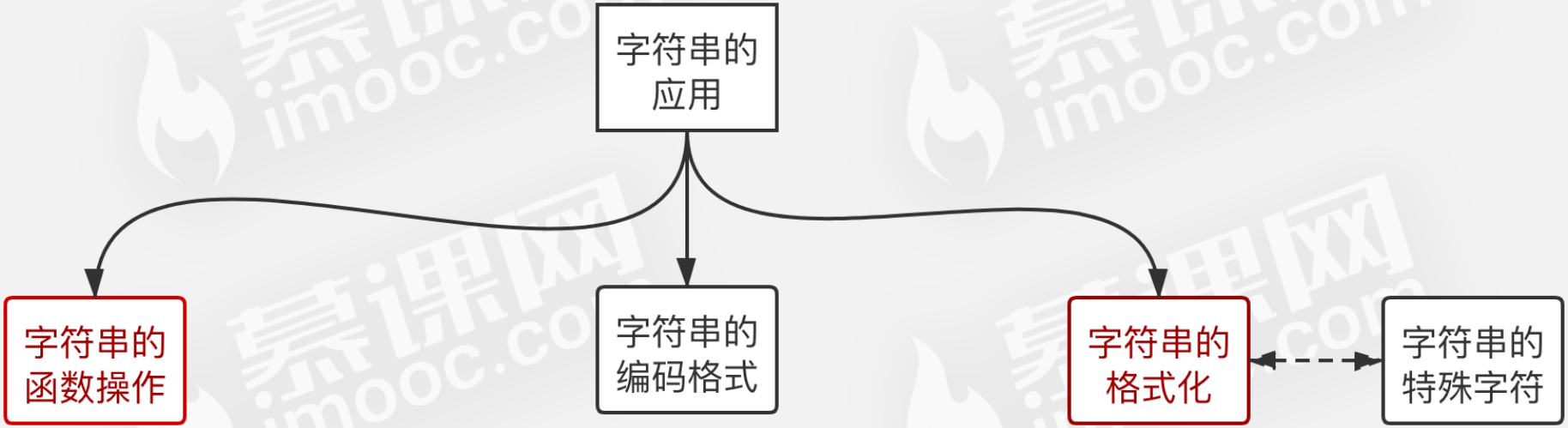
# 三大主流数据类型的操作

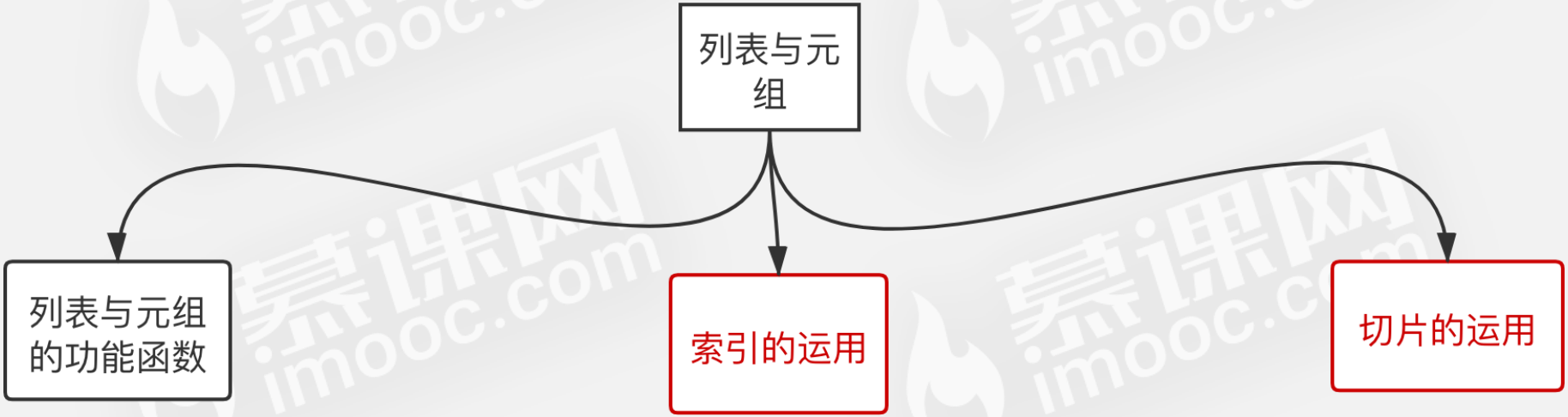
# 本周内容

- Python的对象
- 字符串操作以及他的内置函数
- 列表（元组）的操作以及他的内置函数
- 字典的操作以及他的内置函数

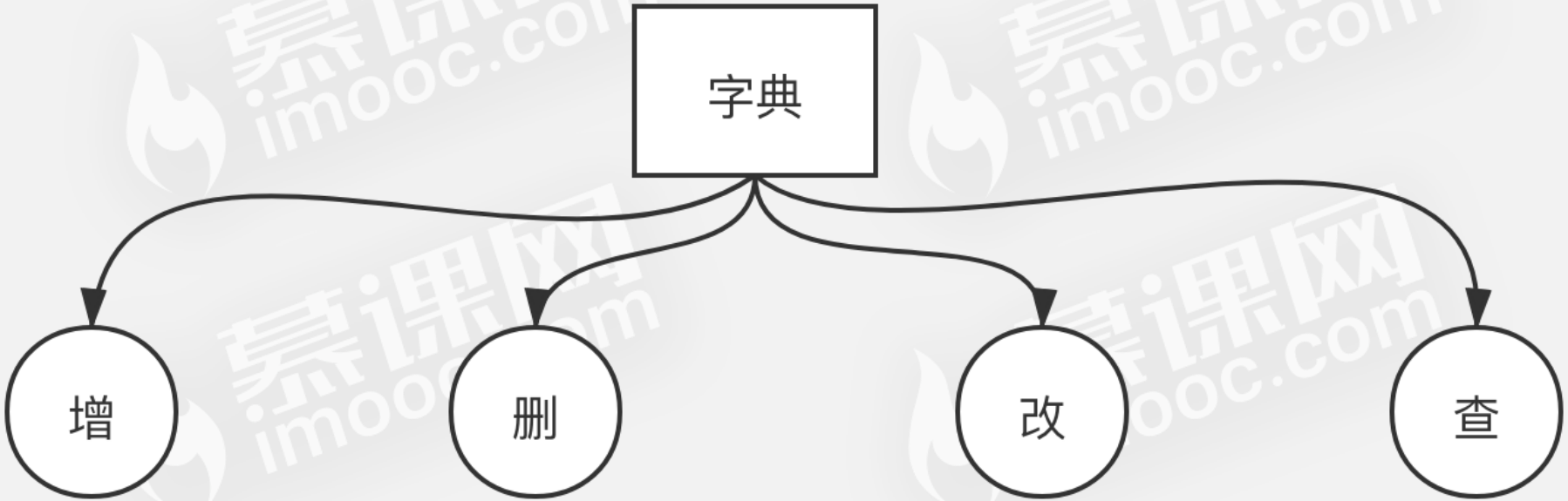
# Python字符串操作与内置函数



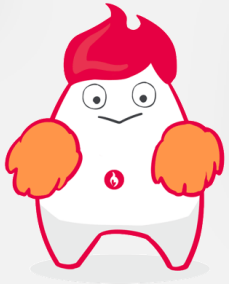
# 列表（元组）操作与内置函数



# 字典操作与内置函数



# 认识python中的对象



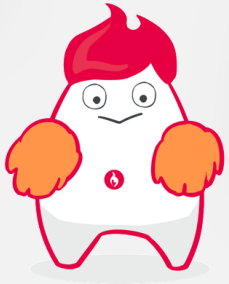
# 本节课内容

- 什么是对象

# 什么是对象

- Python中一切都是对象
- 每个对象都有各自的属性与方法
- 对象里的特点就是它的属性，它的功能就是它的方法

# 字符串的capitalize函数



# 本节课内容

- capitalize的功能
- capitalize的用法
- capitalize的注意事项

# capitalize的功能

◆将字符串的首字母大写,其他字母小写

# capitalize的用法

用法：

**newstr = string.capitalize()**

参数：

**函数括弧内什么都不用填写**

```
In [1]: name = 'xiaoMu'

In [2]: new_name = name.capitalize()

In [3]: print(new_name)
Xiaomu
```

# capitalize的注意事项

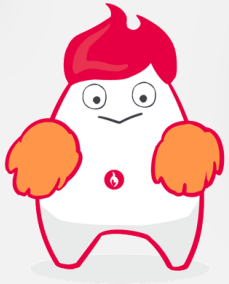
- 只对第一个字母大写有效
- 只对字母有效
- 已经是大写，则无效

number_message = ’1ok’ X

chinese = ’你好小慕’X

had_big = ’Good’X

# 字符串的小写函数



# 本节课内容

- casefold 与 lower 功能
- casefold 与 lower 用法
- casefold 与 lower 的注意事项

# casefold 与lower 的功能

◆将字符串全体小写

# casefold 与lower 的用法

用法：

newstr = string.casefold() -> 函数括弧内什么都不用填写

newstr = string.lower() -> 函数括弧内什么都不用填写

```
In [4]: name = 'DEWEI'

In [5]: new_name = name.lower()

In [6]: print(new_name)
dewei
```

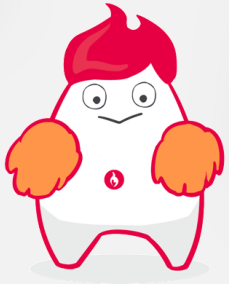
# capitalize的注意事项

◆只对字符串中的字母有效

◆已经是小写，则无效

```
number_message = '1@OK'
chinese = '你好小慕'
had_lower = 'good'
```

# 字符串的upper函数



# 本节课内容

- upper 的功能
- upper 的用法
- upper 的注意事项

# upper的功能

◆将字符串全体大写

# upper的用法

用法：

big_str = string.upper()

参数：

函数括弧内什么都不用填写

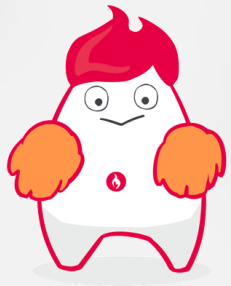
```
In [8]: name = 'xiaomu'

In [9]: big_name = name.upper()

In [10]: print(big_name)
XIAOMU
```

# upper的注意事项

- 只对字符串中的字母有效
- 已经是大写，则无效

```
number_message = ’2$yes'
chinese = '你好小慕'
had_lower = 'GOOD'
```

# 字符串的swapcase函数



# 本节课内容

- swapcase 的功能
- swapcase 的用法
- swapcase  的注意事项

# swapcase的功能

◆将字符串中大小写字母进行转换

# swapcase的用法

用法：

newstr = string.swapcase()

参数：

函数括弧内什么都不用填写

```
In [11]: name = 'DeWei'

In [12]: new_name = name.swapcase()

In [13]: print(new_name)
dEwEI
```

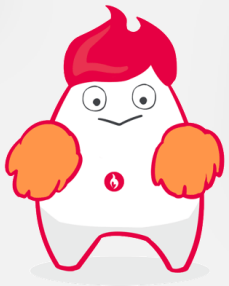
# swapcase的注意事项

- 只对字符串中的字母有效

number_message = '1@oK’ -> '1@Ok'

info = ' python语言很有趣！ '

# 字符串的zfill函数

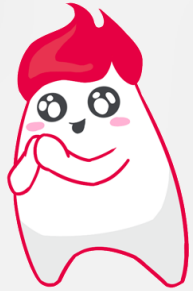


# 本节课内容

- zfill 的功能
- zfill 的用法
- zfill 的注意事项

# zfill的功能

◆为字符串定义长度，如不满足，缺少的部分用 0 填补



# zfill的用法

用法：

newstr = string.zfill(width)

参数：

width: 新字符串希望的宽度

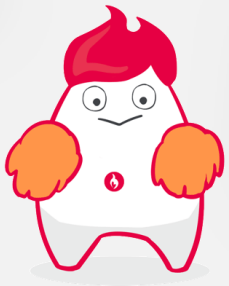
```
In [14]: name = 'xiaomu'

In [15]: new_name = name.zfill(10)

In [16]: print(new_name)
0000xiaomu
```

# zfill的注意事项

- 与字符串的字符无关
- 如果定义长度小于当前字符串长度，则不发生变化

```
test_str = 'my name is dewei'
new_str = test_str.zfill(5)
print(new_str) -> 'my name is dewei'
```

# 字符串的count函数



# 本节课内容

- count 的功能
- count 的用法
- count 的注意事项

# count的功能

◆返回当前字符串中某个成员（元素）的个数

# count的用法

用法：

string.count(sub)

参数：

括弧里需要传一个你想查询个数的元素，返回一个整数

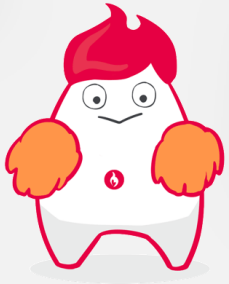
```
In [17]: info = 'my name is dewei'

In [18]: print(info.count('e'))
3
```

# count 的注意事项

- 如果查询的成员（元素）不存在，则返回0
- count 函数可以限制字符串的字段，学习索引后会补上该知识点

```
test_str = 'my name is dewei'
count = test_str.count('s')
print(count) -> 0
```

# 字符串的startswith和endswith函数



# 本节课内容

- startswith和endswith 的功能
- startswitch和endswitch的用法

# startswith和endswith功能

- startswith 判断字符串开始是否是某成员（元素）
- endswith 判断字符串结尾是否是某成员(元素)

# startswith和endswith用法

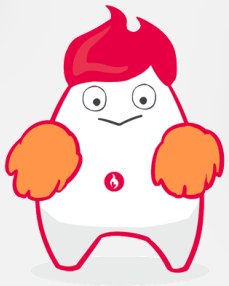
用法：

string.startswith(sub) -> sub：你想查询匹配的元素, 返回一个布尔值

string.endswith(sub) -> sub你想查询匹配的元素, 返回一个布尔值

```
In [20]: 'my name is dewei'.startswith('my')
Out[20]: True

In [21]: 'my name is dewei'.endswith('my')
Out[21]: False
```

# 字符串的find与index函数



# 本节课内容

- find和index 的功能
- find和index的用法
- find与index的区别

# find和index功能

- find 与 index 都是返回你想寻找的成员的位置

# find和index的用法

用法：

string.find(sub) -> sub：你想查询的元素，返回一个整型

string.index(sub) -> sub：你想查询的元素，返回一个整型或者报错

Ps：字符串里的位置是从0开始的

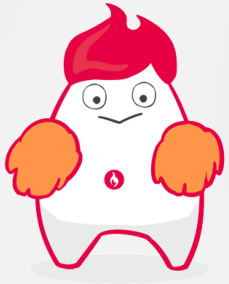
```
In [22]: 'my name is dewei'.find('e')
Out[22]: 6

In [23]: 'my name is dewei'.index('i')
Out[23]: 8
```

# find与index的区别

◆如果find找不到元素，会返回-1

◆如果index 找不到元素，会导致程序报错

# 字符串的strip函数



# 本节课内容

- strip的功能
- strip的用法
- strip的拓展知识

# strip的功能

◆strip 将去掉字符串左右两边的指定元素，默认是空格

# strip的用法

用法：

newstr = string.strip(sub)

参数：

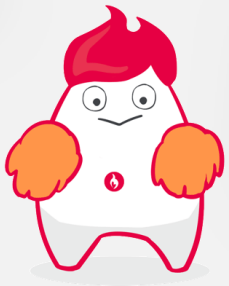
括弧里需要传一个你想去掉的元素，可不填写

```
In [25]: ' hello xiaomu '.strip()
Out[25]: 'hello xiaomu'

In [26]: 'hello xiaomu'.strip('h')
Out[26]: 'ello xiaomu'
```

# strip的拓展

- 传入的元素如果不在开头或结尾则无效
- lstrip 仅去掉字符串开头的指定元素或空格
- rstrip 仅去掉字符串结尾的指定元素或空格

# 字符串的replace函数



# 本节课内容

- replace的功能
- replace的用法

# replace 的功能

◆将字符串中的old（旧元素）替换成new（新元素），并能指定替换的数量

# replace的用法

用法：

newstr = string.replace(old, new, max)

参数：

old： 被替换的元素，

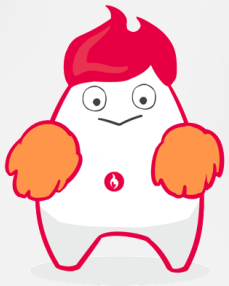
new： 替代old的新元素，

max：可选，代表替换几个,默认全部替换全部匹配的old元素

# replace的用法

```
In [29]: 'hello, dewei'.replace('dewei', 'xiaomu')
Out[29]: 'hello, xiaomu'

In [30]: 'hello, xiaomu'.replace('l', '0', 1)
Out[30]: 'he0lo, xiaomu'
```

# 字符串中返回bool类型的函数集合



# 本节课内容

- isspace
- istitle
- isupper
- islower
- 未完待续的函数 join 与 split

# isspace

功能：

isspace 判断字符串是否是一个由空格组成的字符串

用法：

booltype = string.isspace () -> 无参数可传，返回一个布尔类型

# isspace

```
In [31]: ' '.isspace()
Out[31]: True

In [32]: 'hello xiaomu'.isspace()
Out[32]: False
```

Ps: 由空格组成的字符串，不是空字符串：’ ’ != ’ ’

# istitle

功能：

istitle 判断字符串是否是一个标题类型

用法：

booltype = String.istitle () -> 无参数可传，返回一个布尔类型

```
In [33]: 'Hello Xiaomu'.istitle()
Out[33]: True

In [34]: 'hello xiaomu'.istitle()
Out[34]: False
```

**Ps: 该函数只能用于英文**

# isupper与islower

功能：

isupper 判断字符串中的字母是否都是大写

islower判断字符串中的字母是否都是小写

用法：

booltype = string.isupper () -> 无参数可传，返回一个布尔类型

booltype = string.islower() -> 无参数可传，返回一个布尔类型

# isupper与islower

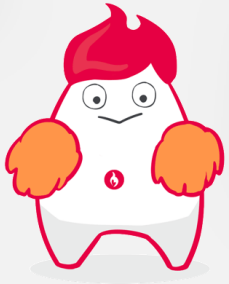
```
In [35]: 'hello xiaomu'.islower()
Out[35]: True

In [36]: 'hello xiaomu'.isupper()
Out[36]: False
```

Ps：只检测字符串里的字母，对其他字符不做判断

# join和split 稍后见

- 我们数据类型转换的时候见

# 字符的编码格式



# 本节课内容

- 什么是编码格式
- 常见的编码格式
- 通用的编码格式

# 什么是编码格式

- 有一定规则的规则

- 使用了这种规则，我们就能知道传输的信息是什么意思

# 常见编码格式

- ascii 英文编码
- gbk是中文编码

```
1  # coding:ascii
2
3  print('哈哈')
4
```

```
print_test
/Users/zhangdewei/.v
??
```

```
# coding:ascii

print('haha')
```

```
print_test
/Users/zhangdewei/.virtuale
haha
```

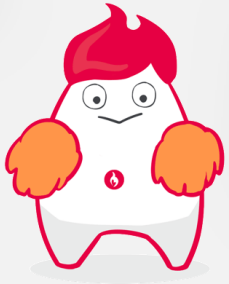
# 通用的编码格式

- uft-8 是一种国际通用的编码格式

```
# coding: utf-8

print('哈哈')
print('haha')
```

```
print_test ×
/Users/zhangdewei/.vi
哈哈
haha
```

# 字符串的格式化



# 本节课内容

- 什么是格式化
- 使用格式化场景和目的
- 格式化的三种方式

# 什么是格式化

- 定义：一个固定的字符串中有部分元素是根据变量的值而改变的字符串

今天是xx，星期xx，大家好

date = ’2020.0101’

Day = ’一’

# 格式化使用场景与目的

- 发送邮件的时候
- 发送短信的时候
- App上发推送的时候
- 对于重复性很多的信息，通过格式化的形式，可以减少代码的书写量

# 根据类型定义的格式化

◆字符串格式化使用操作符 % 来实现

格式符　　格式符

```
'my name is %s, my age is %s' % ('dewei', 33)
```

格式化字符串

对应格式符的变量，变量与格式符按顺序一一对应，数量保持一致，超过1个格式化变量用小括号包裹

格式化字符串与格式符变量之间用 一个 % 连接，% 两边有1个空格

# 字符串格式化函数-format

- string.format 函数用来格式化字符串
- 使用 format的字符串主体使用 {} 大括号来替代格式符
- string.format(data, data, data...)

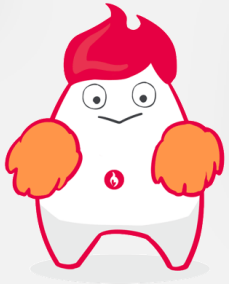
```
In [1]: 'hello {0}, 今天看起来气色{1}'.format('小慕', '不错')
Out[1]: 'hello 小慕, 今天看起来气色不错'
```

# Python3.6加入的新格式化方案—f-strings

- 定义一个变量
- 字符串前加 f 符号
- 需要格式化的位置使用 {变量名}

```
In [6]: f'hello {name}'
Out[6]: 'hello 小慕'
```

# 字符串的格式化



# 本节课内容

- 字符串格式化的符号

# 格式化符号

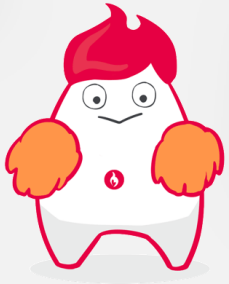
◆用于对应各种数据类型的格式化符号-----格式化符号

| 符号 | 说明 |
| --- | --- |
| **%s** | **格式化字符串，通用类型** |
| %d | 格式化整型 |
| %f | 格式化浮点型 |
| %u | 格式化无符号整型（正整型） |
| %c | 格式化字符 |

# 格式化符号

| 符号 | 说明 |
| --- | --- |
| %o | 格式化无符号八进制数 |
| %x | 格式化无符号16进制数 |
| %e | 科学计数法格式化浮点数 |

# 字符串的转义字符



# 本节课内容

- 什么是转义字符
- Python中的转义字符们
- 转义无效符

# 什么是转义字符

- 字符要转成其他含义的功能，所以我们叫他转义字符
- \ + 字符

# Python中的转义字符们

| 符号 | 说明 |
| --- | --- |
| \n | 换行，一般用于末尾，strip对其也有效 |
| \t | 横向制表符（可以认为是一个间隔符） |
| \v | 纵向制表符（会有一个男性符号） |
| \a | 响铃 |
| \b | 退格符,将光标前移,覆盖（删除前一个） |
| \r | 回车 |
| \f | 翻页（几乎用不到，会出现一个女性符号） |
| \’ | 转义字符串中的单引号 |
| \” | 转义字符串中的双引号 |
| \\ | 转义斜杠 |

# 转义无效符

- 在python中 在字符串前加 r 来将当前字符串的转义字符无效化

```
In [8]: print(r'hello \f')
hello \f
```