高阶函数

廖雪峰
资深软件开发工程师，业余马拉松选手。

高阶函数英文叫Higher-order function。什么是高阶函数？我们以实际代码为例子，一步一步深入概念。

变量可以指向函数

以Python内置的求绝对值的函数 `abs()` 为例，调用该函数用以下代码：

```
>>> abs(-10)
10
```

但是，如果只写 `abs` 呢？

```
>>> abs
<built-in function abs>
```

可见， `abs(-10)` 是函数调用，而 `abs` 是函数本身。

要获得函数调用结果，我们可以把结果赋值给变量：

```
>>> x = abs(-10)
>>> x
10
```

但是，如果把函数本身赋值给变量呢？

```
>>> f = abs
>>> f
<built-in function abs>
```

结论：函数本身也可以赋值给变量，即：变量可以指向函数。

如果一个变量指向了一个函数，那么，可否通过该变量来调用这个函数？用代码验证一下：

```
>>> f = abs
>>> f(-10)
10
```

成功！说明变量 `f` 现在已经指向了 `abs` 函数本身。直接调用 `abs()` 函数和调用变量 `f()` 完全相同。

函数名也是变量

那么函数名是什么呢？函数名其实就是指向函数的变量！对于 `abs()` 这个函数，完全可以把函数名 `abs` 看成变量，它指向一个可以计算绝对值的函数！

如果把 `abs` 指向其他对象，会有什么情况发生？

```
>>> abs = 10
>>> abs(-10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
```

把 `abs` 指向 `10` 后，就无法通过 `abs(-10)` 调用该函数了！因为 `abs` 这个变量已经不指向求绝对值函数而是指向一个整数 `10` ！

当然实际代码绝对不能这么写，这里是为了说明函数名也是变量。要恢复 `abs` 函数，请重启Python交互环境。

注：由于 `abs` 函数实际上是定义在 `import builtins` 模块中的，所以要让修改 `abs` 变量的指向在其它模块也生效，要用 `import builtins; builtins.abs = 10` 。

传入函数

既然变量可以指向函数，函数的参数能接收变量，那么一个函数就可以接收另一个函数作为参数，这种函数就称之为高阶函数。

一个最简单的高阶函数：

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

当我们调用 `add(-5, 6, abs)` 时，参数 `x` ， `y` 和 `f` 分别接收 `-5` ， `6` 和 `abs` ，根据函数定义，我们可以推导计算过程为：

```
x = -5
y = 6
f = abs
f(x) + f(y) ==> abs(-5) + abs(6) ==> 11
return 11
```

用代码验证一下：

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

print(add(-5, 6, abs))
```

编写高阶函数，就是让函数的参数能够接收别的函数。

参考源码

do_f_add.py

小结

把函数作为参数传入，这样的函数称为高阶函数，函数式编程就是指这种高度抽象的编程范式。

« 函数式编程

map/reduce »

Comments

Comments loaded. To post a comment, please Sign In

Rory @ 2025/12/16 02:32:59

指针函数，函数指针，竟然不是同一个东西

Arcueid @ 2025/9/18 02:58:24

c/c++里的函数指针，函数对象

西瓜 @ 2025/9/11 09:01:57

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

print(add(-5,6, abs))
```

Cor Cordium @ 2025/9/9 01:09:16

```
def max(*args):
    return min(*args)
print(max(1,2,3,4,5))   # 1
```

陈彤 @ 2025/8/9 05:01:37

学到这，还没打几行代码，准备从头再来，看看学的到底是啥玩意，不懂的，实践起来。

Nawitar @ 2025/6/21 03:34:05

day 1 keep

😜 @ 2025/6/10 02:38:08

day9

路 @ 2025/5/14 21:32:12

8.1. 高阶函数(Higher-order function) done

日向夏橘 @ 2025/5/5 02:13:53

打卡

寥寥星辰 @ 2025/4/16 08:18:24

day8

阿巴阿巴阿巴 @ 2025/4/11 03:23:14

打卡 day 4

mortal @ 2025/2/2 09:52:05

from functools import reduce def str2float(s): def f1(q): digital= {'0':0,'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7':7,'8':8,'9':9} return digital[q]

```
dot_index=s.find('.')
if dot_index==-1:
        return reduce(lambda x,y:x*10+y,map(f1,s))
else :
        integer_par=s[:dot_index]
        decimal_part=s[dot_index+1:]
        integer_value=reduce(lambda x, y: x*10+y,map(f1,integer_par))
        decmal_value=reduce(lambda x, y: x * 10 + y, map(f1, decimal_part))
        return integer_value+decmal_value/10**len(decimal_part)
```

print('str2float('123.456') =', str2float('123.456')) if abs(str2float('123.456') - 123.456) < 0.00001: print('测试成功!') else: print('测试失败!')

Earth奇人 @ 2020/9/4 23:52:45

```
# -*- coding: utf-8 -*-
# 利用map和reduce编写一个str2float函数，把字符串'123.456'转换成浮点数123.456:
```

```
from functools import reduce

i = 0

def str2float(s):
    def add(s, s1):
        global i
        if isinstance(s, int) and isinstance(s1, int) and (i == 0):
            return s * 10 + s1
        elif isinstance(s, int) and i ==0:
            i = 1
            return s
        elif isinstance(s1, int) or isinstance(s1, float) :
```

Read More ▼

致良知事上练 @ 2021/4/22 09:50:50

eval函数也可以,https://blog.csdn.net/qq_29883591/article/details/53443062

```
print(eval("123.456"))
```

naivety @ 2025/1/5 04:45:54

个人觉得，能不用全局变量，最好不用。上述程序中，i只在add函数中起作用，并不需要在其他函数中传递，建议去掉全局变量，改为默认调用参数i:

def add(s, s1, i=0):

可以达到同样的效果。

楠梓 @ 2024/12/20 04:05:46

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

print(add(-9,73,abs))
82
```

jx @ 2024/11/7 02:30:29

打卡第三天

Fridemn @ 2024/9/14 04:34:12

```
输出=print
输出('结果')
```

杨炎泽 @ 2017/10/19 04:35:28

max, min = min, max

print(max(1, 2, 3, 4, 5))

聚来宝梦寒 @ 2017/10/20 03:21:35

李时珍的皮

Delicious-Zhao @ 2017/11/7 03:25:02

max=min print(max(1,2,3,4,5)) 结果是5 print（min（1,2,3,4,5）） 结果也是5

zz幸福在远方 @ 2017/11/8 00:30:08

max, min = min, max print(max(1, 2, 3, 4, 5)) -> 结果是1 print(min(1, 2, 3, 4, 5)) -> 结果是5

用户0322244315 @ 2017/11/10 08:59:12

你们真会玩。。。

往事随風_辜負了時光 @ 2017/11/14 02:59:57

-
  - 实验证明 max,min = min ,max 这条语句 并不等价于 顺序执行 max = min , min =max 而是同时执行… 有意思~~~

JeromeYLuck @ 2017/12/1 12:25:08

因为 = 代表着 被赋予

用户5260643445 @ 2017/12/17 21:10:13

@往事隨風_辜負了時光 max,min=min,max 对于这个不等价于max=min,min=max，如果你有印象的话，其实在前文"高级特性-生成器"那篇文章中,廖老师有专门拿出来提到过的， 就是使用非递归循环方式编写的斐波那契数列中有一步就与此类似，即： a,b=b,a+b 这里等号右边实际上是一个tuple，因此等号右边的a在定义的时候就已经确定了，并不会因为将b赋值给了左边的a而改变

晴天菜鸟 @ 2017/12/21 03:51:15

max,min=min,max 对于这个不等价于max=min,min=max

对应到其他语言的话, 实际上相当于

```
tmp = max
max = min
min = tmp
```

完成的工作吧

beer_35010 @ 2018/1/28 21:46:06

加上这个就可以完成真正的值传递了；高手~

三木两木加三水 @ 2018/2/2 02:12:02

哈哈，前边的斐波拉有涉及到这点

东林读书人 @ 2018/2/8 03:11:57

就是把变量（函数名）max指向求最小函数，变量（函数名）min指向求最大函数，真皮

kkkkkkkaaaaaa @ 2018/2/9 01:36:49

举个实例吧 def p(): print(2) def o(): print(1) p function p at 0x02C70390 o function o at 0x02C70420 o,p=p,o p function o at 0x02C70420 o function p at 0x02C70390

Srxh1372 @ 2018/3/21 23:43:58

乍一看很神奇 其实非常好理解 厉害厉害

梦想家高先森 @ 2018/4/4 01:12:41

理解的时候记得中间是有个tuple做桥梁就好了！

```
(a, b) = t = (b, a + b)
```

动力橙CHT @ 2018/5/1 02:31:01

max,min = min,max 本质上等价与 t = (min,max) max = t[0] min = t[1] 所以执行该语句后，max变成求最小值，min变成求最大值 故输出为1

CGcn_QQending @ 2018/5/12 06:29:11

那位说=是赋予的，是赋予呢，还是把一个值赋予了此变量，还是将此变量指向了一个值。

用户5996354829 @ 2018/6/25 09:07:35

min,max = max,min

运行过程其实是这样的：

t = (max,min)

min = t (0)

max = t (1)

small_scorpion @ 2018/7/22 13:13:14

是一个tuple，已经被确定

慧眼的安娘娘 @ 2018/11/5 04:38:07

城会玩……

无风之名 @ 2018/11/16 02:36:07

那个delicious_zhao同学：你在说什么啊？？？

大侦探福尔摩一 @ 2024/4/21 05:05:08

函数名就是一个变量，指向一个函数，当然可以赋值给一个新变量，新变量也指向相同的函数。

因为函数的参数可以接收变量，函数名也是一个变量，指向函数，所以函数可以当参数传入进函数，并且可以返回函数的这种操作函数的函数，就叫高阶函数。

以下是个人见解：传入参数的函数如果是非纯函数的话，在call函数的时候，参数里的非纯函数就已经产生副作用了

森 @ 2024/1/29 03:56:41

学习了这个课程几天，定一个小目标，半个月看完它

Ashen One @ 2023/5/27 09:46:56

def add(x,y,z):

return z(x)*z(y)

print(add(-55,30,abs))