Python for Senior

Lesson 2 v1.0.0

2016.9 by David.Yi

复习

- If...elif...else 语句
- 列表 list 的 sort 方法
- 函数式编程 匿名函数 lambda 初步认识
- 多重循环嵌套

函数使用进阶-默认参数

- 位置参数和默认参数的区别
  - 位置参数：必须按照顺序准确传递，如果数量和顺序不对，就会可能造成程序错误；调用函数时候，如果写了参数名称，那么位置就不重要了
  - 默认参数：在参数申明的时候跟一个用于默认值的赋值语句，如果调用函数的时候没有给出值，那么这个赋值语句就执行
  - 注意：所有必须的参数要在默认参数之前
- 默认参数的好处
  - 减少程序复杂度
  - 降低程序错误可能性
  - 更好的兼容性

函数使用进阶-默认参数-举例

```
# 默认参数
def cal_0(money, rate=0.1):
    return money + money * rate

print(cal_0(100))
print(cal_0(100,0.2))
```

```
110.0
120.0
```

```
def cal_1(money, bonus=1000, month=12):
    i = money * month + bonus
    return i

print(cal_1(5000))
print(cal_1(5000, 2000))
print(cal_1(5000, 2000, 10))
```

```
61000
62000
52000
```

- 可以看到默认参数给程序开发带来的灵活性
- 改变函数声明时候的默认参数值，以及改变调用函数的参数，看看不同的结果

函数使用进阶-默认参数-举例

- 举例：显示一个指定层数的由*组成的三角形，如果没有指定层数，则输出为5层

```
    *
   ***
  *****
 *******
*********
***********
*************
```

```
def draw_triangle(n=5):

    for i in range(n+1):
        print(' '*(n-i),'*'*(2*i-1))

draw_triangle(7)
draw_triangle()
```

函数使用进阶-可变长度的参数-元组

- 可变长度的参数，分为不提供关键字和提供关键字两种模式，分别为元组 tuple 和字典 dict
- 下面是一个计算平均值的函数，因为我们不知道用户会输入多少个数字，所以最好的办法就是用可变长度的参数，同时也没有必要为每个数字起名字，所以用不提供关键字方式比较好

```
# 可变长度的参数
def cal_2(kind, *numbers):
    if kind == 'avg':
        n = 0
        for i in numbers:
            n = n + i
        return n / len(numbers)

print(cal_2('avg', 1,2,3,4))
```

```
2.5
```

函数使用进阶-可变长度的参数-举例

- 我们来增加计算平均数和求和这两个功能，并进行优化

```
# 可变长度的参数
def cal_2(kind, *numbers):
    if kind == 'avg':
        n = 0
        for i in numbers:
            n = n + i
        return n / len(numbers)
    if kind == 'sum':
        n = 0
        for i in numbers:
            n = n + i
        return n

print(cal_2('avg', 1,2,3,4))
print(cal_2('sum', 1,2,3,4))
```

```
2.5
10
```

```
# 可变长度的参数
def cal_2(kind, *numbers):

    n = 0
    for i in numbers:
        n = n + i

    if kind == 'avg':
        return n / len(numbers)
    if kind == 'sum':
        return n

print(cal_2('avg', 1,2,3,4))
print(cal_2('sum', 1,2,3,4))
```

```
2.5
10
```

函数使用进阶-可变长度的参数-举例

- 可变长度的参数，如果是提供关键字，就是字典 dict，需要提供 key – value
- 将字典作为参数传递的时候，可以直接传一个字典变量，也可以在参数列表中写明 key 和 value

```
# 可变长度参数 元组 字典
def cal_3(*numbers, **students):

    for i in numbers:
        print(i)
    for k,v in students.items():
        print(k,v)

numbers = [1,2,3,4,5]
students = {'tom':90, 'jerry':95, 'mary':100}
cal_3(*numbers, **students)
```

函数使用进阶-可变长度的参数-举例

- 对于同时传递元组和字典，元组在前，字典在后
- 用字典传递参数，可以让参数非常灵活
- 如果没有传递 kind 这个 key，怎么办？
- 尝试自己增加一种运算形式

```
# base on cal_2,
# 传递元组和字典参数的最合适写法
def cal_5(*numbers, **kind):

    if 'kind' in kind:
        kind_value = kind.get('kind')

    n = 0
    for i in numbers:
        n = n + i

    if kind_value == 'avg':
        return n / len(numbers)
    if kind_value == 'sum':
        return n

print(cal_5(1,2,3,4, kind='avg'))
print(cal_5(1,2,3,4, kind='sum'))
```

```
2.5
10
```

函数使用进阶-可变长度的参数-举例

- 举例，在前面计算平均数或求和程序基础上修改，在 kind 中增加一个 名为max 的key，如果存在这个 key，检查其 value，是 ignore，则计算的时候去掉最大值

```
# kind 中 增加 max key 以及相关功能
def cal_6(*numbers, **kind):

    if 'kind' in kind:
        kind_value = kind.get('kind')

    if 'max' in kind:
        if kind.get('max') == 'ignore':
            numbers = list(numbers)
            numbers.remove(max(numbers))

    n = 0
    for i in numbers:
        n = n + i

    if kind_value == 'avg':
        return n / len(numbers)
    if kind_value == 'sum':
        return n

print(cal_6(1,2,3,4, kind='avg', max='ignore'))
print(cal_6(1,2,3,4, kind='avg'))
print(cal_6(1,2,3,4, kind='sum'))
```

```
2.0
2.5
10
```

函数使用进阶-可变长度的参数-举例

- 举例，在前面程序基础上再修改，在 kind 中增加一个名为 min 的 key，如果 min 的值为 double，则最小值计算两次，来平衡结果

```
# kind 中 增加 min key 以及相关功能
def cal_7(*numbers, **kind):

    numbers = list(numbers)

    if 'kind' in kind:
        kind_value = kind.get('kind')

    if 'max' in kind:
        if kind.get('max') == 'ignore':
            numbers.remove(max(numbers))

    if 'min' in kind:
        if kind.get('min') == 'double':
            numbers.append(min(numbers))

    n = 0
    for i in numbers:
        n = n + i

    if kind_value == 'avg':
        return n / len(numbers)
    if kind_value == 'sum':
        return n

print(cal_7(1,2,3,4, kind='avg', max='ignore', min='double'))
print(cal_7(1,2,3,4, kind='avg'))
print(cal_7(1,2,3,4, kind='sum'))
```

```
1.75
2.5
10
```

函数参数小结

- 位置参数
- 默认参数
- 元组参数，一个星号
- 字典参数，两个星号，需要传递 key 和 value

匿名函数

- Python 允许用 lambda 关键字创造匿名函数
- lambda 匿名函数不需要常规函数的 def 和 return 关键字
- 因为匿名函数代码较短，因此适用于一些简单处理运算的场景
- 下面这样的写法是等价的

```
def add(x, y):
    return x + y

lambda x, y: x + y
```

举例

- 计算一个正整数的因数
- 写一个寻找列表中最大数的函数（不用列表排序方法）