函数

程序分解方法

1. 函数 (function)
2. 对象 (object)
3. 模块 (module)

函数格式

```
def <name>(arg1, arg2, ..., argN):
    <statements>
    return <value>
```

```
def hello():
    print('Hello World!')
    return True


In : hello()
Hello World!
Out: True
```

向函数传递参数

```
In : def hello(name):
...:     print(f'Hello, {name}!')
...:

In : hello('Amy')
Hello, Amy!

In : hello('Chris')
Hello, Chris!
```

形参 是指函数定义中在内部使用的参数，这是函数完成其工作所需的一项信息，在没实际调用的时候，函数用形参来指代

实参 是指调用函数时由调用者传入的参数，这个时候形参指代的内容就是实参了

上面的例子中name就是形参，amy和chris是实参

实参类型

1. 位置参数 (positional argument)
2. 关键字参数 (keyword argument)

位置参数

```
In : def hello(name):
...:     print(f'Hell, {name}!')
...:
```

```
In : def hello(*names):
...:     print(names)
...:

In : hello()
()

In : hello(1)
(1,)

In : hello(1, 2)
(1, 2)
```

强制关键字参数

```
In : def recv(maxsize, *, block):
...:     pass
...:

In : recv(1024, True)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-53-8e61db2ef94b> in <module>()
----> 1 recv(1024, True)

TypeError: recv() takes 1 positional argument but 2 were given

In : recv(1024, block=True)
```

关键字参数

```
In : def hello(name='World'):
...:     print(f'Hello, {name}!')
...:

In : hello()
Hello, World!

In : hello('Amy')
Hello, Amy!

In : hello(name='Chris')
Hello, Chris!
```

** 变长关键字参数

```
In : def run(a, b=1, **kwargs):
...:     print(kwargs)
...:

In : run(1)
{}

In : run(1, b=2)
{}

In : run(1, c=1)
{'c': 1}
```

混合使用参数

```
In : def hello(name, default='World'):
...:     print(f'Hello, {name or default}!')
...:

In : def func(a, b=0, *args, **kwargs):
...:     print('a =', a, 'b =', b, 'args =', args, 'kwargs =', kwargs)
...:

In : func(1, 2)
a = 1 b = 2 args = () kwargs = {}

In : func(1, 2, d=4)
a = 1 b = 2 args = () kwargs = {'d': 4}

In : func(1, 2, 3)
a = 1 b = 2 args = (3,) kwargs = {}

In : func(1, 2, 3, d=4)
a = 1 b = 2 args = (3,) kwargs = {'d': 4}
```

返回值

```
In : def add(a, b):
...:     return a + b
...:

In : add(1, 2)
Out: 3

In : def partition(string, sep):
...:     return string.partition(sep)
...:

In : partition('/home/dongwm/bran', '/')
Out: ('', '/', 'home/dongwm/bran')
```

参数为函数

```
In : def hello(name):
...:     print(f'Hello {name}!')
...:

In : def test(func, name='World'):
...:     func(name)
...:

In : test(hello, 'Amy')
Hello Amy!
```

本地变量

```
In : def run(name):
...:     s = f'{name}'
...:     for x in range(5):
...:         if x == 3:
...:             return
...:     print(s)
...:

In : run('Test')
```

全局变量

```
In : g = 0

In : def run():
...:     print(g)
...:

In : run()
0

In : def run():
...:     g = 2
...:

In : g
Out: 0
```

错误使用全局变量(一)

```
In : g = 0

In : def run():
...:     print(g)
...:     g = 2
...:     print(g)

In : run()
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-14-157c9bda2cd6> in <module>()
----> 1 run()

<ipython-input-13-8b2ff1ac73b1> in run()
1 def run():
----> 2     print(g)
3     g = 2
4     print(g)
...

UnboundLocalError: local variable 'g' referenced before assignment
```

错误使用全局变量(二)

```
In : g = 0

In : def run():
...:     g += 2
...:     print(g)
...:

In : run()
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-16-157c9bda2cd6> in <module>()
----> 1 run()

<ipython-input-15-573471f9c3b9> in run()
      1 def run():
----> 2     g += 2
      3     print(g)
      4

UnboundLocalError: local variable 'g' referenced before assignment
```

解决方案: global关键字

```
In : def run():
...:     global g
...:     g += 2
...:     print(g)
...:

In : run()
2

In : g
Out: 2

In : run()
4

In : g
Out: 4
```

作用域(scope)

B：build-in 系统变量

G：global 全局变量

E：enclosing 嵌套作用域

L：local 本地作用域

系统变量

```
In : import builtins

In : ', '.join((i for i in dir(builtins) if i.islower() and '_' not in i))
Out: 'abs, all, any, ascii, bin, bool, bytearray, bytes, callable, chr,
classmethod, compile, complex, copyright, credits, delattr, dict, dir, divmod,
dreload, enumerate, eval, exec, filter, float, format, frozenset, getattr,
globals, hasattr, hash, help, hex, id, input, int, isinstance, issubclass,
iter, len, license, list, locals, map, max, memoryview, min, next, object,
oct, open, ord, pow, print, property, range, repr, reversed, round, set,
setattr, slice, sorted, staticmethod, str, sum, super, tuple, type, vars, zip'
```

Python 2 系统变量

```
>>> import __builtin__
>>> dir(__builtin__)
...
```

嵌套作用域

```
In : g = 0

In : def run():
...:     g = 2
...:     def run2():
...:         print(g)
...:     return run2
...:

In : f = run()

In : f()
2
```

闭包(Closure)

```
In : def maker(n):
...:     def action(m):
...:         return m * n
...:     return action
...:

In : f = maker(3)

In : f(2)
Out: 6

In : g = maker(10)

In : g(2)
Out: 20
```

闭包指延伸了作用域的函数，其中包含函数定义体中引用，但是不在定义体中定义的非全局变量，它能访问定义体之外定义的非全局变量

nonlocal

```
In : def run():
...:     g = 2
...:     def run2():
...:         g = 4
...:         print('inner ---> ', g)
...:     run2()
...:     print('outer --->', g)
...:

In : run()
inner --->  4
outer ---> 2
```

```
In : def run():
...:     g = 2
...:     def run2():
...:         nonlocal g
...:         g = 4
...:         print('inner ---> ', g)
...:     run2()
...:     print('outer --->', g)
...:

In : run()
inner --->  4
outer ---> 4
```

匿名函数

```
In : def double(n):
...:     return n * 2
...:
In : double(10)
Out: 20
```

```
In : f = lambda n: n * 2

In : f(10)
Out: 20
```

高阶函数 - map

```
In : l1 = [1, 3, 4]

In : l2 = []

In : for i in l1:
...:     l2.append(double(i))
...:

In : l2
Out: [2, 6, 8]
```

```
In : rs = map(double, l1)

In : rs
Out: <map at 0x105986748>

In : list(rs)
Out: [2, 6, 8]
```

高阶函数 - filter

```
In : def is_odd(x):
...:     return x % 2 == 1
...:

In : rs = filter(is_odd, l1)

In : rs
Out: <filter at 0x105986d68>

In : list(rs)
Out: [1, 3]

In : list(filter(None, [1, '', {}, (), False, None, set()]))
Out: [1]
```

高阶函数 - reduce

```
In : def add(a, b):
...:     return a + b
...:

In : from functools import reduce

In : reduce(add, [1, 2, 3])
Out: 6

In : reduce(add, [1, 2, 3], 10)
Out: 16
```

匿名函数(续)

```
In : def double(n):
...:     return n * 2
...:

In : list(map(double, l1))
Out: [2, 6, 8]

In : list(map(lambda x: x * 2, l1))
Out: [2, 6, 8]

In : l = [[2, 4], [1, 1], [9, 3]]

In : sorted(l)
Out: [[1, 1], [2, 4], [9, 3]]

In : sorted(l, key=lambda x:x[1])
Out: [[1, 1], [9, 3], [2, 4]]

In : l3 = ['/boot/grub', '/usr/local', '/home/dongwm']

In : sorted(l3, key=lambda x: x.rsplit('/')[2])
Out: ['/home/dongwm', '/boot/grub', '/usr/local']
```

常用函数 - zip

```
In : a = [1, 2, 3]

In : b = [4, 5, 6]

In : c = [7, 8, 9, 10]

In : zip(a, b)
Out: <zip at 0x10f305a48>

In : list(zip(a, b))
Out: [(1, 4), (2, 5), (3, 6)]

In : list(zip(a, c))
Out: [(1, 7), (2, 8), (3, 9)]

In : list(zip(*zip(a, b)))
Out: [(1, 2, 3), (4, 5, 6)]
```

常用函数 - sum

```
In : sum([1, 2, 3])
Out: 6

In : sum([1, 2, 3], 10)
Out: 16

In : sum([[1, 2], [3, 4]], [])
Out: [1, 2, 3, 4]
```

开发陷阱(一) 可变默认参数

```
In : def append_to(element, to=[]):
...:     to.append(element)
...:     return to
...:

In : my_list = append_to(12)

In : my_list
Out[83]: [12]

In : my_other_list = append_to(42)

In : my_other_list
Out: [12, 42]
```

```
In : def append_to(element, to=None):
...:     if to is None:
...:         to = []
...:     to.append(element)
...:     return to
...:
```

开发陷阱(二) 闭包变量绑定

```
In : def create_multipliers():
...:     return [lambda x : i * x for i in range(5)]
...:

In : for multiplier in create_multipliers():
...:     print(multiplier(2))
...:
8
8
8
8
8
```

解决方案(一): 函数默认值

```
In : def create_multipliers():
....:     return [lambda x, i=i : i * x for i in range(5)]
....:
```

解决方案(二): 偏函数partial

```
In : from functools import partial
In : from operator import mul

In : def create_multipliers():
....:     return [partial(mul, i) for i in range(5)]
....:
```

延伸阅读

1. https://www.learnpython.org/en/Functions
2. http://book.pythontips.com/en/latest/map_filter.html
3. https://docs.python.org/3/library/functools.html