python复习1 数据结构

1. 列表List

- list = ['abcd', 786 , 2.23, 'john', 70.2]
- tinylist = [123, 'john']
  - print (list) # Prints complete list
  - print (list[0]) # Prints first element of the list
  - print (list[1:3]) # Prints elements starting from 2nd t
  - print (list[2:]) # Prints elements starting from 3rd eleme
  - print (tinylist * 2) # Prints list two times
  - print (list + tinylist) # Prints concatenated lists

```
# 输出
print(list[1:3])   包头不包尾
输出:  [786,2.23]
print(list[2:])
输出:[2.23]
print(tinylist * 2)
输出:[123,'john',123,'john']   # *拼接在一起

```

```
list = ['physics', 'chemistry', 1997, 2000]
print (list)

del list[2]
print ("After deleting value at index 2 : ", list)
```

```
del list[2]    # 删除列表第三个元素
输出：['physics', 'chemistry', 2000]
```

```
list.append(obj)  # 末尾添加一个元素
list.extend(seq)  # 在末尾一次性追加另一个序列（可以是列表，元组等）
list.pop(obj = list[-1])  # 移除列表中的一个元素，默认为最后一个元素
 #
my_list = [1, 2, 3, 4]
popped_element = my_list.pop()
# popped_element 的值为 4，my_list 的值为 [1, 2, 3]
 #
list.clear() # 移除列表中所有元素
list.remove(obj) # 移除列表中指定的元素
 #
my_list = [1, 2, 3, 2]
my_list.remove(2)
# 现在 my_list 的值为 [1, 3, 2]
 #
list.insert(index, obj) # 在指定位置（前面）插入一个元素
 #
my_list = [1, 2, 3]
my_list.insert(1, 4)
# 现在 my_list 的值为 [1, 4, 2, 3]
 #
list.reverse() # 反转列表中的元素顺序
 #
my_list = [1, 2, 3]
my_list.reverse()
# 现在 my_list 的值为 [3, 2, 1]
 #
list.sort([func]) # 对列表进行排序
list.count(obj) # 统计列表中指定元素的出现次数
list.index(obj) # 返回列表中第一个匹配给定值的元素的索引
```

🍞 易错题

```
nums = [1, 2, 2, 2, 3, 4, 2]
for num in nums:
    print(num)
    if num == 2:
```

nums.remove(2)

print(nums)

输出：

1

2

2

4

2

[1, 3, 4, 2]

这道题的关键是要理解remove(2)中的参数指的是元素，不是索引。

几个Method的比较

💗 sorted()返回一个排序列表

reversed()返回一个逆序序列的迭代器

```
list.sort(key=None,reverse=False)
#key 参数是一个函数，用于生成用于排序比较的键。
#reverse 参数是一个布尔值，用于指定是否按降序进行排序。
list.reverse() # 反转列表中的元素顺序
sorted(iterable, key=None, reverse=False) # 这是一个内置函数，不是列表的方法。
#iterable 是要排序的可迭代对象，key 和 reverse 参数与 list.sort() 方法相似
reversed(seq) # 这是一个内置函数，返回一个反向的迭代器
```

```
s = 'acbegscvAB'
result = sorted(s, reverse=True)
print(result)
```

```
['v', 's', 'g', 'e', 'c', 'c', 'b', 'a', 'B', 'A']
```

```
my_list = [1, 2, 3]
reversed_list = list(reversed(my_list))
# reversed_list 的值为 [3, 2, 1]
# my_list 保持不变，仍然为 [1, 2, 3]
```

总结：需要注意的是sort和reverse都是list的方法，会改变list。但sorted和reversed都是内置函数，不会改变list。

切片语法

```
lst[::-1]
[::-1] 是切片语法，它包含三个部分：start:stop:step
start 表示起始位置，未指定时默认为列表的开头。
stop 表示结束位置，未指定时默认为列表的末尾。
step 表示步长，未指定时默认为 1。
start 未指定，表示从列表的开头开始。
stop 未指定，表示直到列表的末尾结束。
step 是 -1，表示从右到左逆向遍历列表。
所以，lst[::-1] 返回了一个反向的列表，包含了原列表中的所有元素，但顺序完全相反。
但这个操作不会修改lst。
```

2. 元组tuple

tup = ()创建一个空元组

单元素元组的表示(1,)

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
tiny_tuple = (123, 'john')
print (tuple[1:3])
print (tuple[2:])
print (tiny_tuple*2)
print (tuple + tiny_tuple)
```

元组和列表的区别：

1. 可变性

列表是可变的，但元组是不可变的。

2. 语法表示

列表用[],元组用()

3. 字典Dictionary

由键值对组成，其中每个键必须是唯一的，但值可以重复。字典中的键是不可变的，但值可以是任意类型。

dict = {}　　dict()

```
# 使用花括号创建字典
my_dict = {'key1': 'value1', 'key2': 'value2'}
# 使用dict()构造函数创建字典
another_dict = dict(key1='value1', key2='value2')
```

dict['one'] = "This is one" key='one' value="This is one"

```
tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
print (dict['one']) # Prints value for 'one' key
print (dict[2]) # Prints value for 2 key
print (tinydict) # Prints complete dictionary
print (tinydict.keys()) # Prints all the keys
print (tinydict.values()) # Prints all the values
```

方法：

```
dict.fromkeys()
这个方法使用可迭代对象中的元素作为键，将它们关联到指定的值（默认为 None），创建一个新的字典。
keys = ['key1', 'key2', 'key3']
default_value = 'default_value'
new_dict = dict.fromkeys(keys, default_value)
在这个例子中，new_dict 将会是
{'key1': 'default_value', 'key2': 'default_value', 'key3': 'default_value'}
dict.update(dict2)
这个方法用另一个字典或键值对的可迭代对象中的元素更新当前字典。
dict1 = {'key1': 'value1', 'key2': 'value2'}
dict2 = {'key2': 'new_value', 'key3': 'value3'}
dict1.update(dict2)
在这个操作之后，dict1 将会是 {'key1': 'value1', 'key2': 'new_value', 'key3': 'value3'}。如果 dict2 中的键在 dict1 中已经存在，它们的值将被更新；否则，新的键值对将被添加。
dict.copy()
这个方法创建字典的浅拷贝。对拷贝的修改不会影响原始字典，反之亦然。

dict.setdefault(key, default = None)
这个方法返回指定键的值。如果键不存在，则插入具有指定值的键（默认为 None），并返回默认值。

dict.clear()
这个方法清空字典中的所有元素，使字典变为空字典。
del / dict. pop() / dict. popitem()
# 使用 del 删除指定键
del my_dict['key1']
```

```
# 使用 pop 删除指定键，并返回值，如果不存在返回默认值
value = my_dict.pop('key2', 'default_value')
# 使用 popitem 弹出一对键值对
popped_item = my_dict.popitem()
弹出并返回字典中的一对键值对。字典是无序的，所以弹出的项是不确定的。
dict.get(key, default=None)
这个方法获取字典中指定键的值。如果键不存在，返回指定的默认值（默认为 None）
dict.items()
这个方法返回一个包含字典所有键值对的视图。每个键值对表示为元组
my_dict = {'key1': 'value1', 'key2': 'value2'}
items = my_dict.items()
输出: dict_items([('key1', 'value1'), ('key2', 'value2')])
dict.keys()
这个方法返回一个包含字典所有键的视图
dict.values()
这个方法返回一个包含字典所有值的视图
```

字典dict特点： ▪ 查找和插入的速度极快，不会随着key的增加而变慢； ▪ 需要占用大量的内存，内存浪费多。

列表list特点： ▪ 查找和插入的时间随着元素的增加而增加； ▪ 占用空间小，浪费内存很少。

字典dict是用空间来换取时间的一种方法。

4. 集合Set

由唯一的、不可变的元素组成。集合中的元素没有顺序，不能通过索引访问。

使用花括号 `{}` 表示，元素之间用逗号分隔。

默认情况下，这些大括号用于字典，但如果我们在这些大括号"{}"中使用逗号分隔元素列表，则会将其视为集合。

可以使用set()函数创建集合。

```
# 从列表创建集合
my_list = [1, 2, 3, 3, 4, 5, 5]
my_set = set(my_list)
print(my_set)
输出: {1, 2, 3, 4, 5}
```

```
# 从元组创建集合
my_tuple = (1, 2, 3, 3, 4, 5, 5)
my_set = set(my_tuple)
print(my_set)
输出: {1, 2, 3, 4, 5}
```

```
# 从字符串创建集合
my_string = "hello"
my_set = set(my_string)
print(my_set)
输出：{'h', 'e', 'l', 'o'}
```

方法：

```
add(item)
添加一个新元素到集合中
my_set = {1, 2, 3}
my_set.add(4)

clear()
删除集合中的所有元素使其成为空集合

copy()
创建集合的浅拷贝，返回一个新的集合

difference(other_set)
返回一个新的集合，其中包含只在第一个集合中出现而不在其他集合中的元素
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
diff_set = set1.difference(set2)
输出：{1,2}

discard(item)
从集合中移除指定的元素，如果元素不存在，则不执行任何操作

intersection(other_set)
返回一个新的集合，其中包含两个集合中都存在的元素

issubset(other_set)
检查当前集合是否是另一个集合的子集

issuperset(other_set)
检查当前集合是否是另一个集合的超集

```

5. 总结

- 列表：

list = [1,2,3,4]

有序，可变

可以包含不同类型元素

直接通过索引方式访问，如list[0]

通过循环结构遍历列表

支持切片操作（**包头不包尾**）

允许重复元素

可以进行增删改操作

- 元组：

tuple = (1,2,3)

有序，**不可变**

可以包含不同类型元素

直接通过索引方式访问，如tuple[0]

通过循环结构遍历元祖

切片方式与列表相同

可以包含重复元素

打包与解包

打包

声明元祖的过程就是将各个元素进行打包

```
>>> t = (123, 'python', 1.23)
>>> t
(123, 'python', 1.23)
```

解包

可以使用多个变量承接元祖中的各个元素

```
>>> i, s, d = t
>>> i
123
>>> s
'python'
>>> d
1.23
```

- 字典：

my_dict={'key1': 'value1', 'key2': 'value2'}

无序的键值对集合

键必须是不可变的类型（通常是字符串或数字）

值可以是任意类型

可以通过键访问和修改值

不支持索引和切片

用于表示映射关系

- 集合：

无序、唯一元素的集合

my_set={1,2,3}

元素是唯一的，不能重复

不支持索引和切片

用于去重和集合运算

集合（set）是一个无序的不重复元素序列。可以使用大括号 **{ }** 或者 **set()** 函数创建集合。

空集合只能使用set()创建，{}创建的是空字典。