目 录

[第1章 Python3 基础语法 1](#_Toc480309857)

[1.1 字符编码 1](#_Toc480309858)

[1.1.1 关于中文 2](#_Toc480309859)

[1.2 变量标识符 3](#_Toc480309860)

[1.2.1 变量定义的规则 3](#_Toc480309861)

[1.2.2 python保留字 3](#_Toc480309862)

[1.3 注释 4](#_Toc480309863)

[1.4 行与缩进 4](#_Toc480309864)

[1.5 多行语句 5](#_Toc480309865)

[1.6 数据类型初识 5](#_Toc480309866)

[1.6.1 数字 5](#_Toc480309867)

[1.6.2 布尔值 6](#_Toc480309868)

[1.6.3 字符串 6](#_Toc480309869)

[1.6.4 列表 7](#_Toc480309870)

[1.6.5 元组(不可变列表) 7](#_Toc480309871)

[1.6.6 字典（无序） 7](#_Toc480309872)

[1.7 空行 8](#_Toc480309873)

[1.8 等待用户输入 8](#_Toc480309874)

[1.9 同一行显示多条语句 8](#_Toc480309875)

[1.10 多个语句构成代码组 8](#_Toc480309876)

[1.11 Print 输出 9](#_Toc480309877)

[1.11.1 三级示例 10](#_Toc480309878)

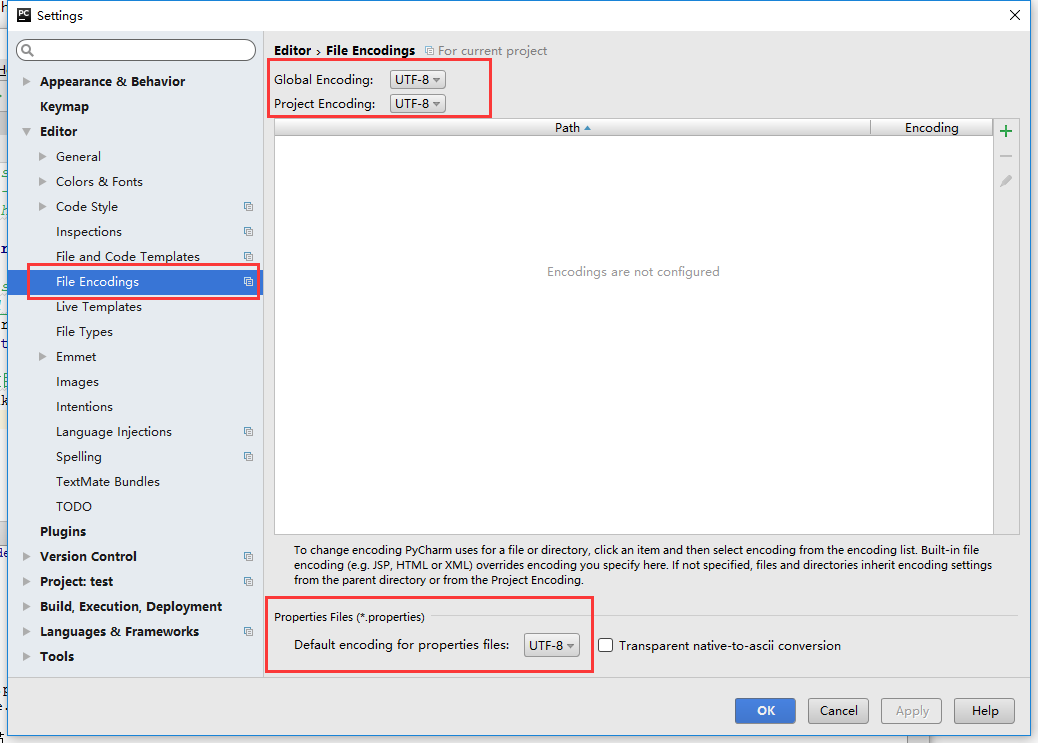
# Python3 基础语法

## 字符编码

默认情况下，Python 3源码文件以 UTF-8 编码，所有字符串都是 unicode 字符串。 当然也可以为源码文件指定不同的编码：

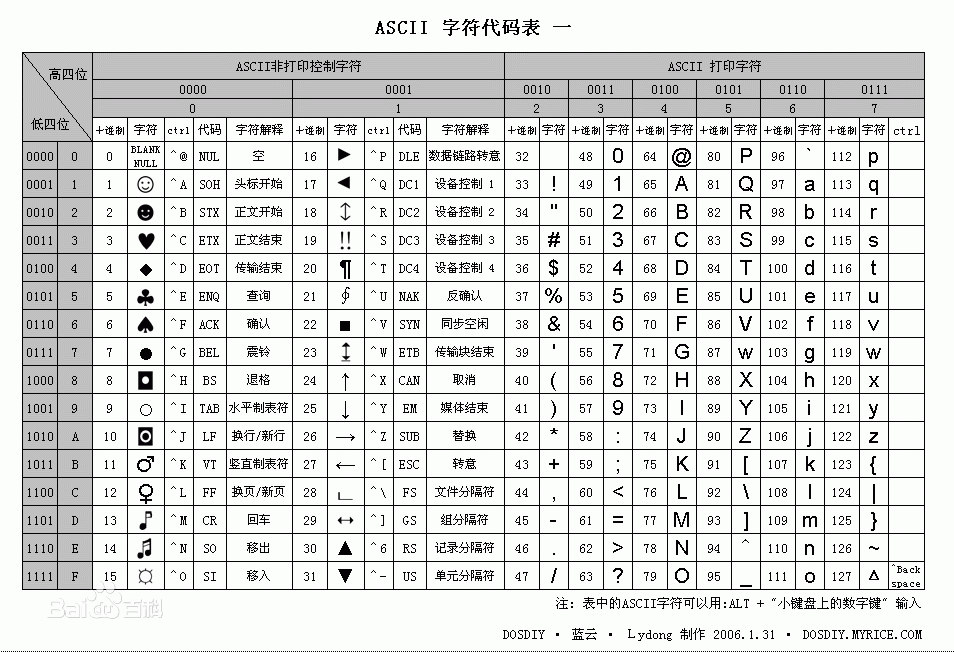
# -\*- coding: cp-1252 -\*-

Windows平台可以在pycharm设置默认编码格式：



python解释器在加载 .py 文件中的代码时，会对内容进行编码（默认ascill）

ASCII（American Standard Code for Information Interchange，美国标准信息交换代码）是基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言，其最多只能用 8 位来表示（一个字节），即：2\*\*8 = 256-1，所以，ASCII码最多只能表示 255 个符号。



### 关于中文

为了处理汉字，程序员设计了用于简体中文的GB2312和用于繁体中文的big5。

GB2312(1980年)一共收录了7445个字符，包括6763个汉字和682个其它符号。汉字区的内码范围高字节从B0-F7，低字节从A1-FE，占用的码位是72\*94=6768。其中有5个空位是D7FA-D7FE。

GB2312 支持的汉字太少。1995年的汉字扩展规范GBK1.0收录了21886个符号，它分为汉字区和图形符号区。汉字区包括21003个字符。2000年的 GB18030是取代GBK1.0的正式国家标准。该标准收录了27484个汉字，同时还收录了藏文、蒙文、维吾尔文等主要的少数民族文字。现在的PC平台必须支持GB18030，对嵌入式产品暂不作要求。所以手机、MP3一般只支持GB2312。

从ASCII、GB2312、GBK 到GB18030，这些编码方法是向下兼容的，即同一个字符在这些方案中总是有相同的编码，后面的标准支持更多的字符。在这些编码中，英文和中文可以统一地处理。区分中文编码的方法是高字节的最高位不为0。按照程序员的称呼，GB2312、GBK到GB18030都属于双字节字符集 (DBCS)。

有的中文Windows的缺省内码还是GBK，可以通过GB18030升级包升级到GB18030。不过GB18030相对GBK增加的字符，普通人是很难用到的，通常我们还是用GBK指代中文Windows内码。

**显然ASCII码无法将世界上的各种文字和符号全部表示，所以，就需要新出一种可以代表所有字符和符号的编码，即：Unicode**

**Unicode（统一码、万国码、单一码）是一种在计算机上使用的字符编码**。Unicode 是为了解决传统的字符编码方案的局限而产生的，它为每种语言中的每个字符设定了统一并且唯一的二进制编码，规定虽有的字符和符号最少由 16 位来表示（2个字节），即：2 \*\*16 = 65536，

注：此处说的的是最少2个字节，可能更多

UTF-8，是对Unicode编码的压缩和优化，他不再使用最少使用2个字节，而是将所有的字符和符号进行分类：ascii码中的内容用1个字节保存、欧洲的字符用2个字节保存，东亚的字符用3个字节保存...

所以，python解释器在加载 .py 文件中的代码时，会对内容进行编码（默认ascill），如果是如下代码的话：

报错：ascii码无法表示中文

#!/usr/bin/env python

print "你好，世界"

改正：应该显示的告诉python解释器，用什么编码来执行源代码，即：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

print "你好，世界"

## 变量标识符

### 变量定义的规则

* 第一个字符必须是字母表中字母或下划线'\_'。
* 标识符的其他的部分有字母、数字和下划线组成。
* 标识符对大小写敏感。

在Python 3中，非-ASCII 标识符也是允许的了。

### python保留字

保留字即关键字，我们不能把它们用作任何标识符名称。Python 的标准库提供了一个 keyword 模块，可以输出当前版本的所有关键字：

>>> import keyword

>>> print(keyword.kwlist)

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

## 注释

　当行注视：# 被注释内容

　多行注释：""" 被注释内容 """,或者是’’’ 被注释内容’’’

## 行与缩进

python最具特色的就是使用缩进来表示代码块，不需要使用大括号({})。

缩进的空格数是可变的，但是同一个代码块的语句必须包含相同的缩进空格数。实例如下：

if True:

print ("True")

else:

print ("False")

以下代码最后一行语句缩进数的空格数不一致，会导致运行错误：

if True:

print ("Answer")

print ("True")

else:

print ("Answer")

print ("False")  **# 缩进不一致，会导致运行错误**

以上程序由于缩进不一致，执行后会出现类似以下错误：

File "test.py", line 6

print ("False") # 缩进不一致，会导致运行错误

^

IndentationError: unindent does not match any outer indentation level #缩进不一致都是报这个错误

## 多行语句

Python 通常是一行写完一条语句，但如果语句很长，我们可以使用反斜杠(\)来实现多行语句，例如：

total = item\_one + \

item\_two + \

item\_three

在 [], {}, 或 () 中的多行语句，不需要使用反斜杠(\)，例如：

total = ['item\_one', 'item\_two', 'item\_three',

'item\_four', 'item\_five']

## 数据类型初识

### 数字

python中数有四种类型：整数、长整数、浮点数和复数。

（1）int 整型

在32位机器上，整数的位数为32位，取值范围为-2\*\*31～2\*\*31-1，即-2147483648～2147483647

在64位系统上，整数的位数为64位，取值范围为-2\*\*63～2\*\*63-1，即-9223372036854775808～9223372036854775807

（2）long（长整型）

跟C语言不同，Python的长整数没有指定位宽，即：Python没有限制长整数数值的大小，但实际上由于机器内存有限，我们使用的长整数数值不可能无限大。

注意，自从Python2.2起，如果整数发生溢出，Python会自动将整数数据转换为长整数，所以如今在长整数数据后面不加字母L也不会导致严重后果了。

（3）float（浮点型）

先扫盲 http://www.cnblogs.com/alex3714/articles/5895848.html

浮点数用来处理实数，即带有小数的数字。类似于C语言中的double类型，占8个字节（64位），其中52位表示底，11位表示指数，剩下的一位表示符号。

（4）complex（复数）

复数由实数部分和虚数部分组成，一般形式为x＋yj，其中的x是复数的实数部分，y是复数的虚数部分，这里的x和y都是实数。

注：Python中存在小数字池：-5 ～ 257

### 布尔值

　　真或假

　　1 或 0

### 字符串

* python中单引号和双引号使用完全相同。
* 使用三引号('''或""")可以指定一个多行字符串。
* 转义符 '\'特殊字符
* 自然字符串， 通过在字符串前加r或R。 如 r"this is a line with \n" 则\n会显示，并不是换行。
* python允许处理unicode字符串，加前缀u或U， 如 u"this is an unicode string"。
* 字符串是不可变的。
* 按字面意义级联字符串，如"this " "is " "string"会被自动转换为this is string。

word = '字符串'

sentence = "这是一个句子。"

paragraph = """这是一个段落，

可以由多行组成"""

万恶的字符串拼接：

python中的字符串在C语言中体现为是一个字符数组，每次创建字符串时候需要在内存中开辟一块连续的空，并且一旦需要修改字符串的话，就需要再次开辟空间，万恶的+号每出现一次就会在内从中重新开辟一块空间。

name = "alex"

print ("i am %s " % name)

#输出: i am alex

**PS: 字符串是 %s;整数 %d;浮点数%f**

**字符串常用功能：**

移除空白

分割

长度

索引

切片

### 列表

创建列表：

name\_list = ['alex', 'seven', 'eric']

或

name\_list ＝ list(['alex', 'seven', 'eric'])

基本操作：

索引

切片

追加

删除

长度

切片

循环

包含

### 元组(不可变列表)

创建元组：

ages = (11, 22, 33, 44, 55)

或

ages = tuple((11, 22, 33, 44, 55))

### 字典（无序）

创建字典：

person = {"name": "mr.wu", 'age': 18}

或

person = dict({"name": "mr.wu", 'age': 18})

常用操作：

索引

新增

删除

键、值、键值对

循环

长度

## 空行

**函数之间或类的方法之间用空行分隔，表示一段新的代码的开始。类和函数入口之间也用一行空行分隔，以突出函数入口的开始。**

空行与代码缩进不同，空行并不是Python语法的一部分。书写时不插入空行，Python解释器运行也不会出错。但是空行的作用在于分隔两段不同功能或含义的代码，便于日后代码的维护或重构。

记住：空行也是程序代码的一部分。

## 等待用户输入

执行下面的程序在按回车键后就会等待用户输入：

#!/usr/bin/python3

input("\n\n按下 enter 键后退出。")

以上代码中 ，"\n\n"在结果输出前会输出两个新的空行。一旦用户按下键时，程序将退出。

## 同一行显示多条语句

Python可以在同一行中使用多条语句，语句之间使用分号(;)分割，以下是一个简单的实例：

#!/usr/bin/python3

import sys; x = 'runoob'; sys.stdout.write(x + '\n')

执行以上代码，输入结果为：

$ python3 test.py

runoob

## 多个语句构成代码组

缩进相同的一组语句构成一个代码块，我们称之代码组。

像if、while、def和class这样的复合语句，首行以关键字开始，以冒号( : )结束，该行之后的一行或多行代码构成代码组。

我们将首行及后面的代码组称为一个子句(clause)。

如下实例：

if expression :

suite

elif expression :

suite

else :

suite

## Print 输出

print 默认输出是换行的，如果要实现不换行需要在变量末尾加上 end=""：

#!/usr/bin/python3

x="a"

y="b"

# 换行输出

print( x )

print( y )

print('---------')

**# 不换行输出**

print( x, end=" " )

print( y, end=" " )

print()

以上实例执行结果为：

a

b

---------

a b

## import 与 from...import

在 python 用 import 或者 from...import 来导入相应的模块。

* 将整个模块(somemodule)导入，格式为： import somemodule
* 从某个模块中导入某个函数,格式为： from somemodule import somefunction
* 从某个模块中导入多个函数,格式为： from somemodule import firstfunc, secondfunc, thirdfunc
* 将某个模块中的全部函数导入，格式为： from somemodule import \*

（1）导入 sys 模块

import sys

print('================Python import mode==========================');

print ('命令行参数为:')

for i in sys.argv:

print (i)

print ('\n python 路径为',sys.path)

（2）导入 sys 模块的 argv,path 成员

from sys import argv,path # 导入特定的成员

print('================python from import===================================')

print('path:',path) # 因为已经导入path成员，所以此处引用时不需要加sys.path

### 三级示例

/usr/sbin/groupadd mysql

/usr/sbin/useradd -g mysql mysql

tar zxf mysql-5.5.3-m3.tar.gz