A L G O R I T H M

P R A C T I C E

深度学习算法与实践

Steven Tang

渡一教育
Duyi Education

函数和代码的复用

函数的基本使用

内置函数

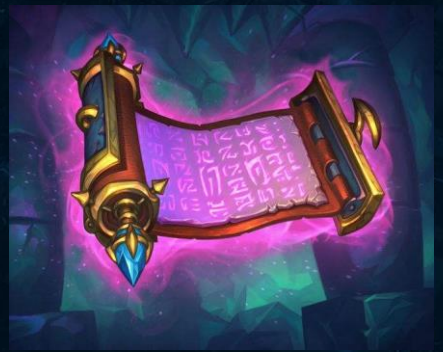
PYTHON 中函数的应用非常广泛，前面章节中我们已经接触过多个函数，比如 INPUT()、PRINT()、RANGE()、LEN() 函数等等，这些都是 PYTHON 的内置函数，可以直接使用。

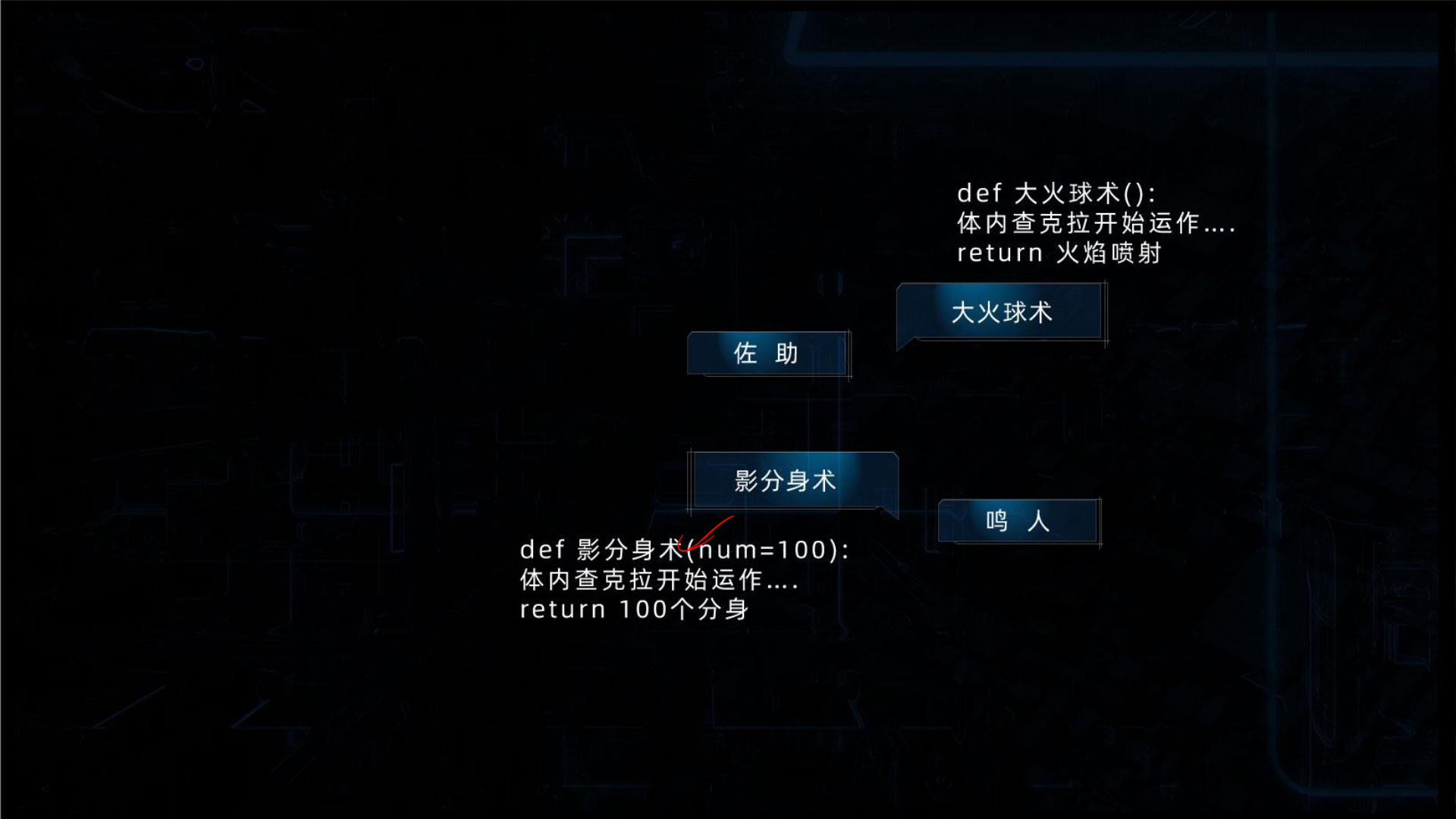
函数的定义

将一段有规律的、可重复使用的代码定义成函数，从而达到一次编写、多次调用的目的。

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>
```




def 大火球术():
体内查克拉开始运作....
return 火焰喷射
大火球术
佐 助
影分身术
鸣 人
def 影分身术(num=100):
体内查克拉开始运作....
return 100个分身

函数的定义

```
In [28]: def happy(name):
    ...:     print(" 新年快乐! {}总!".format(name))
    ...:
    ...:
    ...: happy("王")
    ...: happy("周")
    ...: happy("尊敬的李")
 新年快乐! 王总!
 新年快乐! 周总!
 新年快乐! 尊敬的李总!
```

函数调用的过程

程序调用一个函数需要执行以下四个步骤：

（1）调用程序在调用处暂停执行；

（2）在调用时将实参复制给函数的形参；

（3）执行函数体语句；

（4）函数调用结束给出返回值，程序回到调用前的暂停处继续执行。

函数形参和实参

```
In [8]: def fun(x,y): #形参
   ...:     print(x + y)
   ...:

In [9]: a,b=2,3#实参

In [10]: fun(a,b)
5
```

函数的参数传递

函数的参数

在定义函数时，有些参数可以存在默认值

```
In [11]: def repeat(str, times = 2):
    ...:         print(str*times)
    ...:

In [12]: repeat("Hello")
HelloHello

In [13]: repeat("Hello",4)
HelloHelloHelloHello

In [14]: repeat("Hello",times=4)
HelloHelloHelloHello
```

函数的参数

在定义函数时，有默认值的参数必须写在没有默认参数的后面。

```
In [15]: def repeat(str, times = 2,name):
    ...:         print(str*times)
  File "<ipython-input-15-5111bcbb9683>", line 1
    def repeat(str, times = 2,name):
                                   ^
SyntaxError: non-default argument follows default
argument
```

函数的参数

在函数定义时，可以设计可变数量参数，通过参数前增加星号（*）实现

```
In [23]: def func(x, *y):
    ...:         print("y:",y)
    ...:         print(max(y))
    ...:         for n in y:
    ...:             x += n
    ...:         return x
    ...:

In [24]: func(1,2,3,4)
y: (2, 3, 4)
4
Out[24]: 10
```

函数的参数

在调用函数时也能够使用*语法，在这种情况下，它与函数定义的意思相反，它会解包参数的集合，而不是创建参数的集合。

```
In [29]: def func(a,b,c):
    ...:     print(a+b+c)
    ...:

In [30]: args=(1,2,3)

In [31]: func(*args)
6

In [32]: func(args)
Traceback (most recent call last):

  File "<ipython-input-32-1d8957876bf5>", line 1, in
<module>
    func(args)

TypeError: func() missing 2 required positional
arguments: 'b' and 'c'
```

函数的参数

在python中，还可以将定义好的函数作为参数传给另外一个函数，这种用法在后期项目中十分常见。

```
In [47]: def fun1(x):
    ...:     return x*x
    ...:
    ...: def fun2(a,fun):
    ...:     return fun(a)
    ...:
    ...:
    ...: print(fun2(3,fun1))
9
```

参数的位置和名称传递

PYTHON提供了按照形参名称输入实参的方式，所以参数之间的顺序可以任意调整。

```
In [16]: def f(a,b,c):
    ...:     print(a,b,c)
    ...:
    ...: f(1,2,3)
1 2 3

In [17]: f(c=3,a=1,b=2)
1 2 3
```

变量的返回值

- 函数调用后默认返回NONE。

```
In [52]: def fun(a,b):
    ...:     c=a+b
    ...:

In [53]: fun(2,3)

In [54]: type(fun(2,3))
Out[54]: NoneType
```

变量的返回值

- 多个RETURN 情况。

```
In [80]: def func(l):
    ...:     if len(l)>2:
    ...:         return l[0:2]
    ...:     else:
    ...:         return l
    ...: print(func([1,2,3,4]))
    ...: print(func([1,2]))
[1, 2]
[1, 2]
```

变量的返回值

- RETURN语句用来退出函数并将程序返回到函数被调用的位置继续执行。
- RETURN语句同时可以将0个、1个或多个函数运算完的结果返回给函数被调用处的变量。

```
In [26]: import math
    ...:
    ...: def move(x, y, step, angle=0):
    ...:     nx = x + step * math.cos(angle)
    ...:     ny = y - step * math.sin(angle)
    ...:     return nx, ny
    ...:

In [27]: x, y = move(100, 100, 60, math.pi / 6)

In [28]: print(x, y)
151.96152422706632 70.0

In [29]: move(100, 100, 60, math.pi / 6)
Out[29]: (151.96152422706632, 70.0)
```

函数定义嵌套

函数内部可以再定义函数，这些函数只能在函数内部使用。

```
In [3]: def fun1(x,y):
   ...:
   ...:     def fun2(a):
   ...:         return a*a
   ...:     def fun3(b):
   ...:         return pow(b,0.5)
   ...:     return fun3(fun2(x)+fun2(y))
   ...:

In [4]: fun1(3,4)
Out[4]: 5.0

In [5]: fun2(3)
Traceback (most recent call last):

  File "<ipython-input-5-00452eef5016>", line 1, in <module>
    fun2(3)

NameError: name 'fun2' is not defined
```

全局变量和局部变量

全局变量指在函数之外定义的变量，一般没有缩进，在程序执行全过程有效。

•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存在。

全局变量和局部变量

```
In [33]: x = 1

In [34]: def func(a, b):
    ...:         c = a + b   #c是局部变量，a和b作为函数参数也是局部变量
    ...:         return c
    ...:

In [35]: print(c)
Traceback (most recent call last):

  File "<ipython-input-35-1dd5973cae19>", line 1, in <module>
    print(c)

NameError: name 'c' is not defined
```

这个例子说明，当函数执行完退出后，其内部变量将被释放。如果函数内部使用了全局变量呢？

全局变量和局部变量

```
In [39]: x = 1

In [40]: def func(a, b):
    ...:         x = a + b    #x还是局部变量，不是全局变量
    ...:         return x
    ...:

In [41]: func(2,3)
Out[41]: 5

In [42]: x
Out[42]: 1
```

函数FUNC()内部使用了变量N，并且将变量参数B赋值给变量N，为何全局变量N值没有改变？

全局变量和局部变量

如果希望让FUNC()函数将X当作全局变量，需要在变量X使用前用GLOBAL关键字声明该变量为全局变量，代码如下。

```
In [47]: x = 1

In [48]: def func(a, b):
    ...:         global x
    ...:         x = a + b    #这时候x是全局变量
    ...:         return x
    ...:

In [49]: func(3,4)
Out[49]: 7

In [50]: x
Out[50]: 7
```

全局变量和局部变量

如果此时的全局变量不是整数X，而是列表类型L，会怎么样呢？理解如下代码。

```
In [63]: l = []

In [64]: def func(x, y):
    ...:         l.append(x+y)
    ...:         return x*y
    ...:

In [65]: func(2,3)
Out[65]: 6

In [66]: l
Out[66]: [5]
```

嵌套函数中的全局变量和局部变量。

```
x=1
def fun1():
    x=2
    print("fun1的局部变量x:",x)
    def fun2():
        global x
        x=3
        print("全局变量x:",x)
    fun2()
    print("fun1最后的x值:",x)


print("最开始的x值:",x)
fun1()

print("最后的x值:",x)
```

```
最开始的x值: 1
fun1的局部变量x: 2
全局变量x: 3
fun1最后的x值: 2
最后的x值: 3
```

全局变量和局部变量。

Python函数对变量的作用遵守如下原则：

•单个变量（数字类型）无论是否与全局变量重名，仅在函数内部创建和使用，函数退出后变量被释放；

•单个变量在用global保留字声明后，作为全局变量；

•对于组合数据类型的全局变量，如果在函数内部没有被真实创建的同名变量，则函数内部可直接使用并修改全局变量的值；

•如果函数内部真实创建了组合数据类型变量，无论是否有同名全局变量，函数仅对局部变量进行操作。

lambda函数

- PYTHON语言特有的简洁函数模型。
- PYTHON学习中极为重要的一环！
- 一个合作的PYTHON工程师一定会善用LAMBDA函数！

lambda函数

```
In [30]: c=lambda x,y,z:x*y*z

In [31]: c
Out[31]: <function __main__.<lambda>(x, y, z)>

In [32]: type(c)
Out[32]: function

In [33]: c(2,3,4)
Out[33]: 24

In [34]: c(2,3)
Traceback (most recent call last):

  File "<ipython-input-34-d4885bb50ae6>", line 1, in <module>
    c(2,3)

TypeError: <lambda>() missing 1 required positional argument: 'z'
```

map()和lambda函数组合使用

map()会根据提供的函数对指定序列做映射。

```
In [48]: def sq(x):
    ...:     return x*x
    ...:

In [49]: map(sq,[y for y in range(5)])
Out[49]: <map at 0x1763f620898>

In [50]: list(map(sq,[y for y in range(5)]))
Out[50]: [0, 1, 4, 9, 16]

In [51]: map(lambda x: x*x,[y for y in range(5)])
Out[51]: <map at 0x1763f620c88>

In [52]: map1=map(lambda x: x*x,[y for y in range(5)])

In [53]: list(map1)
Out[53]: [0, 1, 4, 9, 16]
```

lambda函数：练习

1.

```
sentence = "Welcome To Beijing!"
words = sentence.split()
lengths  = map(lambda x:len(x),words)
print(list(lengths))
```

2.

```
A = [1,2,3,4]
B = [5,6,7,8]
PRINT(LIST(MAP(LAMBDA X,Y:X+Y, A,B)))
```

Python内置函数

Python内置函数

PYTHON解释器主要提供了68个内置函数。

abs()	id()	round()	compile()	locals()
all()	input()	set()	dir()	map()
any()	int()	sorted()	exec()	memoryview()
ascii()	len()	str()	enumerate()	next()
bin()	list()	tuple()	filter()	object()
bool()	max()	type()	format()	property()
chr()	min()	zip()	frozenset()	repr()
complex()	oct()		getattr()	setattr()
dict()	open()		globals()	slice()
divmod()	ord()	bytes()	hasattr()	staticmethod()
eval()	pow()	delattr()	help()	sum()
float()	print()	bytearray()	isinstance()	super()
hash()	range()	callable()	issubclass()	vars()
hex()	reversed()	classmethod()	iter()	import()

Python迭代器和生成器

Python迭代器协议

PYTHON中，任意对象，只要定义了__NEXT__方法，它就是一个迭代器。因此，PYTHON中的容器如列表、元组、字典、集合、字符串等组合数据类型都可以被称作迭代器。

可迭代对象就是：实现了迭代器协议的对象。

```
FOR I IN [1, 2, 3, 4]:
    PRINT(I)
```

Python生成器

生成器函数：常规函数定义，但是，使用YIELD语句而不是使用RETURN语句返回结果。YEILD语句执行一次返回一个结果，在每个结果之间函数处于挂起状态，以便重启的时候继续执行。

生成器表达式：类似于列表推导，但是生成器返回按需产生结果的一个对象，而不是构建一个结果列表。

使用普通函数

```
In [64]: def gensquares(N):
    ...:     res=[]
    ...:     for i in range(N):
    ...:         res.append(i*i)
    ...:     return res
    ...: for item in gensquares(5):
    ...:     print(item)
    ...:
0
1
4
9
16
```

使用生成器

```
In [69]: def gensquares(N):
    ...:     for i in range(N):
    ...:         yield i**2
    ...:

In [70]: for item in gensquares(5):
    ...:     print(item)
    ...:
0
1
4
9
16

In [71]: next(gensquares(5))
Out[71]: 0

In [72]: next(gensquares(5))
Out[72]: 0
```

生成器表达式

```
In [20]: sum([x ** 2 for x in range(4)])
Out[20]: 14

In [21]: squares = [x**2 for x in range(5)]

In [22]: squares
Out[22]: [0, 1, 4, 9, 16]

In [23]: squares = (x**2 for x in range(5))

In [24]: squares
Out[24]: <generator object <genexpr> at 0x000001763EFFAC78>

In [25]: sum([x ** 2 for x in range(5)])
Out[25]: 30

In [26]: sum(x ** 2 for x in range(5))
Out[26]: 30
```

课后练习

1. 编写函数，接收n个数字，求这些参数数字的和。