小白必备基础对象练习

13 list 内元素类型要求一致吗？

list 不要求元素类型一致，如下列表 lst 内元素类型有3种：

复制
```
lst = [1,'xiaoming',29.5,'17312662388']
```

复制
```
for _ in lst:
    print(f'{_}的类型为{type(_)}')
```

打印结果如下，列表 lst 内元素类型有3种：

复制
```
1的类型为<class 'int'>
xiaoming的类型为<class 'str'>
29.5的类型为<class 'float'>
17312662388的类型为<class 'str'>
```

14 list 对象怎样实现增加、删除元素？

使用 append 方法增加一个元素到列表最后：

复制
```
lst2 = ['001','2019-11-11',['三文鱼','电烤箱']]
sku = lst2[2] # sku又是一个列表
sku.append('烤鸭')
print(sku)
```

结果：['三文鱼', '电烤箱', '烤鸭']

使用 insert 方法插入指定元素到指定索引处。

使用 remove 方法删除元素：

复制
```
sku.remove('三文鱼')
sku
```

结果：['电烤箱', '烤鸭']

pop 方法默认删除列表最后一个元素：

复制
```
sku.pop()
sku
```

结果：['电烤箱']

15 a 为 list, 使用 `a[:-1]` 返回哪些元素？

使用 `a[:-1]` 获取到除最后一个元素的切片

16 a 为 list, 使用 `a[1:5:2]` 返回哪些元素？

使用 `a[1:5:2]` 生成索引 [1,5) 且步长为 2 的切片

如 a 等于 [1, 4, 7, 10, 13, 16, 19]

返回的元素为：[4,10]

17 a 为 list, 使用 `a[::-3]` 返回哪些元素？

生成逆向索引 [-1，-len(a))且步长为 3 的切片。

如 a 等于 [1, 4, 7, 10, 13, 16, 19]

返回的元素为：[19,10,1]

18 如何去掉最值求平均？

去掉列表中的一个最小值和一个最大值后，计算剩余元素的平均值。

复制
```
def score_mean(lst):
    lst.sort()
    lst2=lst[1:-1]
    return round((sum(lst2)/len(lst2)),1)

lst=[9.1, 9.0,8.1, 9.7, 19,8.2, 8.6,9.8]
score_mean(lst) # 9.1
```

代码执行过程，动画演示：

19 如何打印出 99 乘法表？

复制
```
for i in range(1,10):
    for j in range(1,i+1):
        print('%d*%d=%d'%(j,i,j*i),end='\t')
    print()
```

打印结果：

```
1*1=1
1*2=2   2*2=4
1*3=3   2*3=6   3*3=9
1*4=4   2*4=8   3*4=12  4*4=16
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

20 判断 list 内有无重复元素

`is_duplicated`，使用 list 封装的 count 方法，依次判断每个元素 `x` 在 list 内的出现次数。

如果大于 1，则立即返回 `True`，表示有重复。

如果完成遍历后，函数没返回，表明 list 内没有重复元素，返回 `False`

```
def is_duplicated(lst):
    for x in lst:
        if lst.count(x) > 1:
            return True
    return False
```

调用 `is_duplicated` 方法：

```
a = [1, -2, 3, 4, 1, 2]
print(is_duplicated(a)) # True
```

以上方法实现不简洁，借助 `set` 判断更方便：

```
def is_duplicated(lst):
    return len(lst) != len(set(lst))
```

21 列表反转

一行代码实现列表反转，非常简洁。

`[::-1]`，这是切片的操作。

`[::-1]` 生成逆向索引（负号表示逆向），步长为 1 的切片。

所以，最后一个元素一直数到第一个元素。这样，不正好实现列表反转吗。

```
def reverse(lst):
    return lst[::-1]
```

调用 `reverse`：

```
r = reverse([1, -2, 3, 4, 1, 2])
print(r)
```

结果：[2, 1, 4, 3, -2, 1]

22 求表头

返回列表的第一个元素

注意，列表为空时，返回 `None`

通过此例，学会使用 `if` 和 `else` 的这种简洁表达。

```
In [18]: def head(lst):
    ...:     return lst[0] if len(lst) > 0 else None
```

调用 `head`：

```
In [19]: print(head([]))
    ...: print(head([3, 4, 1]))
None
3
```

23 求表尾

求列表的最后一个元素，同样列表为空时，返回 `None`.

```
In [20]: def tail(lst):
    ...:     return lst[-1] if len(lst) > 0 else None
```

调用 `tail`：

```
In [21]: print(tail([]))
    ...: print(tail([3, 4, 1]))
None
1
```

24 元素对

`t[:-1]`，原列表切掉最后一个元素；

`t[1:]`，原列表切掉第一个元素；

`zip(iter1, iter2)`，实现 `iter1` 和 `iter2` 的对应索引处的元素拼接。

```
In [32]: list(zip([1,2],[2,3]))
Out[32]: [(1, 2), (2, 3)]
```

理解上面，元素组对的实现就不难理解：

```
In [28]: def pair(t):
    ...:     return list(zip(t[:-1],t[1:]))
```

调用 `pair`：

```
In [29]: pair([1,2,3])
Out[29]: [(1, 2), (2, 3)]

In [30]: pair(range(10))
Out[30]: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8),
 (8, 9)]
```

25 一行代码生成 [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

使用列表生成式，创建列表，观察元素出现规律，可得出如下代码：

```
In [97]: a = [2*i+1 for i in range(10)]

In [98]: a
Out[98]: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

26 写一个等差数列

产生一个首项为 10，公差为 12 ，末项不大于 100 的列表

使用列表生成式创建：

```
In [1]: a = list(range(10,100, 12))

In [2]: a
Out[2]: [10, 22, 34, 46, 58, 70, 82, 94]
```

27 一行代码求 1到10000内整数和

提供两种方法

使用 Python 内置函数 `sum` 求和：

```
In [99]: s = sum(range(10000))

In [100]: s
Out[100]: 49995000
```

使用 `functools` 模块中的 `reduce` 求和：

```
In [101]: s = reduce(lambda x,y: x+y, range(10000))

In [102]: s
Out[102]: 49995000
```

28 tuple 对象可以增加、删除元素吗？为什么？

不能增加、删除元素，因为元组 tuple 是不可变对象，不可变的英文：immutable.

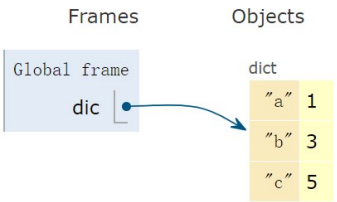
只有不可变对象才能作为字典的键，因此 {[]:'this is an expection'} 会抛出异常。

29 如何创建一个字典？

使用一对花括号 `{}` 另使用冒号 `:`，创建一个 dict 对象：

```
dic = {'a':1, 'b':3, 'c':5} # dict变量
```

字典是一个哈希表，下面的示意图形象的表达出字典的 “形”。



五种创建字典的方法

1 手动创建

```
empty = {}
dic = {'a':1,'c':3,'e':5}
```

2 使用 `dict()` 构造函数

```
In [10]: dict(a=1,b=2,c=3)
Out[10]: {'a': 1, 'b': 2, 'c': 3}
```

3 键值对+关键字参数

第一个参数为字典，后面是一系列关键字参数，如 `c=3`

```
In [9]: dict({'a':1,'b':2},c=3,d=4)
Out[9]: {'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

4 可迭代对象

列表，元素又为一个元组，后面再加一系列关键字参数。

```
In [8]: dict([('a',1),('b',2)],c=3)
Out[8]: {'a': 1, 'b': 2, 'c': 3}
```

5 `fromkeys()` 方法

已知键集合(`keys`)，values 为初始值：

```
In [7]: {}.fromkeys(['k1','k2','k3'],[1,2,3])
Out[7]: {'k1': [1, 2, 3], 'k2': [1, 2, 3], 'k3': [1, 2, 3]}

In [14]: {'a':1,'b':2}.fromkeys(['c','d'],[1,2])
Out[14]: {'c': [1, 2], 'd': [1, 2]}
```

30 字典的基本操作

基本操作包括：

- 创建字典
- 遍历字典
- 获取所有键集合(`keys`)
- 获取所有值集合(`values`)
- 获取某键对应的值
- 添加、修改或删除一个键值对

创建字典 `d` ：

```
In [2]: d = {'a':1,'b':2,'c':3}
```

字典属于容器，遍历容器每一项：

```
In [3]: for key, val in d.items():
   ...:     print(key,val)
```

结果：

```
a 1
b 2
c 3
```

获取所有键集合：

```
# 方法1
In [4]: set(d)
Out[4]: {'a', 'b', 'c'}
# 方法2
In [6]: set(d.keys())
Out[6]: {'a', 'b', 'c'}
```

获取所有值集合：

```
In [7]: set(d.values())
Out[7]: {1, 2, 3}
```

判断键是否在字典中：

```
# 判断键c在d中？
In [8]: if 'c' in d:
   ...:     print('键c在字典d中')
键c在字典d中

# 判断键c不在d中？
In [9]: if 'e' not in d:
   ...:     print('键e不在字典d中')
键e不在字典d中
```

获取某键对应的值：

```
In [10]: d.get('c')
Out[10]: 3
```

添加或修改一个键值对：

```
In [11]: d['d'] = 4
    ...: print(d)
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

删除一个键值对：

```
# 方法1
In [12]: del d['d']
    ...: print(d)
{'a': 1, 'b': 2, 'c': 3}

# 方法2
In [13]: d.pop('c') # 返回3
    ...: print(d)
{'a': 1, 'b': 2}
```

31 字典有哪三个字典视图对象？

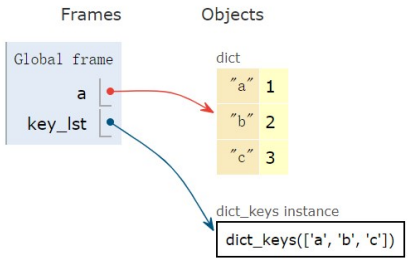
字典自带的三个方法 `d.items()`，`d.keys()`，`d.values()` ，分别返回如下对象：

```
In [14]: d = {'a': 1, 'b': 2, 'c': 3}
In [15]: d.keys()
Out[15]: dict_keys(['a', 'b', 'c'])

In [16]: d.values()
Out[16]: dict_values([1, 2, 3])

In [17]: d.items()
Out[17]: dict_items([('a', 1), ('b', 2), ('c', 3)])
```

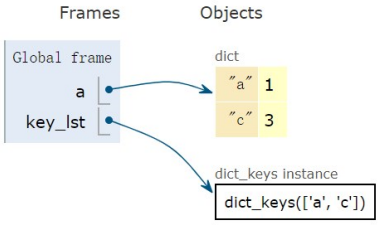
它们都是原字典的视图，修改原字典对象，视图对象的值也会发生改变。

```
a = {'a':1, 'b':2, 'c':3}
key_lst = a.keys() # 创建字典的键集合视图
```

代码可视图如下所示：

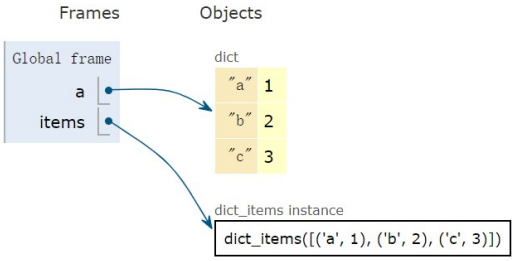


删除键 `b`，由代码执行可视化图可看到，视图对象 `key_lst` 值也发生改变。



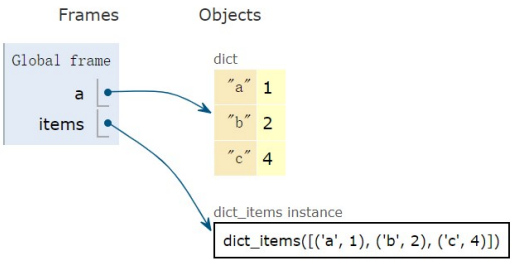
`items()`，也是返回一个视图对象：

```
a = {'a':1, 'b':2, 'c':3}
items = a.items()
```



修改字典，可看到视图对象 `items` 的值也会改变：

```
a['c']=4
```



32 所有对象都能作为字典的键吗？

如果一个列表对象 `lst` 试图作为字典的键，会出现什么问题。

实验一下：

```
In [1]: lst = [1,2]
In [2]: d = {lst:'ok?'}

TypeError: unhashable type: 'list'
```

会抛出如上 `TypeError` 异常：不可哈希的类型 `list`.

因为列表是可变对象，而可变对象是不可哈希的，所以会抛出如上异常。

结论：可哈希的对象才能作为字典的键，不可哈希的对象不能作为字典的键。

33 更新字典常用的三种入参

实际使用字典时，需要批量插入键值对到已有字典中，使用 `update` 方法实现批量插入。

已有字典中批量插入键值对：

```
In [23]: d = {'a': 1, 'b': 2}

# 方法1
In [24]: d.update({'c':3,'d':4,'e':5})
In [25]: d
Out[25]: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

```
# 方法2
```

```
In [31]: d = {'a': 1, 'b': 2}
    ...: d.update([('c',3),('d',4),('e',5)]) # 实现与方法1一样效果

In [32]: d
Out[32]: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

复制

```
# 方法3
In [33]: d = {'a': 1, 'b': 2}
    ...: d.update([('c',3),('d',4)],e=5) # 实现与方法1一样效果

In [34]: d
Out[34]: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

34 字典的 setdefault 方法使用举例

如果仅当字典中不存在某个键值对时，才插入到字典中；

如果存在，不必插入(也就不会修改键值对)。

这种场景，使用字典自带方法 `setdefault`：

复制

```
In [35]: d = {'a':1,'b':2}

In [36]: r = d.setdefault('c',3) # r: 3

In [37]: r
Out[37]: 3

In [38]: d
Out[38]: {'a': 1, 'b': 2, 'c': 3}


In [39]: r = d.setdefault('c',33) # r:3, 已经存在'c':3的键值对, 所以setdefault
时d无改变

In [40]: r
Out[40]: 3

In [41]: d
Out[41]: {'a': 1, 'b': 2, 'c': 3}
```

35 Python 如何创建一个元组对象？

使用一对括号 `()`，创建一个 tuple 型对象：

复制

```
tup = (1,3,5) # tuple 变量
```

但需要注意，含单个元素的元组后面必须保留一个逗号，才被解释为元组。

复制

```
tup = (1,) # 必须保留逗号
```

否则会被认为元素本身：

复制

```
tup=(1)
print(type(tup))
```

结果：

36 集合的最大特点

集合是一种不允许元素出现重复的容器。

判断一个列表中是否含有重复元素，便可借助集合这种数据类型。

复制

```
def duplicated(lst):
    return len(lst)!=len(set(lst)) # 不相等就意味着含重复元素
```

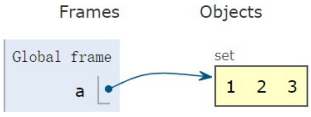
37 创建集合有哪两种方法？

与字典(`dict`)类似，集合(`set`)也是由一对花括号(`{}`)创建。但是，容器内的元素不是键值对。

复制

```
a = {1, 2, 3}
```

示意图，如下：



同字典类似，集合内的元素必须是可哈希类型(`hashable`).

这就意味着 `list`, `dict` 等不可哈希的对象不能作为集合的元素。

复制

```
In [1]: {[1,2]}

TypeError: unhashable type: 'list'
```

另一种创建集合的方法，是通过Python的内置的 `set` 函数，参数类型为可迭代对象 `Iterable`

复制

```
In [1]: set([1,3,5,7])
Out[1]: {1, 3, 5, 7}
```

38 一个集合是另一个的子集吗？

复制

```
In [15]: a = {1,3,5,7}

In [16]: b = {3,4,5,6}

In [17]: a.issubset(b)
```

```
Out[17]: False

In [18]: a.issubset(a)
Out[18]: True

In [19]: a.issubset({1,3,5,7,8})
Out[19]: True
```

下一章

互动评论

说点什么

评论

The Scrapper 2 个月前

day2，继续学习

鼓掌

の娟子 3 个月前

不赖，有几个方法都不知道，基础还是要再牢固下

鼓掌

M 3 个月前

35 Python 如何创建一个集合对象？ 使用一对括号 ()，创建一个 tuple 型对象： 这个35写错了把，能否认证一点

1

无畏 回复 zglg（作者） 1 个月前

写错了为什么不承认呢？

鼓掌

zglg（作者） 3 个月前

你再看看，没有写错。 集合使用 {} ， 而 (,) 创建元组

鼓掌

胃爽 3 个月前

27:一行代码求 1到10000内整数和, 难道不应该是sum(range(1,10001))吗？sum(range(10000))不就是0到10000的和？

1

胃爽 回复 Callen 3 个月前

我说错了， range(10000)是0到9999， 但是它的求和是与1到10000求和结果不同的，所以应该是range(10001) 吧。

鼓掌

Callen 3 个月前

1到10000内整数和，那就是从1加到9999，不包含10000，0不影响最终结果。个人理解。

鼓掌

存