字符串

熟悉快捷键：要在没有选中某行的情况下

In [2]:
```
s = 'hello'
```

这是一个；例子

In [3]:
```
s[0]
```

Out[3]:
```
'h'
```

In [4]:
```
s[0] = 'H'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 s[0] = 'H'

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

不可改变字符串

In [5]:
```
s = 'Hello'
```

In [6]:
```
s
```

Out[6]:
```
'Hello'
```

s为什么就可以直接改变？python的数据结构比较特殊，相当于s是个标签，贴在装着hello的盒子上面，但是字符串hello不可以直接改变 而方法二相当于另外创建了一个对象Hello，再把s标签贴到第二个盒子上。 第一个盒子没人用了没事，因为python有个垃圾清理器，会时不时清理

In [7]:
```
s.upper()
```

Out[7]:
```
'HELLO'
```

In [8]:
```
s
```

Out[8]:
```
'Hello'
```

其实没动s，标签还贴在那里，盒子里面的东西没变，字符串不可改变。upper相当于另外创建了有一块地方，放了转化后的，而且一段时间就被清理了

In [13]:
```
long_string = '''hello
hello again
hello
'''
```

In [14]:
```
long_string
```

Out[14]:
```
'hello\nhello again\nhello\n'
```

In [15]:
```
s = 'hello
world'
```

```
  Cell In[15], line 1
    s = 'hello
        ^
SyntaxError: unterminated string literal (detected at line 1)
```

长字符串用三引号'''，普通引号不允许转行

In [16]:
```
1 + 2 + 3
+4
```

Out[16]:
```
4
```

py语句一般不允许换行，可以用续行号

In [17]:
```
1+2+3\
+4
```

Out[17]:
```
10
```

还有一种情况，可以用（），建议用括号不用续行号

In [18]:
```
(1+2+3
 +4)
```

Out[18]:
```
10
```

In [21]:
```
print("hello\nword.")      #\n换行符
```

```
hello
word.
```

In [22]:
```
print("path:c:\\documents")
```

```
path:c:\documents
```

In [23]:
```
print('you must input "c:\\documents" for "c:\documents"')
```

```
you must input "c:\documents" for "c:\documents"
```

In [25]:
```
print('you must input "c:\\\documents" for "c:\documents"')#巧合，恰好\d不是转义字符
```

```
you must input "c:\\documents" for "c:\documents"
```

In [26]:
```
print(r'you must input "c:\\documents" for "c:\documents"')
```

```
you must input "c:\\documents" for "c:\documents"
```

raw，用r实现打啥样出来啥样！重要，回忆上节课格式化f

第二节：List

In [27]:
```
my_list = [0, 1, 2, 3, 4, 5]
```

In [28]:
```
my_list
```

Out[28]:
```
[0, 1, 2, 3, 4, 5]
```

In [29]:
```
my_list[0]
```

Out[29]:
```
0
```

In [30]:
```
my_list[-1]
```

Out[30]:
```
5
```

py里面list可以负数，-1，倒数第一，但埋下了陷阱

In [31]:
```
my_list[0] = 100
```

In [32]:
```
my_list
```

Out[32]:
```
[100, 1, 2, 3, 4, 5]
```

list和string最大的不同：list可以改变，string不行

In [33]:
```
my_list[6]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[33], line 1
----> 1 my_list[6]

IndexError: list index out of range
```

In [36]:
```
my_list[2] = 'hello'
my_list
```

Out[36]:
```
[100, 1, 'hello', 3, 4, 5]
```

list和数组不同，它可以是非均一的序列。但是确点，它没法像数组（均一）一样访问迅速。后来又为此设计了numpy

In [37]:
```
my_list[2] = [7, 8, 9]
```

In [38]:
```
my_list
```

Out[38]:
```
[100, 1, [7, 8, 9], 3, 4, 5]
```

甚至可以用它模仿二维数组!

In [39]:
```
my_list[2][1]
```

Out[39]:
```
8
```

嵌套-一层一层访问

In [41]:
```
my_list = [0, 1, 2, 3, 4, 5]
```

In [44]:
```
my_list[1:5]  #1-5
```

Out[44]:
```
[1, 2, 3, 4]
```

py的半开区间，一般区间指[a,b)

In [45]:
```
my_list[1:]
```
Out[45]:
```
[1, 2, 3, 4, 5]
```

In [47]:
```
my_list[0:6]
```
Out[47]:
```
[0, 1, 2, 3, 4, 5]
```

In [48]:
```
my_list[0:6:2]
```
Out[48]:
```
[0, 2, 4]
```

最后一个数字代表步长（间隔），甚至间隔还可以是负数

In [50]:
```
my_list[3:0:-1]
```
Out[50]:
```
[3, 2, 1]
```

In [51]:
```
my_list[5:0:-1]
```
Out[51]:
```
[5, 4, 3, 2, 1]
```

走不到最后，差点就完美了，完全转置

In [55]:
```
my_list[5:-1:-1]#可不行，-1不是0前面
```
Out[55]:
```
[]
```

In [52]:
```
my_list[5::-1]
```
Out[52]:
```
[5, 4, 3, 2, 1, 0]
```

但可以不写！直接完整

In [53]:
```
my_list[5:-1]
```
Out[53]:
```
[]
```

不行，5和-1值的都是最后一个，还是开区间

还能不能其他方法转置，以下方法更好，不需要数数

In [54]:
```
my_list[-1::-1]
```
Out[54]:
```
[5, 4, 3, 2, 1, 0]
```

In [56]:
```
my_list = []
```

In [57]:
```
my_list = list()
```

类&对象 （抽象概念&一个实例）

In [58]:
```
[1, 2, 3]+[4, 5, 6]
```

Out[58]:
```
[1, 2, 3, 4, 5, 6]
```

In [59]:
```
[1]*5
```

Out[59]:
```
[1, 1, 1, 1, 1]
```

In [62]:
```
alist = list()#空的
x = 0
while x< 10:
    alist.append(x)
    x = x + 1
```

In [63]:
```
alist
```

Out[63]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

养成习惯，循环n次，那就从0开始到<n即可，符合py的半开区间习惯，不用≤

In [64]:
```
alist = list('hello')
```

In [66]:
```
alist#思考？里面有几个元素？1个or5个？
```

Out[66]:
```
['h', 'e', 'l', 'l', 'o']
```

用一种sequence去构造另一个sequence，和原来的元素一样

In [68]:
```
'*'.join(alist)
```

Out[68]:
```
'h*e*l*l*o'
```

''*join拼在一起，用*把alist里面每个元素链接起来

''.join直接拼在一起（用空字符串）

In [69]:
```
''.join(alist)
```

Out[69]:
```
'hello'
```

推荐方法

In [70]:
```
alist = list(range(0, 10))
alist
```

Out[70]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

range不是函数，功能是将在范围[a,b)的数字（对象）一个一个地给

In [71]:
```
for number in range(0, 10):
    print('hello')#为什么不用写换行？因为print缺省默认换行，如果不换行，最后要“ ，
#    print(f'{number}^3 = {number ** 3}')
```

```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

for number in range(0, n): 想重复n次，就写(0, n); range每次给你一个数字，且会帮你+1；

In [72]:
```
for number in range(0, 10):
    print(f'{number}^3 = {number ** 3}')#为什么不用写换行？
```

```
0^3 = 0
1^3 = 1
2^3 = 8
3^3 = 27
4^3 = 64
5^3 = 125
6^3 = 216
7^3 = 343
8^3 = 512
9^3 = 729
```

打印*十行

In [74]:
```
for index in range(0,10):
    print(' ' * (10 - index - 1), (2 * index + 1) * '*')
```

```
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
  *****************
 *******************
```

py不必二重循环

打印乘法表（上三角）： 1.先简单点，全部填满（正方形），要做循环9*9=8次

In [78]:
```
for x in range(1, 10):
    for y in range(1, 10):
        print(f'{x}x{y}={x*y}')#缺省直接换行
```

```
1x1=1
1x2=2
1x3=3
1x4=4
1x5=5
1x6=6
1x7=7
1x8=8
1x9=9
2x1=2
2x2=4
2x3=6
2x4=8
2x5=10
2x6=12
2x7=14
2x8=16
2x9=18
3x1=3
3x2=6
3x3=9
3x4=12
3x5=15
3x6=18
3x7=21
3x8=24
3x9=27
4x1=4
4x2=8
4x3=12
4x4=16
4x5=20
4x6=24
4x7=28
4x8=32
4x9=36
5x1=5
5x2=10
5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
6x1=6
6x2=12
6x3=18
6x4=24
6x5=30
6x6=36
6x7=42
6x8=48
6x9=54
7x1=7
7x2=14
7x3=21
7x4=28
7x5=35
7x6=42
7x7=49
7x8=56
7x9=63
8x1=8
```

```
8x2=16
8x3=24
8x4=32
8x5=40
8x6=48
8x7=56
8x8=64
8x9=72
9x1=9
9x2=18
9x3=27
9x4=36
9x5=45
9x6=54
9x7=63
9x8=72
9x9=81
```

In [79]:
```
for x in range(1, 10):
    for y in range(1, 10):
        print(f'{x}x{y}={x*y}',end='\t')#改进1
```

```
1x1=1   1x2=2   1x3=3   1x4=4   1x5=5   1x6=6   1x7=7   1x8=8   1x9=9   2x1=2   2x2=
4       2x3=6   2x4=8   2x5=10  2x6=12  2x7=14  2x8=16  2x9=18  3x1=3   3x2=6   3x3=
9       3x4=12  3x5=15  3x6=18  3x7=21  3x8=24  3x9=27  4x1=4   4x2=8   4x3=12  4x4=
16      4x5=20  4x6=24  4x7=28  4x8=32  4x9=36  5x1=5   5x2=10  5x3=15  5x4=20  5x5=
25      5x6=30  5x7=35  5x8=40  5x9=45  6x1=6   6x2=12  6x3=18  6x4=24  6x5=30  6x6=
36      6x7=42  6x8=48  6x9=54  7x1=7   7x2=14  7x3=21  7x4=28  7x5=35  7x6=42  7x7=
49      7x8=56  7x9=63  8x1=8   8x2=16  8x3=24  8x4=32  8x5=40  8x6=48  8x7=56  8x8=
64      8x9=72  9x1=9   9x2=18  9x3=27  9x4=36  9x5=45  9x6=54  9x7=63  9x8=72  9x9=
81
```

In [80]:
```
for x in range(1, 10):
    for y in range(1, 10):
        print(f'{x}x{y}={x*y}',end='\t')
    print('')#换行？为啥不用换行符？
```

```
1x1=1   1x2=2   1x3=3   1x4=4   1x5=5   1x6=6   1x7=7   1x8=8   1x9=9
2x1=2   2x2=4   2x3=6   2x4=8   2x5=10  2x6=12  2x7=14  2x8=16  2x9=18
3x1=3   3x2=6   3x3=9   3x4=12  3x5=15  3x6=18  3x7=21  3x8=24  3x9=27
4x1=4   4x2=8   4x3=12  4x4=16  4x5=20  4x6=24  4x7=28  4x8=32  4x9=36
5x1=5   5x2=10  5x3=15  5x4=20  5x5=25  5x6=30  5x7=35  5x8=40  5x9=45
6x1=6   6x2=12  6x3=18  6x4=24  6x5=30  6x6=36  6x7=42  6x8=48  6x9=54
7x1=7   7x2=14  7x3=21  7x4=28  7x5=35  7x6=42  7x7=49  7x8=56  7x9=63
8x1=8   8x2=16  8x3=24  8x4=32  8x5=40  8x6=48  8x7=56  8x8=64  8x9=72
9x1=9   9x2=18  9x3=27  9x4=36  9x5=45  9x6=54  9x7=63  9x8=72  9x9=81
```

In [83]:
```
for x in range(1, 10):
    for y in range(1, 10):
        if x<=y:#改进：左下不输出
            print(f'{x}x{y}={x*y}',end='\t')
        else:
            print('\t')#没有写全
    print('')
```

1x1=1 1x2=2 1x3=3 1x4=4 1x5=5 1x6=6 1x7=7 1x8=8 1x9=9

2x2=4 2x3=6 2x4=8 2x5=10 2x6=12 2x7=14 2x8=16 2x9=18

3x3=9 3x4=12 3x5=15 3x6=18 3x7=21 3x8=24 3x9=27

4x4=16 4x5=20 4x6=24 4x7=28 4x8=32 4x9=36

5x5=25 5x6=30 5x7=35 5x8=40 5x9=45

6x6=36 6x7=42 6x8=48 6x9=54

7x7=49 7x8=56 7x9=63

8x8=64 8x9=72

9x9=81

for x in range(1, 10):

```
    for y in range(1, 10):
        if x<=y:#改进：左下不输出
            print(f'{x}x{y}={x*y}',end='\t')
        else:
            print('\t',end='\t')#又错了。。
    print('')
```

In [85]:

```
for x in range(1, 10):
    for y in range(1, 10):
        if x<=y:#改进：左下不输出
            print(f'{x}x{y}={x*y}',end='\t')
        else:
            print('',end='\t')#又错了。。
    print('')
```

```
1x1=1   1x2=2   1x3=3   1x4=4   1x5=5   1x6=6   1x7=7   1x8=8   1x9=9
        2x2=4   2x3=6   2x4=8   2x5=10  2x6=12  2x7=14  2x8=16  2x9=18
                3x3=9   3x4=12  3x5=15  3x6=18  3x7=21  3x8=24  3x9=27
                        4x4=16  4x5=20  4x6=24  4x7=28  4x8=32  4x9=36
                                5x5=25  5x6=30  5x7=35  5x8=40  5x9=45
                                        6x6=36  6x7=42  6x8=48  6x9=54
                                                7x7=49  7x8=56  7x9=63
                                                        8x8=64  8x9=72
                                                                9x9=81
```

list可以用一个操作符in来判断某元素在不在其中

In [87]:
```
4 in [1, 2, 3]
```

Out[87]:
```
False
```

False是py内置的

In [90]:
```
ch = input('please input an alphabet: ')
if ch =='a' or ch =='e' or ch =='i' or ch =='o' or ch =='u':
    print('vowel')
else:
    print('not vowel')
```

```
not vowel
```

哈哈哈，不智能。漂亮的写法如下：

In [91]:
```
ch = input('please input an alphabet: ')
if ch in 'aeiouAEIOU':
    print('vowel')
else:
    print('not vowel')
```

```
vowel
```

In [92]:
```
a = list(range(0, 10))
a
```

Out[92]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

清除：clear和del，但注意能修改的东西

In [93]:
```
a.clear()
```

In [94]:
```
a
```

Out[94]:
```
[]
```

In [96]:
```
a = list(range(0, 10))
a
```

Out[96]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [98]:
```
del a[0]
```

In [99]:
```
a
```

Out[99]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [101…
```
a = [1, 2, 3]
b = a
```

In [102…
```
a
```

Out[102]:
```
[1, 2, 3]
```

In [103…
```
b
```

Out[103]:
```
[1, 2, 3]
```

In [104…
```
b[0] = 100
```

In [105…
```
b
```

Out[105]:
```
[100, 2, 3]
```

In [106…
```
a
```

Out[106]:
```
[100, 2, 3]
```

a,b都是标签，贴在同一个盒子上

In [107…
```
x = 3
x = 'hello'
```

In [108…
```
x
```

Out[108]:
```
'hello'
```

x没有类型，只是标签

如果想让a,b有所区别，贴在不同的盒子,用copy函数

In [110…
```
a = [1, 2, 3]
b = a.copy()
b.append(4)
```

In [111…
```
a
```

Out[111]:
```
[1, 2, 3]
```

In [112…
```
b
```

Out[112]:
```
[1, 2, 3, 4]
```

然而问题还没结束...

In [114…
```
a = [1, 2, [3, 4]]
b = a.copy()
b[2][0] = 0
print(a)
print(b)
```

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

[3,4]其实在另一个盒子里面，标签写的是item[2];b复制a的时候，只是复制一层， 把a[0]（即1）,a[1]（即2）,a[2]（即item[2]复制过来了）； 所以b[2]也是item[2]，两个标签都贴在同一个盒子上

所以要深度复制deepcopy

In [168…
```
from copy import deepcopy  #写法有两种，另一种import copy 然后b = copy.deepcopy(a)
```

In [169…
```
a = [1, 2, [3, 4]]
b = deepcopy(a)    #???
b[2][0] = 0
print(a)
print(b)
```

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

In [118…
```
a = [1, 2, 3]
len(a)
```

Out[118]:
```
3
```

extend和append不同

In [119…
```
a.extend([4, 5, 6])
```

In [120…
```
a
```

Out[120]:
```
[1, 2, 3, 4, 5, 6]
```

In [121…
```
a.append([4, 5, 6])
```

In [122…
```
a
```

Out[122]:
```
[1, 2, 3, 4, 5, 6, [4, 5, 6]]
```

In [124…
```
a.count(4)#数4
```

Out[124]:
```
1
```

In [125…
```
a.index(8)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[125], line 1
----> 1 a.index(8)

ValueError: 8 is not in list
```

In [126…
```
a = list(range(0, 10))
```

In [127…
```
a
```

Out[127]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [128…
```
a.reverse()#转置
```

In [129…
```
a
```

Out[129]:
```
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

list可变转置成功，string不可变，之前upper'Hello'没成功

sort和sorted:a.sort改变a;sorted不改变a，只返回排序后的值；而会改变的sort函数一定不返回值

In [131…
```
import random
a = list()
for x in range(0,10):
    a.append(random.randint(0,100))#random.randint随机取值？
a
```

Out[131]:
```
[53, 29, 27, 80, 99, 62, 40, 51, 70, 56]
```

In [134…
```
a.sort()#排序
```

In [133…
```
a
```

Out[133]:
```
[27, 29, 40, 51, 53, 56, 62, 70, 80, 99]
```

In [137…
```
import random
a = list()
for x in range(0,10):
    a.append(random.randint(0,100))#random.randint随机取值？
a
```

Out[137]:
```
[76, 27, 84, 53, 87, 22, 84, 48, 74, 81]
```

In [138…
```
sorted(a)
```

Out[138]:
```
[22, 27, 48, 53, 74, 76, 81, 84, 84, 87]
```

In [139…
```
a
```

Out[139]:
```
[76, 27, 84, 53, 87, 22, 84, 48, 74, 81]
```

会改变的sort函数一定不返回值

In [142…
```
b = a.sort()
```

In [143…
```
type(b)
```

Out[143]:
```
NoneType
```

前面这个例子有缺陷， import random a = list() for x in range(0,10): a.append(random.randint(0,100)) 可能后面list空间会不够，更好的做法：先定好空间大小

In [144…
```
a = [None]*100
for i in range(0, 100):
    a[i] = random.randint(0,100)
```

In [145…

```
a
```

Out[145]:
```
[13,
 0,
 75,
 80,
 62,
 85,
 26,
 50,
 16,
 93,
 90,
 41,
 32,
 43,
 4,
 6,
 93,
 80,
 76,
 18,
 20,
 3,
 83,
 43,
 27,
 78,
 35,
 0,
 1,
 97,
 51,
 67,
 29,
 17,
 90,
 25,
 47,
 86,
 0,
 3,
 1,
 97,
 3,
 54,
 68,
 11,
 5,
 98,
 30,
 18,
 38,
 100,
 4,
 32,
 50,
 91,
 32,
 5,
 2,
 79,
 79,
 16,
 65,
 64,
```

```
 40,
 43,
 16,
 7,
 22,
 80,
 70,
 88,
 31,
 16,
 94,
 49,
 33,
 34,
 41,
 30,
 39,
 47,
 94,
 38,
 37,
 21,
 50,
 48,
 82,
 13,
 80,
 46,
 67,
 16,
 48,
 28,
 89,
 68,
 25,
 90]
```

你可能会怀疑这种做法有什么好处吗？可以用%%timeit检验运行时间

In [149…

```
%%timeit

a = list()
for x in range(0,1000):
    a.append(random.randint(0,100))
```

```
842 µs ± 81.1 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

In [151…

```
%%timeit

a = [None]*1000
for i in range(0, 1000):
    a[i] = random.randint(0,100)
```

```
852 µs ± 150 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

介绍表格编程

比如问8月15日是第几天

month = int(input('Please input month:')) day = int(input('Please input day:'))

count = day

if month > a: count+=31 #麻烦

In [153…
```
a=[1, 2, 3]
sum(a)
```

Out[153]: 6

In [165…
```
sum(a[0:])
```

Out[165]: 6

In [4]:
```
month = int(input('Please input month: '))
day = int(input('Please input day: '))

days_in_month = [31, 28, 31, 30 ,31, 30, 31, 31, 30, 31, 30, 31]
print(f'This is the {sum(days_in_month[0:month-1]) + day}th day of the year.')
```

This is the 365th day of the year.

In [10]:
```
grades = 'E' * 40 + 'D' * 20 + 'C' * 15 + 'B' * 15 + 'A' * 11
# print(grade)
score = int(input('Please input your score:'))
if score > 100 or score < 0:
    print('error.')
else:
    print(f"your garde is {grades[score]}.")
```

your garde is A.

In [ ]: