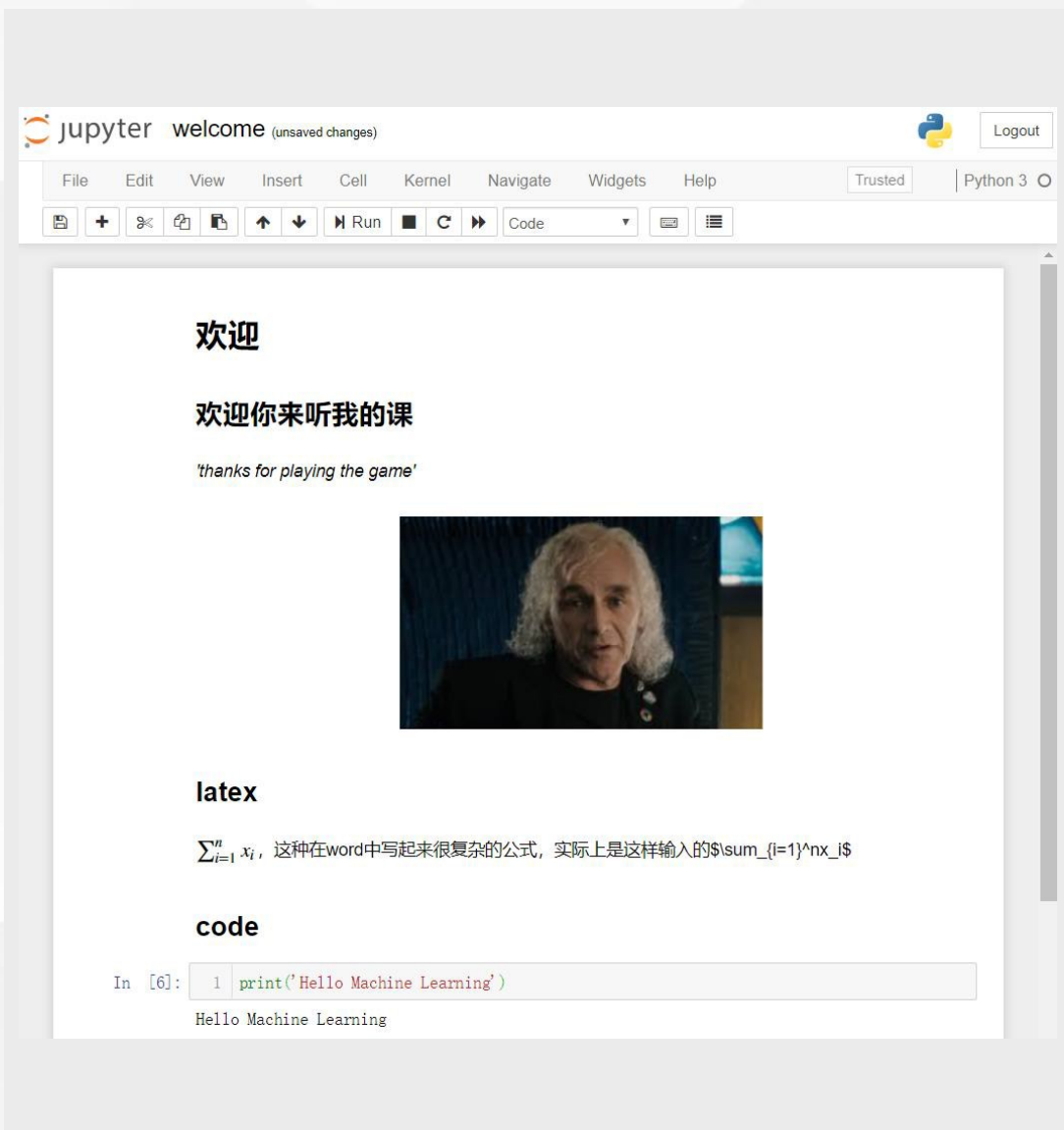


3.集合

Jupyter简介



Jupyter简介

通俗的说，jupyter就是在浏览器上运行的，可以跑代码的记事本。它具有以下有优点：

支持代码种类繁多

轻量化的编辑器，安装简单

支持markdown语法

支持latex语法

逐代码块执行，完美搭配机器学习

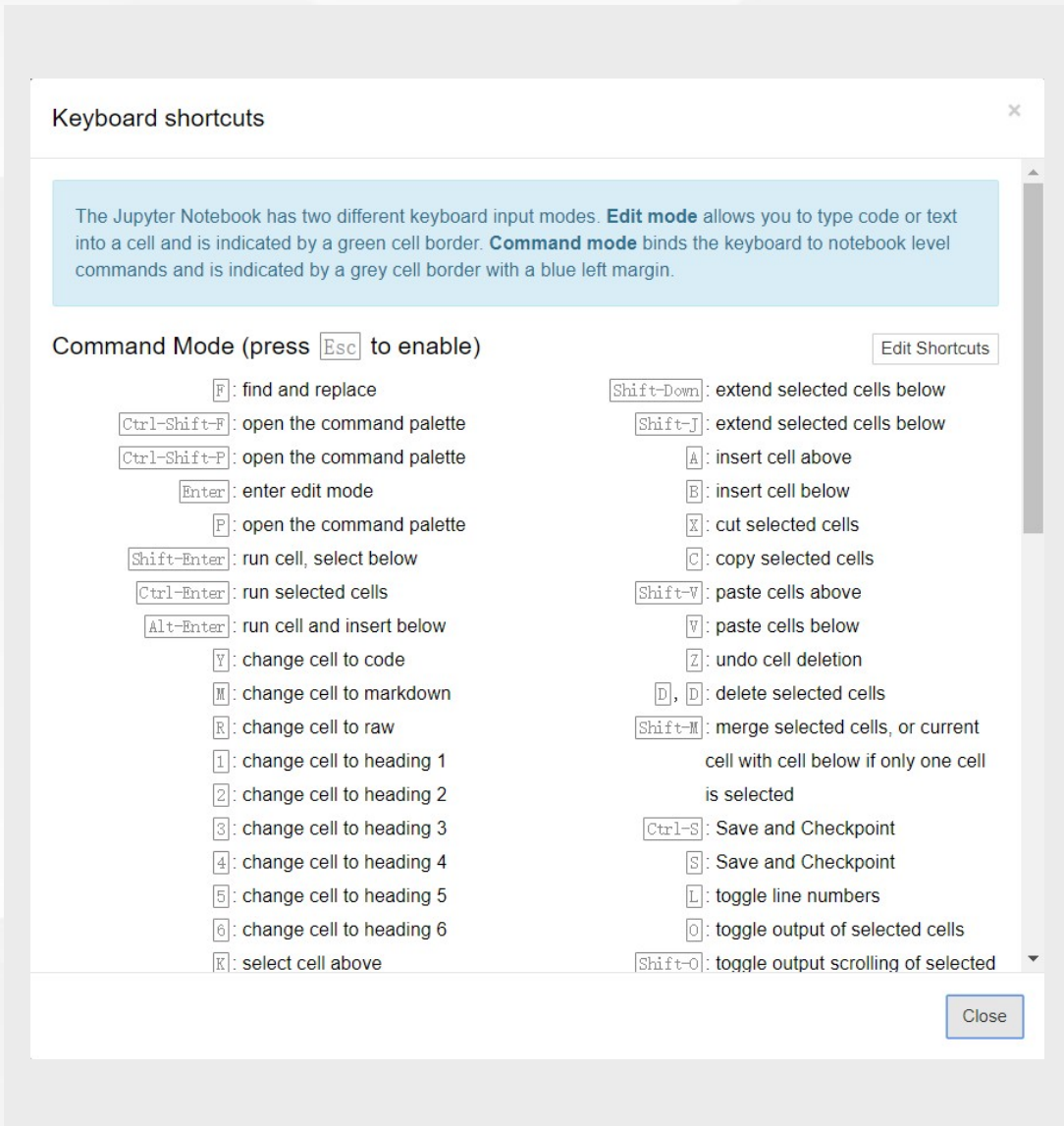
Jupyter的缺点之一是不方便转换为word格式，但是可以转换为html、pdf、md等格式。

Jupyter简介

Jupyter快捷键

Jupyter的大部分快捷键，都需要在非编辑模式下输入，按 h 可以查看快捷键帮助，常用快捷键如下：

- 编辑模式下ctrl+enter运行当前cell
- 编辑模式下shift+enter运行当前cell并移动到下个cell
- a，在当前cell前插入一个cell
- b，在当前cell后插入一个cell
- dd，删除当前cell
- m，将当前cell转化为markdown
- y，将当前cell转化为code（cell默认为code）
- z，撤销上一次对cell的操作



The screenshot shows the 'Keyboard shortcuts' dialog box in Jupyter. It has a title bar 'Keyboard shortcuts' with a close button. Below the title bar is a light blue informational box with text: 'The Jupyter Notebook has two different keyboard input modes. **Edit mode** allows you to type code or text into a cell and is indicated by a green cell border. **Command mode** binds the keyboard to notebook level commands and is indicated by a grey cell border with a blue left margin.' Below this box, the text 'Command Mode (press `Esc` to enable)' is followed by an 'Edit Shortcuts' button. The main area contains two columns of keyboard shortcuts. The left column includes: `F`: find and replace; `Ctrl-Shift-F`: open the command palette; `Ctrl-Shift-P`: open the command palette; `Enter`: enter edit mode; `P`: open the command palette; `Shift-Enter`: run cell, select below; `Ctrl-Enter`: run selected cells; `Alt-Enter`: run cell and insert below; `Y`: change cell to code; `M`: change cell to markdown; `R`: change cell to raw; `1`: change cell to heading 1; `2`: change cell to heading 2; `3`: change cell to heading 3; `4`: change cell to heading 4; `5`: change cell to heading 5; `6`: change cell to heading 6; `K`: select cell above. The right column includes: `Shift-Down`: extend selected cells below; `Shift-J`: extend selected cells below; `A`: insert cell above; `B`: insert cell below; `X`: cut selected cells; `C`: copy selected cells; `Shift-V`: paste cells above; `V`: paste cells below; `Z`: undo cell deletion; `D, D`: delete selected cells; `Shift-M`: merge selected cells, or current cell with cell below if only one cell is selected; `Ctrl-S`: Save and Checkpoint; `S`: Save and Checkpoint; `L`: toggle line numbers; `O`: toggle output of selected cells; `Shift-O`: toggle output scrolling of selected. At the bottom right is a 'Close' button.

Keyboard shortcuts

The Jupyter Notebook has two different keyboard input modes. **Edit mode** allows you to type code or text into a cell and is indicated by a green cell border. **Command mode** binds the keyboard to notebook level commands and is indicated by a grey cell border with a blue left margin.

Command Mode (press `Esc` to enable)

Edit Shortcuts

`F`: find and replace
`Ctrl-Shift-F`: open the command palette
`Ctrl-Shift-P`: open the command palette
`Enter`: enter edit mode
`P`: open the command palette
`Shift-Enter`: run cell, select below
`Ctrl-Enter`: run selected cells
`Alt-Enter`: run cell and insert below
`Y`: change cell to code
`M`: change cell to markdown
`R`: change cell to raw
`1`: change cell to heading 1
`2`: change cell to heading 2
`3`: change cell to heading 3
`4`: change cell to heading 4
`5`: change cell to heading 5
`6`: change cell to heading 6
`K`: select cell above

`Shift-Down`: extend selected cells below
`Shift-J`: extend selected cells below
`A`: insert cell above
`B`: insert cell below
`X`: cut selected cells
`C`: copy selected cells
`Shift-V`: paste cells above
`V`: paste cells below
`Z`: undo cell deletion
`D, D`: delete selected cells
`Shift-M`: merge selected cells, or current cell with cell below if only one cell is selected
`Ctrl-S`: Save and Checkpoint
`S`: Save and Checkpoint
`L`: toggle line numbers
`O`: toggle output of selected cells
`Shift-O`: toggle output scrolling of selected

Close

Jupyter简介

一级标题

二级标题

三级标题

这是一段斜体文字

这是一段加粗文字

这是一段斜体加粗文字

~~这是删除线~~

```
import numpy as np
```

- 无序列表1
- 无序列表2

1. 有序列表
2. 有序列表

thanks for play the game

[这是某网站的超链接](#)

MarkDown

常用markdown语法如下：

一级标题

斜体

****加粗****

******撕佐功夫******

~~删除线~~

- 无序列表

1. 有序列表

> 引用

`code`

![图片注释](图片地址)

[超链接名](超链接地址)

注：markdown不支持直接输入换行、空格

Jupyter简介

x_1^2 : `x_1^2`

$\sum_{i=1}^n x_i^2$: `$\sum_{i=1}^n x_i^2$`

$x \geq y \leq z \neq s \times t$: `$x \geq y \leq z \neq s \times t$`

$x \in y \notin z \cap s \cup t$: `$x \in y \notin z \cap s \cup t$`

$\alpha\beta\gamma\sigma\delta\epsilon\Delta$: `$\alpha \beta \gamma \sigma \delta \epsilon \Delta$`

$\frac{\partial y}{\partial x}$: `$\frac{\partial y}{\partial x}$`

$Loss = (\hat{y} - y)^2 \times \sqrt{y}$: `$Loss=(\hat{y}-y)^2 \times \sqrt{y}$`

$\vec{a} \mathcal{XY} \mathbf{X}$: `$\vec{a} \mathcal{XY} \mathbf{X}$`

Latex

Latex是一种排版系统，尤其擅长公式排版，

latex公式需要使用\$将公式内容包围起来，两个\$表示居中

x_1 表示下标 x_1 ， x^2 表示上标 x^2 ，

`\neq`表示 \neq ， `\geq`表示 \geq ， `\leq`表示 \leq ， `\times`表示 \times

`\in`表示 \in ， `\notin`表示 \notin ， `\cap`表示 \cap ， `\cup`表示 \cup

`\alpha`表示 α ， `\delta`表示 δ ， `\phi`表示 ϕ ， `\Delta`表示 Δ

`\hat{y}`表示 \hat{y} ， `\bar{y}`表示 \bar{y} ， `\sqrt{y}`表示 \sqrt{y}

`\vec{a}`表示 \vec{a} ， `\mathcal{X}`表示 \mathcal{X} ， `\mathbf{X}`表示 \mathbf{X}

本课程目标



掌握List的常用方法



掌握Tuple的常用方法



掌握Set集合的常用方法



掌握Dictionary的常用方法



掌握集合的高级应用

课程目录

Course catalogue

1/ 列表

2/ 元组

3/ Set集合

4/ 字典

5/ 高级应用

创建列表

```
1 list = [1,2,3,4,5]
2
3 print(list)
```

列表是python以及其他语言中最常用到的数据结构之一。

Python中使用[]来创建列表

遍历列表

```
1  list = [1,2,3,4,5]
2  print(list)
3  for ele in list:
4      print(ele)
5
6  index = 0
7  while index < len(list):
8      print(list[index])
9      index += 1
10
```

使用循环来遍历一个列表，可以通过foreach循环，也可以循环生成索引来访问列表中的每一个元素

获取元素

```
1 list = [1, 2, 3, 4, 5]
2
3 print(list[0])
4 print(list[3])
5 #print(list[5])
6
7 print(list[-2])
8 print(list[-5])
9 #print(list[-6])
10
```

通过索引即可以获取列表中的相应位置的元素，索引由0开始且可以为负数。

当索引为整数时，索引的最大值不能超过列表的长度-1。

当索引为负数时，索引的最小值不能超过负的列表长度

新增元素

```
1  list = [1,2,3,4,5]
2
3  print(id(list))
4  list.append(4)
5
6  print(list)
7  print(id(list))
8
9  list.insert(3,9)
10 print(list)
11 print(id(list))
12
```

list提供了`append`方法，可以方便的向列表中追加元素。若需要向指定位置添加元素，可以使用列表的`insert`方法，传递索引以及待插入的元素作为参数。新增元素后的列表依然为原列表

修改元素

```
1 list = [1,2,3,4,5]
2
3 print(id(list))
4
5 list[3] = 8
6 print(list)
7 print(id(list))
8
```

list中的元素可以进行修改，只需通过索引获取到待修改的元素，为其重新赋值即可。修改后的列表依然为原列表

删除元素

```
1  list = [1,2,3,4,5]  
2  
3  print(id(list))  
4  
5  # 3是index  
6  del list[3]  
7  print(list)  
8  print(id(list))  
9  
10 #3是元素  
11 list.remove(3)  
12 print(list)  
13 print(id(list))  
14  
15 list.remove(4)  
16
```

list中的元素可以进行删除，只需通过索引获取到待删除的元素，即可使用del关键字进行删除。或者可以使用列表的remove方法实现删除，remove方法接收一个元素，若该元素存在于列表中则在调用remove方法时会被删除，否则抛出异常。
删除元素后的列表依然为原列表

查找元素

```
1 list = [1,2,3,4,5]
2
3 print(list[4])
4
```

list中的元素可以通过列表提供的index方法进行查找

拼接列表

```
1 listA = [1,2,3,4,5]
2 listB = [4,3,2,1]
3
4 print(listA+listB)
5
6 listA.extend(listB)
7 print(listA)
8
```

当有两个列表需要进行组合时，
可以通过加法运算进行拼接，
或者通过列表的extend方法实现拼接

获取元素出现次数

```
1 tupleA = [1,2,3,4,5,3]
2
3 print(tupleA.count(3))
4 print(tupleA.count(5))
5
```

列表提供了count方法，该方法可以返回传入的元素参数在列表中出现的次数

排序与反转

```
1  list = [3,9,5,1,2,7]
2
3  list.reverse()
4  print(list)
5
6  list = sorted(list)
7  print(list)
8
9  list.sort(reverse=True)
10 print(list)
11
```

通过sorted或列表提供的sort方法可以实现升序排序，如需进行降序排序，可以向sort方法中传入参数reverse
反转列表则通过列表的reverse方法实现

支持的内置函数操作

```
1  list = [1,2,3,4,5]
2
3  print(len(list))
4  print(max(list))
5  print(min(list))
6  print(sum(list))
7
```

除了列表本身提供的方法外，
列表还支持其它的内置函数操作

课程目录

Course catalogue

1/ 列表

2/ 元组

3/ Set集合

4/ 字典

5/ 高级应用

创建元组

```
1 tupleA = (1,2,3,4,5)
2 tupleB = (1,)
3 tupleC = ()
4
5 print(tupleA)
6 print(tupleB)
7 print(tupleC)
```

Python中使用()来创建元组，
若创建元素个数为1的元组，则
需要在唯一的元素后面增加一个逗号

遍历列表

```
1 tuple = (1,2,3,4,5)
2
3 for ele in tuple:
4     print(ele)
```

可以通过foreach循环遍历一个元组

查找元素

```
1 tupleA = (1,2,3,4,5)
2
3 #查找元素所在的位置
4 print(tupleA.index(2))
```

元组可以通过index函数查找元素所在位置

获取元素

```
1 tupleA = (1,2,3,4,5)
2
3 print(tupleA[0])
4 print(tupleA[3])
5 print(tupleA[-5])
6
7 #print(tupleA[5])
8 #print(tupleA[-6])
```

通过索引即可以获取元组中的相应位置的元素，索引由0开始且可以为负数。

当索引为整数时，索引的最大值不能超过列表的长度-1。

当索引为负数时，索引的最小值不能超过负的列表长度

修改元素

```
1 tupleA = (1,2,3,4,5)
2
3 tupleA[0] = 0
```

```
tupleA[0] = 0
TypeError: 'tuple' object does not support item assignment
```

元组为不可变类型，元组中的元素一旦定义不能修改

删除元素

```
1 tupleA = (1,2,3,4,5)
2
3 del tupleA[3]
```

```
TypeError: 'tuple' object doesn't support item deletion
```

元组为不可变类型，元组中的元素一旦定义不能删除

拼接元组

```
1 tupleA = (1,2,3,4,5)
2 tupleB = (3,4,5)
3
4 tupleC = tupleA + tupleB
5 print(tupleC)
6
```

当有两个元组需要进行组合时，
可以通过加法运算进行拼接

排序

```
1 tupleA = (1,5,4,2,3)
2
3 print(sorted(tupleA))
4
```

通过sorted可以实现对元组的排序，排序后的返回结果为列表

获取元素出现次数

```
1 tupleA = (1,5,3,2,3)
2
3 print(tupleA.count(3))
4
```

元组提供了count方法，该方法可以返回传入的元素参数在元组中出现的次数

支持的内置函数操作

```
1 tupleA = (1,5,3,2,3)
2
3 print(len(tupleA))
4 print(max(tupleA))
5 print(min(tupleA))
6 print(sum(tupleA))
```

除了元组本身提供的方法外，列表还支持其它的内置函数操作

课程目录

Course catalogue

1/ 列表

2/ 元组

3/ Set集合

4/ 字典

5/ 高级应用


创建Set集合

```
1 setA = set((1,2,3,4))
2 setB = set([1,2,3,4])
3 setC = {1,2,3,4}
4
5 print(setA)
6 print(setB)
7 print(setC)
8
```

```
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4}
```

Set集合可以通过set函数将元组或列表进行数据类型转换，实现创建，也可以直接通过{}进行创建

遍历Set集合

```
1 setA = set((1,2,3,4))
2
3 for i in setA:
4     print(i)
5 
```

可以通过foreach循环遍历一个Set集合

新增元素

```
1 setA = set((1,2,3,4))
2
3 setA.add(6)
4 setA.update((7,8,9))
5 print(setA)
```

可以使用add以及update方法
向集合中添加元素

删除元素

```
1 setA = {1,2,3,4,5}
2
3 setA.remove(3)
4 print(setA)
5 print(setA.pop())
6 setA.discard(5)
7 setA.discard(6)
8 print(setA)
9 #setA.remove(6)
10
```

Set集合中的元素可以使用remove方法实现删除，除此之外，还支持通过pop方法以及discard方法删除元素。remove与discard的区别在于，当待删除元素不存在与集合中时，remove方法会抛出异常，而discard方法将不做任何处理。

交集、并集、差集

```
1 setA = {1,2,3,4,5}
2 setB = {2,3,5}
3
4 print(setA.intersection(setB))
5 print(setA.union(setB))
6 print(setA.difference(setB))
```

通过intersection方法可以获取两个集合的交集

通过union方法可以获取两个集合的并集

通过difference方法可以获取两个集合的差集

课程目录

Course catalogue

1/ 列表

2/ 元组

3/ Set集合

4/ 字典

5/ 高级应用

创建字典

```
1
2 dictA = {'name': 'tom', 'age': 10, 'weight': 8}
3 dictB = {}
4
5 print(dictA, dictB)
6
```

字典由键值对元素组成，字典可以使用{}来定义，字典中的每个元素key与value间使用:进行分隔

获取元素值

```
1 dictA = {'name': 'tom', 'age': 10, 'weight': 8}
2
3
4 print(dictA['name'])
5 print(dictA.get('name'))
6
```

字典中的元素值，可以通过[]
加上待获取的值所对应的key
进行取值，也可以通过get方法
进行取值

获取key、value、item集合


```
1
2 dictA = {'name': 'tom', 'age': 10, 'weight': 8}
3
4 print(dictA.keys())
5 print(dictA.values())
6 print(dictA.items())
-
```

```
dict_keys(['name', 'age', 'weight'])
dict_values(['tom', 10, 8])
dict_items([('name', 'tom'), ('age', 10), ('weight', 8)])
```

```
Process finished with exit code 0
```

字典中可以通过keys、values以及items分别获取元素的key集合、元素的value集合以及键值对集合

添加、修改元素

```
1 dictA = {'name': 'tom', 'age': 10, 'weight': 8}
2
3
4 dictA['height'] = 20
5 dictA['weight'] = 10
6 
7 print(dictA)
```

可以直接通过向字典中不存在的key赋值，实现添加元素，当key存在时，则赋值操作将修改原key对应的value值

删除元素

```
1 dictA = {'name': 'tom', 'age': 10, 'weight': 8}
2
3
4 print(dictA.popitem())
5 print(dictA)
6
7 print(dictA.pop('age'))
8 print(dictA)
9
10 del dictA['name']
11 print(dictA)
```

使用pop以及popitem方法可以实现移除元素，pop方法需要指定待移除元素的key值，popitem方法则默认移除最后一个元素
除此之外，还可以使用del关键字实现元素的删除操作

课程目录

Course catalogue

1/ 列表

2/ 元组

3/ Set集合

4/ 字典

5/ 高级应用

切片

```
2 mylist = [1,2,3,4,5,6,7]
3
4 print(mylist[1:4])
5 print(mylist[1:])
6 print(mylist[:5])
7 print(mylist[1:6:2])
8 print(mylist[::-1])
```

切片是一种快速截取序列中部分元素的方法。切片可以通过指定的区间，将待截取元素以列表形式返回，也可以通过指定步长跳跃性的截取元素

生成器

```
#用列表生成式, 生成一个列表 []
```

```
L = [x*x for x in range(10)]
```

```
L
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
#用列表生成式, 生成一个生成器, generator:
```

```
#只要把一个列表生成式的[]改成()
```

```
g = (x*x for x in range(5))
```

```
#g中保存的是算法, 每次调用next(), 计算出g下一个元素值
```

```
print(next(g))
```

```
#生成器也是可迭代对象, 可以用for
```

```
for n in g:
```

```
    print(n)
```

```
0
```

```
1
```

```
4
```

```
9
```

```
16
```

在Python中, 一边循环一边计算的机制, 称为生成器: generator。

generator非常强大。若推算的算法比较复杂, 用类似列表生成式的for循环无法实现的时候, 还可以用函数来实现。

生成器

```
1 def fib(max):  
2     n,a,b = 0,0,1  
3     while n < max:  
4         yield b #print(b) 函数变成生成器  
5         a,b = b,a+b  
6         n = n+1  
7     return 'done'  
8  
9 f = fib(10)  
10 print(next(f),end=' ')  
11 print(next(f),end=' ')  
12  
13 for i in f:  
14     print(i,end=' ')
```

生成器是迭代器的一种

如果一个函数定义中包含**yield**关键字，那么这个函数就不再是一个普通函数，而是一个generator。

而变成generator的函数，在每次调用**next()**的时候执行，遇到**yield**语句返回，再次执行时从上次返回的yield语句处继续执行。