python线性结构和切片

列表， 元组， 字符串， bytes， bytearray

都是顺序存储， 顺序访问的， 都是可迭代对象， 都可以通过索引访问

线性结构

- 可迭代
- len获取长度
- 可以使用下标操作符通过索引访问
- 可以切片

In [1]:

```
for i in [1, 2, 3]:print(i)
```

```
1
2
3
```

In [2]:

```
for i in (1, 2, 3):print(i)
```

```
1
2
3
```

In [3]:

```
for c in 'i love python':print(c)
```

```
i

l
o
v
e

p
y
t
h
o
n
```

In [5]:

```
list(enumerate([1, 2, 3]))
```

Out[5]:

```
[(0, 1), (1, 2), (2, 3)]
```

In [6]:

```
def enumerate(iterator):i = 0for v in iterator:yield i, vi += 1
```

In [7]:

```
def enumerate(iterator):ret = []i = 0for v in iterator:ret.append((i, v))i += 1return ret
```

In [9]:

```
len(range(5)) # 可迭代对象都可以用len获取长度
```

Out[9]:

```
5
```

In [13]:

```
r = range(5)
```

In [14]:

```
list(r)
```

Out[14]:

```
[0, 1, 2, 3, 4]
```

In [16]:

```
it = iter(r)
```

In [17]:

```
type(it)
```

Out[17]:

```
range_iterator
```

In [18]:

```
list(it)
```

Out[18]:

```
[0, 1, 2, 3, 4]
```

In [20]:

```
it = iter(r)
```

In [21]:

```
next(it)
```

Out[21]:

0

In [22]:

```
next(it)
```

Out[22]:

1

In [24]:

```
next(it)
```

Out[24]:

2

In [27]:

```
next(it)
```

```
---------------------------------------------------------------------------
---StopIteration                             Traceback (most recent call
last)
<ipython-input-27-2cdb14c0d4d6> in <module>()----> 1 next(it)StopIterati
on:
```

In [28]:

```
it = iter([1, 2, 3])
```

In [29]:

```
type(it)
```

Out[29]:

list_iterator

切片操作

In [30]:

```
lst = list(range(10))
```

In [31]:

```
lst
```

Out[31]:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [32]:

```
lst[3]
```

Out[32]:

3

lst[start:stop] 可以访问这个list一段， 从start开始， 到stop结束， 不包含stop

In [37]:

```
lst[3:7] # 从索引3开始， 到索引7结束， 不包含7， 返回新的list，不会对原有的list做任何修改
```

Out[37]:

[3, 4, 5, 6]

In [35]:

```
lst
```

Out[35]:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

当start为0时可以省略

In [38]:

```
lst[:4]
```

Out[38]:

[0, 1, 2, 3]

当stop为-0时可以省略

In [39]:

```
lst[3:]
```

Out[39]:

[3, 4, 5, 6, 7, 8, 9]

In [41]:

```
lst[:] # 等效于copy方法
```

Out[41]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [43]:

```
lst[-5:-3] # 支持负数索引
```

Out[43]:

```
[5, 6]
```

In [44]:

```
lst[:100]
```

Out[44]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [45]:

```
lst[-100:]
```

Out[45]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [46]:

```
lst[-100: 100]
```

Out[46]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- start 超出索引范围： start = 0
- stop 超出索引范围： stop = -0

In [48]:

```
lst[100: ] # lst[100: -1]
```

Out[48]:

```
[]
```

In [49]:

```
lst[:-100] # lst[0: -100]
```

Out[49]:

```
[]
```

In [50]:

```
lst[100: -100]
```

Out[50]:

```
[]
```

In [52]:

```
lst[3: 1] # 当start >= stop 时， 返回空列表
```

Out[52]:

```
[]
```

In [53]:

```
lst[3:3]
```

Out[53]:

```
[]
```

In [54]:

```
lst[3: -1]   #负数索引 实际上等于  len(lst) + index   10 + (-1) = 9
```

Out[54]:

```
[3, 4, 5, 6, 7, 8]
```

In [55]:

```
len(lst)
```

Out[55]:

```
10
```

- 负数索引 实际上可以转化为 len(lst) + index
- 当start为0时可以省略 当stop为-0时可以省略
- 当stop <= start时 返回空列表
- 当start超出索引范围 start = 0, 当stop超出索引范围 stop 为-0

In [67]:

```
def slice(lst, start=0, stop=0):if start < 0:start = len(lst) + startif stop <= 0:stop
= len(lst) + stopif stop <= start:return []if stop > len(lst):stop = len(lst)if start
< 0:start = 0ret = []for i, v in enumerate(lst):if i >= start and i < stop:ret.appen
d(v)return ret
```

In [71]:

```
slice(lst, -100, 100)
```

Out[71]:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

lst[start: stop: step] step 参数表示一次增加多少

In [72]:

```
lst[3:8:2]
```

Out[72]:

[3, 5, 7]

In [73]:

```
lst
```

Out[73]:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [74]:

```
lst[8:3:-2]
```

Out[74]:

[8, 6, 4]

In [1]:

```
def slice(lst, start=0, stop=0, step=1):ret = []current = startif step > 0:while curre
nt < stop:try:ret.append(lst[current])except IndexError:passcurrent += stepif step <
0:while current > stop:try:ret.append(lst[current])except IndexError:passcurrent +=
stepreturn ret
```

In [2]:

```
lst = list(range(10))
```

In [3]:

```
lst[8: 5: -1]
```

Out[3]:

[8, 7, 6]

In [4]:

```
slice(lst, 8, 5, -1)
```

Out[4]:

[8, 7, 6]

In [5]:

```
slice(lst, 5, 8, 1)
```

Out[5]:

[5, 6, 7]

In [6]:

```
slice(lst, -10, 20, 1)
```

Out[6]:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [8]:

```
slice(lst, 20, -10, -1)
```

Out[8]:

[9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 9, 8, 7, 6, 5, 4, 3, 2, 1]

In [ ]:

```

```