[python 推导式](https://www.cnblogs.com/songdanlee/p/11104725.html)

推导式 comprehensions（又称解析式），是 python 的一种独有特性。推导式是可以从一个数据序列构建另一个新的数据序列。

**1.1列表推导式**

语法：

* 变量名 = [表达式 for 变量 in 列表 for 变量 in  xxx]
* 变量名 = [表达式 for 变量 in 列表 if 条件]

语义：

* 遍历出列表中的内容给变量，表达式根据变量值进行逻辑运算。或者遍历列表中的内容给变量，然后进行判断，符合的值在给表达式。

练习：首字母大写

str=[**"tom"**, **"jack"**, **"bob"**, **"nash"**]  
str2=[]  
for i in str:  
 str2.append(i.capitalize())

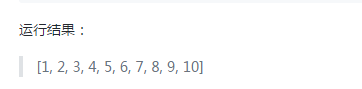
---------------------------------------------

* str=[**"tom"**, **"jack"**, **"bob"**, **"nash"**]  
  str2=[i.capitalize() for i in str if len(str2)>3]

例： t=[type(item) for item in [True, 123,1.0, **'abc'**]]

练习1：快速创建一个包含元素1-10的列表

|  |  |
| --- | --- |
| 1  2 | list1 = [i for i in range(1, 11)]  print(list1) |



练习2：快速创建一个包含1-10之间所有偶数的列表

|  |  |
| --- | --- |
| 1  2 | list1 = [i for i in range(1, 11) if i % 2 == 0]  print(list1) |



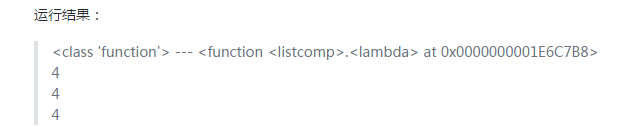
练习3：现在有一列表 lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] ，求出 1/4/7 和 1/5/9元素，思考如何取出

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | # 使用列表推导式：  lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  lst1 = [lst[i][0] for i in range(len(lst))]  print(lst1)  lst2 = [lst[i][i] for i in range(len(lst))]  print(lst2) |



面试题：求

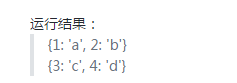
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | a = [lambda x: x \* i for i in range(3)]  # 此处定义了一个装了 3 个匿名函数函数的列表，尚未调用匿名函数，                                           # 调用时全局变量 i 变为 2 相当于 a = [lambda x:x\*2, lambda x:x\*2, lambda x:x\*2]  print(type(a[0]), "---", a[0])  # -> x为函数形参，i为全局变量，程序执行到此处i变为2  print(a[0](2))  print(a[1](2))  print(a[2](2)) |



**2字典推导式**

* 字典推导式列表推导式思想的延续，语法差不多，只不过产生的是字典而已。
* 字典推导式格式：
* 变量名 = {....}
* 练习:将字典中的 key 和 value 进行对换。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 2.2字典推导式    字典推导式列表推导式思想的延续，语法差不多，只不过产生的是字典而已。  字典推导式格式：  变量名 = {....}  练习:将字典中的 key 和 value 进行对换。 |



**3集合推导式**

* 集合推导式跟列表推导式非常相似，唯一区别在于用 { } 代替 [ ]
* 练习：
* 创建一个存储10个偶数的集合

|  |  |
| --- | --- |
| 1  2 | set1 = {x for x in range(10) if x % 2 == 0}  print(set1) |

reversed(str)

str.reverse() #原地改变，面向对象方法，就地修改数据

txt=**" Sir Thomas, poor Sir Thomas, a parent, and conscious of errors in his own conduct as a parent, was the longest to suffer. He felt that he of six-and-twenty, with no want of sense or good companions, was durable18 in its happy effects. He became what he ought to be: useful to his father, steady and quiet, and not living merely for himself."**txts=txt.split()  
txt3=[(len(i),i) for i in txts]  
print(max(txt3))

print(max(txt.split(),key=len))

t2=[i for i in txt.split()]  
t3=filter(lambda x:x[0]==**"a"**,t2)  
print(list(t3))