第2天两行代码下载网站数据

一、 创建

```
s1 = 'lenovo'
s2 = "QF"
s3 = """hello lenovo"""
s4 = '''hello 千锋云计算'''
s5 = """hello
shark
"""
s6 = '''hello
world'''
```


二、简单使用

1. \ 转义符

```
testimony = 'This shirt doesn\'t fit me'
```

```
words = 'hello \nshark'
```

2. + 拼接

```
In [1]: file_name= "成功的21个信念"

In [2]: suffix = '.txt'

In [3]: file_name = file_name + suffix

In [4]: file_name
Out[4]: '成功的21个信念.txt'

```

拼接只能是 字符串和字符串进行操作，不可以用 字符串和 一个非字符串类型的对象相加

```
In [5]: '西瓜甜' + 1    ## 这会报错的
```


3. `*` 复制

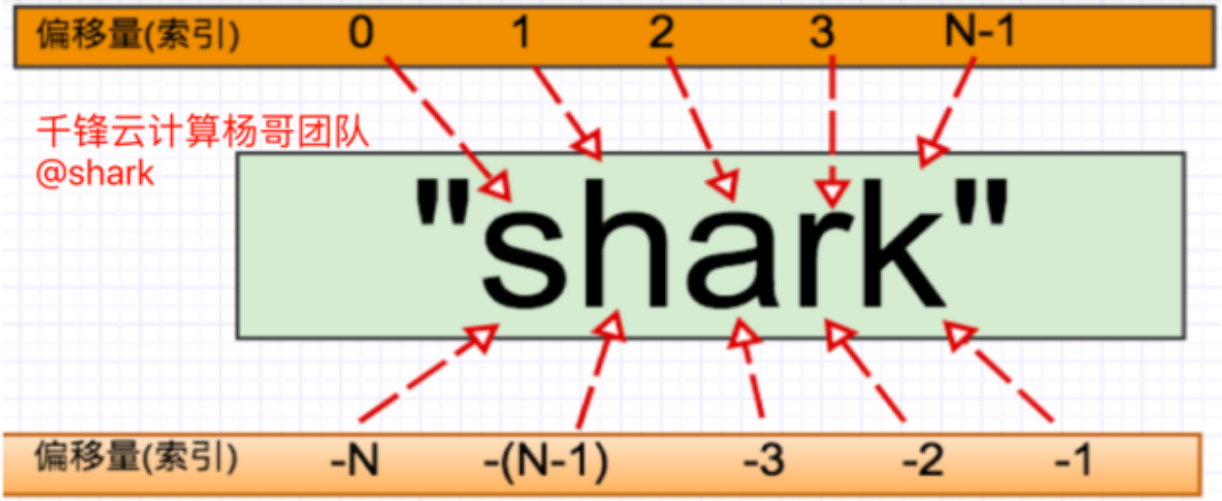
```
In [6]: "-" * 10
Out[6]: '----------'

In [7]: print('*' * 10)
**********

```

三、取值和切片

1. 字符串 是 Python 中的一个 序列类型 的数据结构

- 存放的数据，在其内是有序的。

```
s1 = "shark"
```



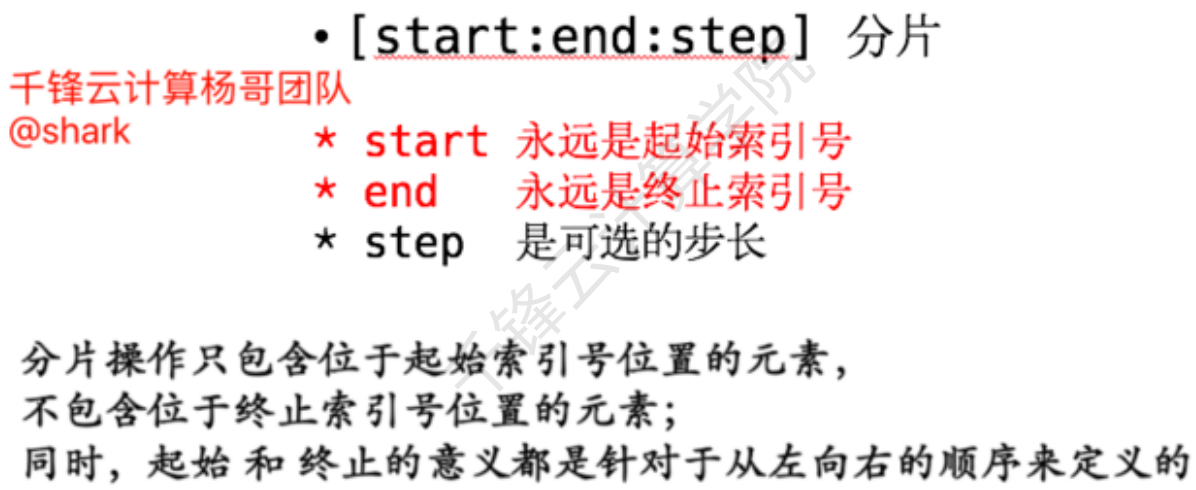
序列类型的特点

- 序列里的每个数据被称为序列的一个元素
- 元素在序列里都是有个自己的位置的，这个位置被称为索引或者叫偏移量，也有叫下标的， 从 `0` 开始，从左到右依次递增
- 序列中的每一个元素可以通过这个元素的索引来获取到
- 获取到序列类型数据中的多个元素需要用切片的操作来获取

2. 通过索引取值，获取单个元素

```
In [10]: s1 = "shark"

In [11]: s1[0]
Out[11]: 's'

In [12]: s1[-1]
Out[12]: 'k'

In [13]: s1[3]
Out[13]: 'r
```

3. 切片，获取多个元素



3.1 一般操作

```
# 使用切片获取多个元素
In [15]: s1[0:3]
Out[15]: 'sha'

# 起始和结尾的索引号可以省略
In [16]: s1[:3]
Out[16]: 'sha'

In [17]: s1[1:]
```

```
Out[17]: 'hark'

# 索引可以使用 负数
In [18]: s1[2:-1]
Out[18]: 'ar'

In [19]:
```

下面这样的操作，臣妾做不到

```
>>> s1[-1:-3]
''
>>>
```

> 因为默认的切片是从左向右开始操作， 索引号 `-1` 的右边没有任何索引号了 `-3` 在 `-1` 的左边

3.2 使用步长

> - 步长就是每数几个取一个的意思
> - 步长是正数时，是从左向右开始操作
> - 步长是负数时，是从右向左操作

```
In [19]: s2 = 'abcdefg'

In [20]: s2[::2]
Out[20]: 'aceg'

In [21]: s2[::-1]
Out[21]: 'gfedcba'

In [22]: s2[::-2]
Out[22]: 'geca'

```

四、字符串方法

1. 统计序列数据的长度

> 就是获取一个序列数据的元素个数，这个适用于所有的序列类型的数据，比如 字符串、列表、元组。

```
# 获取字符串的长度，包含空格和换行符
In [25]: s3 = "a \n\t"

In [26]: len(s3)
Out[26]: 4
```

> `\n` 是一个换行符 `\t` 是一个 Tab 键

2. in 成员判断

```
In [39]: line = 'Size: 8192 MB'

In [40]: if 'Size' in line:
    ...:     print(line)
    ...:
    Size: 8192 MB
```

> 注意： 空的字符串总是被视为任何其他字符串的子串，因此 `"" in "abc"` 将返回 `True`。

3. strip() 去除字符串两端的空白字符（空格、`\t`、`\n`）

```
Out[41]: line = '    Size: 8192 MB'

In [42]: line.strip()
Out[42]: 'Size: 8192 MB'
```

4. split() 分割

> 默认使用空白字符作为分隔符（空格、`\t`、`\n`） 和 shell 中的 awk 一样道理

```
In [47]: line
Out[47]: '    Size: 8192 MB'

In [48]: line.split()
Out[48]: ['Size:', '8192', 'MB']

In [49]: s
Out[49]: '\tab\n'

In [50]: s.split()
Out[50]: ['ab']

```


可以指定分隔符

```
In [51]: line.split(':')
Out[51]: ['    Size', ' 8192 MB']

In [52]: line.split(': ')
Out[52]: ['    Size', '8192 MB']

```

5. strip() 移除字符串两端的空白字符**

```
In [71]: line = '    Size: 8192 MB'

In [72]: line.strip()
Out[72]: 'Size: 8192 MB'

```

`strip()` 返回的是字符串，所以可以连续操作

```
In [73]: line.strip().split(': ')
Out[73]: ['Size', '8192 MB']

In [74]: line
Out[74]: '    Size: 8192 MB'

In [75]: k, v = line.strip().split(': ')

In [76]: k
Out[76]: 'Size'

In [77]: v
Out[77]: '8192 MB'

```


6. replace() 替换

```
In [65]: line = '    <strong>10、命运在自己手里，而不是在别人的嘴里</strong></p>'

In [66]: line.strip()    ## 先去除两端空白字符
Out[66]: '<strong>10、命运在自己手里，而不是在别人的嘴里</strong></p>'

In [67]: line.strip().replace('strong>', '')    ## 将字符串 strong> 替换为空
Out[67]: '<10、命运在自己手里，而不是在别人的嘴里</</p>'

In [68]: line.strip().replace('strong>', '')[1:-6]
Out[68]: '10、命运在自己手里，而不是在别人的嘴里'

```

7. startswith() 判断字符串以什么为开头

```
In [85]: line = 'Locator: DIMM_A2'

In [86]: line.startswith("Locator:")
Out[86]: True
```

8. endswith() 判断字符串以什么为结尾

```
In [87]: line = 'Size: 8192 MB'

In [88]: line.endswith('MB')
Out[88]: True

```

五、 字符串的判断（扩展自修）

```
In [1]: s = '123'

In [2]: s.isdigit()   # 判断是否是纯数字
Out[2]: True

In [3]: s1 = '123adb'

In [4]: s1.isalnum()  # 判断是否是数字和字母
Out[4]: True

In [5]: s2 = 'adb'

In [6]: s2.isalpha()  # 判断是否是纯字母
Out[6]: True

In [7]: s2.encode()   # 转换为二进制 bytes 类型
Out[7]: b'adb'

In [8]: s4 = "云计算"

In [9]: s4.encode()   # 转换为二进制 bytes 类型，默认编码 utf-8
Out[9]: b'\xe4\xba\x91\xe8\xae\xa1\xe7\xae\x97'

In [16]: b = s4.encode()

In [17]: b.hex()      # bytes 转换成 16 进制
Out[17]: 'e4ba91e8aea1e7ae97'

In [18]: b.decode()   # bytes 转换成 str，默认编码 utf-8
Out[18]: '云计算'
```