

Python程序设计

字符串 - 基本方法

刘安
苏州大学, 计算机科学与技术学院

<http://web.suda.edu.cn/anliu/>

字符串字面量

- 位于单引号或者双引号之间的字符序列

```
>>> s = 'python'
>>> s
'python'
>>> s = "Python"
>>> s
'Python'
>>> s = "python"
SyntaxError: EOL while scanning string literal
```

- 也可以位于三引号之间, 此时可以跨越多行

```
>>> s = '''how
are
you'''
>>> print(s)
how
are
you
```

2

字符串方法

- 字符串和列表类似, 都属于序列对象
 - 很多列表方法都可以用在字符串上, 比如索引、分片
- 字符串也有专属的方法, 请参考下方网址
- 🤖 • <https://docs.python.org/3/library/stdtypes.html#string-methods>
- 列表属于可变类型, 其中的元素可以改变
- 字符串属于**不可变类型**, 任何对字符串的修改操作一定产生一个新的字符串

3

字符串的连接和重复

- Python自动连接相邻的字符串字面量
- 运算符+连接两个字符串
- 运算符*重复字符串

```
>>> s = 'py' 'tho' 'n'
>>> s
'python'
>>> s = 'hi, ' + s
>>> s
'hi, python'
>>> s = 'hi' * 3
>>> s
'hihihi'
```

4

使用split方法对字符串进行分割

- split方法的参数是一个分隔符，用其对字符串进行分割，如果不指定，使用默认值空白字符（空格，换行，制表）
- 返回值是子串构成的列表

```
>>> s = 'aaa bb ccc'
>>> s.split() #没有指定分隔符
['aaa', 'bb', 'ccc']
>>> s
'aaa bb ccc'
>>> s = 'Lutz,Mark'
>>> s.split(',') #分隔符是,
['Lutz', 'Mark']
>>> s = 'docsdotpythondotcom'
>>> s.split('dot') #分隔符是dot
['docs', 'python', 'com']
```

5

使用join方法对子串进行合并

- 在一个字符串上调用join方法，注意该字符串是分隔符
- 参数是一个列表，返回值是用分隔符将列表中的子串连接起来的字符串

```
>>> L = list('python')
>>> L
['p', 'y', 't', 'h', 'o', 'n']
>>> s = ''.join(L) #分隔符是空字符
>>> s
'python'
>>> L = ['docs', 'python', 'com']
>>> s = 'dot'.join(L)
>>> s
'docsdotpythondotcom'
```

🤔 list函数接受一个序列并用该序列的所有元素构造一个列表
该序列可以是range、列表、字符串等

6

变位词 (anagram)

- 变位词是通过重新排列一个单词的字母形成的新单词，比如date、eat、tea就是一组变位词
- 如何判断两个词是同一组变位词
 - 两个单词含有的字母完全一样，分别把两个单词的字母升序排列，如果完全一致，它们就是同一组变位词

```
>>> sorted('eat')      字符串是不可变类型，所以没有sort方法 🤔
['a', 'e', 't']      sorted函数可以对可迭代对象的所有元素进行排序
>>> sorted('tea')      并返回包含这些元素的有序列表
['a', 'e', 't']
>>> sorted('dealer') == sorted('leader')
True
>>> sorted('leader') == sorted('ladder')
False
```

🤖 两个列表是如何比较大小的?

7

in/count/find/replace

- 运算符in测试一个子串是否存在
- count方法返回子串的出现次数（子串不允许重叠）
- find方法返回子串第一次出现的索引
- replace方法用新子串替换所有的旧子串

```
>>> s = 'docsdotpythondotcom'
>>> 'dot' in s
True
>>> s.count('dot')
2
>>> s.find('dot')
4
>>> s.replace('dot', '.')
'docs.python.com'
>>> s
'docsdotpythondotcom'
```

😎 字符串属于不可变类型，所以即便是调用replace方法后，也没有修改原字符串

8

部分常用的字符串方法

- 请自行了解下列字符串方法的功能和用法
 - isalnum、isalpha、isdecimal、isdigit、isnumeric
 - islower、isupper、lower、upper
 - startswith、endswith
 - strip、lstrip、rstrip
- 参考网址
 -  <https://docs.python.org/3/library/stdtypes.html#string-methods>

字符串和数值之间的转换

- +运算符：实现数值的相加以及字符串的连接
- 不能直接将字符串和数值进行相加

```
>>> 1 + '42'
Traceback (most recent call last):
  File "<pyshell#151>", line 1, in <module>
    1 + '42'
TypeError: unsupported operand type(s) for +:
'int' and 'str'
```
- int、float、eval函数将字符串转换成数值
- str函数将数值转换成其字符串表示

```
>>> 1 + int('42') + float('3.14')

46.14
```

单个字符和ASCII码之间的转换

- ord函数将单个字符转换成其对应的ASCII码
- chr函数将整数（有效的ASCII码）转换成其对应的字符

```
>>> n = ord('a')
>>> n
97
>>> for i in range(26):
    print(chr(n+i), end = ' ')

abcdefghijklmnopqrstuvwxyz
```

A 65	N 78	a 97	n 110
B 66	O 79	b 98	o 111
C 67	P 80	c 99	p 112
D 68	Q 81	d 100	q 113
E 69	R 82	e 101	r 114
F 70	S 83	f 102	s 115
G 71	T 84	g 103	t 116
H 72	U 85	h 104	u 117
I 73	V 86	i 105	v 118
J 74	W 87	j 106	w 119
K 75	X 88	k 107	x 120
L 76	Y 89	l 108	y 121
M 77	Z 90	m 109	z 122

Pig Latin (儿童黑话)

- Pig Latin是一种英语语言游戏，在英语上使用一些规则使得发音改变，多半被儿童用来瞒着大人沟通秘密
- 常见规则：1) 如果一个单词以元音字母开头，直接在该单词末尾加上way (one -> oneway)；2) 如果一个单词以辅音字母开头，将该辅音字母移至单词末尾，然后在其后加上ay (be -> ebay)
- 挑战规则：1) 如果一个单词开头的辅音字母不止一个，将这些连续的辅音字母移至单词末尾，然后在其后加上ay (string -> ingstray)；2) 对字母y的判断：如果y后面是元音字母，y被看成辅音字母；反之，y被看成元音字母

Pig Latin简单版

- 编写一个函数pig_latin_simple, 接受一个字符串s, 使用常见规则对其进行处理, 返回处理以后得到的字符串。如果s为空, 返回空字符串。不考虑大小写

```
1 def pig_latin_simple(s):
2     vowels = 'aeiou'
3     if not s:
4         return s
5     else:
6         if s[0] in vowels:
7             return s + 'way'
8         else:
9             return s[1:] + s[0] + 'ay'
```

13

Pig Latin挑战版

- 编写一个函数pig_latin, 接受一个字符串s, 使用常见规则和挑战规则对其进行处理, 返回处理以后得到的字符串。如果s为空, 返回空字符串。不考虑大小写
- 如果一个单词开头的辅音字母不止一个, 将这些连续的辅音字母移至单词末尾, 然后在其后加上ay (string -> ingstray)
- 如何找出连续的辅音字母

string tring ring ing

首字母辅音, 考察剩余部分 (递归情况)
首字母元音, 忽略剩余部分 (基本情况)

14

Pig Latin挑战版

- 如何找出连续的辅音字母

```
1 def initial_consonants(s):
2     vowels = 'aeiou'
3     if not s:
4         return ''
5     elif s[0] in vowels:
6         return ''
7     else:
8         return s[0] + initial_consonants(s[1:])
```

string tring ring ing

首字母辅音, 考察剩余部分 (递归情况)
首字母元音, 忽略剩余部分 (基本情况)

15

Pig Latin挑战版

- 1) 如果一个单词以元音字母开头, 直接在该单词末尾加上way; 2) 如果一个单词以辅音字母开头, 将该辅音字母移至单词末尾, 然后在其后加上ay; 3) 如果一个单词开头的辅音字母不止一个, 将这些连续的辅音字母移至单词末尾, 然后在其后加上ay; 4) 对字母y的判断: 如果y后面是元音字母, y被看成辅音字母; 反之, y被看成元音字母
- 空字符串: 直接返回
- 元音字母 and 元音y: 末尾加way
- 其他情况: 通过辅助函数求出开头的辅音字母串, 将其移至末尾, 再加ay

16

Pig Latin挑战版

```

1 def pig_latin(s):
2     vowels = 'aeiou'
3     if not s:
4         return s
5     elif s[0] in vowels:
6         return s + 'way'
7     elif s[0] == 'y' and s[1] not in vowels:
8         return s + 'way'
9     else:
10        prefix = initial_consonants(s)
11        idx = len(prefix)
12        return s[idx:] + prefix + 'ay'

```

 针对某些特定的输入，上述函数不能正常工作！！
找出这样的输入，分析原因并给出解决方案

17

凯撒加密

- 加密的基本模型



- 凯撒加密：密钥k是0~25的整数，对于每个英文字母，将其在字母表中后移k个位置，得到的英文字母作为其密文

- 假设密钥k为3

a	b	c	d	e	f	g	h	i	j	k	l	m	n
d	e	f	g	h	i	j	k	l	m	n	o	p	q

o	p	q	r	s	t	u	v	w	x	y	z
r	s	t	u	v	w	x	y	z	a	b	c



18

凯撒加密

- 编写一个函数，接受两个参数，第一个参数是一个字符串（明文），第二个参数是一个位于0到25的整数（密钥），返回凯撒加密得到的密文。注意，仅处理英文字母，区分大小写，非英文字母不做处理
- 如何实现加密：利用字母ASCII码

a	97		100	d
b	98		101	e
w	119	+3	122	z
x	120		123	97+26
y	121		124	98+26
z	122		125	99+26

97	a
98	b
99	c

19

```

1 def encrypt(s, k):
2     res = []
3     for c in s:
4         if c.islower():
5             d = ord(c) + k
6             if d > ord('z'):
7                 d = d - 26
8             res.append(chr(d))
9         elif c.isupper():
10            d = ord(c) + k
11            if d > ord('Z'):
12                d = d - 26
13            res.append(chr(d))
14        else:
15            res.append(c)
16    return ''.join(res)

```

```

>>> encrypt('Alice needs an X-ray exam!', 3)
'Dolph qhgv dq A-udb hadp!'
>>>
>>> encrypt('Caesar cipher? I prefer Caesar salad.', 25)
'Bzdrzq bhogdq? H oqdedq Bzdrzq rzkzc.'

```

20

凯撒加密的另一种实现 不使用字母的ASCII码，利用字母的相对顺序

```
1 def encrypt_v1(s, k):
2     alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
3     res = []
4     for c in s:
5         idx = alphabet.find(c.upper())
6         if idx != -1:
7             idx = idx + k
8             idx = idx % len(alphabet)
9             if c.isupper():
10                res.append(alphabet[idx])
11            else:
12                res.append(alphabet[idx].lower())
13        else:
14            # -1 means c.upper() was not found
15            res.append(c)
16    return ''.join(res)
```

21

回文字符串

- 正读和反读都一样的字符串，比如level, noon
- 编写一个函数，接受一个字符串，判断其是否是回文串，如是，返回True，否则返回False。空串认为是回文串
- 递归情况

l e v e l 是回文串当且仅当

l 和 l 相等 and e v e 是回文串

- 基本情况：字符串为空或者只有一个字符（均为回文串）

22

回文字符串

- 正读和反读都一样的字符串，比如level, noon
- 编写一个函数，接受一个字符串，判断其是否是回文串，如是，返回True，否则返回False。空串认为是回文串

```
1 def is_palindrome(s):
2     n = len(s)
3     if n <= 1:
4         return True
5     else:
6         a, *b, c = s #first, rest, last
7         return a == c and is_palindrome(b)
```

23

格式化字符串

- 在单个步骤中对一个字符串执行多个特定类型的替换
- C语言printf风格的字符串格式化方法（本课件简单介绍）
 - <https://docs.python.org/3/library/stdtypes.html#old-string-formatting>
- 字符串对象的format方法（自行学习）
 - <https://docs.python.org/3/library/stdtypes.html#str.format>
 - <https://docs.python.org/3/library/string.html#formatstrings>
 - <https://docs.python.org/3/library/string.html#string-formatting>

24

printf风格的字符串格式化

- 使用二元运算符% (没错, 就是那个求余运算符😁)
- 在运算符%的左侧是需要进行格式化的字符串, 其中有一个或者多个待转换的目标, 以%开头 (比如%d)
- 在运算符%的右侧是一个对象或者多个嵌入到元组中的对象, 这些对象将会插入到左侧需要格式化的字符串, 替换其中的待转换目标

```
>>> 'hello, %s' % 'world'
'hello, world'
>>> '%d %s %g' % (1, 'hi', 3.142)
'1 hi 3.142'
>>> '%s -- %s -- %s' % (3.142, [1, 2, 3], {'a': 1, 'b': 2})
"3.142 -- [1, 2, 3] -- {'a': 1, 'b': 2}"
```

25

printf风格的字符串格式化

- 常用的类型码tcode
 - s : 字符串, d : 十进制数字, f : 十进制浮点数
- %号和类型码之间的结构 - %[flag][width][.precision]tcode
 - flag : - (左对齐), 0 (零填充), + (数值符号)
 - width : 总的最小字段宽度
 - precision : 浮点数小数点后面的位数

```
>>> '%d|%-6d|06d' % (1234, 1234, 1234)
'1234|1234 |001234'
>>>
>>> '%-6.2f|05.2f|+06.1f' % (1.2345, 1.2345, 1.2345)
'1.23 |01.23|+001.2'
```

26