字符串

- 机构名称：西部开源技术中心
- 课程方向: python全栈开发与大数据分析

目录

CONTENTS

1. 字符串的创建和赋值
2. 字符串的基本特性
3. 字符串内建函数
4. 字符串相关模块

01 字符串的创建和赋值

字符串类型是 Python 里面最常见的类型。 可以简单地通过在引号间(单引号,双引号和三引号)包含字符的方式创建它。

第一种方式:

```
str1 = 'our company is westos'
```

第二种方式:

```
str2 = "our company is westos"
```

第三种方式:

```
str3 = """our company is westos"""
```

01 字符串的创建和赋值

转义符号

一个反斜线加一个单一字符可以表示一个特殊字符，通常是不可打印的字符

转义字符	名　　称	功　　能
\a	响铃	用于输出响铃
\b	退格（Backspace键）	用于退回一个字符
\f	换页	用于输出
\n	换行符	用于输出
\r	回车符	用于输出
\t	水平制表符（Tab键）	用于输出
\v	纵向制表符	用于制表
\\	反斜杠字符	用于表示一个反斜杠字符
\'	单引号	用于表示一个单引号字符
\"	双引号	用于表示一个双引号字符
\ddd	ddd是ASCII码的八进制值，最多三位	用于表示该ASCII码代表的字符
\xhh或\Xhh	hh是ASCII码的十六进制值，最多两位	用于表示该ASCII码代表的字符

```
>>> say = 'let's go'
  File "<stdin>", line 1
    say = 'let's go'
              ^
SyntaxError: invalid syntax
>>> say = 'let\'s go'
>>> print say
let's go
>>> say = 'hello python\n'
>>> print say
hello python

>>> say = '\thello python\t'
>>> print say
        hello python
```

01 字符串的创建和赋值

三重引号

作用一：进行多行注释

Python中单行注释是#，多行注释的时候每行都写一个#，或者采用连续的三个双引号。

```
# 这是单行注释

"""
这是多行注释第一行
这是多行注释第二行
这是多行注释第三行
"""
```

作用二：定义多行字符串

为避免使用转义换行符 \n，通常会用在定义SQL语句的表达式中没有变量的时候使用.

```
# SQL建表语句
sql_create_table = """CREATE TABLE CUSTOMER (
         FULL_NAME  CHAR(20) NOT NULL,
         AGE INT,
         SEX CHAR(1),
         BALANCE FLOAT )"""
```

01 字符串的创建和赋值

删除字符和字符串(了解即可)

字符串是**不可变的**,只能通过赋一个空字符串或者使用 del 语句来清空或者删除一个字符串

但是没有必要显式的删除字符串。定义这个字符串的**代码结束**时会**自动释放**这些字符串

```
aString = ''
del aString
```

02 字符串的基本特性

连接符(+), 重复操作符(*)，计算长度

连接操作符： 从原有字符串获得一个新的字符串

重复操作符: 创建一个包含了原有字符串的多个拷贝的新串

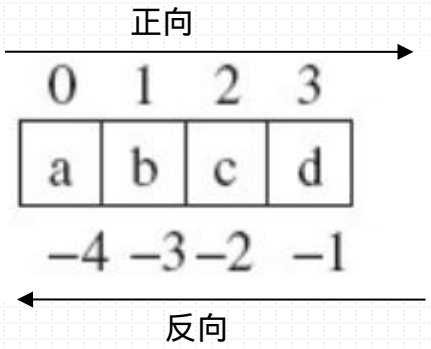
"*"*10 + "学生管理系统" + "*"*10

```
>>> print '===='*10
========================================
>>> print 'hello'+'world'
helloworld
>>> print 'hello'+' world'
hello world
>>> s = 'hello world'
>>> len(s)
11
```

02 字符串的基本特性

序列操作符(索引与切片)

1. 索引(s[i])：获取特定偏移的元素
2. 索引的分类:正向索引, 反向索引



02 字符串的基本特性

序列操作符(索引与切片)

1. 切片S[i:j]提取对应的部分作为一个序列：
2. 如果没有给出切片的边界，切片的下边界默认为0，上边界为字符串的长度;

扩展的切片S[i:j:k],其中i,j含义同上，k为递增步长;

3. s[:]获取从偏移量为0到末尾之间的元素，是实现有效拷贝的一种方法;
4. s[::-1]是实现字符串反转的一种方法;

```
>>> s = 'hello'
>>> s[1:3]
'el'
>>> s[1:]
'ello'
>>> s[:3]
'hel'
>>> s[:-1]
'hell'
>>> s[:]
'hello'
```

02 字符串的基本特性

成员操作符(in ,not in)

成员操作符用于判断一个字符或者一个子串(中的字符)是否出现在另一个字符串中。

出现则返回 True,否则返回 False.

```
>>> s = 'westos'
>>> 's' in s
True
>>> 'hel' in s
False
>>> 'hel' not in s
True
```

02 字符串的基本特性

成员操作符(in ,not in)

string 模块预定义的字符串：

string.ascii_letters

string.ascii_lowercase

string.ascii_uppercase

string.digits

string.whitespace

string.punctuation

02 字符串的基本特性

验证回文串

LeetCode

给定一个字符串，验证它是否是回文串，只考虑字母和数字字符，可以忽略字母的大小写。

说明：本题中，我们将空字符串定义为有效的回文串。

示例 1:

```
输入: "A man, a plan, a canal: Panama"
输出: true
```

示例 2:

```
输入: "race a car"
输出: false
```

02 字符串的基本特性

标识符合法性检查

编写一个检查 Python 有效标识符的小脚本,名字是 idcheck.py。

要求:Python 标识符必须以字母或下划线开头

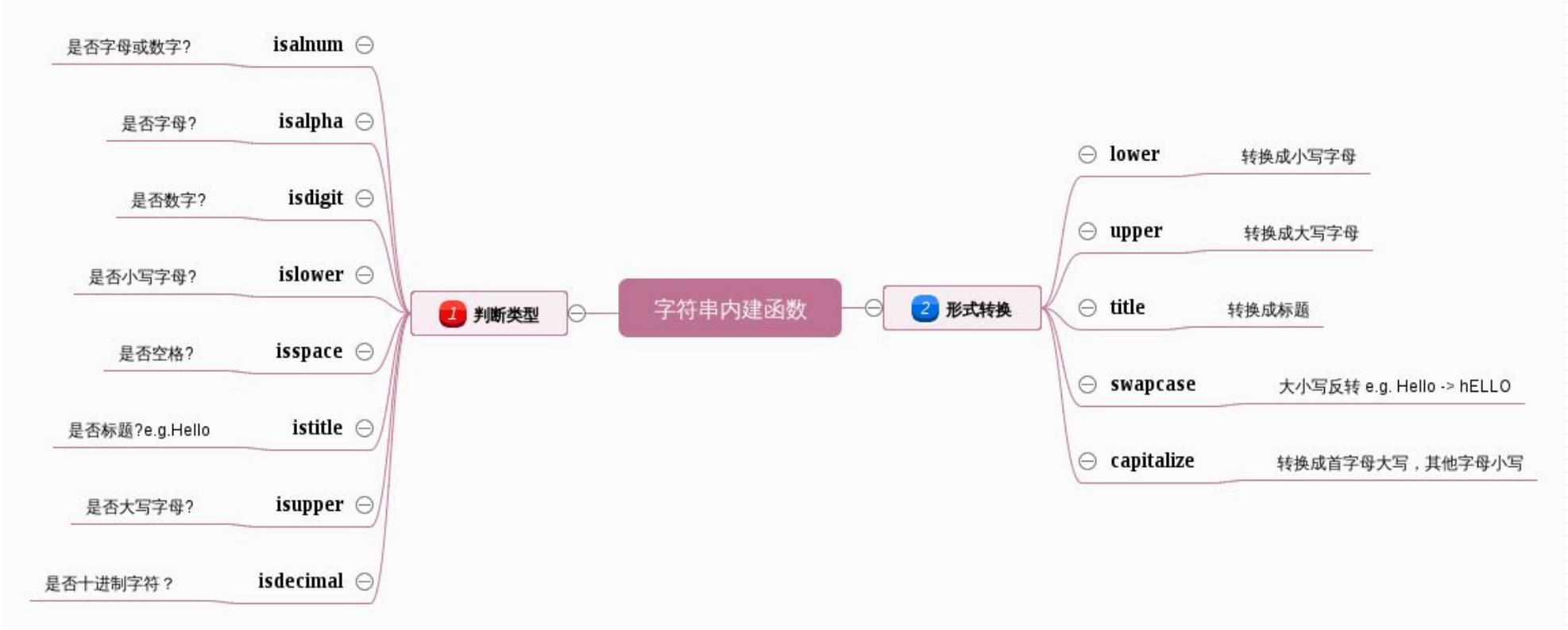
1). 只检查长度大于等于 2 的标识符

2). 以字母或者下划线开始

3). 后面要跟字母,下划线或者或数字

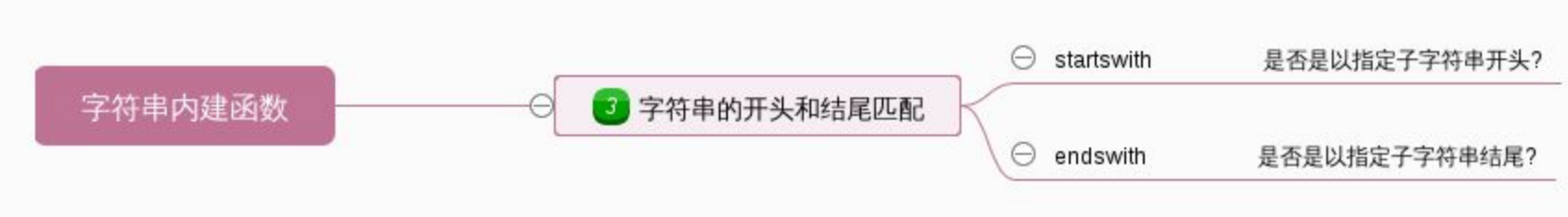
03 字符串内建函数

字符串的判断与转换



03 字符串内建函数

字符串的开头和结尾匹配



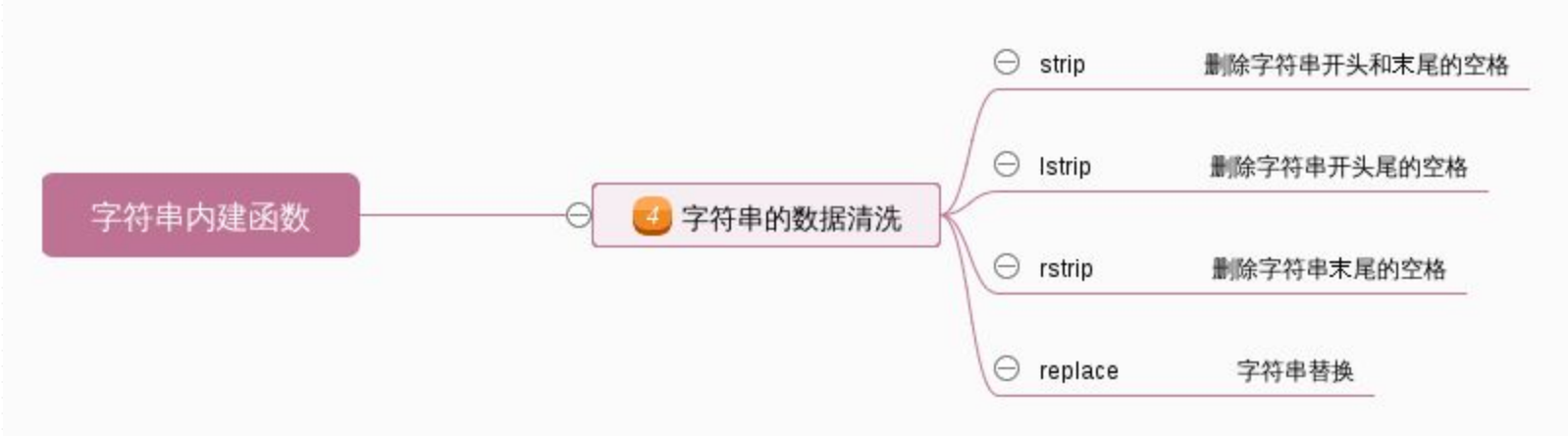

```
filename = "hello.log"
if filename.endswith(".log"):
        print(filename)
else:
        print("error file")


url1 = "file:///mnt"
url2 = "ftp://172.25.254.250/pub/"
url3 = "http://172.25.254.250/index.html"


if url1.startswith("http://"):
        print("爬取网页.......")
else:
        print("不能爬取网页")
```

03 字符串内建函数

字符串的数据清洗



03 字符串内建函数

字符串的位置调整

03 字符串内建函数

字符串的搜索,统计

03 字符串内建函数

字符串的分离与拼接

03 字符串内建函数

小练习

编写一个函数来验证输入的字符串是否是有效的 IPv4 ？

1). IPv4 地址由十进制数和点来表示，每个地址包含4个十进制数，其范围为 0 - 255， 用(".")分割。

比如，172.16.254.1；

2). IPv4 地址内的数不会以 0 开头。比如，地址 172.16.254.01 是不合法的。

示例 1:

输入: "172.16.254.1"

输出: "IPv4"

示例 3:

输入: "256.256.256.256"

输出: "Neither"

03 字符串内建函数

其他内置方法

cmp()	根据字符串的 ASCII 码值进行比较
len()	返回字符串的字符数
max() and min()	返回最大或者最小的字符,(按照 ASCII 码值排列)
enumerate()	枚举对象同时列出数据和数据下标
zip()	将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表

```
# 枚举： 返回索引值和对应的value值;
>>> for i,v in enumerate('hello'):
...     print(str(i) +" -----> " + v)
...
0 -----> h
1 -----> e
2 -----> l
3 -----> l
4 -----> o
```

```
# zip
>>> s1 = 'abc'
>>> s2 = "123"
>>> for i in zip(s1,s2):
...     print(i)
...
('a', '1')
('b', '2')
('c', '3')
>>> for i in zip(s1,s2):
...     print("".join(i))
...
a1
b2
c3
>>> for i in zip(s1,s2):
...     print("-".join(i))
...
a-1
b-2
c-3
```

04 字符串相关模块

感兴趣自己下去了解

string　　字符串操作相关函数和工具

```
>>> from string import Template
>>> s = Template('$who likes $what')
>>> s.substitute(who='tim', what='kung pao')
'tim likes kung pao'
```

base64　　一种“防君子不防小人”的编码方式，用于加密

```
>>> import base64
>>> encoded = base64.b64encode(b'data to be encoded')
>>> encoded
b'ZGF0YSB0byBiZSBlbmNvZGVk'
>>> data = base64.b64decode(encoded)
>>> data
b'data to be encoded'
```

05 本节练习一

检测大写字母 detect-capital

LeetCode

给定一个单词，你需要判断单词的大写使用是否正确。

我们定义，在以下情况时，单词的大写用法是正确的：

1. 全部字母都是大写，比如"USA"。
2. 单词中所有字母都不是大写，比如"leetcode"。
3. 如果单词不只含有一个字母，只有首字母大写， 比如 "Google"。

否则，我们定义这个单词没有正确使用大写字母。

示例 1:

```
输入: "USA"
输出: True
```

示例 2:

```
输入: "FlaG"
输出: False
```

注意: 输入是由大写和小写拉丁字母组成的非空单词。

05 本节练习二

学生出勤记录 student-attendance-record

LeetCode

给定一个字符串来代表一个学生的出勤记录，这个记录仅包含以下三个字符：

1. **'A'** : Absent，缺勤
2. **'L'** : Late，迟到
3. **'P'** : Present，到场

如果一个学生的出勤记录中不**超过一个'A'(缺勤)**并且**不超过两个连续的'L'(迟到)**,那么这个学生会被奖赏。

你需要根据这个学生的出勤记录判断他是否会被奖赏。

示例 1:

```
输入: "PPALLP"
输出: True
```

示例 2:

```
输入: "PPALLL"
输出: False
```

05 本节练习三

机器人能否返回原点 robot-return-to-origin

在二维平面上，有一个机器人从原点 (0, 0) 开始。给出它的移动顺序，判断这个机器人在完成移动后是否在 **(0, 0) 处结束**。

移动顺序由字符串表示。字符 move[i] 表示其第 i 次移动。机器人的有效动作有 R（右），L（左），U（上）和 D（下）。如果机器人在完成所有动作后返回原点，则返回 true。否则，返回 false。

注意：机器人"面朝"的方向无关紧要。 "R" 将始终使机器人向右移动一次，"L" 将始终向左移动等。此外，假设每次移动机器人的移动幅度相同。

示例 1:

输入: "UD"
输出: true
解释：机器人向上移动一次，然后向下移动一次。所有动作都具有相同的幅度，因此它最终回到它开始的原点。因此，我们返回 true。

示例 2:

输入: "LL"
输出: false
解释：机器人向左移动两次。它最终位于原点的左侧，距原点有两次 "移动" 的距离。我们返回 false，因为它在移动结束时没有返回原点。

也是某年腾讯笔试编程题(比本题简单但是思路相同)

05 本节练习四

小学生计算能力测试系统

设计一个程序，用来实现帮助小学生进行算术运算练习，它具有以下功能：提供基本算术运算(加减乘)的题目，每道题中的操作数是随机产生的，练习者根据显示的题目输入自己的答案，程序自动判断输入的答案是否正确并显示出相应的信息。最后显示正确率。

思路:

运行程序， 输入测试数字的大小范围

输入测试题目数量

任意键进入测试

系统进行测试并判断对错

系统根据得分情况进行总结，退出程序

感谢聆听！

THANK YOU!