# Python3

## 参考资料

<http://www.runoob.com/python3/python3-tutorial.html>

## 基础

### 查看版本

python -v

### Hello,World

#!/usr/bin/python3

print("Hello, World!");

（1）如果调用python脚本时，使用:

python script.py

#!/usr/bin/python 被忽略，等同于注释。

（2）如果调用python脚本时，使用:

./script.py

#!/usr/bin/python 指定解释器的路径。

再解释一下第一行代码#!/usr/bin/python3

这句话仅仅在linux或unix系统下有作用，在windows下无论在代码里加什么都无法直接运行一个文件名后缀为.py的脚本，因为在windows下文件名对文件的打开方式起了决定性作用。

#!/usr/bin/env python3

这种用法先在 env（环境变量）设置里查找 python 的安装路径，再调用对应路径下的解释器程序完成操作。

### 编码

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

源文件未utf-8编码。

### 注释

#### 单行注释

# 单行注释以#开头

#### 多行注释

多个#号

‘’’

‘’’

“””

“””

### 缩进

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。

## 基本类型

### Number

#### int (整数)

如 1, 只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。

#### bool (布尔)

如 True。

#### float (浮点数)

如 1.23、3E-2

#### complex (复数)

如 1 + 2j、 1.1 + 2.2j

### 字符串

1. python中单引号和双引号使用完全相同。
2. 使用三引号('''或""")可以指定一个多行字符串。
3. 转义符 '\'
4. 反斜杠可以用来转义，使用r可以让反斜杠不发生转义。。 如 r"this is a line with \n" 则\n会显示，并不是换行。
5. 按字面意义级联字符串，如"this " "is " "string"会被自动转换为this is string。
6. 字符串可以用 + 运算符连接在一起，用 \* 运算符重复。
7. Python 中的字符串有两种索引方式，从左往右以 0 开始，从右往左以 -1 开始。
8. Python中的字符串不能改变。
9. Python 没有单独的字符类型，一个字符就是长度为 1 的字符串。
10. 字符串的截取的语法格式如下：变量[头下标:尾下标]。

### List

列表可以完成大多数集合类的数据结构实现。列表中元素的类型可以不相同，它支持数字，字符串甚至可以包含列表（所谓嵌套）。

列表是写在方括号([])之间、用逗号分隔开的元素列表。

和字符串一样，列表同样可以被索引和截取，列表被截取后返回一个包含所需元素的新列表。

列表截取的语法格式如下：

变量[头下标:尾下标]

索引值以 0 为开始值，-1 为从末尾的开始位置。

加号（+）是列表连接运算符，星号（\*）是重复操作。

#!/usr/bin/python3

list = [ 'abcd', 786 , 2.23, 'runoob', 70.2 ]

tinylist = [123, 'runoob']

print (list) # 输出完整列表

print (list[0]) # 输出列表第一个元素

print (list[1:3]) # 从第二个开始输出到第三个元素

print (list[2:]) # 输出从第三个元素开始的所有元素

print (tinylist \* 2) # 输出两次列表

print (list + tinylist) # 连接列表

### Tuple

元组（tuple）与列表类似，不同之处在于元组的元素不能修改。元组写在小括号 () 里，元素之间用逗号隔开。

元组中的元素类型也可以不相同

#!/usr/bin/python3

tuple = ( 'abcd', 786 , 2.23, 'runoob', 70.2 )

tinytuple = (123, 'runoob')

print (tuple) # 输出完整元组

print (tuple[0]) # 输出元组的第一个元素

print (tuple[1:3]) # 输出从第二个元素开始到第三个元素

print (tuple[2:]) # 输出从第三个元素开始的所有元素

print (tinytuple \* 2) # 输出两次元组

print (tuple + tinytuple) # 连接元组

构造包含 0 个或 1 个元素的元组比较特殊，所以有一些额外的语法规则：

tup1 = () # 空元组

tup2 = (20,) # 一个元素，需要在元素后添加逗号

### Set

集合（set）是一个无序不重复元素的序列。

基本功能是进行成员关系测试和删除重复元素。

可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。

创建格式：

parame = {value01,value02,...}

或者

set(value)

#!/usr/bin/python3

student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}

print(student) # 输出集合，重复的元素被自动去掉

# 成员测试

if('Rose' in student) :

print('Rose 在集合中')

else :

print('Rose 不在集合中')

# set可以进行集合运算

a = set('abracadabra')

b = set('alacazam')

print(a)

print(a - b) # a和b的差集

print(a | b) # a和b的并集

print(a & b) # a和b的交集

print(a ^ b) # a和b中不同时存在的元素

### Dictionary

字典（dictionary）是Python中另一个非常有用的内置数据类型。

列表是有序的对象集合，字典是无序的对象集合。两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。

字典是一种映射类型，字典用"{ }"标识，它是一个无序的键(key) : 值(value)对集合。

键(key)必须使用不可变类型。

在同一个字典中，键(key)必须是唯一的。

#!/usr/bin/python3

dict = {}

dict['one'] = "1 - 菜鸟教程"

dict[2] = "2 - 菜鸟工具"

tinydict = {'name': 'runoob','code':1, 'site': 'www.runoob.com'}

print (dict['one']) # 输出键为 'one' 的值

print (dict[2]) # 输出键为 2 的值

print (tinydict) # 输出完整的字典

print (tinydict.keys()) # 输出所有键

print (tinydict.values()) # 输出所有值

构造函数 dict() 可以直接从键值对序列中构建字典。另外，字典类型也有一些内置的函数，例如clear()、keys()、values()等。

### 不可变数据

Number（数字）、String（字符串）、Tuple（元组）、Sets（集合）。

### 可变数据

List（列表）、Dictionary（字典）。

### 数据类型转换



## 变量

Python 中的变量不需要声明。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。

在 Python 中，变量就是变量，它没有类型，我们所说的"类型"是变量所指的内存中对象的类型。

a =1

a = b = c = 1

a, b, c = 1, 2, "runoob"

### 交互模式变量\_

在交互模式中，最后被输出的表达式结果被赋值给变量 \_ 。

## 运算符

### 算术运算符



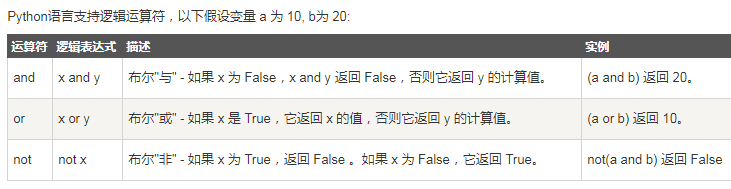
### 比较（关系）运算符



### 赋值运算符



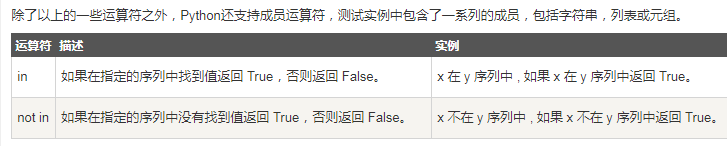
### 逻辑运算符



### 位运算符



### 成员运算符



### 身份运算符



### 运算符优先级



## 输出

### print输出

# 换行输出

print( x )

# 不换行输出

print( x, end=" " )

## 模块管理

### import

将整个模块(somemodule)导入，格式为： import somemodule。

### form ... import ...

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 \*

## 函数

### 定义

def 函数名（参数列表）:

函数体

## 常用函数

### type

a, b, c, d = 20, 5.5, True, 4+3j

print(type(a), type(b), type(c), type(d))

type()不会认为子类是一种父类类型。

### isinstance

a = 111

isinstance(a, int)

isinstance()会认为子类是一种父类类型。

### del

使用del语句删除一些对象引用。

del语句的语法是：

del var1[,var2[,var3[....,varN]]]]

## 流程控制语句

### 条件控制

### 循环控制

## 迭代器和生成器

### 迭代器iter

### 生成器yield

# Python2与Python3

## 参考资料

<https://www.cnblogs.com/kendrick/p/7478304.html>

<https://www.cnblogs.com/hanggegege/p/5840005.html>

<http://www.runoob.com/python/python-2x-3x.html>

<http://python.jobbole.com/87372/>

## 语法差异

### 整数的除法

#### Python2

a = 5 / 2

print a

输出2

#### Python3

a = 5 / 2

print(a)

输出2.5

b = 5 // 2 输出2

### Unicode支持

#### Python2

Python 2 默认使用 ASCII 字母表，因此当您输入“Hello，Sammy！”时， Python 2 将以 ASCII 格式处理字符串。被限定为在多种扩展形式上的数百个字符，用ASCII 进行字符编码并不是一种非常灵活的方法，特别是使用非英语字符时。

要使用更通用和更强大的Unicode字符编码，这种编码支持超过128,000个跨越现今和历史的脚本和符号集的字符，你必须输入

u“Hello，Sammy！”

前缀 u 代表 Unicode。

#### Python3

Python 3 默认使用 Unicode，这节省了程序员多余的开发时间，并且您可以轻松地在程序中直接键入和显示更多的字符。因为 Unicode 支持更强大的语言字符多样性以及 emoji 的显示，所以将它作为默认字符编码来使用，能确保全球的移动设备在您的开发项目中都能得到支持。

如果你希望你的 Python 3 代码向后兼容 Python 2，你可以通过在你的字符串的前面保留 “u” 来实现。

### 异常

#### Python2

#### Python3

### 八进制字面量

#### Python2

八进制数必须写成0o777，原来的形式0777不能用了；二进制必须写成0b111。

新增了一个bin()函数用于将一个整数转换成二进制字串。 Python 2.6已经支持这两种语法。

#### Python3

在Python 3.x中，表示八进制字面量的方式只有一种，就是0o1000。

### 不等运算符

#### Python2

Python 2.x中不等于有两种写法 != 和 <>。

#### Python3

Python 3.x中去掉了<>, 只有!=一种写法，还好，我从来没有使用<>的习惯。

### 去掉数据类型long

Py3.X去除了long类型，现在只有一种整型——int，但它的行为就像2.X版本的long。

### 新增数据类型bytes

新增了bytes类型，对应于2.X版本的八位串，定义一个bytes字面量的方法如下：

>>> b = b'china'

>>> type(b)

<type 'bytes'>

str对象和bytes对象可以使用.encode() (str -> bytes) or .decode() (bytes -> str)方法相互转化。

### Dict修改

dict的.keys()、.items 和.values()方法返回迭代器，而之前的iterkeys()等函数都被废弃。同时去掉的还有 dict.has\_key()，用 in替代它吧 。

## 函数差异

### print

#### Python2

print 被视为一个语句而不是一个函数，这是一个典型的容易弄混的地方，因为在 Python 中的许多操作都需要括号内的参数来执行。

print “123”

#### Python3

print（）会被显式地视为一个函数。

print(“123”)

### range与xrange

#### Python2

在 Python 2 中 xrange() 创建迭代对象的用法是非常流行的。比如： for 循环或者是列表/集合/字典推导式。

这个表现十分像生成器（比如。"惰性求值"）。但是这个 xrange-iterable 是无穷的，意味着你可以无限遍历。

由于它的惰性求值，如果你不得仅仅不遍历它一次，xrange() 函数 比 range() 更快（比如 for 循环）。尽管如此，对比迭代一次，不建议你重复迭代多次，因为生成器每次都从头开始。

import timeit

n = 10000

def test\_range(n):

return for i in range(n):

pass

def test\_xrange(n):

for i in xrange(n):

pass

#### Python3

在 Python 3 中，range() 是像 xrange() 那样实现以至于一个专门的 xrange() 函数都不再存在（在 Python 3 中 xrange() 会抛出命名异常）。

### repr

#### Python2

Python 2.x 中反引号``相当于repr函数的作用。

#### Python3

Python 3.x 中去掉了``这种写法，只允许使用repr函数，这样做的目的是为了使代码看上去更清晰么？不过我感觉用repr的机会很少，一般只在debug的时候才用，多数时候还是用str函数来用字符串描述对象。

### 打开文件

原：

file( ..... )

或

open(.....)

改为只能用

open(.....)

### Input

从键盘录入一个字符串

原:

raw\_input( "提示信息" )

改为:

input( "提示信息" )

在python2.x中raw\_input()和input( )，两个函数都存在，其中区别为：

raw\_input()---将所有输入作为字符串看待，返回字符串类型

input()-----只能接收"数字"的输入，在对待纯数字输入时具有自己的特性，它返回所输入的数字的类型（int, float ）

在python3.x中raw\_input()和input( )进行了整合，去除了raw\_input()，仅保留了input()函数，其接收任意任性输入，将所有输入默认为字符串处理，并返回字符串类型。

### map、filter和reduce

#### Python2



#### Python3



## 类差异

## 库差异

# 后续发展

## Python2

Python 2.7 将在 2020 年失去后续的支持。

## Python3