Python 程序设计基础

Python Programming

字符编码

- Unicode 编码是目前最广泛使用的字符编码。
- Unicode 编码的实现方式称为 Unicode 转换格式（ Unicode Transformation Format ， UTF ）。
- UTF-8（ 8-bit Unicode Transformation Format ）是在互联网上使用最广泛的一种 Unicode 编码的实现方式。
- UTF-8 是一种针对 Unicode 的可变长度字符编码，它可以用来表示 Unicode 标准中的任何字符，且其编码中的第一个字节仍与 ASCII 兼容，这使得原来处理 ASCII 字符的软件无须或只须做少部份修改，即可继续使用。

字符编码

- **Python 3 默认使用 UTF-8 编码。**

```
>>> import sys
>>> print(sys.getdefaultencoding())
utf-8
>>>
```

字符串

- **字符串是一个字符序列。**
- **字符串字面量可以表示为是以单引号 ' 或双引号 " 括起来的一个字符序列。**

  起始和末尾的引号必须是一致的（要么是两个双引号，要么是两个单引号）

```
>>> print("Welcome to Python")
Welcome to Python
>>> print('Programming is fun')
Programming is fun
>>>
```

字符串

- 单引号可以出现在由双引号包围的字符串中。双引号可以出现在由单引号包围的字符串中。

```
>>> print("What's your name?")
What's your name?
>>> print('He said, "Python program is easy to read"')
He said, "Python program is easy to read"
>>>
```

- 若字符串内部既包含 ' 又包含 " ，可以用转义字符来标识。
- 还可以使用连续三个单引号 ''' 或三个双引号 """ 创建字符串字面量，多用于创建多行字符串。

字符串

- **字符串是对象。当将一个字符串字面量赋值给变量时，就会为这个字符串字面量创建新对象，然后将这个新对象的引用赋值给这个变量。**

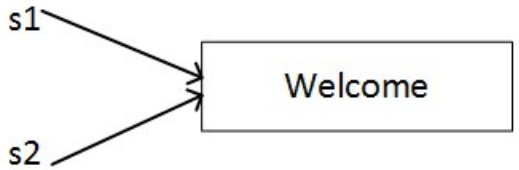
```
>>> s1 = ''
>>> type(s1)
<class 'str'>
>>> s2 = "Python"
>>> type(s2)
<class 'str'>
>>>
```

```
>>> s1 = str()
>>> type(s1)
<class 'str'>
>>> s2 = str("Python")
>>> type(s2)
<class 'str'>
>>> s3 = str(123)
>>> type(s3)
<class 'str'>
>>>
```

字符串

- 为了优化性能，减少字符串对象的重复创建，引入了字符串常量池。
- 创建字符串对象时，首先会对这个字符串进行检查。如果字符串常量池中存在相同内容的字符串对象，则返回该对象的引用；否则新的字符串对象被创建，然后将这个字符串对象放入字符串常量池中，并返回该对象的引用。

```
>>> s1 = "Welcome"
>>> s2 = "Welcome"
>>> id(s1)
2077797526752
>>> id(s2)
2077797526752
```



  - s1 和 s2 指向字符串常量池中同一个字符串对象，它们都有相同的 id 。

字符串的基本操作

- **通过下标访问字符串中的字符。**
  - **s = "Welcome" ，下标从 0 开始。字符串 s 的下标范围从 0 到 len(s)-1 ，即 0 到 6 。**
  - **字符串变量名 [下标] 或字符串字面量 [下标] 来访问字符串中的字符。例如， s[0] 是字符串 s 的第一个字符，而 s[6] 是字符串 s 的最后一个字符。**
  - **“ IndexError” 异常**

```
>>> s = "Welcome"
>>> s[0]
'W'
>>> s[6]
'e'
>>> "Welcome"[3]
'c'
>>> s[7]
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    s[7]
IndexError: string index out of range
>>>
```

字符串的基本操作

- **允许使用负数作为下标来引用相对于字符串末端的位置。将字符串长度和负数下标相加就可以得到实际的位置。**

```
>>> "Welcome"[-1]
'e'
>>> "Welcome"[-1 + len("Welcome")]
'e'
>>> "Welcome"[-7]
'W'
>>> "Welcome"[-7 + len("Welcome")]
'W'
>>>
```

- **字符串是不可变对象，不能通过下标改变它的内容**

```
>>> s = "Welcome"
>>> s[0] = 'w'
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    s[0] = 'w'
TypeError: 'str' object does not support item assignment
>>>
```

字符串的基本操作

- **通过切片操作获得字符串的子串**
  - **字符串变量名 [start:end:step] 或字符串字面量 [start:end:step] ，默认情况下 step 为 1 ，返回下标从 start 到 end-1 的字符构成的一个子串。**

```
>>> s = "Welcome"
>>> s[1:4]
'elc'
>>> s[0:6:2]
'Wlo'
>>>
```

  - **start 和 end 可以省略。若省略 start ， start 默认为 0 ；若省略 end ， end 默认为字符串长度；若 start 和 end 都省略，则切片就是整个字符串的一个拷贝。**

```
>>> s = "Welcome"
>>> s[:6]
'Welcom'
>>> s[0:6]
'Welcom'
```

```
>>> s[4:]
'ome'
>>> s[4:len(s)]
'ome'
```

```
>>> s[::]
'Welcome'
>>> s[::2]
'Wloe'
>>>
```

字符串的基本操作

- 若 start 大于或等于 end ，将返回一个空字符串。若 end 指定了一个超出字符串末尾的位置，将使用字符串长度替代 end 。

```
>>> s = "Welcome"
>>> s[3:3]
''
>>> s[2:10]
'lcome'
>>> s[2:len(s)]
'lcome'
>>>
```

- 切片也可以使用负数下标。

```
>>> s = "Welcome"
>>> s[1:-1]
'elcom'
>>> s[1:-1 + len(s)]
'elcom'
>>>
```

字符串的基本操作

- **下面的切片操作将字符串反转（逆序）。**

```
>>> s = "Welcome"
>>> s
'Welcome'
>>> s[::-1]
'emocleW'
>>>
```

字符串的基本操作

- **运算符**
  - **使用 * 运算符以给定的次数重复一个字符串。**

```
>>> s = "Welcome"
>>> 3 * s
'WelcomeWelcomeWelcome'
>>> s * 3
'WelcomeWelcomeWelcome'
>>> 'a' * 4
'aaaa'
>>> 4 * 'a'
'aaaa'
>>>
```

  - **使用 in 或 not in 运算符来判断一个字符串是否在另一个字符串中。**

```
>>> s = "Welcome to Python"
>>> "Python" in s
True
>>> "come" not in s
False
>>>
```

字符串的基本操作

- **使用 is 或 is not 来判断两个字符串是否是同一个对象。**

```
>>> s1 = "Welcome"
>>> s2 = "Welcome"
>>> id(s1)
1461213127600
>>> id(s2)
1461213127600
>>> s1 is s2
True
>>>
```

- **可以使用关系运算符对字符串进行比较。通过比较字符串中对应的字符（字典顺序）决定字符串大小。**

```
>>> s1 = "green"
>>> s2 = "glow"
>>> s1 > s2
True
>>>
```

字符串的基本操作

- **遍历字符串。**
  - **使用简捷 for 语句。**

```
>>> s = "Welcome"
>>> for ch in s:
        print(ch, end=' ')


W e l c o m e
>>>
```

  - **使用 for 语句，结合内置函数 range 和 len ，通过下标访问。**

```
>>> s = "Welcome"
>>> for i in range(len(s)):
        print(s[i], end=' ')


W e l c o m e
>>>
```

字符串的基本操作

- **测试字符串**
  - **isalnum() 方法。若字符串中至少有一个字符且所有字符是由字母数字组成的，返回 True ，否则返回 False 。**

```
>>> "CS101".isalnum()
True
```

  - **isalpha() 方法。若字符串中至少有一个字符且所有字符是由字母组成的，返回 True ，否则返回 False 。**

```
>>> "Welcome".isalpha()
True
```

  - **isdigit() 方法。若字符串中至少有一个字符且所有字符是由数字组成的，返回 True ，否则返回 False 。**

```
>>> "2017".isdigit()
True
```

字符串的基本操作

- **isidentifier() 方法。若字符串符合 Python 标识符规则，返回 True ，否则返回**

```
>>> "radius".isidentifier()
True
>>> "100_bottles".isidentifier()
False
```

- **islower() 方法。若字符串中至少有一个区分大小写的字符且这些字符全是小写的，返**

```
>>> "python is fun".islower()
True
```

- **isupper() 方法。若字符串中至少有一个区分大小写的字符且这些字符全是大写的，**

```
>>> "HELLO PYTHON".isupper()
True
```

字符串的基本操作

- **isspace() 方法。若字符串中只包含空白字符，返回 True ，否则返回 False 。**

```
>>> "  \n  \t  ".isspace()
True
```

字符串的基本操作

- **转换字符串**
  - **capitalize() 方法。返回第一个单词首字母大写的新字符串。**

```
>>> s = "welcome to python".capitalize()
>>> s
'Welcome to python'
```

  - **title() 方法。返回每个单词首字母大写的新字符串。**

```
>>> s = "welcome to python".title()
>>> s
'Welcome To Python'
```

  - **swapcase() 方法。返回小写字母变成大写字母、大写字母变成小写字母后的新字符串。**

```
>>> "Python".swapcase()
'pYTHON'
```

字符串的基本操作

- **replace(old, new[, count]) 方法。返回用 new 替换 old 后的新字符串。 count 可选，若指定了 count ，则 new 替换 old 最多 count 次。**

```
>>> "Old China".replace("Old", "New")
'New China'
>>> "This is string example...wow!!! This is really string".replace("is", "was", 3)
'Thwas was string example...wow!!! Thwas is really string'
```

字符串的基本操作

- **删除字符串中的空白**
  - **lstrip([chars]) 方法。 chars 可选，返回去掉左端空白字符或 chars 字符的新字符串。**
  - **rstrip([chars]) 方法。 chars 可选，返回去掉右端空白字符或 chars 字符的新字符串。**
  - **strip([chars]) 方法。 chars 可选，返回去掉左右两端空白字符或 chars 字符的新字符串**

```
>>> s = "  Welcome to Python\n  \t"
>>> s.lstrip()
'Welcome to Python\n  \t'
>>> s.rstrip()
'  Welcome to Python'
>>> s.strip()
'Welcome to Python'
>>> s = "0000000This is string...wow!!!0000000"
>>> s.strip("0!")
'This is string...wow'
```

字符串的基本操作

- **格式化字符串**
  - **center(width[, fillchar]) 方法。 fillchar 可选，默认以空格填充，返回在给定宽度 width 上居中对齐的新字符串。**
  - **ljust(width[, fillchar]) 方法。 fillchar 可选，默认以空格填充，返回在给定宽度 width 上左对齐的新字符串。**
  - **rjust(width[, fillchar]) 方法。 fillchar 可选，默认以空格填充，返回在给定宽度 width 上右对齐的新字符串。**

```
>>> s = "Python"
>>> s.center(10, '*')
'**Python**'
>>> s.ljust(10, '*')
'Python****'
>>> s.rjust(10, '*')
'****Python'
```

字符串的基本操作

- **搜索子串**
  - endswith(suffix[, start[, end]]) 方法。 start 和 end 参数可选，用于指定搜索范围。默认情况下， start 为 0 ， end 为字符串长度。若字符串以子串 suffix 结尾返回 True ，否则返回 False 。
  - startswith(prefix[, start[, end]]) 方法。 start 和 end 参数可选，用于指定搜索范围。默认情况下， start 为 0 ， end 为字符串长度。若字符串以子串 prefix 开头返回 True ，否则返回 False 。

```
>>> s = "Welcome to Python"
>>> s.endswith("thon")
True
>>> s.startswith("we")
False
```

字符串的基本操作

- find(sub[, start[, end]) 方法。 start 和 end 参数可选，用于指定搜索范围。默认情况下，start 为 0 ， end 为字符串长度。返回子串 sub 在字符串中首次出现的位置（下标），否则返回 -1 。
- rfind(sub[, start[, end]) 方法。 start 和 end 参数可选，用于指定搜索范围。默认情况下，start 为 0 ， end 为字符串长度。返回子串 sub 在字符串中最后出现的位置（下标），否则返回 -1 。

```
>>> s = "Welcome to Python"
>>> s.find("come")
3
>>> s.find("become")
-1
>>> s.find('o', 5)
9
>>> s.find('o', 10, len(s))
15
```

```
>>> s.rfind('o')
15
>>> s.rfind('o', 0, 5)
4
```

字符串的基本操作

- index(sub[, start[, end]) 方法。类似于 find 方法。返回子串 sub 在字符串中首次出现的位置（下标），否则抛出 "ValueError" 异常。
- rindex(sub[, start[, end]) 方法。类似于 rfind 方法。返回子串 sub 在字符串中最后出现的位置（下标），否则抛出 "ValueError" 异常。
- count(sub[, start[, end]) 方法。 start 和 end 参数可选，用于指定搜索范围。默认情况下，start 为 0 ， end 为字符串长度。返回子串 sub 在字符串中出现的次数。

```
>>> s = "Welcome to Python"
>>> s.count('o')
3
```

字符串的基本操作

- **拼接字符串**
  - **join 方法将一个字符串列表的元素拼接起来。需要在一个分隔符上调用它，并传入一个列表作为参数。**

```
>>> lst = ['I', 'am', 'a', 'student']
>>> ' '.join(lst)
'I am a student'
```

  - **这里分隔符是一个空格， join 方法在单词之间添加一个空格。还可以使用空字符 '' 或其他字符串作为分隔符。**

字符串的基本操作

- **逆序和排序字符串**
  - **s[::-1] 可以将字符串 s 中的字符逆序。**
  - **reversed(seq) 函数是 Python 提供的内置函数。使用 reversed 函数可以将字符串 seq 中的所有字符逆序，返回由逆序后的所有字符构成的一个可迭代对象，原字符串 seq 保持不变。使用 join 方法可以将可迭代对象转换为一个字符串并返回该字符串。**

```
>>> s = "Python"
>>> r_s = reversed(s)
>>> s
'Python'
>>> r_s
<reversed object at 0x000001543722B2E8>
>>> ''.join(r_s)
'nohtyP'
```

字符串的基本操作

- sorted(seq, key=None, reverse=False) 函数是 Python 提供的内置函数。使用 sorted 函数可以将字符串 seq 中的所有字符升序（默认）或降序（ reverse 参数为 True ）排序。若 key 参数为一个函数名，则按该函数指定的规则进行排序。返回由排序后的所有字符构成的一个列表，使用 join 方法可以将列表转换为一个字符串并返回该字符串。原字符串 seq 保持不变。

```
>>> s = "Python"
>>> s_s = sorted(s)
>>> s
'Python'
>>> s_s
['P', 'h', 'n', 'o', 't', 'y']
>>> ''.join(s_s)
'Phnoty'
>>> s_s = sorted(s, reverse=True)
>>> ''.join(s_s)
'ytonhP'
```

例子

- **回文串**
  - **判断输入的一个字符串是否为回文串，若是输出“ Yes”，否则输出“ No”。回文串是指正读和反读都一样的字符串，如 level 。不区分字母的大小写。**
  - **解题思路：将输入的字符串逆序，和原来的字符串比较，若相等，则为回文串。**

```
def is_palindrome(s):
    '''
    参数为字符串
    返回True(回文串)或False(非回文串)
    '''
    return s.lower() == s[::-1].lower()

def main():
    s = input("请输入一个字符串: ")
    print("Yes" if is_palindrome(s) else "No")

main()
```

不区分字母的大小写，将字符串中的字母统一转换为小写。 s[::-1] 将字符串 s 逆序。

```
请输入一个字符串: Level
Yes

请输入一个字符串: abadcba
No
```

例子

- **元音字母的个数**
  - **输入一个字符串，统计并输出该字符串中元音字母的个数（元音字母：A、E、I、O、U）。不区分字母的大小写。**

```
def total_vowels(s):
    '''
    参数为字符串
    返回字符串中元音字母的个数
    '''
    count = 0
    s = s.lower()
    vowels = "aeiou"
    for char in s:
        if char in vowels:
            count = count + 1
    return count

def main():
    s = input("请输入一个字符串: ")
    print(total_vowels(s))

main()
```

不区分字母的大小写，将字符串中的字母统一转换为小写。将元音字母组合成元音字符串 vowels。对于输入字符串中的每个字符，判断是否在元音字符串中，若在则计数器加 1。

```
请输入一个字符串: I am a student.
5
```