🡪

Python的lambda关键字匿名函数：

**lambda <parameter\_list>: <expression>**

lambda关键字用于申明匿名函数

parameter\_list以,分隔的匿名函数接收的参数

：分隔符号，用于分隔parameter\_list以及expression

expression表达式，用于表示匿名函数逻辑

lambda匿名函数将返回一个函数对象，可以使用变量进行收纳。

**lambda匿名函数只用于构建基本逻辑。无法在lambda函数中使用赋值语句，return关键字，也无法使用for, while循环,无法使用完整的判断语句if else（但可以使用Python三元运算符x if 条件 else y）,无法使用关键字def定义方法.**

**🡪**

**在filter, reduce, map函数中，如果需要传入的函数引用所代表的逻辑复杂，那么先额外定义一个函数，再传入该函数应用刀filter, reduce, map中作为其参数**

例子：

lambda s: s[::-1] # 申明匿名函数

callable(lambda s: s[::-1]) # 申明匿名函数，并且使用callable验证其返回为可以调用对象

(lambda s: s[::-1])("I am a string")# 申明匿名函数并且调用，返回值为'gnirts a ma I'

animals = ["ferret", "vole", "dog", "gecko"]   
sorted(animals, key=lambda s: -len(s)) # lambda申明一个函数，于sorted连用

forty\_two\_producer = lambda: 42 # 不使用任何参数的匿名函数  
forty\_two\_producer() # 输出为42

🡪

概念：Python本质是面向对象编程语言。

与之相对应的是面向函数编程语言。面向函数编程的语言特点是将功能逻辑包裹在一个个方法当中，这些方法接收参数返回结果，并且在函数执行过程中不会改变程序的状态。因为不会改变程序的状态，并且对于给定的输入将会返回给定的输出，函数的返回值只和函数内的逻辑有关，所以易于调式理解。

Python本质是面向对象编程语言。Python在1991年开发成型，在1993年加入了支持函数编程的特性，在Python的内置函数当中,filter, map, reduce函数体现了Python提供的函数编程特性支持。

Python的常用内置函数

|  |  |
| --- | --- |
| any( iterable object) | 当这个可迭代对象的所有元素有一个可判断为真的，any内置函数返回为真。即是集合或。 |
| all( iterable object) | 当这个可迭代对象的所有元素都可判断为真的，any内置函数返回为真。即是集合并。 |
| dir(class)  help(class) | Python内置帮助函数dir(class)将列出该类内的所有方法，但是没有简短解释。  Python内置帮助函数help(class)将列出该类内的所有方法，并且含有简短解释。 |
| divmod(num1, num2) | Divmod内置函数方法需要两个参数，第一个是被除数，第二个是除数，返回一个tuple,第一个是商，第二个是余数。 |
| Compile(str)  exec(complie\_object)  eval(str) | Python网络开发框架Django, Flask在模板上用的都是Jinja模板。它可以在Html的模板中加入控制循环的语句，并且Python可以执行。从本质上，能实现这一点是利用了Python的内置函数compile, exec 函数。并且与之相似的还有eval内置函数。  --Compile(str) 的作用是Python将此字符串编译成Python代码并且返回compile实例。然后exec(compile\_obj)接受该compile实例，然后Python可以执行该语句。所以Jinja模板也是一样的，将在HTML模板中先编译，然后再exec内置函数执行。  --eval(str)函数，eval函数用于传入一个表达式字符串，Python将执行并且返回运算值。例如eval(“3+3”) 返回6. 相比于exec, exec内置函数也可以执行表达式，但是exec执行完以后不返回数值，而eval只能传入表达式，不允许像exec那样执行print语句，但是它可以返回数值。 |
| isinstance(obj,class)  issubclass(obj,class) | isinstance(obj,class)判断obj是否为class的一个实例  issubclass(obj,class)判断obj是否为class的一个派生类 |
| filter(function\_obj, iterable\_object) | Filter接受两个参数。  第一个参数是一个**指定函数（非调用），**该指定函数接受一个参数。  第二个参数是一个可迭代对象。  作用是，将可迭代对象中每一个元素取出，作为第一个函数的参数，如果此元素在第一个函数中***为真则保留，否则抛弃***。最后filter函数返回过滤完的可迭代元素。  Note: 可以结合lambda使用  E\_list = [1,2,3,4,5,6]  Result = filter(lambda a: a>3,E\_list) # Result = [4,5,6] |
| map(function, iterable[, iterable1,.., iterableN]) | Map接受两个参数。  第一个参数是一个**指定函数（非调用），**该指定函数接受的参数数量和iterable\_object的个数相同。  第二个参数是一个可迭代对象。  作用是，将从所有可迭代对象中每一个元素取出，作为第一个函数的参数。进行处理，并且自动返回，放在一个新的map对象然后返回。  Note: 可以结合lambda使用  例子：  E\_list = [1,2,3,4,5,6]  Result = list(map(lambda a: a+3,E\_list)) # 传入一个可迭代对象E\_list,所以每次为前面的函数传入一个参数。Result = [4,5,6,7,8,9],最后使用list对map对象进行转化  例子：  list(map(lambda x, y: x - y, [2, 4, 6], [1, 3, 5]))  # 传入两个可迭代对象E\_list,所以每次为前面的函数传入两个参数，分别为每个可迭代对象的每个元素。Result = [1,1,1],最后使用list对map对象进行转化 |
| reduce(function\_obj, iterable\_object) | reduce() 函数会对参数序列中元素进行累积。  函数将一个数据集合（链表，元组等）中的所有数据进行下列操作：用传给 reduce 中的函数 function（有两个参数）先对集合中的第 1、2 个元素进行操作，得到的结果再与第三个数据用 function 函数运算，最后得到一个结果。  第一个参数是一个**指定函数（非调用），**该指定函数接受两个参数。  第二个参数是一个可迭代对象。  Note: 可以结合lambda使用  Note: reduce函数已经从标准内置函数中取消，需要从functools中引入才能使用  Example:  reduce (lambda x,y: x+y, [1,2,3]) |
| global()  local() | global()函数返回所有的全局变量，以字典的方式返回。  local()函数返回所有的全局变量，以字典的方式返回。 |
| hash(str) | 把一个字符串转为哈希值。  Note: 字典在Python的key在底层并不是以字符串存储，Python是将key计算成哈希值，然后作为字典的key |
| Sum(iterable\_obj), max(iterable\_obj), min(iterable\_obj) | Sum返回可迭代序列中的和  max返回可迭代序列中的最大的元素  min返回可迭代序列中的最小的元素 |
| bin(十进制数), hex(十进制数), int(字符串), oct(十进制数) | bin将十进制变为二进制  hex将十进制变为16进制  int 将字符串转为十进制数  oct 将是十进制数转为八进制 |
| reversed(list) | 将列表反转并且返回，不改变源列表 |
| sorted(list) | 将列表排序并且返回，不改变源列表 |