PYTHON BASICS
容器型数据

2020-04-14

STEVEN WANG

上节总结

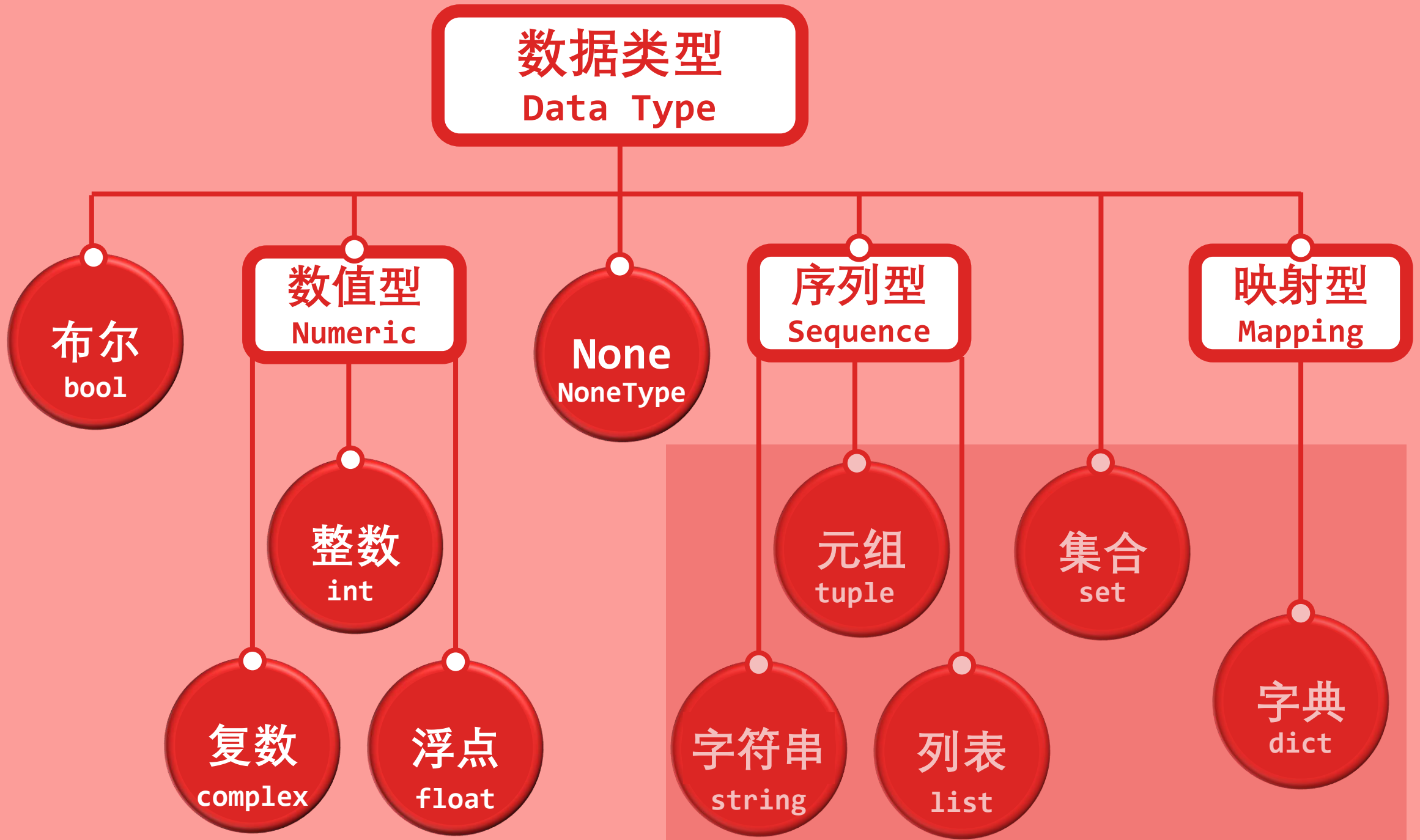
七大运算符

赋值
算术
比较
按位
逻辑
身份
成员

→

数据类型	含义	用处
int	整数型变量	整数运算
float	浮点型变量	实数运算
bool	布尔型变量	流程控制
NoneType	None 关键字	初始化值

→ 表达式


数据类型
Data Type
布尔
bool
数值型
Numeric
整数
int
复数
complex
浮点
float
None
NoneType
序列型
Sequence
元组
tuple
字符串
string
列表
list
集合
set
映射型
Mapping
字典
dict

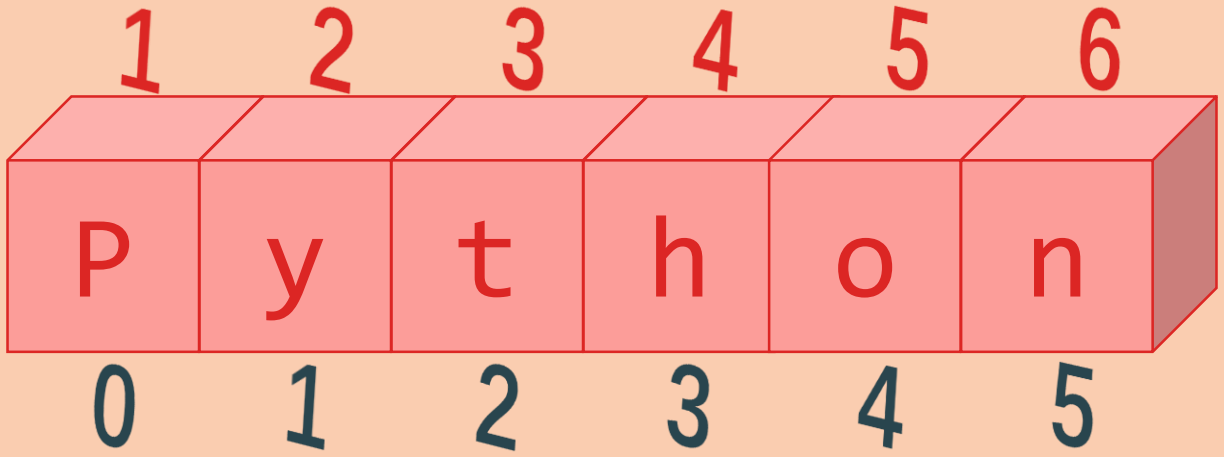
容器型数据

- 字符串
- 列表
- 元组
- 字典
- 集合
- 其他

创建字符串

字符串 (string) 是若干字符的序列，也可把它当成是字符 (char) 的容器。
创建单行和多行字符串的几种方法：

1. 单引号 '，双引号 "：　　s = 'Python', s = "Python"
2. 三单引号 '''，三双引号 """：　创建多行字符串
3. 内置函数 str()



字符串三件套

内存地址

`id()` 函数用于字符串的内存地址。

```
id(s)
1924357357896
```

类型

`type()` 函数返回字符串的类型。

```
type(s)
<class 'str'>
```

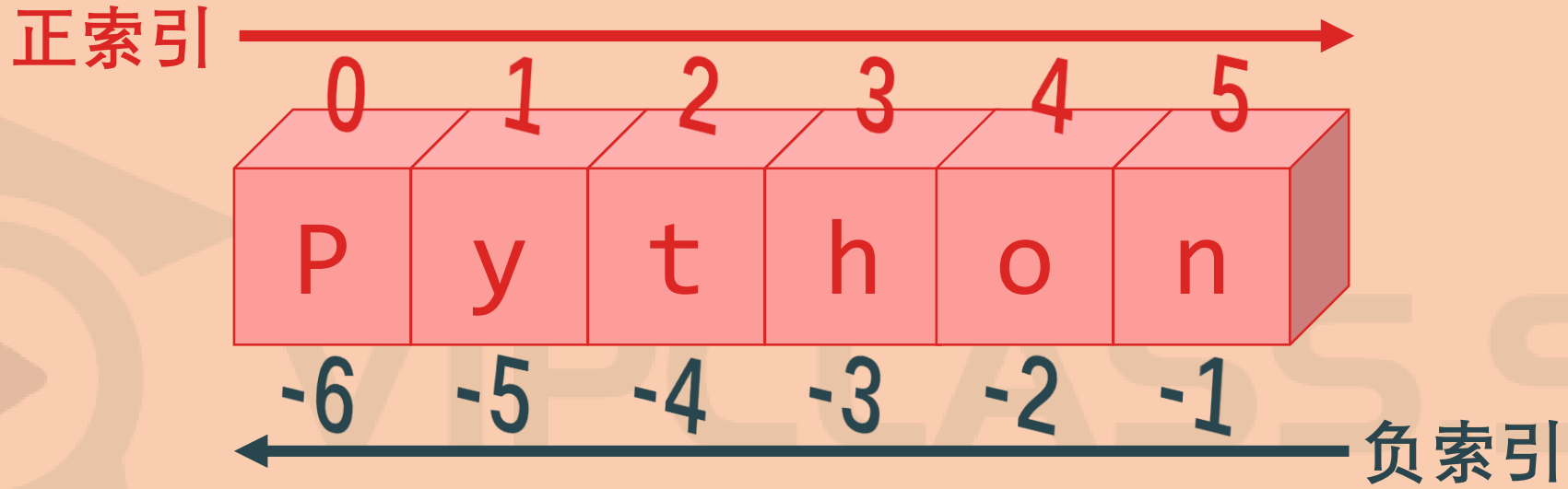
值

`print()` 函数返回字符串的值。

```
print(s)
Python
```

索引字符串

字符串也是可迭代的，可使用索引查找字符串中的每个字符。



索引从 0 开始

```
s[0]
'P'
```

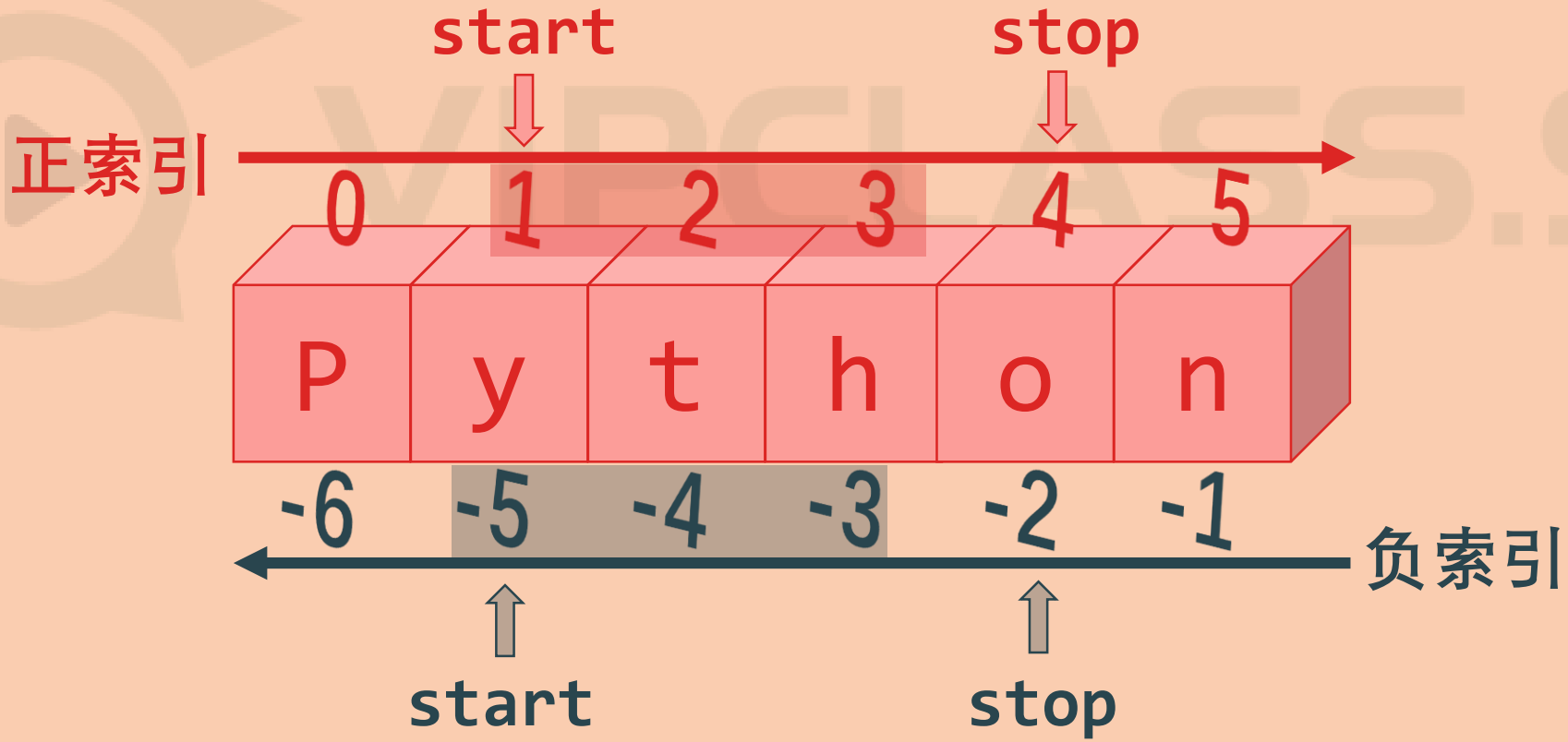
索引值可正可负

```
s[-5]
'y'
```

切片字符串

切片可将字符串的子集，创建为一个新字符串，语法是 s[start:stop]

- start 是起始索引，切片时包括起始索引位置的元素
- stop 是结束索引，切片时不包括结束索引位置的元素



切片字符串

起始索引　结束索引　步长

字符串长度
N = len(s)

→ s[start : stop : step]

```
s[:] = s[0:N:1]
s[start:] = s[start:N:1]
s[:stop] = s[0:stop:1]
s[start:stop] = s[start:stop:1]
s[::-1] = s[0:N:-1]
```

步长留空 ⇨ step = 1

起始索引留空 ⇨ start = 0

结束索引留空 ⇨ stop = N

字符串常见方法

分

分隔 →

`s.partition(d)`	以 d 为界
`s.split(d)`	d 分隔符
`s.splitlines()`	句子分行

合

合并 →

`d.join(l)`	用 d 连 l

查

查找 →

`s.find(x)`	s 中找 x

替

替换 →

`s.replace(x,y)`	y 替换 x

删

删除 →

`s.strip(d)`	删句首末
`s.rstrip(d)`	删句末
`s.lstrip()`	删句首

补

补齐 →

`s.center(n,d)`	补句首末
`s.rjust(n,d)`	补句末
`s.ljust(n,d)`	补句首

格式

`s.upper()`	全部大写
`s.lower()`	全部小写
`s.capitalize()`	大写句首
`s.title()`	大写词首

用 f-string 格式化

格式化字符串

范式 `f'{value:type}'` 用 f-string

```
f'this is an integer {15:d}'
'this is an integer 15'
```

```
f'this is an integer {15:4d}'
'this is an integer   15'
```

2 个空格

```
f'this is an integer {15:04d}'
'this is an integer 0015'
```

```
f'this is a float {83.1031:.2f}'
'this is a float 83.10'
```

```
f'this is a float {83.1031:8f}'
'this is a float 83.103100'
```

```
f'this is a float {83.1031:8.2f}'
'this is a float    83.10'
```

3 个空格

```
f'this is a string {'love':10s}'
'this is a string       love'
```

6 个空格

字符串 + 运算符

算术运算符

\+

```
'love' + ' ' + 'you'
```

```
love you
```

*

```
'love ' * 3
```

```
love love love
```

比较运算符

== !=

> <

>= <=

```
'Python' >= 'Matlab'
```

```
True
```

```
'Python'!='P'+'ython'
```

```
False
```

成员运算符

in

not in

```
'f' not in 'finance'
```

```
False
```

身份运算符

is

is not

```
'P' is not 'Python'
```

```
True
```

容器型数据

- 字符串
- 列表
- 元组
- 字典
- 集合
- 其他

列表

列表 (list) 是存储有序对象的一种容器，其定义语法为

```
l = [元素1, 元素2, ..., 元素n]
```

需要注意的是中括号 [] 和逗号 ,

- 中括号把所有元素绑一起
- 逗号将每个元素一一分开

```
l1 = [1, 10.31, 'Python']
```

```
l2 = list((1, 10.31, 'Python'))
```

```
l3 = list('Python')
```

还可以使用内置函数 list() 将字符串和元组转换成列表。

- l = list(字符串)
- l = list(元组)

列表三件套

内存地址

`id()` 函数返回列表的内存地址。

```
id(l)
```

```
1831016236936
```

类型

`type()` 函数返回列表的类型。

```
type(l)
```

```
list
```

值

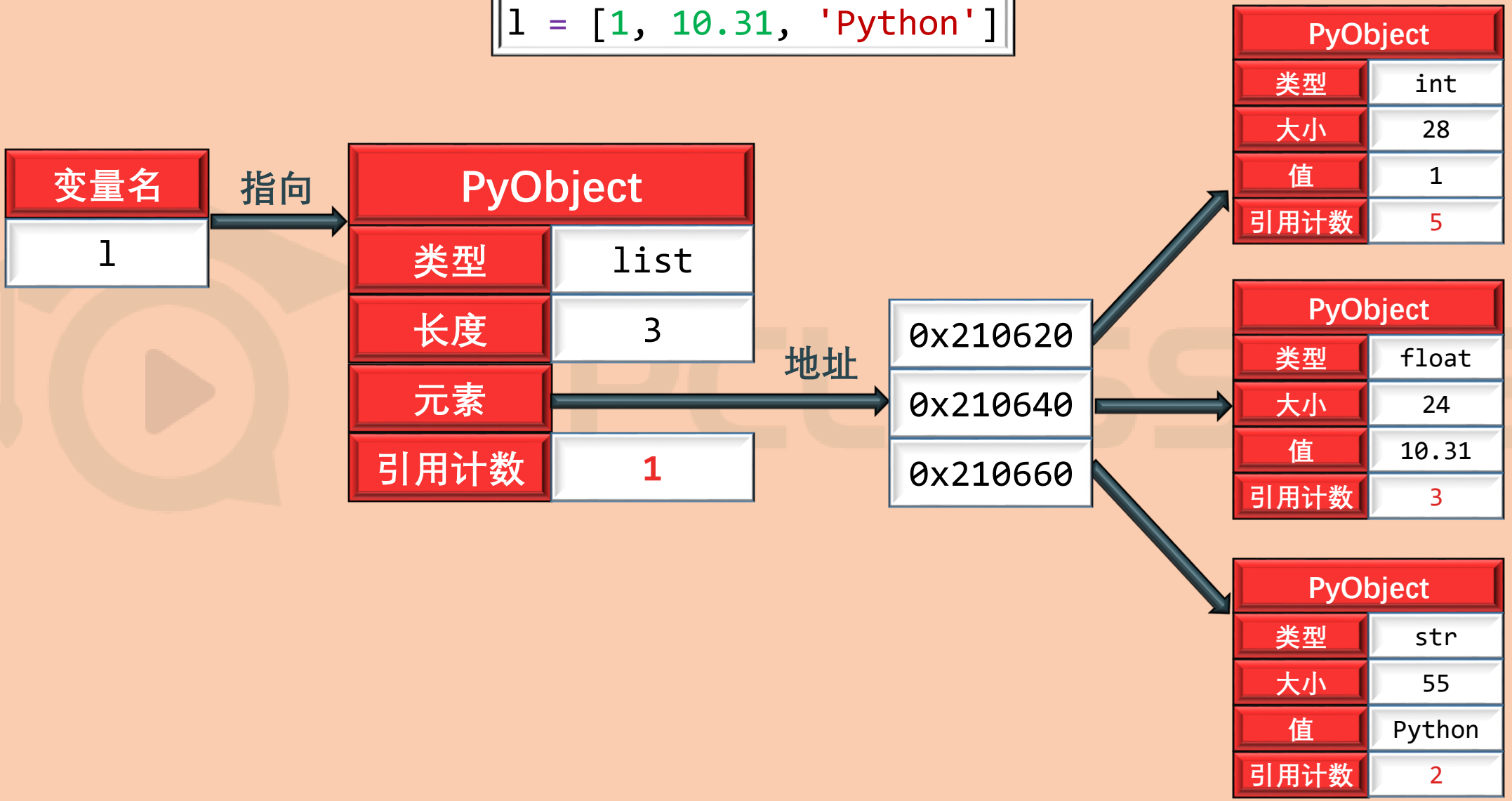
`print()` 函数返回列表的值。

```
print(l)
```

```
[1, 10.31, 'Python']
```

内存中的列表


```
l = [1, 10.31, 'Python']
```



索引和切片

列表索引

列表元素的访问与字符串中字符的访问语法是一样的，使用方括号操作符。方括号内的表达式指定索引位置，而且索引从 0 开始。

列表切片

列表长度 N=len(l)

起始索引　结束索引　步长

```
l[ start : stop : step ]
```

```
l[:] = l[0:N:1]
l[start:] = l[start:N:1]
l[:stop] = l[0:stop:1]
l[start:stop] = l[start:stop:1]
l[::-1] = l[0:N:-1]
```

步长留空 ⇨ step = 1

起始索引留空 ⇨ start = 0

结束索引留空 ⇨ stop = N

切片例子 1

7	2	9	10	1	3	7	2	0	1

```
l[:] = l[0:N:1]
```

```
l[:]
```

7	2	9	10	1	3	7	2	0	1

切片例子 2

7	2	9	10	1	3	7	2	0	1

```
l[start:] = l[start:N:1]
```

`l[3:]`

7	2	9	**10**	**1**	**3**	**7**	**2**	**0**	**1**

`l[-4:]`

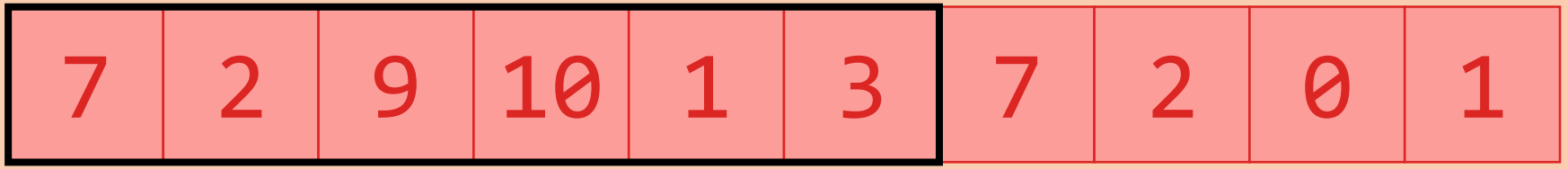
7	2	9	10	1	3	**7**	**2**	**0**	**1**

切片例子 3


```
l[:stop] = l[0:stop:1]
```

```
l[:2]
```

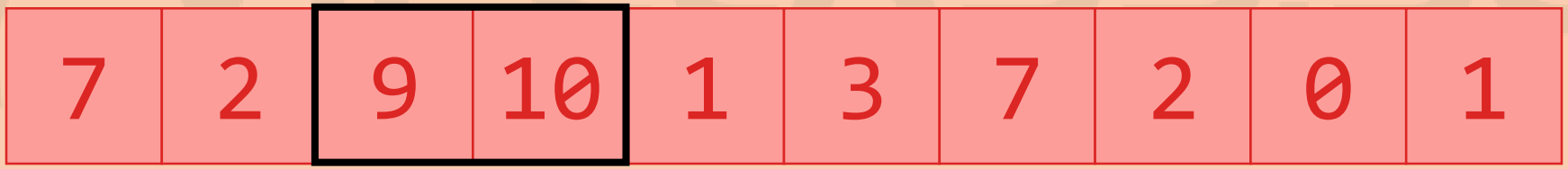

```
l[:-4]
```



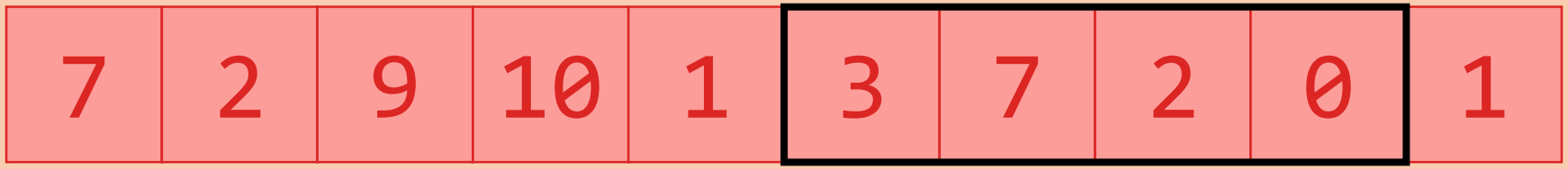
切片例子 4


```
l[start:stop] = l[start:stop:1]
```

```
l[2:4]
```




```
l[-5:-1]
```



切片例子 5


```
l[start:stop:step]
```

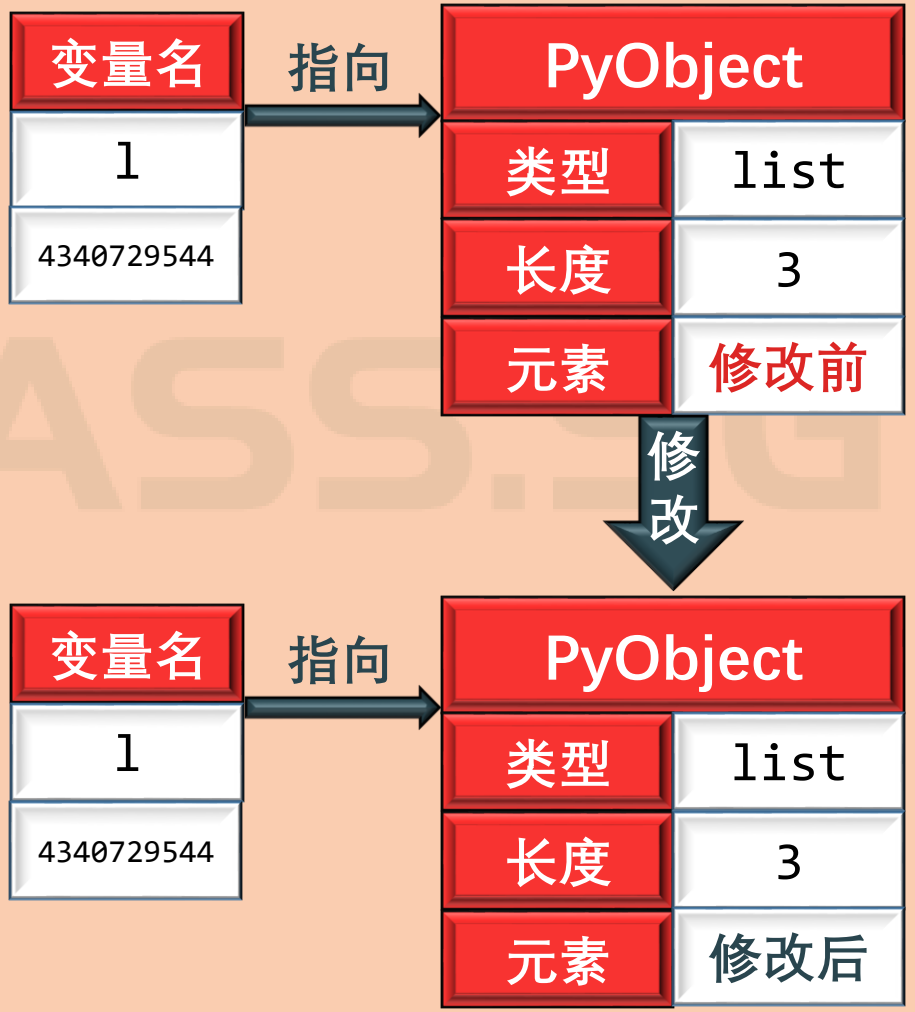
`l[1:5:2]`	7	**2**	9	**10**	1	3	7	2	0	1
`l[:5:2]`	**7**	2	**9**	10	**1**	3	7	2	0	1
`l[1::2]`	7	**2**	9	**10**	1	**3**	7	**2**	0	**1**
`l[::2]`	**7**	2	**9**	10	**1**	3	**7**	2	**0**	1
`l[::-1]`	1	0	2	7	3	1	10	9	2	7

列表可修改 (mutable)

可添加或删减列表中的元素，可改变元素顺序，可对元素重新赋值。

加减排更

附加	l.append(x)	在尾部添加 x 整体
	l.extend(x)	在尾部添加 x 个体
插入	l.insert(i,x)	在索引 i 前插入 x
删除	l.remove(x)	删除元素 x
	l.pop(i)	删除索引 i 的元素
排序	l.sort()	按元素大小排序
	l.reverse()	按索引大小排序
更新	l[i] = x	更新单值 x
	l[i:j] = x	更新多值 x



列表+ 运算符

算术运算符

\+

```
[1, 2] + ['OK']
```

```
[1, 2, 'OK']
```

*

```
['OK'] * 3
```

```
['OK', 'OK', 'OK']
```

比较运算符

== !=

> <

>= <=

```
['Finance', 11] >
['Data', 1234567]
```

```
True
```

成员运算符

in

not in

```
1 in [1, 2]
```

```
True
```

身份运算符

is

is not

```
[1, 3] is not [1, 2]
```

```
True
```

容器型数据

- 字符串
- 列表
- 元组
- 字典
- 集合
- 其他

元组

元组 (tuple) 是存储有序对象的一种容器，其定义语法为

```
t = (元素1, 元素2, ..., 元素n)
```

需要注意的是小括号 () 和逗号 ,

- 小括号把所有元素绑一起
- 逗号将每个元素一一分开

```
t1 = (1, 10.31, 'Python')
```

```
t2 = 1, 10.31, 'Python'
```

```
t3 = tuple((1, 10.31, 'Python'))
```

定义元组甚至可以不需要小括号，或者使用内置函数 tuple()

- t = 元素1, 元素2, ..., 元素n
- t = tuple((元素1, 元素2, ..., 元素n))

元组三件套

内存地址

`id()` 函数返回元组的内存地址。

```
id(t)
```

```
1924399961432
```

类型

`type()` 函数返回元组的类型。

```
type(t)
```

```
<class 'tuple'>
```

值

`print()` 函数返回元组的值。

```
print(t)
```

```
(1, 10.31, 'Python')
```

索引和切片

元组索引

元组元素的访问与列表元素的访问、字符串中字符的访问语法是一样的，使用方括号操作符。方括号内的表达式指定索引位置，而且索引从 0 开始。

元组切片

列表长度 N=len(t)

起始索引　结束索引　步长

```
→t[ start : stop : step ]
```

```
t[:] = t[0:N:1]
t[start:] = t[start:N:1]
t[:stop] = t[0:stop:1]
t[start:stop] = t[start:stop:1]
t[::-1] = t[0:N:-1]
```

步长留空 ⇨ step = 1

起始索引留空 ⇨ start = 0

结束索引留空 ⇨ stop = N

元组不可修改 (immutable)

不可添加或删减元组中的元素，不可改变元素顺序，不可对元素重新赋值。

```
t = (3, 2, 1)
```

```
t[0] = 5
```

```
TypeError: 'tuple' object does not support item assignment
```

```
t.append(0)
```

```
AttributeError: 'tuple' object has no attribute 'append'
```

```
t.remove(2)
```

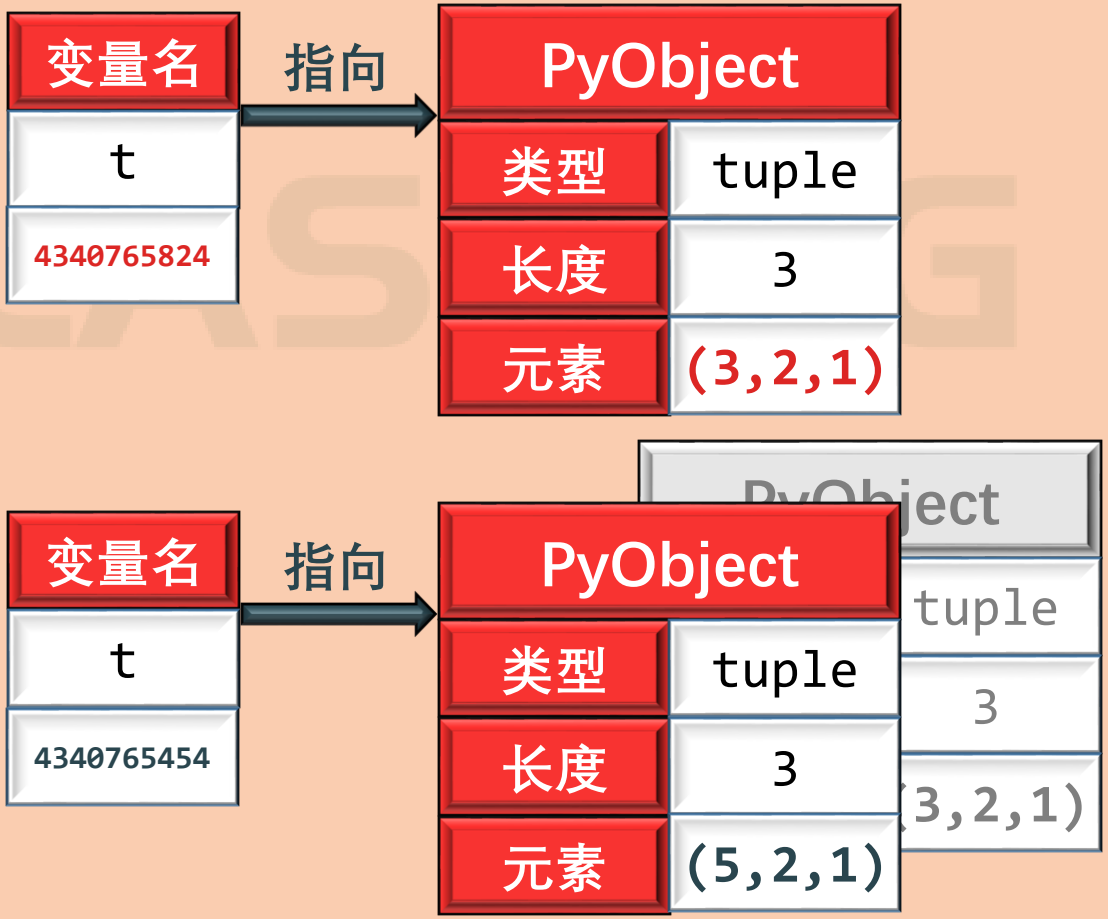
```
AttributeError: 'tuple' object has no attribute 'remove'
```

```
t.sort()
```

```
AttributeError: 'tuple' object has no attribute 'sort'
```

可对元组重新赋值

```
t = (5, 2, 1)
```



元组 + 运算符

算术运算符

\+

```
(1, 2) + ('OK',)
(1, 2, 'OK')
```

*

```
('OK',) * 3
('OK', 'OK', 'OK')
```

比较运算符

== !=

> <

>= <=

```
('Finance', 11) > ('Data', 11)
True
```

成员运算符

in

not in

```
1 in (1, 2)
True
```

身份运算符

is

is not

```
(1, 3) is not (1, 2)
True
```

元组 Vs 列表

运行效率

```
%%timeit
l = []
x = range(10000)
for item in x:
    l.append(item)
```

```
980µs ± 425µs per loop
```

```
%%timeit
t = []
for item in x:
    t = t + (item,)
```

```
142ms ± 3.5ms per loop
```

列表胜

查错难度

```
a = [1, 3, 5, 7]
b = a
b[0] = -10
a
```

```
[-10, 3, 5, 7]
```

元组胜

内存存储

```
from sys import getsizeof
l = list(range(10000))
t = tuple(range(10000))
print( getsizeof(l) )
print( getsizeof(t) )
```

```
90112
80048
```

元组胜

容器型数据

- 字符串
- 列表
- 元组
- 字典
- 集合
- 其他

字典

字典 (dictionary) 是存储无序对象的一种容器，其定义语法为

```
d = {元素1, 元素2, ..., 元素n}
```

映射

其中每一个元素是一个键值对 (key-value pair)，元素i = 键i:值i

需要注意的是大括号 {}，逗号 , 和分号 :

- 大括号把所有元素绑一起
- 逗号将每个元素一一分开
- 分号将键和值分开

定义字典还可使用内置函数 dict()。

字典三件套

内存地址

id() 函数返回字典的内存地址。

```
id(d)
1924357357896
```

类型

type() 函数返回字典的类型。

```
type(d)
str
```

值

print() 函数返回字典的值。

```
print(d)
Python
```

字典 Vs 列表

字典就像列表，前者用键来获取值，后者用数字来获取值。

```
l = list()
l.append('京东')
l.append('JD')
l.append(41)
print(l)
['京东', 'JD', 41]
print(l[2])
41
l[2] = 40
print(l)
['京东', 'JD', 40]
```

```
d = dict()
d['公司'] = '京东'
d['代号'] = 'JD'
d['价格'] = 41
print(d)
{'公司':'京东', '代号':'JD', '价格':41}
print(d['价格'])
41
d['价格'] = 40
print(d)
{'公司':'京东', '代号':'JD', '价格':40}
```

常见方法


```
d = {'名字': '海底捞',
'代号': '06862',
'行业': '食品',
'价格': 29.1,
'单位': 'HKD'}
```

```
d.keys()
d.values()
d.items()
```

索引

```
d['价格']
```

更新

```
d.update({'价格': 30.2,
          '董事长': '张勇'})
```

添加

```
d['国家'] = 中国
```

删除

```
p = d.pop('董事长')
d.clear()
```

获取

```
d.get('代号')
```

容器型数据

- 字符串
- 列表
- 元组
- 字典
- 集合
- 其他

集合

集合 (set) 是存储无序但唯一对象的一种容器，其定义语法为，

```
s = {元素1, 元素2, ..., 元素n}
```

需要注意的是大括号 {} 和逗号 ,

- 大括号把所有元素绑一起
- 逗号将每个元素一一分开

集合的元素是不可修改的

int float bool None tuple

list dict set

此外还可以将 set()函数用在字符串、列表和元组上来创建集合

```
set( string ), set( list ), set( tuple )
```

集合三件套

内存地址

`id()` 函数返回集合的内存地址。

```
id(s)
```

```
623473103144
```

类型

`type()` 函数返回集合的类型。

```
type(s)
```

```
set
```

值

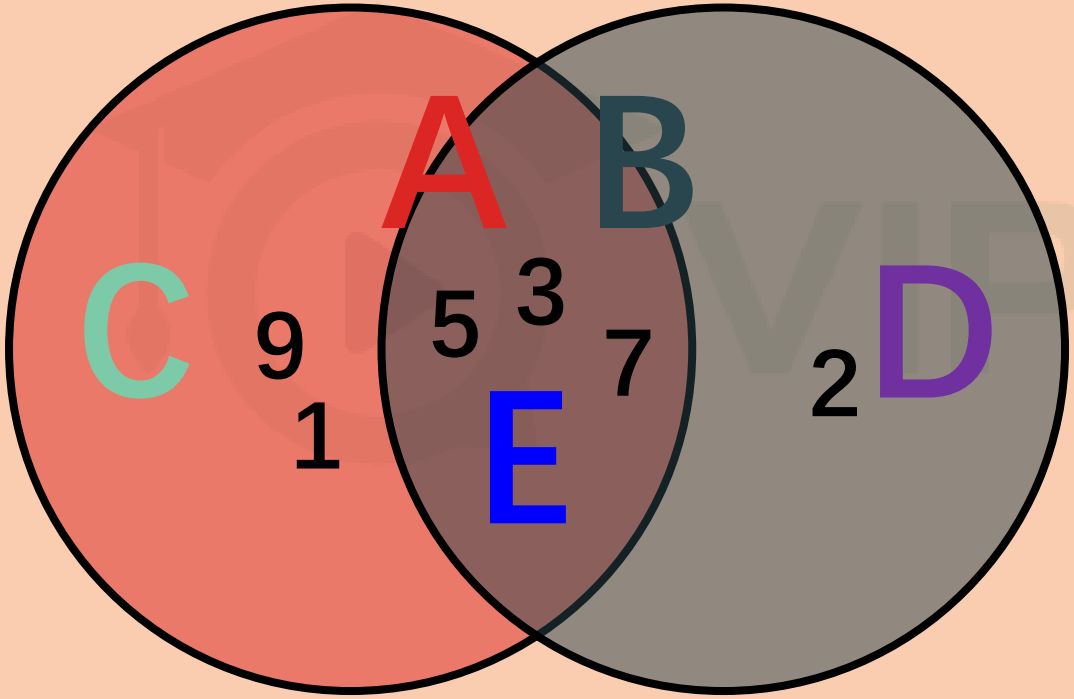
`print()` 函数返回集合的值。

```
print(s)
```

```
{1, 2, 3}
```

集合运算


```
A = {1, 3, 5, 7, 9}
B = {2, 3, 5, 7}
```



并集

```
A.union(B)
A | B
```

C+D+E

交集

```
A.intersection(B)
A & B
```

E

差集

```
A.difference(B)
A - B
```

C

```
B.difference(A)
B - A
```

D

对称差集

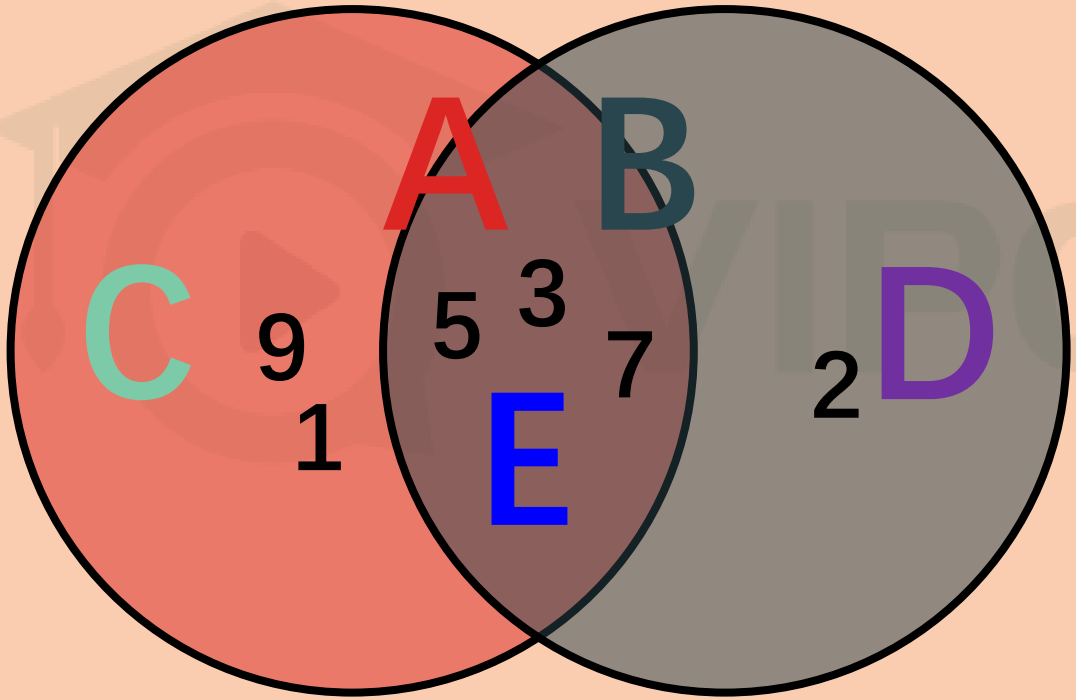
```
A.symmetric_difference(B)
A ^ B
```

C+D

比较运算


```
A = {1, 3, 5, 7, 9}
B = {2, 3, 5, 7}
```

```
C = A - B
D = B - A
```



互斥

```
C.isdisjoint(D)    True
A.isdisjoint(B)    False
```

超集

```
A.issuperset(C)    True
A >= C             True
```

子集

```
D.issubset(B)      True
D <= B             True
```

集合可修改 (mutable)

可更新、添加或删减集合中的元素。

添加

`s.add(x)`	添加 x

删减

`s.remove(x)`	删除元素 x，s 为空报错
`s.discard(x)`	删除元素 x, s 为空不报错
`s.pop()`	删除任意元素
`s.clear()`	删除全部元素

更新

`A.update(B)` `A |= B`	求 A, B 并集并更新 A
`A.symmetric_difference_update(B)` `A ^= B`	求 A, B 对称差集并更新 A
`A.intersection_update(B)` `A &= B`	求 A, B 交集并更新 A
`A.difference_update(B)` `A -= B`	求 A, B 差集并更新 A

容器型数据

- 字符串
- 列表
- 元组
- 字典
- 集合
- 其他

六大常用 Collection

namedtuple
命名元组

赋予每个元素含义，而且元素不可修改

defaultdict
默认字典

字典子类，对于不存在的键，赋个默认值

Counter
计数器

字典子类，提供了可哈希对象的计数功能

deque
双端队列

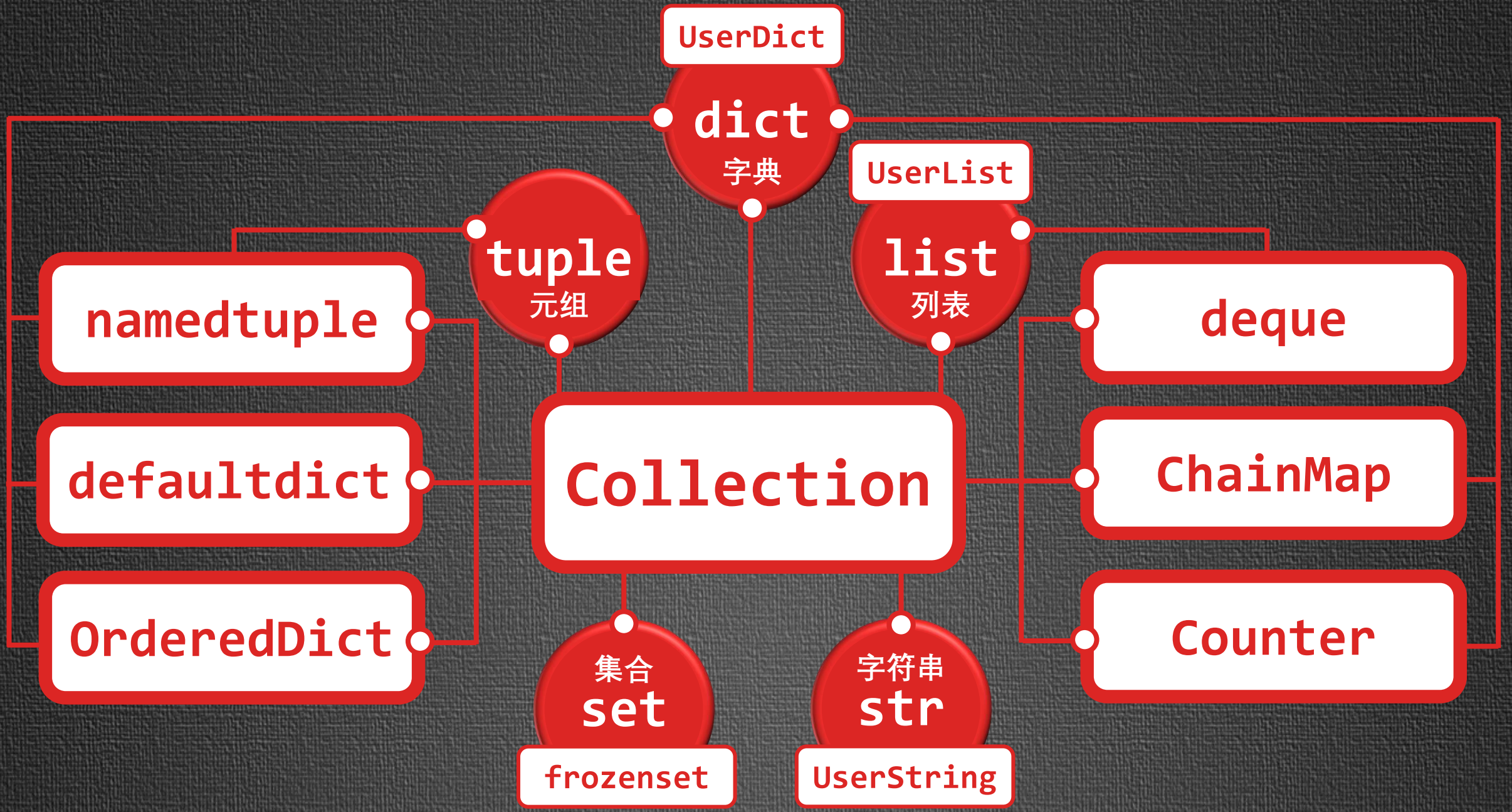
在其尾部和头部进行插入和删除操作

ChainMap
链式映射

字典的容器，在一个单元上操作更为高效

OrderedDict
有序字典

字典子类，记录放入元素的顺序


UserDict
dict
字典
UserList
tuple
元组
list
列表
namedtuple
deque
defaultdict
Collection
ChainMap
OrderedDict
Counter
集合
set
字符串
str
frozenset
UserString

总结

	字符串	列表	元组	字典	集合
创建	'string'	[1, 2]	(1, 2)	{'a':1,'b':2}	{1, 2}
修改?	否	是	否	是	是
重复?	是	是	是	是	否
有序?	是	是	是	否	否
索引?	是	是	是	是	否
切片?	是	是	是	否	否
+,*?	是	是	是	否	否
in?	是	是	是	否	否

下节预告：流程控制


VIPCLASS.SG
small class, world class
终身学习
快乐学习
王圣元 Steven Wang
微信公众号：王的机器