

实验六 Python函数

班级： 21计科04

学号： B20210302412

姓名： 刘巍

Gitee地址： (<https://github.com/smile-lv/smile>)

CodeWars地址： <https://www.codewars.com/users/Strugglingtt>

实验目的

1. 学习Python函数的基本用法
2. 学习lambda函数和高阶函数的使用
3. 掌握函数式编程的概念和实践

实验环境

1. Git
2. Python 3.10
3. VSCode
4. VSCode插件

实验内容和步骤

第一部分

Python函数

完成教材《Python编程从入门到实践》下列章节的练习：

- 第8章 函数
-

第二部分

在[Codewars网站](#)注册账号，完成下列Kata挑战：

第一题：编码聚会1

难度： 7kyu

你将得到一个字典数组，代表关于首次报名参加你所组织的编码聚会的开发者的数据。 你的任务是返回来自欧洲的JavaScript开发者的数量。 例如，给定以下列表：

```
lst1 = [
  { 'firstName': 'Noah', 'lastName': 'M.', 'country': 'Switzerland', 'continent':
'Europe', 'age': 19, 'language': 'JavaScript' },
  { 'firstName': 'Maia', 'lastName': 'S.', 'country': 'Tahiti', 'continent':
'Oceania', 'age': 28, 'language': 'JavaScript' },
  { 'firstName': 'Shufen', 'lastName': 'L.', 'country': 'Taiwan', 'continent':
'Asia', 'age': 35, 'language': 'HTML' },
  { 'firstName': 'Sumayah', 'lastName': 'M.', 'country': 'Tajikistan',
'continent': 'Asia', 'age': 30, 'language': 'CSS' }
]
```

你的函数应该返回数字1。如果，没有来自欧洲的JavaScript开发人员，那么你的函数应该返回0。

注意：字符串的格式将总是"Europe"和"JavaScript"。所有的数据将始终是有有效的和统一的，如上面的例子。

这个卡塔是Coding Meetup系列的一部分，其中包括一些简短易行的卡塔，这些卡塔是为了让人们掌握高阶函数的使用。在Python中，这些方法包括：`filter`, `map`, `reduce`。当然也可以采用其他方法来解决这些卡塔。

[代码提交地址](#)

第二题：使用函数进行计算

难度：5kyu

这次我们想用函数来写计算，并得到结果。让我们看一下一些例子：

```
seven(times(five())) # must return 35
four(plus(nine())) # must return 13
eight(minus(three())) # must return 5
six(divided_by(two())) # must return 3
```

要求：

- 从0 ("零") 到9 ("九") 的每个数字都必须有一个函数。
- 必须有一个函数用于以下数学运算：加、减、乘、除。
- 每个计算都由一个操作和两个数字组成。
- 最外面的函数代表左边的操作数，最里面的函数代表右边的操作数。
- 除法应该是整数除法。

例如，下面的计算应该返回2，而不是2.666666....。

```
eight(divided_by(three()))
```

代码提交地址：<https://www.codewars.com/kata/525f3eda17c7cd9f9e000b39>

第三题：缩短数值的过滤器(Number Shortening Filter)

难度：6kyu

在这个kata中，我们将创建一个函数，它返回另一个缩短长数字的函数。给定一个初始值数组替换给定基数的X次方。如果返回函数的输入不是数字字符串，则应将输入本身作为字符串返回。

例子：

```
filter1 = shorten_number(['','k','m'],1000)
filter1('234324') == '234k'
filter1('98234324') == '98m'
filter1([1,2,3]) == '[1,2,3]'
filter2 = shorten_number(['B','KB','MB','GB'],1024)
filter2('32') == '32B'
filter2('2100') == '2KB';
filter2('pippi') == 'pippi'
```

代码提交地址：<https://www.codewars.com/kata/56b4af8ac6167012ec00006f>

第四题：编码聚会7

难度：6kyu

您将获得一个对象序列，表示已注册参加您组织的下一个编程聚会的开发人员的数据。

您的任务是返回一个序列，其中包括最年长的开发人员。如果有多个开发人员年龄相同，则将他们按照在原始输入数组中出现的顺序列出。

例如，给定以下输入数组：

```
list1 = [
  { 'firstName': 'Gabriel', 'lastName': 'X.', 'country': 'Monaco', 'continent':
'Europe', 'age': 49, 'language': 'PHP' },
  { 'firstName': 'Odval', 'lastName': 'F.', 'country': 'Mongolia', 'continent':
'Asia', 'age': 38, 'language': 'Python' },
  { 'firstName': 'Emilija', 'lastName': 'S.', 'country': 'Lithuania', 'continent':
'Europe', 'age': 19, 'language': 'Python' },
  { 'firstName': 'Sou', 'lastName': 'B.', 'country': 'Japan', 'continent': 'Asia',
'age': 49, 'language': 'PHP' },
]
```

您的程序应该返回如下结果：

```
[
  { 'firstName': 'Gabriel', 'lastName': 'X.', 'country': 'Monaco', 'continent':
'Europe', 'age': 49, 'language': 'PHP' },
```

```
{ 'firstName': 'Sou', 'lastName': 'B.', 'country': 'Japan', 'continent': 'Asia',  
  'age': 49, 'language': 'PHP' },  
]
```

注意：

- 输入的列表永远都包含像示例中一样有效的正确格式的数据，而且永远不会为空。

代码提交地址：<https://www.codewars.com/kata/582887f7d04efdaae3000090>

第五题：Currying versus partial application

难度：4kyu

[Currying versus partial application](#)是将一个函数转换为具有更小arity(参数更少)的另一个函数的两种方法。虽然它们经常被混淆，但它们的工作方式是不同的。目标是学会区分它们。

Currying

是一种将接受多个参数的函数转换为以每个参数都只接受一个参数的一系列函数链的技术。

Currying接受一个函数：

```
f: X × Y → R
```

并将其转换为一个函数：

```
f': X → (Y → R)
```

我们不再使用两个参数调用f，而是使用第一个参数调用f'。结果是一个函数，然后我们使用第二个参数调用该函数来产生结果。因此，如果非curried f被调用为：

```
f(3, 5)
```

那么curried f'被调用为：

```
f'(3)(5)
```

示例 给定以下函数：

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

我们可以以普通方式调用：

```
add(1, 2, 3) # => 6
```

但我们可以创建一个curried版本的add(a, b, c)函数：

```
curriedAdd = lambda a: (lambda b: (lambda c: add(a,b,c)))  
curriedAdd(1)(2)(3) # => 6
```

Partial application 是将一定数量的参数固定到函数中，从而产生另一个更小arity(参数更少)的函数的过程。

部分应用接受一个函数：

```
f: X × Y → R
```

和一个固定值x作为第一个参数，以产生一个新的函数

```
f': Y → R
```

f'与f执行的操作相同，但只需要填写第二个参数，这就是其arity比f的arity少一个的原因。可以说第一个参数绑定到x。

示例:

```
partialAdd = lambda a: (lambda *args: add(a,*args))  
partialAdd(1)(2, 3) # => 6
```

你的任务是实现一个名为curryPartial()的通用函数，可以进行currying或部分应用。

例如：

```
curriedAdd = curryPartial(add)  
curriedAdd(1)(2)(3) # => 6  
  
partialAdd = curryPartial(add, 1)  
partialAdd(2, 3) # => 6
```

我们希望函数保持灵活性。

所有下面这些例子都应该产生相同的结果：

```
curryPartial(add)(1)(2)(3) # =>6
curryPartial(add, 1)(2)(3) # =>6
curryPartial(add, 1)(2, 3) # =>6
curryPartial(add, 1, 2)(3) # =>6
curryPartial(add, 1, 2, 3) # =>6
curryPartial(add)(1, 2, 3) # =>6
curryPartial(add)(1, 2)(3) # =>6
curryPartial(add)()(1, 2, 3) # =>6
curryPartial(add)()(1)()(2)(3) # =>6

curryPartial(add)()(1)()(2)(3, 4, 5, 6) # =>6
curryPartial(add, 1)(2, 3, 4, 5) # =>6

curryPartial(curryPartial(curryPartial(add, 1), 2), 3) # =>6
curryPartial(curryPartial(add, 1, 2), 3) # =>6
curryPartial(curryPartial(add, 1), 2, 3) # =>6
curryPartial(curryPartial(add, 1), 2)(3) # =>6
curryPartial(curryPartial(add, 1)(2), 3) # =>6
curryPartial(curryPartial(curryPartial(add, 1)), 2, 3) # =>6
```

代码提交地址：<https://www.codewars.com/kata/53cf7e37e9876c35a60002c9>


第三部分

使用Mermaid绘制程序流程图

安装VSCode插件：

- Markdown Preview Mermaid Support
- Mermaid Markdown Syntax Highlighting

使用Markdown语法绘制你的程序绘制程序流程图（至少一个），Markdown代码如下：

 程序流程图

显示效果如下：

```
flowchart LR
    A[Start] --> B{Is it?}
    B -->|Yes| C[OK]
    C --> D[Rethink]
    D --> B
    B ---->|No| E[End]
```

查看Mermaid流程图语法-->[点击这里](#)

使用Markdown编辑器（例如VScode）编写本次实验的实验报告，包括[实验过程与结果](#)、[实验考查](#)和[实验总结](#)，并将其导出为 **PDF格式** 来提交。

实验过程与结果

请将实验过程与结果放在这里，包括：

- [第一部分 Python函数](#)
- [第二部分 Codewars Kata挑战](#)

第一题：编码聚会1

```
def count_developers(lst):  
    count = 0  
  
    for developer in lst:  
        if developer['continent'] == 'Europe' and developer['language'] ==  
            'JavaScript':  
            count += 1  
  
    return count
```

第二题：使用函数进行计算

```
def zero(operation=None):  
    if operation is None:  
        return 0  
    else:  
        return operation(0)  
  
def one(operation=None):  
    if operation is None:  
        return 1  
    else:  
        return operation(1)  
  
def two(operation=None):  
    if operation is None:  
        return 2  
    else:  
        return operation(2)  
  
def three(operation=None):  
    if operation is None:  
        return 3  
    else:  
        return operation(3)  
  
def four(operation=None):
```

```
    if operation is None:
        return 4
    else:
        return operation(4)

def five(operation=None):
    if operation is None:
        return 5
    else:
        return operation(5)

def six(operation=None):
    if operation is None:
        return 6
    else:
        return operation(6)

def seven(operation=None):
    if operation is None:
        return 7
    else:
        return operation(7)

def eight(operation=None):
    if operation is None:
        return 8
    else:
        return operation(8)

def nine(operation=None):
    if operation is None:
        return 9
    else:
        return operation(9)

def plus(x):
    def operation(y):
        return x + y
    return operation

def minus(x):
    def operation(y):
        if(x-y>0):
            return -(x - y)
        else:
            return abs(x-y)
    return operation

def times(x):
    def operation(y):
        return x * y
    return operation

def divided_by(x):
```



```
def operation(y):
    return y // x
return operation
```

第三题：缩短数值的过滤器(Number Shortening Filter)

```
def shorten_number(suff, base):
    def wrapper(n):
        try:
            n = int(n)
            for i in range(len(suff)):
                if 0 < n // (base ** i) < base:
                    return f'{n // base ** i}{suff[i]}'
            return f'{n // base ** i}{suff[i]}'
        except (ValueError, TypeError):
            return f'{n}'
    return wrapper
```

第四题：编码聚会7

```
def find_senior(lst):
    ans = []
    age = 0
    for x in lst:
        if x['age'] >= age:
            if x['age'] > age:
                age = x['age']
                ans = []
            ans.append(x)
    return ans
```

第五题：Currying versus partial application

- [第三部分 使用Mermaid绘制程序流程图](#)

```
flowchart LR
    A[lis列表] ---> B{language=Javascript?} -->|False| A
    B -->|True| C{continent=Europe?} -->|False| A
    C -->|True| D[sum+1]
```

1. 什么是函数式编程范式？

面向对象编程的思维方式：把现实世界中的事物抽象成程序世界中的类和对象，通过封装、继承和 多态来演示事物事件的联系 函数式编程的思维方式：把现实世界的事物和事物之间的联系抽象到程序世界

(对运算过程进行抽象) 程序的本质: 根据输入通过某种运算获得相应的输出, 程序开发过程中会涉及很多有输入和输出的函数 $x \rightarrow f(\text{联系、映射}) \rightarrow y$, $y=f(x)$ 函数式编程中的函数指的是不是程序中的函数(方法), 而是数学中的函数即映射关系, 例如: $y = \sin(x)$, x 和 y 的关系 相同的输入始终要得到相同的输出(纯函数) * 函数式编程用来描述数据(函数)之间的映射

2. 什么是lambda函数? 请举例说明。lambda函数有如下特性:

lambda函数是匿名的: 所谓匿名函数, 通俗地说就是没有名字的函数。lambda函数没有名字。

lambda函数有输入和输出: 输入是传入到参数列表argument_list的值, 输出是根据表达式expression计算得到的值。

lambda函数一般功能简单: 单行expression决定了lambda函数不可能完成复杂的逻辑, 只能完成非常简单的功能。由于其实现的功能一目了然, 甚至不需要专门的名字来说明。

下面是一些lambda函数示例:

lambda x, y: xy; 函数输入是x和y, 输出是它们的积xy

lambda: None; 函数没有输入参数, 输出是None

lambda *args: sum(args); 输入是任意个数的参数, 输出是它们的和(隐性要求是输入参数必须能够进行加法运算)

lambda **kwargs: 1; 输入是任意键值对参数, 输出是1

3. 什么是高阶函数? 常用的高阶函数有哪些? 这些高阶函数如何工作? 使用简单的代码示例说明。1、若a函数, 接收的参数是一个函数, 那么a就可以称之为高阶函数

2、若a函数, 调用的返回值依然是一个函数, 那么a就可以称之为高阶函数。常见的高阶函数有: Promise、setTimeout、arr.map()、filter、reduce等等

实验总结

总结一下这次实验你学习和使用到的知识, 例如: 编程工具的使用、数据结构、程序语言的语法、算法、编程技巧、编程思想。