《Python程序设计》

Python函数编程

刘潇
机械科学与工程学院

2023年10月26日

2023秋季

本节要点

- 掌握匿名函数的定义和使用
- 掌握映射、过滤、规约等典型函数的使用
- 理解过程编程与函数编程的区别

主要内容

1. 匿名函数

2. eval和exec函数

3. 映射、过滤和规约

4. 函数编程应用实例

面向过程编程

面向过程编程是首先分析解决问题所需要的步骤，然后用函数或者模块把这些步骤一步一步实现， 通过依次调用达到目的

Process Oriented（PO）程序设计步骤：

- **分析程序从输入到输出的各步骤**
- **按照执行过程从前到后编写程序**
- **将高耦合部分封装成模块或函数**
- **输入参数，按照程序执行过程调试**

1. **番茄**切块；
2. **鸡蛋**磕入碗搅匀；
3. 淀粉加水调开；
4. 蒜瓣切片；
5. 炒菜锅放少量的油，然后放蒜片；
6. 待出香味，**放番茄翻炒**到皮软出汁，放适量的水、鸡精、盐炒匀；
7. 水开了慢慢把湿淀粉倒进锅中，并不停搅动；
8. 锅再次开后，用筷子搅拌着**鸡蛋慢慢转着倒进去**；
9. 再次开锅即可关火，装汤盆，撒葱花。

棋盘游戏

```python
# 棋盘游戏

def createpoint(startposition): # 落子
    pos = startposition
    def go(direction, step): # 走子
        new_x = pos[0] + direction[0]*step
        new_y = pos[1] + direction[1]*step
        pos[0] = new_x
        pos[1] = new_y
        return pos
    return go

point1 = createpoint([0,0]) # 落第一个棋子
print(point1([10,0], 1))
point2 = createpoint([10,10]) # 落第二个棋子
print(point2([5, 0], 1))
print(point1([0,10], 1))
print(point2([0, 5], 1))
print(point1([-1,0], 5))
```

```
[10, 0]
[15, 10]
[10, 10]
[15, 15]
[5, 10]
```


```python
print(point1, point2)
```

```
<function createpoint.<locals>.go at 0x00000248AA9C1310> <function createpoint.<locals>.go at 0x00000248AA9C1550>
```

point1，point2为函数对象

面向函数编程

面向函数编程是一种抽象程度很高的编程范式，函数是一种对象，程序依靠函数对象的引用和执行来完成

```
def func1(x, y):
    return abs(x)+abs(y)
print(func1(-5, -3))
```

面向过程编程

```
8
```

```
def func1(x, y, f):
    return f(x)+f(y)
print(func1(-5, -3, abs))
print(func1(-5.2, -3.6, round))
```

面向函数编程（函数式编程）

```
8
-9
```

```
print(abs, round)
```

abs，round为函数对象，可以作为参数传入func1（高阶函数）

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

核心思想仍然是面向过程的

匿名函数

□ lambda表达式返回一个函数对象，用于定义匿名函数

lambda 参数1, 参数2, ... : 表达式　　一行完成

```
func1 = lambda x, y : x+y
print(func1)
print(func1(1, 2))
```

```
<function <lambda> at 0x00000248AA8FD280>
3
```

func1指向lambda表达式返回的匿名函数

```
def func1(x, y):
    return x+y
print(func1)
print(func1(1,2))
```

```
<function func1 at 0x00000248AAA66940>
3
```

x, y是传递给表达式x+y的参数

lambda表达式即用即得，执行完即释放，代码简洁高效

匿名函数应用

求1到50能被5和7整除的数

```
def func1(number, end, filter_func):
    if number > end:
        return []
    if filter_func(number):
        return [number]+func1(number+1, end, filter_func)
    else:
        return func1(number+1, end, filter_func)

list1 = func1(1, 50, lambda x: x%5==0 or x%7==0)
print(list1)
```

filter_func

```
[5, 7, 10, 14, 15, 20, 21, 25, 28, 30, 35, 40, 42, 45, 49, 50]
```

```
# 利用for循环语句实现
list1 = []
for number in range(1, 51):
    if number%5 == 0 or number%7 ==0:
        list1.append(number)
print(list1)
```

```
[5, 7, 10, 14, 15, 20, 21, 25, 28, 30, 35, 40, 42, 45, 49, 50]
```

关键字参数和可变参数

```
# 全局变量
x = 1
func1 = lambda x, y=2: x**y
func2 = lambda x, *y: y*x

# 关键字参数
print(func1(x = 2))

# 元组可变参数
print(func2(2, 3, 4))
print(func2(2, *tuple(range(5))))
```

```
4
(3, 4, 3, 4)
(0, 1, 2, 3, 4, 0, 1, 2, 3, 4)
```

Lambda表达式中不支持global语句，无法访问全局变量

***形参：打包**
***实参：解包**

eval和exec函数

- eval和exec为Python内置函数，用于执行一个字符串形式的代码

eval(字符串代码, globals=None, locals=None)

用于求值（evaluate），有返回值

exec(字符串代码, globals=None, locals=None)

用于执行（execute），无返回值

globals：管控全局的命名空间，如果无 globals 参数，则使用Python的全局命名空间
locals：管控局部的命名空间，当它和 globals 中有重复或冲突时，以 locals 的为准，如果 locals 没有被提供，则默认为 globals

eval和exec的区别

```
x = 10
def func1():
    y = 20
    a = eval("x+y")
    print("a:", a)
    b = eval("x+y", {"x":1, "y":2})
    print("b:", b)
    c = eval("x+y", {"x":1, "y":2}, {"y":3, "z":4})
    print("c:", c)
    d = eval("print(x, y)")
    print("d:", d)

func1()
```

```
a: 30
b: 3
c: 4
10 20
d: None
```

eval求表达式的值

exec始终返回None

```
x = 10
def func2():
    y = 20
    a = exec("x+y")
    print("a:", a)
    b = exec("x+y", {"x":1, "y":2})
    print("b:", b)
    c = exec("x+y", {"x":1, "y":2}, {"y":3, "z":4})
    print("c:", c)
    d = exec("print(x, y)")
    print("d:", d)

func2()
```

```
a: None
b: None
c: None
10 20
d: None
```

eval和exec的区别

```
func3 = lambda x, y, z: x+y+z
result = eval("func3(1, 2, 3)")
print(result)
result = exec("sum_number = func3(1, 2, 3)")
print(result, sum_number)

result = eval("sum_number = func3(1, 2, 3)")
print(result, sum_number)
```

```
6
None 6

Traceback (most recent call last):

  File "C:\Users\xiaol\anaconda3\lib\site-packages\IPython\core\interactiveshell.py", line 3437, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  File "<ipython-input-30-011799b2f210>", line 7, in <module>
    result = eval("sum_number = func3(1, 2, 3)")

  File "<string>", line 1
    sum_number = func3(1, 2, 3)

SyntaxError: invalid syntax
```

eval表达式中不能包含赋值符号

Python语言的动态性

```
# 创建动态变量名的变量
for number in range(2, 5):
    exec("var_name{} = {}".format(number, number**2)) # 动态变量赋值

print(var_name2)

# 使用动态名称调用变量
x = 3
exec("print(var_name{})".format(x))
y = 4
number = eval("var_name{}+var_name{}".format(x, y))
print(number)
```

```
4
9
25
```

动态变量名var_name2, var_name3, var_name4

Python内置高阶函数

```
print(dir(__builtin__))
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarnin
g', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarnin
g', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureW
arning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryErro
r', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImp
lemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'Recursi
onError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'Synt
axWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'Unicod
eEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivis
ionError', '__IPYTHON__', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'ab
s', 'all', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyr
ight', 'credits', 'delattr', 'dict', 'dir', 'display', 'divmod', 'enumerate', 'eval', 'exec', 'filter', 'float', 'format', 'frozenset', 'get
_ipython', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'licens
e', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'range', 'rep
r', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
```

map: 映射

fliter: 过滤

```
from functools import reduce
print(reduce.__doc__)
```

```
reduce(function, sequence[, initial]) -> value

Apply a function of two arguments cumulatively to the items of a sequence,
from left to right, so as to reduce the sequence to a single value.
For example, reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]) calculates
((((1+2)+3)+4)+5).  If initial is present, it is placed before the items
of the sequence in the calculation, and serves as a default when the
sequence is empty.
```

reduce: 规约

映射函数map

- map为Python内置高阶函数，根据提供的函数对指定的序列（字符串、列表、元组等）做映射

map(function, 序列1, ...)

可以是多个序列，
返回值为新的序列

```
def power2(x):
    return x**2
```

```
list1 = []
for i in range(10):
    list1.append(power2(i))

print(list1)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

面向过程编程

```
print(list(map(power2, range(10))))
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

半函数编程风格

```
print(list(map(lambda x: x**2, range(10))))
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

全函数编程风格

用于序列类型转换

map函数用于序列类型转换

```
# 字符串转换为列表
print(list("123456"))
print(list(map(int, "123456")))
```

```
['1', '2', '3', '4', '5', '6']
[1, 2, 3, 4, 5, 6]
```

```
# 元组转换为列表
print(list((1, 2, 3, 4, 5, 6)))
print(list(map(float, (1, 2, 3, 4, 5, 6))))
```

```
[1, 2, 3, 4, 5, 6]
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
```

```
# 提取字典的键
print(list(map(float, {"1":1, "2":2, "3":3})))
```

```
[1.0, 2.0, 3.0]
```

多个序列的映射

多个序列的映射

```
1  print(list(map(lambda x, y: x*y, [1, 2, 3, 4], (5, 6, 7, 8))))
```

```
[5, 12, 21, 32]
```

```
1  print(list(map(lambda x, y: x*y, [1, 2, 3, 4], (5, 6, 7, 8, 10))))
2  print(list(map(lambda x, y: x*y, [1, 2, 3, 4, 10], (5, 6, 7, 8))))
```

```
[5, 12, 21, 32]
[5, 12, 21, 32]
```

数据类型动态性

```
1  print(list(map(lambda x, y: x*y, ["1", "2", "3", "4", 10], (5, 6, 7, 8))))
```

```
['11111', '222222', '3333333', '44444444']
```

规范用户名为首字母大写格式

```
user_name_list = []
for i in range(3):
    user_name = input("请输入用户名：")
    user_name_list.append(user_name)

def name_normalize(name):
    return name[0].upper()+name[1:].lower()

print(user_name_list)
print(list(map(name_normalize, user_name_list)))
```

```
请输入用户名：FDAFREAfasdfsadf
请输入用户名：dsfsdfFSDFDs
请输入用户名：fsfeFSDGSsdf
['FDAFREAfasdfsadf', 'dsfsdfFSDFDs', 'fsfeFSDGSsdf']
['Fdafreafasdfsadf', 'Dsfsdffsdfds', 'Fsfefsdgssdf']
```

创建等差浮点数序列

```
xlst=list(map(lambda x:float(x)/100,range(628)))
print(xlst)
```

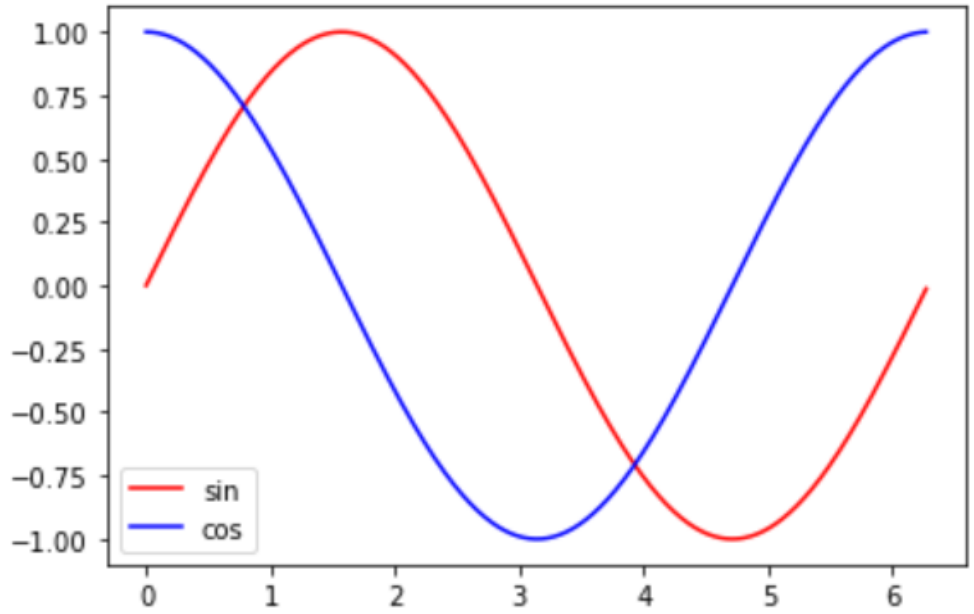
```
[0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2, 0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3, 0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4, 0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5, 0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6, 0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7, 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8, 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.0, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07, 1.08, 1.09, 1.1, 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18, 1.19, 1.2, 1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29, 1.3, 1.31, 1.32, 1.33, 1.34, 1.35, 1.36, 1.37, 1.38, 1.39, 1.4, 1.41, 1.42, 1.43, 1.44, 1.45, 1.46, 1.47, 1.48, 1.49, 1.5, 1.51, 1.52, 1.53, 1.54, 1.55, 1.56, 1.57, 1.58, 1.59, 1.6, 1.61, 1.62, 1.63, 1.64, 1.65, 1.66, 1.67, 1.68, 1.69, 1.7, 1.71, 1.72, 1.73, 1.74, 1.75, 1.76, 1.77, 1.78, 1.79, 1.8, 1.81, 1.82, 1.83, 1.84, 1.85, 1.86, 1.87, 1.88, 1.89, 1.9, 1.91, 1.92, 1.93, 1.94, 1.95, 1.96, 1.97, 1.98, 1.99, 2.0, 2.01, 2.02, 2.03, 2.04, 2.05, 2.06, 2.07, 2.08, 2.09, 2.1, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19, 2.2, 2.21, 2.22, 2.23, 2.24, 2.25, 2.26, 2.27, 2.28, 2.29, 2.3, 2.31, 2.32, 2.33, 2.34, 2.35, 2.36, 2.37, 2.38, 2.39, 2.4, 2.41, 2.42, 2.43, 2.44, 2.45, 2.46, 2.47, 2.48, 2.49, 2.5, 2.51, 2.52, 2.53, 2.54, 2.55, 2.56, 2.57, 2.58, 2.59, 2.6, 2.61, 2.62, 2.63, 2.64, 2.65, 2.66, 2.67, 2.68, 2.69, 2.7, 2.71, 2.72, 2.73, 2.74, 2.75, 2.76, 2.77, 2.78, 2.79, 2.8, 2.81, 2.82, 2.83, 2.84, 2.85, 2.86, 2.87, 2.88, 2.89, 2.9, 2.91, 2.92, 2.93, 2.94, 2.95, 2.96, 2.97, 2.98, 2.99, 3.0, 3.01, 3.02, 3.03, 3.04, 3.05, 3.06, 3.07, 3.08, 3.09, 3.1, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.17, 3.18, 3.19, 3.2, 3.21, 3.22, 3.23, 3.24, 3.25, 3.26, 3.27, 3.28, 3.29, 3.3, 3.31, 3.32, 3.33, 3.34, 3.35, 3.36, 3.37, 3.38, 3.39, 3.4, 3.41, 3.42, 3.43, 3.44, 3.45, 3.46, 3.47, 3.48, 3.49, 3.5, 3.51, 3.52, 3.53, 3.54, 3.55, 3.56, 3.57, 3.58, 3.59, 3.6, 3.61, 3.62, 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.69, 3.7, 3.71, 3.72, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.79, 3.8, 3.81, 3.82, 3.83, 3.84, 3.85, 3.86, 3.87, 3.88, 3.89, 3.9, 3.91, 3.92, 3.93, 3.94, 3.95, 3.96, 3.97, 3.98, 3.99, 4.0, 4.01, 4.02, 4.03, 4.04, 4.05, 4.06, 4.07, 4.08, 4.09, 4.1, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.2, 4.21, 4.22, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.3, 4.31, 4.32, 4.33, 4.34, 4.35, 4.36, 4.37, 4.38, 4.39, 4.4, 4.41, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47, 4.48, 4.49, 4.5, 4.51, 4.52, 4.53, 4.54, 4.55, 4.56, 4.57, 4.58, 4.59, 4.6, 4.61, 4.62, 4.63, 4.64, 4.65, 4.66, 4.67, 4.68, 4.69, 4.7, 4.71, 4.72, 4.73, 4.74, 4.75, 4.76, 4.77, 4.78, 4.79, 4.8, 4.81, 4.82, 4.83, 4.84, 4.85, 4.86, 4.87, 4.88, 4.89, 4.9, 4.91, 4.92, 4.93, 4.94, 4.95, 4.96, 4.97, 4.98, 4.99, 5.0, 5.01, 5.02, 5.03, 5.04, 5.05, 5.06, 5.07, 5.08, 5.09, 5.1, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.2, 5.21, 5.22, 5.23, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.3, 5.31, 5.32, 5.33, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.4, 5.41, 5.42, 5.43, 5.44, 5.45, 5.46, 5.47, 5.48, 5.49, 5.5, 5.51, 5.52, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.6, 5.61, 5.62, 5.63, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.7, 5.71, 5.72, 5.73, 5.74, 5.75, 5.76, 5.77, 5.78, 5.79, 5.8, 5.81, 5.82, 5.83, 5.84, 5.85, 5.86, 5.87, 5.88, 5.89, 5.9, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 6.0, 6.01, 6.02, 6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.1, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.17, 6.18, 6.19, 6.2, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27]
```

创建等差浮点数序列

```
import math
sinlst=list(map(lambda x:math.sin(x),xlst))
coslst=list(map(lambda x:math.cos(x),xlst))
```

```
import matplotlib.pyplot as plt
plt.plot(xlst,sinlst,color='r',label='sin')
plt.plot(xlst,coslst,color='b',label='cos')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x27a16bb3910>
```



过滤函数filter

- filter用于过滤一个序列，序列的每个元素作为参数传递给函数进行判断，过滤掉不符合条件的元素

filter(function, 序列)

function返回Ture或False，返回值为新的序列

```
# 过滤序列中的奇数
def is_even(x):
    return x%2 == 0

even_list = []
for i in range(10):
    if is_even(i):
        even_list.append(i)

print(even_list)
```

Ture或False

面向过程编程

```
[0, 2, 4, 6, 8]
```

```
print(list(filter(is_even, range(10))))
```

半函数编程风格

```
[0, 2, 4, 6, 8]
```

```
print(list(filter(lambda x: x%2==0, range(10))))
```

全函数编程风格

```
[0, 2, 4, 6, 8]
```

```
#不被2整除(奇数)
print(list(filter(lambda x:x%2,range(10))))
```

```
[1, 3, 5, 7, 9]
```

```
#不被2,3整除
print(list(filter(lambda x:x%2 and x%3,range(10))))
```

```
[1, 5, 7]
```

```
#同时被2,3整除
print(list(filter(lambda x:x%2==0 and x%3==0,range(10))))
```

```
[0, 6]
```

```
#滤掉所有的2
list(filter(lambda x:x!=2,[2, 5, 2, 2, 5, 0, 99, 2, 2, 2]))
```

```
[5, 5, 0, 99]
```

C盘中的系统配置文件

```
import re
def fimg(fname):
    return re.search(r"\.sys$",fname.lower()) #is not None
print(fimg('abc.sys'))

import os
l=os.listdir("c:\\")
#print(l)

print(list(filter(fimg,l)))
```

```
<re.Match object; span=(3, 7), match='.sys'>
['hiberfil.sys', 'pagefile.sys', 'swapfile.sys']
```

规约函数reduce

- reduce对参数序列中元素进行累积，对序列中的前两个元素进行操作，得到的结果再与第三个元素用 function 函数运算，最后得到一个结果

reduce(function, 序列, 初始值)

function为二元函数，返回值为最终的结果，初始值为可选参数

```
from functools import reduce

def add_func(x, y):
    print(x, y)
    return x+y

print(reduce(add_func, [1, 2, 3, 4, 5]))
```

Python3中reduce在functools模块

add_func为二元函数

```
1 2
3 3
6 4
10 5
15
```

等价于((((1+2)+3)+4)+5)，得到的结果传入第一个参数

```
print(reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]))
```

```
15
```

无初始值时从列表的前两个元素开始计算

```
print(reduce(lambda x, y: x+y, range(1,6), 100))
```

```
115
```

有初始值时，传入二元函数的第一个参数

```
print(reduce(lambda x, y: x+y, range(1,6), 100, 50))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-65e50c4c98c1> in <module>
----> 1 print(reduce(lambda x, y: x+y, range(1,6), 100, 50))

TypeError: reduce expected at most 3 arguments, got 4
```

只能有一个初始值

求100~200里面所有的素数

```
# 定义判断是否为素数的函数

def isPrime(num):
    if num == 1:
        return True
    for i in range(2, num):
        if num%i == 0:
            return False
    else:
        return True

for i in range(100, 200):
    if isPrime(i):
        print(i, end = ",")
```

```
101,103,107,109,113,127,131,137,139,149,151,157,163,167,173,179,181,191,193,197,199,
```

```
# 不能被质数整除的数
print(reduce(lambda l,y:not 0 in list(map(lambda x: y%x,1)) and l+[y] or l,range(2,200),[]))
```

```
print(list(filter(lambda x: x > 100, reduce(lambda l,y:not 0 in list(map(lambda x: y%x,1)) \
                                            and l+[y] or l,range(2,200),[]))))
```

```
[101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199]
```

函数编程示例

打印出10以内的前三个偶数

```
1 i = 1
2 for number in range(10):
3     if i > 3:
4         break
5     elif number%2 == 0:
6         print(number, end = " ")
7         i += 1
```

```
0 2 4
```

可以在break和continue语句前打印提示信息

```
1 list(filter(lambda number:not number%2, range(10)))[:3]
```

```
[0, 2, 4]
```

打印出10以内除了2的偶数

```
1 for number in range(10):
2     if number%2 != 0 or number == 2:
3         continue
4     print(number, end = " ")
```

```
0 4 6 8
```

```
1 list(filter(lambda number:not number%2 and number!=2, range(10)))
```

```
[0, 4, 6, 8]
```

打印出2到20内的斐波那契数列 F(1) = 1, F(2) = 1, F(n) = F(n-1) + F(n-2) (n>=2 , n∈N*)

```
list1 = [1, 1]
i = len(list1)
for number in range(2, 20):
    if number == list1[i-1] + list1[i-2]:
        list1.append(number)
        i += 1
print(list1)
```

```
[1, 1, 2, 3, 5, 8, 13]
```

```
from functools import reduce
reduce(lambda list1, number: number == list1[-1] + list1[-2] and list1 + [number] or list1, \
       range(2,20), [1,1])
```

```
[1, 1, 2, 3, 5, 8, 13]
```

```
reduce(lambda list1, number: list1 + [number] if number == list1[-1] + list1[-2] else list1, \
       range(2,20), [1,1])
```

```
[1, 1, 2, 3, 5, 8, 13]
```

三目运算

利用莱布尼茨公式计算π

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \cdots$$

```
1  i = 0
2  for n in range(10**6):
3      i += (-1)**n/(2*n+1)
4  print(4*i)
```

3.1415916535897743

```
1  i = 0
2  n = 0
3  while True:
4      i += (-1)**n/(2*n+1)
5      n += 1
6      if n > 10**6: break
7  print(4*i)
```

3.1415936535887745

用while实现

```
1  print(4*reduce(lambda i, n:i+(-1)**n/(2*n+1), range(10**6), 0))
```

3.1415916535897743

用reduce实现

小结

- 利用lambda表达式创建匿名函数
- eval和exec函数的动态执行，体现python语言的动态性
- 使用map、filter、reduce等函数进行函数编程
- 函数式编程可以使代码简洁高效，会影响可读性

下一节：类与对象