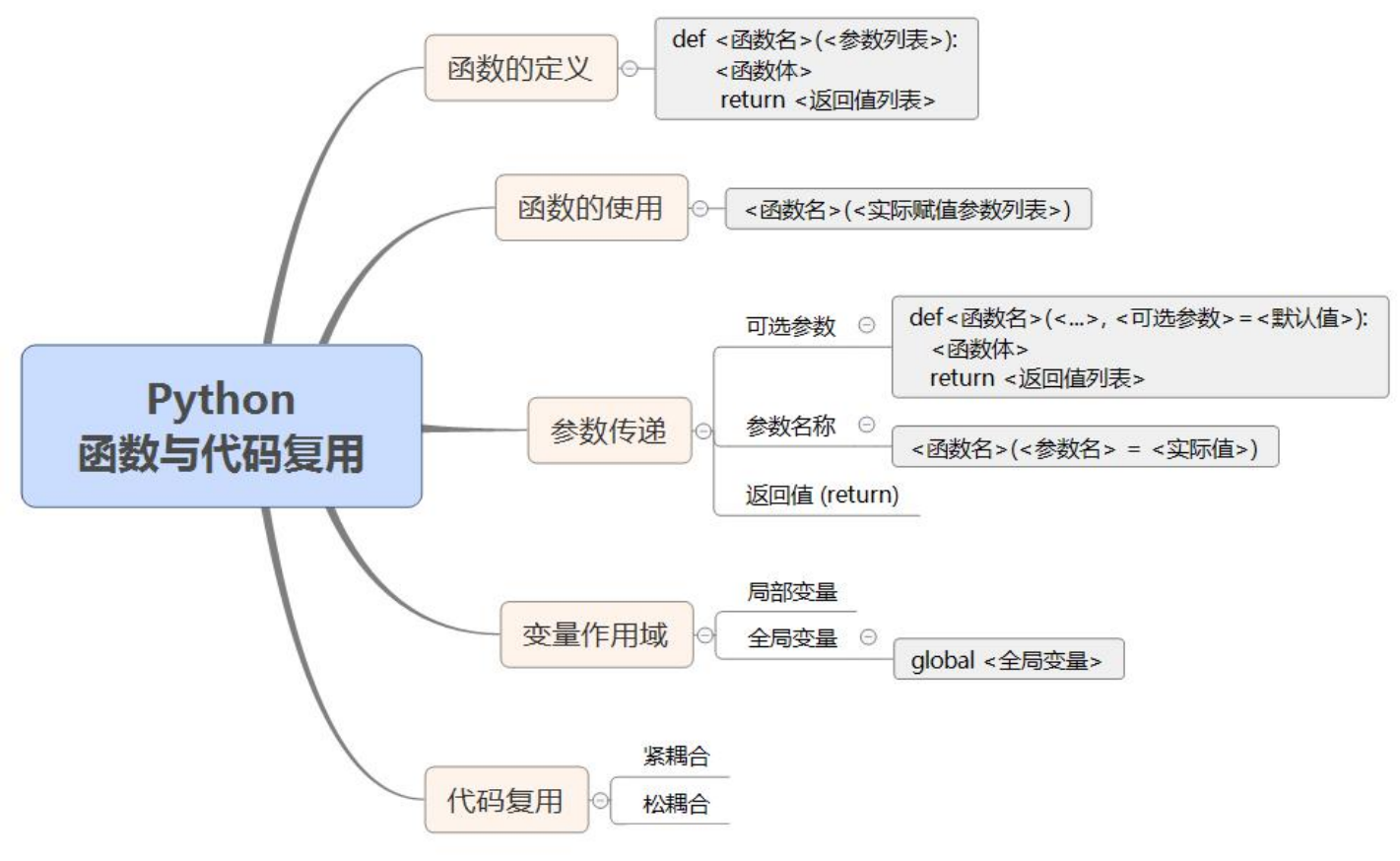
全国计算机等级考试二级教程
Python语言程序设计
(2018年版)

【第5章】
函数和代码复用

考纲考点

- 函数的定义和使用
- 函数的参数传递: 可选参数传递、参数名称传递、函数的返回值
- 变量的作用域: 局部变量和全局变量

知识导图


Python
函数与代码复用
函数的定义
def <函数名>(<参数列表>):
<函数体>
return <返回值列表>
函数的使用
<函数名>(<实际赋值参数列表>)
参数传递
可选参数
def<函数名>(<...>, <可选参数>=<默认值>):
<函数体>
return <返回值列表>
参数名称
<函数名>(<参数名> = <实际值>)
返回值 (return)
变量作用域
局部变量
全局变量
global <全局变量>
代码复用
紧耦合
松耦合

函数的基本使用

函数的定义

- 函数是一段具有特定功能的、可重用的语句组，通过函数名来表示和调用。经过定义，一组语句等价于一个函数，在需要使用这组语句的地方，直接调用函数名称即可。
- 因此，函数的使用包括两部分：**函数的定义**和**函数的使用**。
- 函数是一种功能抽象。

函数的定义

- Python定义一个函数使用def保留字，语法形式如下：

```
def <函数名>(<参数列表>):
    <函数体>
    return <返回值列表>
```

函数的定义

- 函数名可以是任何有效的Python标识符
- 参数列表是调用该函数时传递给它的值，可以有零个、一个或多个，当传递多个参数时各参数由逗号分隔，当没有参数时也要保留圆括号。
- 函数体是函数每次被调用时执行的代码，由一行或多行语句组成。

函数的定义

```
1  # 定义一个对整数n求阶乘的函数
2  def fact(n):
3      s = 1
4      for i in range(1, n+1):
5          s *= i
6      return s
```

- 如果需要返回值，使用保留字return和返回值列表。函数可以没有return语句，函数体结束后会将控制权返回给调用者。

函数的使用

- 定义后的函数不能直接运行，需要经过“调用”才能运行。调用函数的基本方法如下：

<函数名>(<实际赋值参数列表>)

```
1  # 定义一个对整数n求阶乘的函数
2  def fact(n):
3      s = 1
4      for i in range(1, n+1):
5          s *= i
6      return s
7  # 调用整数阶乘的函数
8  print(fact(100))
```

函数的使用

- 具体来说，函数的使用一共分为四个步骤：

1. 函数定义
2. 函数调用
3. 函数执行
4. 函数返回

函数的使用

1. 函数定义

使用def保留字将一段代码定义为函数，需要确定函数的名字、参数的名字、参数的个数，使用参数名称作为形式参数（占位符）编写函数内部的功能代码。

函数的使用

2. 函数调用

通过函数名调用函数功能，对函数的各个参数赋予实际值，实际值可以是实际数据，也可以是在调用函数前已经定义过的变量。

函数的使用

3.函数执行

函数被调用后，使用实际参数（赋予形式参数的实际值）参与函数内部代码的运行，如果有结果则进行输出。

函数的使用

4.函数返回

函数执行结束后，根据return保留字的指示决定是否返回结果，如果返回结果，则结果将被放置到函数被调用的位置，函数使用完毕，程序继续运行。

函数的参数传递

可选参数传递

- 函数的参数在定义时可以指定默认值，当函数被调用时，如果没有传入对应的参数值，则使用函数定义时的默认值替代，函数定义时的语法形式如下：

```
def <函数名>(<非可选参数列表>, <可选参数> = <默认值>):
    <函数体>
    return <返回值列表>
```

可选参数传递

- 需要注意，可选参数一般都放置在非可选参数的后面，即定义函数时，先给出所有非可选参数，然后再分别列出每个可选参数及对应的默认值。

```
>>>def multiply(x, y = 10):
        print(x*y)
>>>multiply(99)
990
>>>multiply(99, 2)
198
```

参数名称传递

- Python语言同时支持函数按照参数名称方式传递参数，语法形式如下：

<函数名>(<参数名>=<实际值>)

```
>>>def multiply(x, y = 10):
        print(x*y)
>>>multiply(x = 99)
990
>>>multiply(y = 2, x = 99)
198
```

函数的返回值

- return语句用来结束函数并将程序返回到函数被调用的位置继续执行。
- return语句可以出现在函数中的任何部分，同时可以将0个、1个或多个函数运算的结果返回给函数被调用处的变量。

```
>>>def multiply(x, y = 10):
        return x*y
>>>s = multiply(99, 2)
>>>print(s)
198
```

函数的返回值

- 函数可以没有return，此时函数并不返回值。当函数使用return返回多个值，可以使用一个变量或多个变量保存结果。

```
>>>def multiply(x, y = 10):
        return x*y, x+y
>>>s = multiply(99, 2)
>>>print(s)
(198, 101)
>>>a,b = multiply(99, 2)
>>>print(a)
198
>>>print(b)
101
```

变量的作用域

函数的返回值

- 根据程序中变量所在的位置和作用范围，变量分为**局部变量**和**全局变量**。
- 局部变量仅在函数内部，且作用域也在函数内部，全局变量的作用域跨越多个函数。

局部变量

- 局部变量指在函数内部使用的变量，仅在函数内部有效，当函数退出时变量将不再存在。

```
>>>def multiply(x, y = 10):
        z = x*y     # z是函数内部的局部变量
        return z
>>>s = multiply(99, 2)
>>>print(s)
198
>>>print(z)
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    print(z)
NameError: name 'z' is not defined
```

- 变量z是函数multiple()内部使用的变量，当函数调用后，变量z将不存在。

全局变量

- 全局变量指在函数之外定义的变量，在程序执行全过程有效。全部变量在函数内部使用时，需要提前使用保留字global声明，语法形式如下：

global <全局变量>

全局变量

```
>>>n = 2     #n是全局变量
>>>def multiply(x, y = 10):
        global n
        return x*y*n      # 使用全局变量n
>>>s = multiply(99, 2)
>>>print(s)
396
```

- 上例中，变量n是全局变量，在函数multiply()中使用时需要在函数内部使用global声明，定义后即可使用。

全局变量

- 如果未使用保留字global声明，即使名称相同，也不是全局变量。

```
>>>n = 2     #n是全局变量
>>>def multiply(x, y = 10):
        n = x*y
        return n      # 此处的n不是全局变量
>>>s = multiply(99, 2)
>>>print(s)
198
>>>print(n)    #不改变外部全局变量的值
2
```

代码复用

代码复用

- 函数是程序的一种基本抽象方式，它将一系列代码组织起来通过命名供其他程序使用。
- 函数封装的直接好处是**代码复用**，任何其他代码只要输入参数即可调用函数，从而避免相同功能代码在被调用处重复编写。代码复用产生了另一个好处，当更新函数功能时，所有被调用处的功能都被更新。

代码复用

- 模块化设计指通过函数的封装功能将程序划分成主程序、子程序和子程序间关系的表达。模块化设计是使用函数设计程序的思考方法，以功能块为基本单位，一般有两个基本要求：
  - ✓**紧耦合**：尽可能合理划分功能块，功能块内部耦合紧密；
  - ✓**松耦合**：模块间关系尽可能简单，功能块之间耦合度低。

代码复用

- 耦合性指程序结构中各模块之间相互关联的程度，它取决于各模块间接口的复杂程度和调用方式。
- 紧耦合指模块或系统间关系紧密，存在较多或复杂的相互调用。紧耦合的缺点在于更新一个模块可能导致其它模块变化，复用较困难。
- 松耦合一般基于消息或协议实现，系统间交互简单。

代码复用

- 使用函数只是模块化设计的必要非充分条件，根据计算需求合理划分函数十分重要。一般来说，完成特定功能或被经常复用的一组语句应该采用函数来封装，并尽可能**减少函数间参数和返回值的数量**。

实例解析：软文的诗词风

软文的诗词风

- 软文的诗词风将原有文章根据标点符号重新切分成短句并居中排版，对小屏幕阅读十分有利。使用程序将普通文章变成软文的诗词风十分有趣

软文的诗词风

```
txt = '''
人生得意须尽欢，莫使金樽空对月。
天生我材必有用，千金散尽还复来。
'''
linewidth = 30   # 预定的输出宽度

def lineSplit(line):
    plist = [',', '!', '?', ', ', '。', '! ', '? ']
    for p in plist:
        line = line.replace(p, '\n')
    return line.split('\n')

def linePrint(line):
    global linewidth
    print(line.center(linewidth, chr(12288)))

newlines = lineSplit(txt)
for newline in newlines:
    linePrint(newline)
```

软文的诗词风

- 原始文本使用变量txt保存，程序运行效果如下。

```
>>>

人生得意须尽欢
莫使金樽空对月

天生我材必有用
千金散尽还复来
```

软文的诗词风

- 原始文本使用变量txt保存，程序运行效果如下。

```
txt = '''
三国演义 上卷
罗贯中

滚滚长江东逝水，浪花淘尽英雄。是非成败转头空。青山依旧在，几度夕阳红。
白发渔樵江渚上，惯看秋月春风。一壶浊酒喜相逢。古今多少事，都付笑谈中。
--调寄《临江仙》
第一回　宴桃园豪杰三结义　斩黄巾英雄首立功
话说天下大势，分久必合，合久必分。周末七国分争，并入于秦。及秦灭之后，
楚、汉分争，又并入于汉。汉朝自高祖斩白蛇而起义，一统天下，后来光武中兴，
传至献帝，遂分为三国。
'''
```

软文的诗词风

>>>

三国演义 上卷

罗贯中

滚滚长江东逝水
浪花淘尽英雄
是非成败转头空
青山依旧在
几度夕阳红

白发渔樵江渚上
惯看秋月春风
一壶浊酒喜相逢
古今多少事
都付笑谈中

--调寄《临江仙》

第一回　宴桃园豪杰三结义　斩黄巾英雄首立功

话说天下大势
分久必合
合久必分
周末七国分争
并入于秦
及秦灭之后
楚、汉分争
又并入于汉
汉朝自高祖斩白蛇而起义
一统天下
后来光武中兴
传至献帝
遂分为三国

软文的诗词风

- 当每句长度超过变量linewdith后，显示效果并不好。这需要修改函数linePrint()，当一个短句行数超过限制时，分行居中显示。

```
1  def linePrint(line):
2      global linewidth
3      while len(line) > linewidth:
4          print(line[0:linewidth])
5          line = line[linewidth:]
6    print(line.center(linewidth, chr(12288)))
```

本章小结

本章讲解了函数的基本使用方法，包括函数的定义和调用。进一步具体讲解了函数的参数传递方法和变量的作用域，包括可选参数传递、参数名称传递和函数的返回值，初步介绍了函数的作用和代码复用。通过软文的诗词风实例帮助读者理解函数的定义和使用。

古代的诗词歌赋是填词怡情，当代的诗词歌赋则是风格怡情，快来造个自己风格吧？！