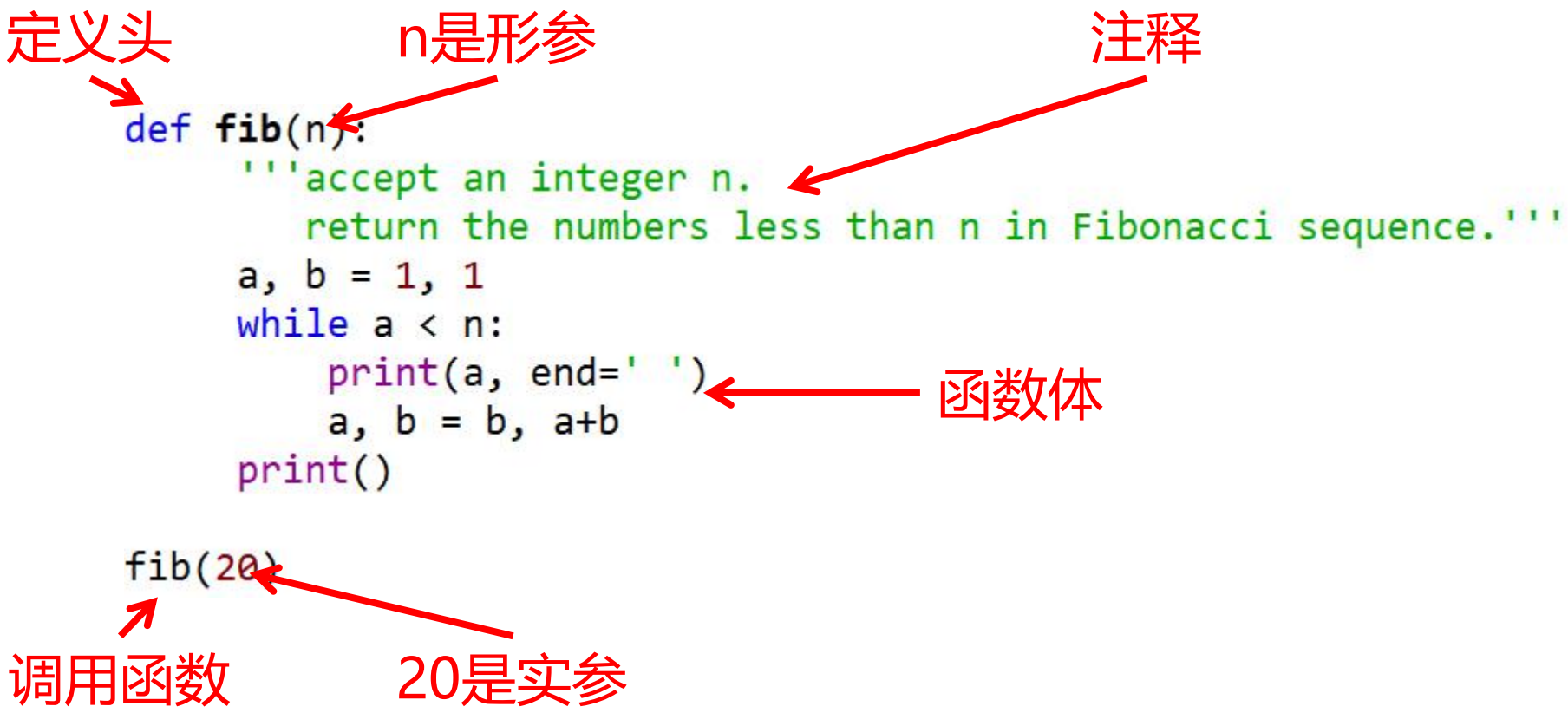
```
def 函数名([参数列表]):
    '''注释'''
    函数体
```

  - 函数参数不需要声明类型，也不需要指定返回值类型
  - 即使不需要接收参数，括号仍需保留
  - 括号后的冒号不可省略
  - def 关键字与函数体需保持缩进

问题：python 定义函数，为什么要冒号？

Python函数设计：基础

- 函数的基本格式



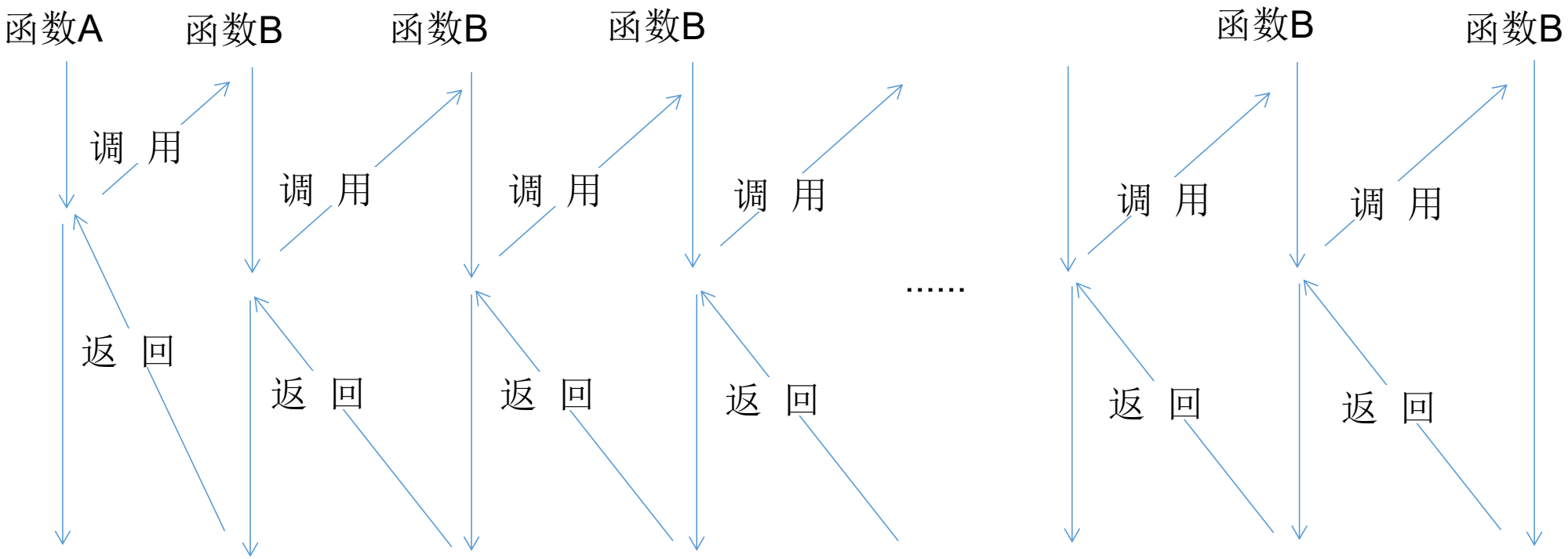
Python函数设计：基础

- 函数设计的一些思想
  - 设计函数时，应注意提高模块的内聚性，同时降低模块之间的耦合
  - 在实际开发中，通常会把一些项目常用函数封装到一个模块中，置于顶层文件夹，更方便管理
  - 注意: Python中允许函数递归调用

```
def func(n):
'''计算阶乘'''
        if n == 0:
                return 1
        else:
                return n * func(n - 1)
print(func(5))
```

Python函数设计：基础

- 函数的**递归调用**
  - 函数调用自己，自己再调用自己. . .
  - 当某个条件得到满足后停止调用，逐层返回结构直到第一次调用



Python函数设计：参数

- 形参与实参
  - 函数定义时括弧内为形参，一个函数可以没有形参，但是括弧必须要有，表示该函数不接受参数
  - 函数调用时向其传递实参，将实参的值或引用传递给形参
  - 大多数情况下，在函数内直接修改形参的值不影响实参

```
>>> def addOne(a):
        print(a)
        a = a+1
        print(a)

>>> a = 1
>>> addOne(a)
1
2
>>> a
1
```

Python函数设计：参数

- 在有些情况下，可以通过特殊的方式在函数内部修改实参的值

```
>>> def modifylist(a):
        a[0] = a[0] + 1
        print(a)

>>> a = [1]
>>> modifylist(a)
[2]
>>> a
[2]
```

Python函数设计：参数

- 参数类型
  - 普通参数、默认值参数、关键参数、可变长度参数等
  - 定义函数时不需要指定参数的类型，也不需要指定函数的类型，完全由调用者决定，类似于重载和泛型
  - 定义函数，参数个数没有限制，多个形参用逗号分隔
  - 函数编写如果有问题，只有在调用时才能被发现，传递某些参数时执行正确，而传递另一些类型的参数时则出现错误

```
29  def printMax(a, b):
30      if a > b:
31          print(a, 'is greater than', b)
32      else:
33          print(b, 'is greater than', a)
```

这里有什么问题?

Python函数设计：参数

- 参数类型：普通参数
  - 调用时形参和实参的顺序必须一致，且实参和形参的数量必须相同

```
29  def printMax(a, b):
30      if a > b:
31          print(a, 'is greater than', b)
32      else:
33          print(b, 'is greater than', a)
34  printMax(1, 2) # 2 is greater than 1
35  printMax(1, 'a')
36  #'>' not supported between instances of 'int' and 'str'
37  printMax(1, 2, 3)
38  #printMax() takes 2 positional arguments but 3 were given
```

Python函数设计：参数

- 参数类型：默认值参数
  - 默认值参数必须出现在函数参数列表的最右端，且任何一个默认值参数右边不能有非默认值参数

```
>>> def func(a=3, b, c):
        print(a, b, c)

SyntaxError: non-default argument follows default argument


>>> def func(a=3, b, c=5):
        print(a, b, c)

SyntaxError: non-default argument follows default argument
```

Python函数设计：参数

- 参数类型：默认值参数
  - 调用带有默认值参数的函数时，可以不对默认值参数赋值，也可以进行赋值

```
>>> def Join(List,sep=None):
        return (sep or ' ').join(List)

>>> a_List = ['a', 'b', 'c']
>>> Join(a_List)
'a b c'
>>> Join(a_List, ',')
'a,b,c'
```

Python函数设计：参数

- 参数类型：默认值参数
  - 默认值参数如果使用不当，会导致很难发现的逻辑错误

```
>>> def demo(newitem,old_list=[]):
        old_list.append(newitem)
        return old_list


>>> print(demo('5',[1,2,3,4])) #right
[1, 2, 3, 4, '5']
>>> print(demo('aaa',['a','b'])) #right
['a', 'b', 'aaa']
>>> print(demo('a')) #right
['a']
>>> print(demo('a')) #wrong
['a', 'a']
>>> print(demo('b')) #wrong
['a', 'a', 'b']
```

Python函数设计：参数

- 参数类型：默认值参数
  - 默认参数只在函数定义时被解释一次
  - 使用可变序列作为参数默认值时，一定慎重

```
>>> def demo(newitem,old_list=[]):
        old_list.append(newitem)
        return old_list

>>> print(demo('5',[1,2,3,4])) #right
[1, 2, 3, 4, '5']
>>> print(demo('aaa',['a','b'])) #right
['a', 'b', 'aaa']
>>> print(demo('a')) #right
['a']
>>> print(demo('a')) #wrong
['a', 'a']
>>> print(demo('b')) #wrong
['a', 'a', 'b']
```

这里old_list并不是空集了

典型错误示例：默认参数使用可变对象

python Copy code

```python
def add_item(item, container=[]):
    container.append(item)
    return container

print(add_item(1))   # 预期: [1], 实际: [1]
print(add_item(2))   # 预期: [2], 实际: [1, 2]
print(add_item(3))   # 预期: [3], 实际: [1, 2, 3]
```

❗ 出错原因

默认参数 `container=[]` 只在函数定义阶段创建了一次。
后续所有调用都**共享同一个 list**，导致内容不断累积。

python

```python
def counter(x, data={"count": 0}):
    data["count"] += x
    return data["count"]

print(counter(1))   # 1
print(counter(1))   # 2 (意外共享字典)
print(counter(5))   # 7
```

这里 `data` 字典作为默认参数，只创建了一次，被所有调用共享。

✅ 正确写法（使用 None）

python Copy code

```python
def add_item(item, container=None):
    if container is None:
        container = []
    container.append(item)
    return container

print(add_item(1))   # [1]
print(add_item(2))   # [2]
print(add_item(3))   # [3]
```

Python函数设计：参数

- 参数类型：默认值参数
  - 可以使用函数名.__defaults__查看默认参数的当前值

```
>>> def demo(newitem,old_list=[]):
        old_list.append(newitem)
        return old_list

>>> print(demo('aaa',['a','b']))
['a', 'b', 'aaa']
>>> demo.__defaults__
([],)
>>> print(demo('a'))
['a']
>>> demo.__defaults__
(['a'],)
```

Python函数设计：参数

- 参数类型：关键参数
  - 关键参数主要指实参，即调用函数时的参数传递方式
  - 通过关键参数传递，实参顺序可以和形参顺序不一致，但不影响传递结果，避免了用户需要牢记位置参数顺序的麻烦

```
>>> def demo(a, b, c=5):
        print(a, b, c)

>>> demo(3, 7)
3 7 5
>>> demo(a=7, b=3, c=6)
7 3 6
>>> demo(c=8, a=9, b=0)
9 0 8
```

Python函数设计：参数

- 参数类型：可变长度参数
  - 可变长度参数主要有两种形式：*parameter和**parameter
    - 前者：接受多个实参并将其放在一个元组中
    - 后者：接受多个关键参数，并存到字典中

Python函数设计：参数

- 参数类型：可变长度参数
  - *parameter用法

```
>>> def demo(*p):
        print(p)

>>> demo(1, 2, 3)
(1, 2, 3)
>>> demo(1, 2, 3, 4, 5, 6, 7)
(1, 2, 3, 4, 5, 6, 7)
```

Python函数设计：参数

- 参数类型：可变长度参数
  - * * parameter用法

```
>>> def demo(**p):
        for item in p.items():
            print(item)

>>> demo(x=1, y=2, z=3)
('x', 1)
('y', 2)
('z', 3)
```

Python函数设计：参数

- 参数类型：可变长度参数
  - 几种不同类型的参数可以混合使用，但是不建议这样做

```
>>> def func_4(a, b, c=4, *aa, **bb):
        print(a, b, c)
        print(aa)
        print(bb)

>>> func_4(1, 2, 3, 4, 5, 6, 7, 8, 9, xx='1', yy='2', zz=3)
1 2 3
(4, 5, 6, 7, 8, 9)
{'xx': '1', 'yy': '2', 'zz': 3}
```

Python函数设计：参数

- 参数传递的序列解包
  - 传递参数时，可以通过在实参序列前加星号将其解包，然后传递给多个单变量形参

```
>>> def demo(a,b,c):
        print(a+b+c)

>>> seq=[1,2,3]
>>> demo(*seq)
6
>>> dic = {1:'a',2:'b',3:'c'}
>>> demo(*dic.values())
abc
```

Python函数设计：参数

- 参数传递的序列解包
  - 如果实参是字典，可以在前面加两个*进行解包，等价于关键参数

```
>>> def demo(a,b,c):
        print(a+b+c)

>>> dic = {1:'a',2:'b',3:'c'}

>>> demo(**dic)
Traceback (most recent call last):
  File "<pyshell#91>", line 1, in <module>
    demo(**dic)
TypeError: demo() keywords must be strings
>>> dic = {'a':1,'b':2,'c':3}
>>> demo(**dic)
6
```

Python知识复习：序列解包/多变量赋值

- 多变量赋值

```
>>> a, b = 0, 1
>>> a
0
>>> b
1
```

```
>>> a, b = 0, 1
>>> a, b = b, a
>>> a
1
>>> b
0
```

```
>>> a = 2
>>> a, b = 1, a
>>> a
1
>>> b
2
```

存的是指向2的引用

```
>>> (r, g, b) = ["Red","Green","Blue"]
>>> r
'Red'
>>> g
'Green'
>>> b
'Blue'
```

Python知识复习：序列解包/多变量赋值

- 多变量赋值和“序列解包”

```
>>> x, y, z = 1, 2, 3
>>> a_tuple = (1, 2, 3)
>>> (x, y, z) = a_tuple
>>> x, y, z = a_tuple
>>> x, y, z = range(3)   #对range对象进行序列解包

>>> x, y, z = iter([1,2,3])      #对迭代器对象进行序列解包
>>> x, y, z = map(int, range(3))#对map对象进行序列解包
>>> x, y, z = sorted([1, 3, 2]) #sorted()方法返回的列表进行解包
>>> x, y, z = '123'
>>> x = [1, 2, 3, 4, 5]
>>> x[:3] = map(str, range(3))   #切片也支持序列解包
>>> x
['0', '1', '2', 4, 5]
```

Python函数设计：参数

- return语句
  - return语句用来从一个函数中返回，即跳出函数，也可用return语句从函数中返回一个值
  - 如果函数没有return语句，Python将认为该函数以return None结束
  - 在调用内置数据类型的方法时，一定要注意该方法有没有返回值

```
>>> a_list = [1, 5, 3]
>>> print(sorted(a_list))          #return一个新列表
[1, 3, 5]
>>> print(a_list.sort())           #return none
None
```

Python函数设计：变量

- 变量作用域
  - 变量起作用的范围称为变量的作用域，不同作用域内的变量可以重名，互不影响
  - 一个变量在函数外部定义和在函数内部定义，其作用域是不同的
  - 在函数内部定义的普通变量只在函数内部起作用，成为局部变量。当函数执行结束后，局部变量自动删除
  - 局部变量的引用比全局变量速度快

Python函数设计：变量

- 变量作用域
  - 全局变量可通过关键字global 定义，这分两种情况
    - 一个变量已在函数外定义，如果在函数内需要为这个变量赋值，并要将这个赋值反映到函数外，可在函数内使用global关键字将其声明为全局变量
    - 一个变量在函数外没有定义，在函数内部也可以声明为新的全局变量，该函数执行后，将增加一个新的全局变量

Python函数设计：变量

- 变量作用域

```
>>> def demo():
        global x
        x = 3
        y = 4
        print(x,y)

>>> x = 5
>>> demo()
3 4
>>> x
3
>>> y
Traceback (most recent call last):
  File "<pyshell#106>", line 1, in <module>
    y
NameError: name 'y' is not defined
```

Python函数设计：变量

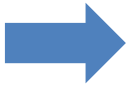
- 变量作用域
  - 在某个作用域内任意位置有为变量赋值的操作，该变量在这个作用域内就是局部变量，除非有global为其声明

```
>>> x = 3
>>> def func():
        print(x)

>>> func()
3
>>> def func():
        print(x)
        x = 5

>>> func()
Traceback (most recent call last):
  File "<pyshell#120>", line 1, in <module>
    func()
  File "<pyshell#119>", line 2, in func
    print(x)
UnboundLocalError: local variable 'x' referenced before assignment
```

Python函数设计：变量

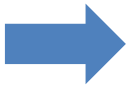
- 变量作用域
  - Python for循环中的索引变量会泄露围合(enclosing)的函数作用域

```
for i in [1, 2, 3]:
        pass
print(i)
```



3

```
def foo(lst):
        a = 0
        for i in lst:
                a += i
        b = 1
        for t in lst:
                b *= i
        return a, b
print(foo([2,3,4]))
```

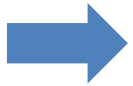


?

Python函数设计：变量

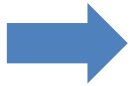
- 变量作用域

```
def foo():
        lst = []
        for i in range(4):
                lst.append(i)
        print([f for f in lst])
foo()
```



?

```
def foo():
        lst = []
        for i in range(4):
                lst.append(lambda: i)
        print([f() for f in lst])
foo()
```



?

Python函数设计：特殊函数

- lambda表达式
  - lambda表达式可以用来声明匿名函数，即没有函数名字的临时使用的小函数，适合一个函数作为另一个函数参数的场景，也可以用来定义具名函数
  - lambda表达式只可以包含一个表达式，且该表达式的计算结果为函数的返回值，不允许包含其他复杂的语句，但在表达式中可以调用其他函数

```
>>> f=lambda x, y, z:x+y+z
>>> f(1, 2, 3)
6
>>> g=lambda x, y=2, z=3:x+y+z
>>> g(1)
6
>>> g(2, z=4, y=5)
11
```

Python函数设计：特殊函数

- lambda表达式

```
>>> def demo(n):
        return n*n

>>> demo(5)
25
>>> a_list=[1,2,3,4,5]
>>> map(lambda x:demo(x),a_list)
<map object at 0x00986270>
>>> list(_)
[1, 4, 9, 16, 25]
```

Python函数设计：特殊函数

- lambda表达式

```
>>> import random
>>> data=list(range(20))
>>> random.shuffle(data)
>>> data
[6, 13, 8, 12, 17, 14, 9, 1, 15, 0, 11, 3, 4, 2, 19, 18, 7, 5, 16, 10]
>>> data.sort(key=lambda x:len(str(x)), reverse=True)
>>> data
[13, 12, 17, 14, 15, 11, 19, 18, 16, 10, 6, 8, 9, 1, 0, 3, 4, 2, 7, 5]
>>> data.sort(key=lambda x:x)
>>> data
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

Python函数设计：案例

- 编写函数计算圆的面积

```
>>> from math import pi as PI
>>> import types
>>> def CircleArea(r):
        if isinstance(r,int) or isinstance(r,float): #确保接收的参数为数值
            return PI*r*r
        else:
            print('You must give me an integer or float as radius.')

>>> print(CircleArea(3))
28.274333882308138
```

Python函数设计：案例

- 编写函数，接收任意多个实数，返回一个元组，其中第一个元素为所有参数的平均值，其他元素为所有参数中大于平均值的实数

```
>>> def demo(*para):
        avg = sum(para)/len(para)
        g = [i for i in para if i>avg]
        return (avg,)+tuple(g)

>>> print(demo(1,2,3,4))
(2.5, 3, 4)
```

Python函数设计：案例

- 编写函数，接收包含20个整数的列表lst和一个整数k作为参数，返回新列表
- 处理规则为：将列表lst中下标k之前的元素逆序，下标k之后的元素逆序，然后将整个列表lst中的所有元素逆序

```
>>> def demo(lst,k):
        x = lst[:k]
        x.reverse()
        y = lst[k:]
        y.reverse()
        r = x+y
        r.reverse()
        return r

>>> lst = list(range(1,21))
>>> print(lst)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
>>> print(demo(lst,5))
[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 1, 2, 3, 4, 5]
```

Python函数设计：案例

- 编写函数，接收一个包含若干整数的列表参数lst，返回一个元组，其中第一个元素为列表lst中的最小值，其余元素为最小值在列表lst中的下标

```
>>> import random
>>> def demo(lst):
        m = min(lst)
        result = (m,)
        for index, value in enumerate(lst):
            if value==m:
                result = result+(index,)
        return result

>>> x = [random.randint(1,20) for i in range(50)]
>>> print(x)
[16, 13, 8, 15, 10, 11, 11, 8, 11, 13, 8, 14, 9, 20, 3, 20, 11, 2, 16, 9, 17, 7, 10, 3, 6,
16, 3, 12, 6, 3, 15, 14, 17, 9, 9, 12, 19, 19, 4, 13, 4, 8, 7, 7, 1, 19, 4, 13, 10, 19]
>>> print(demo(x))
(1, 44)
```

Python函数设计：小结

- 函数的基本含义
  - 定义头、函数名、参数、函数体
- 参数与变量
  - 普通参数、默认值参数、关键参数、可变长度参数
  - 全局变量与局部变量
- 特殊函数
  - lambda表达式

作业题

- 删除文本中的重复单词
  - 编写一个函数，输入参数为由单词组成的文本，不同单词由空格分隔，删除文本中的重复单词并输出
  - 在命令行下输入文本，输出处理后的文本
- 计算两个矩形交叉区域的面积
  - 平面上有两个矩形，它们的边平行于直角坐标系的两个坐标轴，给定这两个矩形左下角和右上角的坐标，计算交叉区域的面积
- 将嵌套列表转成单个元素的列表
  - 输入：[1, 2, [3, 4, [5, 6], 7], 8]，输出：[1, 2, 3, 4, 5, 6, 7, 8]

作业题

- 列表中元素按照某个值排序
  - 假设列表[{ ‘name':'a','age':20},{'name':'b','age':30},{'name':'c','age':25}]
  - 按照年龄从小到大的顺序排列该列表的元素
  - 至少采用3种不同方法实现

- 将字典作为函数的可变参数
  - 定义一个函数，令其包含普通参数、默认值参数和可变长度参数
  - 定义一个字典并将其作为函数的默认值参数和可变长度参数