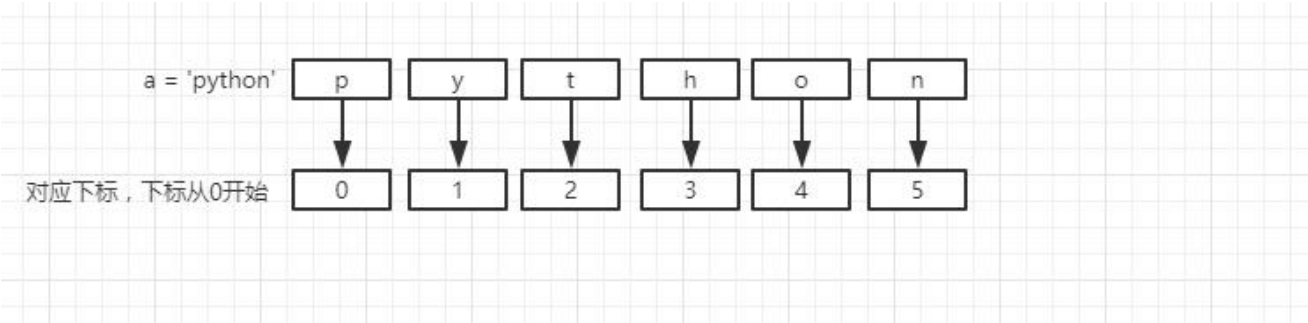
高级数据类型

一、字符串下标与切片

1. 下标索引

生活中我们经常坐大巴车，每个座位一个编号，一个位置对应一个下标。　字符串中也有下标



2. 要取出字符串中的部分数据，可以用下标取。

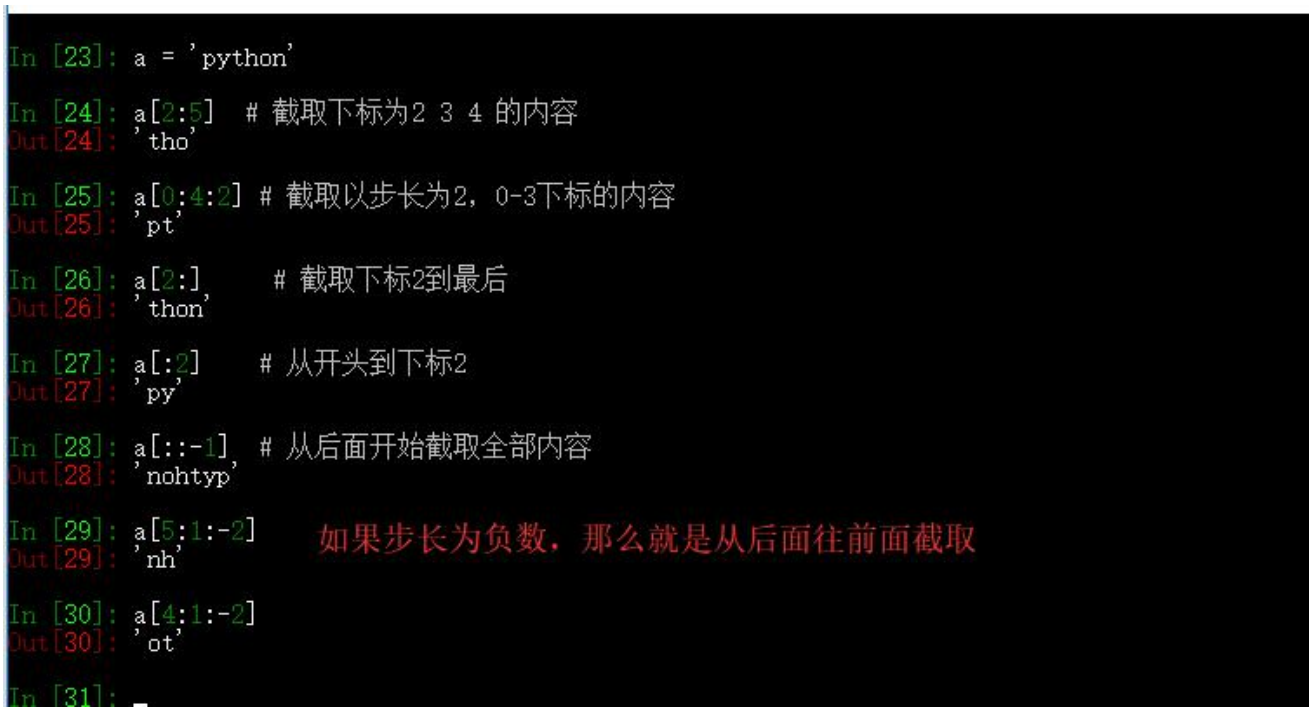
```
>>> a = 'python'
>>> a[0] # 下标0取出p
'p'
>>> a[2]  # 下标2取出t
't'
>>>
```

3. 下标越界

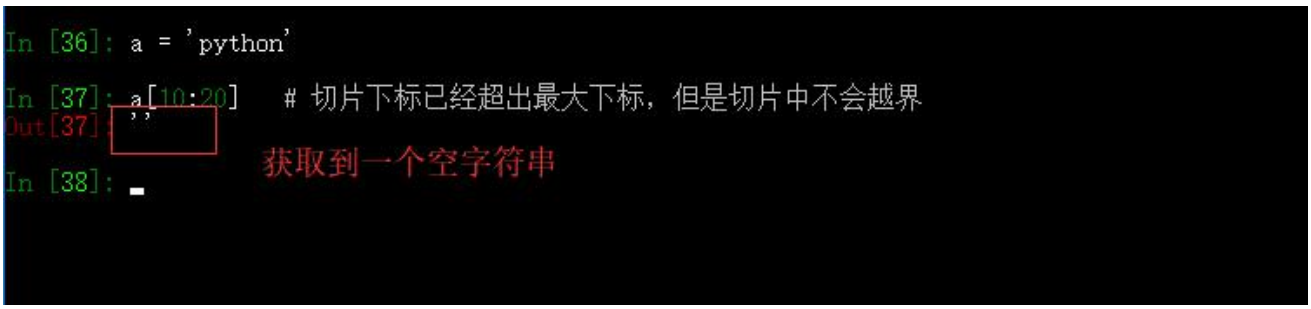
越界是指采用下标获取字符串内容时，超出最大下标。如果下标越界会报错。

4.切片

切片是指截取字符串中的其中一段内容。 切片使用语法：[起始下标：结束下标：步长] 切片截取的内容不包含结束下标对应的数据，步长指的是隔几个下标获取一个字符。



5 切片中是否有越界？



二、字符串常用方法

1、capitalize() # 首字母变大写

```
>>> a = 'hello world'
>>> a.capitalize()
'Hello world'
```

2、endswith()　　# 是否以 xxx 结束

```
>>> a = 'hello world'
>>> a.endswith('ld')   # 以ld结尾，返回True
True
>>> a.endswith('hello') # 不是以hello结尾，返回False
False
>>>
```

3、startswith() # 是否以xxx开始

```
>>> a = 'hello world'
>>> a.startswith('world') # 不是以world开头，返回False
False
>>> a.startswith('hello') # 以hello 开头，返回True
True
>>>
```

4、fi

检测str是否包含在mystr中，如果是返回开始的索引值，否则返回-1

mystr.find(str, start=0, end=len(mystr))

```
>>> mystr = 'hello world,life is short,you need python'
>>> mystr.find('world') # 如果找到返回字符串所在下标位置
6
>>> mystr.find('is',0,5)  #没有找到返回-1
-1
>>> mystr.find('is',0,20)
17
```

5、) 判断字符串是否是字母和数字

```
>>> mystr ='python321'
>>> mystr.isalnum()   # 字符串中只包含数字和字母返回True
True
>>> mystr1 ='python 321'
>>> mystr1.isalnum() # 字符串中有空格，不全是字母或者数字，返回False
False
>>>
```

6、 判断字符串是否是字母

```
>>> mystr= 'abc'
>>> mystr.isalpha()   # 字符串中是纯字母，返回True
True
>>> mystr= 'abc123' # 字符串中不是纯字母，返回Fals
>>> mystr.isalpha()
False
>>>
```

7、 判断字符串是否是数字

```
>>> mystr = 'abc123'
>>> mystr.isdigit()   #   判断字符串是否是纯数字，是返回True,否则返回False
False
>>> mystr = '123'
>>> mystr.isdigit()
True
```

8、islower() #是否是小写

```
>>> mystr= 'Hello'
>>> mystr.islower() # 判断字符串是否是小写，是返回True，否则返回False
False
>>> mystr='hello'
>>> mystr.islower()
True
```

9、　　字符串连接，循环取出字符串的值用　去连接

```
>>> mystr='hello'
>>> '-'.join(mystr)
'h-e-l-l-o'
```

10、　　将字符串中的大写字母全部变成小写的。

```
>>> mystr='Python HELLO world'
>>> mystr.lower()
'python hello world'
```

11、　　将字符串中的小写字母全部变成小写字母

```
>>> mystr='Python HELLO world'
>>> mystr.upper()
'PYTHON HELLO WORLD'
```

12、 小写变大写，大写变小写

```
>>> mystr='Python HELLO world'
>>> mystr.swapcase()
'pYTHON hello WORLD'
```

```
>>> mystr='     python'
>>> mystr.lstrip()
python'
```

13、 lstrip 移除左侧空白

14、rstrip 移除右侧空白

```
>>> mystr='   python     '
>>> mystr.rstrip()
'   python'
>>>
```

15、strip 移除两侧空白

```
>>> mystr='   python     '
>>> mystr.strip()
'python'
```

16、replace(old, new, count=None)：old需要替换的字符串，new替换成这个字符串，count替换多少个。不传count表示全部替换。

```
>>> mystr='my name is lilei'
>>> mystr.replace('lilei','ligang')
'my name is ligang'
```

17、　　切割字符串

```
>>> mystr='my name is lilei'
>>>
>>> mystr.split(" ")
['my', 'name', 'is', 'lilei']
```

18、　　把每个单词的首字母变成大写

```
>>> mystr='my name is lilei'
>>> mystr.title()
'My Name Is Lilei'
```

19、count()　统计出现的次数

```
>>> mystr='my name is lilei'
>>> mystr.count('i')
3
```

三、列表下标与切片

1、列表也是　　的基本数据类型之一。

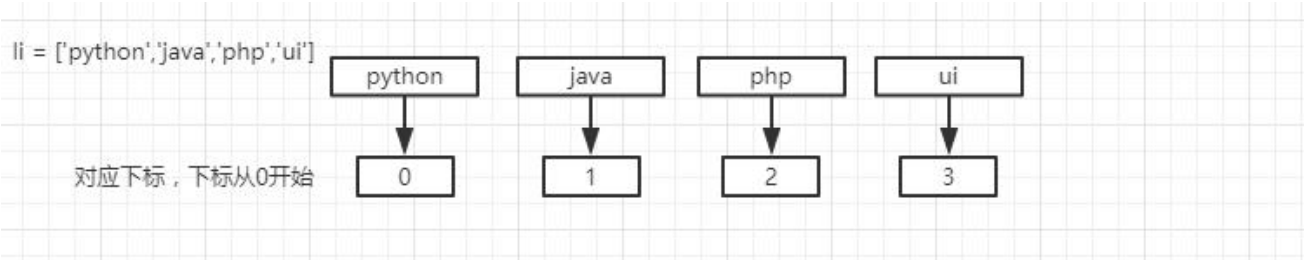
创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来即可。如下所示：

```
>>> li =['python','java','php','ui']
>>> li2 = [1, 2, 'python', 1.3 ,[1,2,3]]
>>>
```

列表可以保存不同的数据类型，字典，列表，元组，字符串，数字都可以。

2、下标索引

列表跟字符串一样，也是由索引的



3、获取　中的元素

```
>>> li =['python','java','php','ui']
>>>
>>> li[0]
'python'
>>> li[1]          # 用下标获取list元素
'java'
>>> li[3]
'ui'
```

4、越界

跟字符串一样，list也是在用下标获取数据的时候超过索引最大值时，会报越界错误

```
>>> li =['python','java','php','ui']
>>> li2 = [1, 2, 'python', 1.3 ,[1,2,3]]
>>>
>>>
>>> li[0]
'python'
>>> li[1]
'java'
>>> li[3]
'ui'
>>>
>>>
>>>
>>>
>>> li[20]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>      #   当使用下标获取元素时，下标超出最大值报越界错误
IndexError: list index out of range
>>>
```

5、切片同样是跟字符串操作一样，区别就是列表是获取元素。

```
>>> li
['python', 'java', 'php', 'ui']
>>>
>>> li[1:3]
['java', 'php']
>>> li[1:]
['java', 'php', 'ui']
>>> li[::-1]
['ui', 'php', 'java', 'python']
```

四、列表循环遍历

昨天讲的for循环时候，提到过可以用for循环遍历获取列表中的元素。

用for循环可将列表中的元素一个一个取出，取完后退出循环。

```
>>> li = ['python', 'java', 'php', 'ui']
>>> for i in li:
...     print(i)
...
python
java
php
ui
>>>
```

使用while循环遍历获取列表元素

```
>>> li = ['python', 'java', 'php', 'ui']
>>> count = len(li)
>>> while i < count:
...     print(li[i])
...     i+=1
...
python
java
php
ui
```

五、列表常用方法

1、 在列表后面追加元素

```
>>> li = ['python','java']
>>> li.append('c++')
>>> li
['python', 'java', 'c++']
>>>
```

2、 统计元素出现的次数

```
>>> li = ['python','java','python','c','c++','linux']
>>> li.count('python')
2
```

3、 扩展，相当于批量添加

```
>>> li =['python','linux']
>>> newli =['java','mysql']
>>> li.extend(newli)
>>> li
['python', 'linux', 'java', 'mysql']
```

4、 获取指定元素索引号

```
>>> li
['python', 'linux', 'java','mysql']
>>> li.index('java')
2
>>> li.index('python')
0
```

5、 在指定位置插入

```
>>> li = ['python', 'linux', 'java', 'mysql']
>>>
>>> li.insert(1,'php')
>>> li
['python', 'php', 'linux', 'java', 'mysql']
>>>
```

6、 删除最后一个元素

```
>>> li = ['python', 'php', 'linux', 'java', 'mysql']
>>>
>>> li.pop()
'mysql'
>>> li
['python', 'php', 'linux','java']
```

7、 移除左边找到的第一个元素

```
>>> li
['python', 'php', 'linux','java']
>>> li.remove('php')
>>> li
['python', 'linux', 'java']
```

8、reverse 反转列表

```
>>> li=['python', 'linux','java']
>>>
>>> li.reverse()
>>> li
['java', 'linux', 'python']
```

9、sort列表排序

```
>>> li = ['python', 'php', 'linux', 'java', 'mysql']
>>> li.sort()
>>> li
['java', 'linux', 'mysql', 'php', 'python']

>>> li.sort(reverse=True)
>>> li
['python', 'php', 'mysql', 'linux', 'java']
```

六、元组

1.Python 元组

> Python的元组与列表类似，不同之处在于元组的元素不能修改。元组使用小括号，列表使用方括号。

2、创建一个元组

```
>>> atuple = ('a','b','c')
>>> atuple
('a', 'b', 'c')
```

3、访问元组内容，跟列表一样使用下标访问

```
>>> atuple = ('a','b','c')
>>> atuple
('a', 'b', 'c')
>>>
>>> atuple[1]  #使用下标获取元祖里的元素
'b'
>>> atuple[3]
Traceback (most recent call last):
File "<stdin>", line 1, in <module>     # 超出最大值报越界错误
IndexError: tuple index out of range
>>>
```

4、元组的内置方法　　，统计元素在元组中出现的次数

```
>>> atuple = ('a','b','c','a')
>>> atuple.count('a')
2
```

5、元组的内置方法　　查找指定元素在元组中的下标索引

```
>>> atuple = ('a','b','c','a')
>>> atuple.index('a')
0
```

七、字典

1 字典的概念：

字典是Python的中重要的一种数据类型，可以存储任意对像。

字典是以键值对的形式创建的{'key':'value'}利用大括号包裹着。

创建字典:

```
mydict ={'key1':'value1','key2':'value2'}
```

2 字典列表相同点和不同点

字典和列表一样，也能够存储多个数据列

表中找某个元素时，是根据下标进行的

字典中找某个元素时，是根据键值（就是冒号:前面的那个值，例如下面代码中的'name','age'）

字典的每个元素由2部分组成，键:值。例如 'name':'xiaoming' ,'name'为键，'xiaoming'为值

```
dic ={'name':'xiaoming','age':18}
```

3 根据键访问值，如果键不存在会报错

```
>>> dic ={'name':'xiaoming','age':18}
>>> dic['name']
'xiaoming'
>>> dic['age']
18
>>> dic['hometown']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'hometown'
>>>
```

4 访问值的安全方式　方法，在我们不确定字典中是否存在某个键而又想获取其值时，可以使用　方法，还可以设置默认值

```
>>> dic ={'name':'xiaoming','age':18}
>>> dic['name']
'xiaoming'
>>> dic['age']
18
>>> dic['hometown']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'hometown'
>>> dic.get('hometown')
>>> dic.get('hometown','shanghai')
'shanghai'
```

5 注意点：

1、字典的键（key）不能重复，值（value）可以重复。

2、字典的键（key）只能是不可变类型，如数字，字符串，元组。

八、字典常用方法

1、修改元素

字典中的值是可以修改的，通过键找到对应值修改

```
>>> info ={'name':'xiaoming','age':18}
>>> info['age'] = 20
>>> info['age']
20
>>> info
{'name': 'xiaoming', 'age': 20}
```

2、新增元素

如果在使用 变量名['键'] = 数据 时，这个“键”在字典中，不存在，那么就会新增这个元素

```
>>> info = {'name': 'xiaoming', 'age': 20}
>>> info['hometown']='广州'
>>> info
{'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
```

3、删除元素

del 删除指定元素 clear 清空字典

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> del info['age']
>>> info
{'name': 'xiaoming', 'hometown':'广州'}
>>> info.clear()
>>> info
{}
```

4、 查看字典中有几个键值对

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> len(info)
3
```

5、keys python3中返回包含字典所有key值的dict_keys对象，用for循环取出每个key值。

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> info.keys()
dict_keys(['name', 'age', 'hometown'])   # python3版本返回一个dict_keys对象，可以迭代取出里面元素
>>> for i in info.keys():
...     print(i)
...
name
age
hometown
>>>

# python2版本直接生成列表

>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> info.keys()
['hometown', 'age', 'name']
```

python2中直接返回一个包含字典所有key值的列表。

6、values 获取字典的所有值（value）

python3版本中使用values返回一个包含所有值（value）的dict_values对象

python2版本中使用values直接返回包含字典中说有值（value）的列表

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> info.values()
dict_values(['xiaoming', 20, '广州'])
>>> for i in info.values():
...     print(i)
...
xiaoming
20
广州



# python2 版本
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> info.values()
['\xe5\xb9\xbf\xe5\xb7\x9e', 20, 'xiaoming']
```

7、items　python2返回一个包含所有（键，值）元组的列表

python3 返回一个包含所有（键，值）元组的列表dict_items对象

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> info.items()
dict_items([('name', 'xiaoming'), ('age', 20), ('hometown', '广州')])


# python2版本
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> info.items()
[('hometown', '\xe5\xb9\xbf\xe5\xb7\x9e'), ('age', 20), ('name', 'xiaoming')]
```

8、pop('name') 删除指定键

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> info.pop('name')
'xiaoming'
>>> info
{'age': 20, 'hometown': '广州'}
```

9、判断一个键是否存在字典中

判断一个键是否存在字典中

python2版本中使用has_key，dict.has_key(key)如果key在字典中，返回True，否则返回False

python3版本中删除了has_key这个方法，使用in代替用来判断key是否存在。

python3版本：

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> if 'name' in info:
...     print('name 存在字典中')
... else:
...     print('name 不存在字典中')
...
name 存在字典中
```

python2版本：

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> info.has_key('name')
True
>>> info.has_key('gender')
False
>>>
```

九、字典的遍历

通过for ... in ...:的语法结构，我们可以遍历字符串、列表、元组、字典等数据结构。

1、遍历字典的所有

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> for i in info.keys():
...     print(i)
...
name
age
hometown
>>>
```

2、遍历字典的左右

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> for i in info.values():
...     print(i)
...
xiaoming
20
广州
>>>
```

3、遍历字典元素

```
>>> info = {'name': 'xiaoming', 'age': 20, 'hometown': '广州'}
>>> for i in info.items():
...     print(i)
...
('name', 'xiaoming')
('age', 20)
('hometown', '广州')
```

4、遍历字典的key-value

```
>>> for key,value in info.items():
...     print(key,value)
...
name xiaoming
age 20
hometown 广州
>>>
```

十、共有操作

1、合并操作 ，适用于字符串，列表，元组

```
>>> str1 = 'hello'
>>> str2 = 'world'
>>> str1 + str2   # 字符串
'helloworld'

>>> [1,2,3] + [4,5,6]   # 列表
[1, 2, 3, 4, 5, 6]

>>> ('a','b')+('c','d')  #元祖
('a', 'b', 'c', 'd')
>>>
```

2、 复制 适用于字符串、列表、元组

```
>>> str1 = 'hello'
>>> str1*3
'hellohellohello'   # 字符串
>>> [1,2,3]*3
[1, 2, 3, 1, 2, 3, 1, 2, 3] # 列表
>>> (1,2,3)*3
(1, 2, 3, 1, 2, 3, 1, 2, 3) # 元祖
```

3、 判断元素是否存在 适用于 字符串、列表、元组字典

```
>>> 'my' in 'my name is lilei'
True
>>> 1 in [1,2,3]
True
>>> 'age' in{'name':'xiaoming','age':18}
True
```