Python 程序设计教案

<table>
<tr><td>本次授课内容</td><td colspan="2">7.1 字符串概述
7.2 字符串编码格式
7.3 转义字符与原始字符串
7.4 字符串格式化
7.5 字符串常用方法与操作</td></tr>
<tr><td>本次课的
教学目的</td><td colspan="2">了解 ASCII、UTF-8、GBK、CP936 等常见字符编码格式
了解转义字符和原始字符串的概念和用法
掌握字符串格式化方法 format()的用法
熟练运用字符串常用方法
熟练运用运算符和内置函数对字符串的操作</td></tr>
<tr><td>本次课教学
重点与难点</td><td colspan="2">字符串编码格式
字符串常用方法</td></tr>
<tr><td>教学方法
教学手段</td><td colspan="2">PPT、边讲边练</td></tr>
<tr><td rowspan="3">课堂教学
时间分配</td><td>教学内容</td><td>时间分配（分）</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>课堂教学设计</td><td colspan="2">首先介绍字符串编码格式之间的区别，然后介绍转义字符与原始字符串的概念和用法，接下来介绍字符串格式化的几种方法，最后重点介绍字符串本身提供的常用操作和操作。</td></tr>
<tr><td>实　验</td><td colspan="2">字符串格式化，字符串常用方法</td></tr>
<tr><td>思考题及作业题</td><td colspan="2">本章所有课后习题。</td></tr>
<tr><td>备　注</td><td colspan="2"></td></tr>
<tr><td>教学后记</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">第　一　节　课</td></tr>
</table>

课堂重点内容详解

UTF-8 对全世界所有国家需要用到的字符进行了编码，以 1 个字节表示英语字符(兼容 ASCII)，以 3 个字节表示中文。GB2312 是我国制定的中文编码，使用 1 个字节表示英语，2 个字节表示中文；GBK 是 GB2312 的扩充，而 CP936 是微软在 GBK 基础上开发的编码方式。GB2312、GBK 和 CP936 都是使用 2 个字节表示中文。

Python 3.x 默认使用 UTF8 编码格式，完全支持中文。在统计字符串长度时，无论是一个数字、英文字母，还是一个汉字，都按一个字符对待和处理。

```
>>> s = '中国山东烟台'
>>> len(s)                          #字符串长度，或者包含的字符个数
6
>>> s = '中国山东烟台 ABCDE'     #中文与英文字符同样对待，都算一个字符
>>> len(s)
11
```

除了支持 Unicode 编码的 str 类型之外，Python 还支持字节串类型 bytes，str 类型字符串可以通过 encode()方法使用指定的字符串编码格式编码成为 bytes 对象，而 bytes 对象则可以通过 decode()方法使用指定编码格式解码成为 str 字符串。

```
>>> type('Python 是个好语言')
<class 'str'>
>>> type('山东'.encode('gbk'))       #编码成字节串，采用 GBK 编码格式
<class 'bytes'>
>>> '中国'.encode()                    #默认使用 utf8 进行编码
b'\xe4\xb8\xad\xe5\x9b\xbd'
>>> _.decode()                         #默认使用 utf8 进行解码
'中国'
```

表 7-1 常用转义字符

转义字符	含义
\b	退格，把光标移动到前一列位置
\f	换页符
\n	换行符
\r	回车
\t	水平制表符
\v	垂直制表符
\\	一个斜线\
\’	单引号’

\”	双引号”
\ooo	3 位八进制数对应的字符
\xhh	2 位十六进制数对应的字符
\uhhhh	4 位十六进制数表示的 Unicode 字符

字符串格式化方法 format()提供了更加强大的功能，不要求待格式化的内容和格式字符之间的顺序严格一致，更加灵活。该方法中可以使用的格式主要有 b（二进制格式）、c（把整数转换成 Unicode 字符）、d（十进制格式）、o（八进制格式）、x（小写十六进制格式）、X（大写十六进制格式）、e/E（科学计数法格式）、f/F（固定长度的浮点数格式）、%（使用固定长度浮点数显示百分数）。Python 3.6.x 开始支持在数字常量的中间位置使用单个下划线作为分隔符来提高数字可读性，相应的，字符串格式化方法 format()也提供了对下划线的支持。

```
>>> 1/3
0.3333333333333333
>>> print('{0:.3f}'.format(1/3))          #保留 3 位小数
0.333
>>> '{0:%}'.format(3.5)                    #格式化为百分数
'350.000000%'
>>>  print("The number {0:,} in hex is: {0:#x}, in oct is {0:#o}".format(55))
The number 55 in hex is: 0x37, in oct is 0o67
>>> print("The number {0:,} in hex is: {0:x}, the number {1} in oct is {1:o}".format(5555, 55))
The number 5,555 in hex is: 15b3, the number 55 in oct is 67
>>> print("The number {1} in hex is: {1:#x}, the number {0} in oct is {0:#o}".format(5555, 55))
The number 55 in hex is: 0x37, the number 5555 in oct is 0o12663
>>>  print("my name is {name}, my age is {age}, and my QQ is {qq}".format(name = "Dong", qq = "306467355", age = 38))
my name is Dong, my age is 38, and my QQ is 306467355
>>> position = (5, 8, 13)
>>> print("X:{0[0]};Y:{0[1]};Z:{0[2]}".format(position))
X:5;Y:8;Z:13
>>> '{0:_},{0:_x}'.format(1000000)       #Python 3.6.0 及更高版本支持
'1_000_000,f_4240'
>>> '{0:_},{0:_x}'.format(10000000)      #Python 3.6.0 及更高版本支持
'10_000_000,98_9680'
```

第 二 节 课

7.5.1 find()、rfind()、index()、rindex()、count()

```
>>> s="apple,peach,banana,peach,pear"
>>> s.find("peach")                    #返回第一次出现的位置
```

```
6
>>> s.find("peach", 7)              #从指定位置开始查找
19
>>> s.find("peach", 7, 20)          #在指定范围中进行查找
-1
>>> s.rfind('p')                    #从字符串尾部向前查找
25
>>> s.index('p')                    #返回首次出现位置
1
>>> s.index('pe')
6
>>> s.index('pear')
25
>>> s.index('ppp')                  #指定的子字符串不存在时抛出异常
ValueError: substring not found
>>> s.count('p')                    #统计子字符串出现次数
5
```

7.5.2 split()、rsplit()、partition()、rpartition()

```
>>> s = "apple,peach,banana,pear"
>>> s.split(",")                      #使用逗号进行分隔
["apple", "peach", "banana", "pear"]
>>> s = "2014-10-31"
>>> t = s.split("-")                  #使用指定字符作为分隔符
>>> t
['2014', '10', '31']
>>> list(map(int, t))                 #将分隔结果转换为整数
[2014, 10, 31]
>>> s = '\n\nhello\t\t world \n\n\n My name\t is Dong   '
>>> s.split()
['hello', 'world', 'My', 'name', 'is', 'Dong']
>>> 'a\t\t\tbb\t\tccc'.split('\t')  #每个制表符都被作为独立的分隔符
['a', '', '', 'bb', '', 'ccc']
>>> 'a\t\t\tbb\t\tccc'.split()      #连续多个制表符被作为一个分隔符
['a', 'bb', 'ccc']
>>> 'a,,,bb,,ccc'.split(',')         #每个逗号都被作为独立的分隔符
['a', '', '', 'bb', '', 'ccc']
>>> s = "apple,peach,banana,pear"
>>> s.partition(',')                  #从左侧使用逗号进行切分
('apple', ',', 'peach,banana,pear')
>>> s.rpartition(',')                 #从右侧使用逗号进行切分
('apple,peach,banana', ',', 'pear')
```

7.5.3 join()

```
>>> li = ["apple", "peach", "banana", "pear"]
>>> ","sep.join(li)              #使用逗号作为连接符
"apple,peach,banana,pear"
>>> ':'.join(li)                  #使用冒号作为连接符
'apple:peach:banana:pear'
>>> ''.join(li)                   #使用空字符作为连接符
'applepeachbananapear'
```

7.5.4 lower()、upper()、capitalize()、title()、swapcase()

```
>>> s = "What is Your Name?"
>>> s.lower()                    #返回小写字符串
'what is your name?'
>>> s.upper()                    #返回大写字符串
'WHAT IS YOUR NAME?'
>>> s.capitalize()               #字符串首字符大写
'What is your name?'
>>> s.title()                     #每个单词的首字母大写
'What Is Your Name?'
>>> s.swapcase()                  #大小写互换
'wHAT IS yOUR nAME?'
```

7.5.5 replace()、maketrans()、translate()

```
>>> s = "Python 是一门非常优秀的编程语言"
>>> s.replace('编程', '程序设计')          #两个参数都各自作为整体对待
'Python 是一门非常优秀的程序设计语言'
>>> print('abcdabc'.replace('abc', 'ABC'))
ABCdABC
#创建映射表，将字符"abcdef123"一一对应地转换为"uvwxyz@#$"
>>> table = ''.maketrans('abcdef123', 'uvwxyz@#$')
>>> s = 'Beautiful is better than ugly.'
#按映射表进行转换
>>> s.translate(table)
'Byuutizul is vyttyr thun ugly.'
```

7.5.6 strip()、rstrip()、lstrip()

```
>>> '\n\nhello world   \n\n'.strip()     #删除两侧的空白字符
'hello world'
>>> "aaaassddf".strip("a")                 #删除两侧的指定字符
"ssddf"
>>> "aaaassddfaaa".rstrip("a")             #删除字符串右侧指定字符
'aaaassddf'
>>> "aaaassddfaaa".lstrip("a")             #删除字符串左侧指定字符
'ssddfaaa'
```

7.5.7 startswith()、endswith()

```
>>> s = 'Beautiful is better than ugly.'
>>> s.startswith('Be')            #检测整个字符串
True
>>> s.startswith('Be', 5)         #指定检测范围起始位置
False
>>> s.startswith('Be', 0, 5)      #指定检测范围起始和结束位置
True
>>> import os
>>> [filename
     for filename in os.listdir(r'D:\\')
     if filename.endswith(('.bmp', '.jpg', '.gif'))]
```

7.5.8 isalnum()、isalpha()、isdigit()、isspace()、isupper()、islower()

```
>>> '1234abcd'.isalnum()
True
>>> '\t\n\r '.isspace()            #测试是否全部为空白字符
True
>>> 'aBC'.isupper()                #测试是否全部为大写字母
False
>>> '1234abcd'.isalpha()          #全部为英文字母时返回 True
False
>>> '1234abcd'.isdigit()           #全部为数字时返回 True
False
>>> '1234.0'.isdigit()             #不能测试浮点数
False
>>> '1234'.isdigit()               #只能测试整数
True
```

7.5.9 center()、ljust()、rjust()

```
>>> 'Main Menu'.center(20)          #居中对齐，两侧默认以空格进行填充
'     Main Menu      '
>>> 'Main Menu'.center(20, '-')     #居中对齐，两侧以减号进行填充
'-----Main Menu------'
>>> 'Main Menu'.ljust(20, '#')      #左对齐，右侧以井号进行填充
'Main Menu###########'
>>> 'Main Menu'.rjust(20, '=')      #右对齐，左侧以等号进行填充
'==========Main Menu'
```

7.5.10 字符串支持的运算符

```
>>> 'Hello ' + 'World!'
'Hello World!'
>>> 'abcd' * 3
'abcdabcdabcd'
>>> "a" in "abcde"            #测试一个字符中是否存在于另一个字符串中
True
>>> 'ac' in 'abcde'           #关键字 in 左边的字符串作为一个整体对待
False
```

7.5.11 适用于字符串的内置函数

```
>>> x = 'Hello world.'
>>> len(x)                     #字符串长度
12
>>> max(x)                     #最大字符
'w'
>>> min(x)                     #最小字符
' '
>>> list(zip(x,x))            #zip()也可以作用于字符串
[('H', 'H'), ('e', 'e'), ('l', 'l'), ('l', 'l'), ('o', 'o'), (' ', ' '), ('w', 'w'), ('o', 'o'), ('r', 'r'), ('l', 'l'), ('d', 'd'), ('.', '.')]
>>> sorted(x)                  #对所有字符进行排序，返回列表
```

```
    [' ', '.', 'H', 'd', 'e', 'l', 'l', 'l', 'o', 'o', 'r', 'w']
    >>> ''.join(reversed(x))     #翻转字符串
    '.dlrow olleH'
    >>> list(enumerate(x))       #枚举字符串
    [(0, 'H'), (1, 'e'), (2, 'l'), (3, 'l'), (4, 'o'), (5, ' '), (6,
'w'), (7, 'o'), (8, 'r'), (9, 'l'), (10, 'd'), (11, '.')]
```

7.5.12 字符串切片

```
    >>> 'Explicit is better than implicit.'[:8]
    'Explicit'
    >>> 'Explicit is better than implicit.'[9:23]
    'is better than'
    >>> path = 'C:\\Python35\\test.bmp'
    >>> path[:-4] + '_new' + path[-4:]
    'C:\\Python35\\test_new.bmp'
```

Python 程序设计教案

本次授课内容	7.6 字符串常量 7.7 中英文分词 7.8 汉字到拼音的转换 7.9 综合案例解析	
本次课的 教学目的	了解字符串常量的用法 了解分词扩展库 jieba 的用法 了解拼音扩展库 pypinyin 的用法	
本次课教学 重点与难点	字符串应用	
教学方法 教学手段	PPT、边讲边练	
课堂教学 时间分配	**教学内容**	**时间分配（分）**
课堂教学设计	首先介绍字符串常量、中英文分词、拼音转换的用法，重点讲解几个例题演示字符串的应用。	
实　验	教材例题 7-5 至 7-7	
思考题及作业题	所有课后习题	

备　注	
教学后记	

第　一　节　课

课堂重点内容详解

例 7-5 使用 string 模块提供的字符串常量，模拟生成指定长度的随机密码。

```
from random import choice
from string import ascii_letters, digits

characters = digits + ascii_letters

def generatePassword(n):
    return ''.join((choice(characters) for _ in range(n)))

print(generatePassword(8))
print(generatePassword(15))
```

扩展库 jieba 和 snownlp 可以进行中英文分词。

```
>>> import jieba                          #导入 jieba 模块
>>> x = '分词的准确度直接影响了后续文本处理和挖掘算法的最终效果。'
>>> jieba.cut(x)                          #使用默认词库进行分词
<generator object Tokenizer.cut at 0x000000000342C990>
>>> list(_)
['分词', '的', '准确度', '直接', '影响', '了', '后续', '文本处理', '和',
 '挖掘', '算法', '的', '最终', '效果', '。']
>>> list(jieba.cut('花纸杯'))
['花', '纸杯']
>>> jieba.add_word('花纸杯')          #增加词条
>>> list(jieba.cut('花纸杯'))         #使用新词库进行分词
['花纸杯']
>>> import snownlp                        #导入 snownlp 模块
>>> snownlp.SnowNLP('学而时习之，不亦说乎').words
['学而', '时习', '之', '，', '不亦', '说乎']
>>> snownlp.SnowNLP(x).words
['分词', '的', '准确度', '直接', '影响', '了', '后续', '文本', '处理', '和',
 '挖掘', '算法', '的', '最终', '效果', '。']
```

Python 扩展库 pypinyin 支持汉字到拼音的转换，并且可以和分词扩展库配合使用。

```
>>> from pypinyin import lazy_pinyin, pinyin
>>> lazy_pinyin('董付国')            #返回拼音
['dong', 'fu', 'guo']
>>> lazy_pinyin('董付国', 1)         #带声调的拼音
['dǒng', 'fù', 'guó']
>>> lazy_pinyin('董付国', 2)         #数字表示前面字母的声调
['do3ng', 'fu4', 'guo2']
>>> lazy_pinyin('董付国', 3)         #只返回拼音首字母
['d', 'f', 'g']
>>> lazy_pinyin('重要', 1)           #能够根据词组智能识别多音字
['zhòng', 'yào']
>>> lazy_pinyin('重阳', 1)
['chóng', 'yáng']
>>> pinyin('重阳')                     #返回拼音
[['chóng'], ['yáng']]
```

第 二 节 课

例 7-6 编写函数实现字符串加密和解密，循环使用指定密钥，采用简单的异或算法。

```
def crypt(source, key):
    from itertools import cycle
    func = lambda x, y: chr(ord(x)^ord(y))
    return ''.join(map(func, source, cycle(key)))

source = 'Beautiful is better than ugly.'
key = 'Python'

print('Before Encrypted:'+source)
encrypted = crypt(source, key)
print('After Encrypted:'+encrypted)
decrypted = crypt(encrypted, key)
print('After Decrypted:'+decrypted)
```

例 7-7 编写程序，统计一段文字中每个词出现的次数。

```
from collections import Counter
from jieba import cut

def frequency(text):
    return Counter(cut(text))

text = '''八百标兵奔北坡，北坡八百炮兵炮。
标兵怕碰炮兵炮，炮兵怕把标兵碰。'''
print(frequency(text))
```

例 7-8 检查并判断密码字符串的安全强度。

```
import string
```

```
def check(pwd):
    #密码必须至少包含 6 个字符
    if not isinstance(pwd, str) or len(pwd)<6:
        return 'not suitable for password'

    #密码强度等级与包含字符种类的对应关系
    d = {1:'weak', 2:'below middle', 3:'above middle', 4:'strong'}
    #分别用来标记 pwd 是否含有数字、小写字母、大写字母和指定的标点符号
    r = [False] * 4

    for ch in pwd:
        #是否包含数字
        if not r[0] and ch in string.digits:
            r[0] = True
        #是否包含小写字母
        elif not r[1] and ch in string.ascii_lowercase:
            r[1] = True
        #是否包含大写字母
        elif not r[2] and ch in string.ascii_uppercase:
            r[2] = True
        #是否包含指定的标点符号
        elif not r[3] and ch in ',.!;?<>':
            r[3] = True
    #统计包含的字符种类，返回密码强度
    return d.get(r.count(True), 'error')

print(check('a2Cd,'))
print(check('1234567890'))
print(check('abcdERGj'))
```