多行注释可以用多个 **#** 号，还有 **'''** 和 **"""**

IndentationError: unindent does not match any outer indentation level

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

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

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

python中数字有四种类型：整数、布尔型、浮点数和复数。

* **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
* 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 '\'
* 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。
* 按字面意义级联字符串，如"this " "is " "string"会被自动转换为this is string。
* 字符串可以用 + 运算符连接在一起，用 \* 运算符重复。
* 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。
* Python中的字符串不能改变。
* Python 没有单独的字符类型，一个字符就是长度为 1 的字符串。
* 字符串的截取的语法格式如下：**变量[头下标:尾下标:步长]**

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"""

#!/usr/bin/python3 str='Runoob' print(str) # 输出字符串 print(str[0:-1]) # 输出第一个到倒数第二个的所有字符 print(str[0]) # 输出字符串第一个字符 print(str[2:5]) # 输出从第三个开始到第五个的字符 print(str[2:]) # 输出从第三个开始的后的所有字符 print(str \* 2) # 输出字符串两次 print(str + '你好') # 连接字符串 print('------------------------------') print('hello\nrunoob') # 使用反斜杠(\)+n转义特殊字符 print(r'hello\nrunoob') # 在字符串前面添加一个 r，表示原始字符串，不会发生转义

这里的 r 指 raw，即 raw string。

输出结果为：

Runoob

Runoo

R

noo

noob

RunoobRunoob

Runoob你好

------------------------------

hello

runoob

hello\nrunoob

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')

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

**getopt模块**

getopt模块是专门处理命令行参数的模块，用于获取命令行选项和参数，也就是sys.argv。命令行选项使得程序的参数更加灵活。支持短选项模式（-）和长选项模式（--）。

该模块提供了两个方法及一个异常处理来解析命令行参数。

**getopt.getopt 方法**

getopt.getopt 方法用于解析命令行参数列表，语法格式如下：

getopt.getopt(args, options[, long\_options])

方法参数说明：

* **args**: 要解析的命令行参数列表。
* **options**: 以字符串的格式定义，**options**后的冒号(:)表示该选项必须有附加的参数，不带冒号表示该选项不附加参数。
* **long\_options**: 以列表的格式定义，**long\_options** 后的等号(=)表示如果设置该选项，必须有附加的参数，否则就不附加参数。
* 该方法返回值由两个元素组成: 第一个是 **(option, value)** 元组的列表。 第二个是参数列表，包含那些没有'-'或'--'的参数。

另外一个方法是 getopt.gnu\_getopt，这里不多做介绍。

**Exception getopt.GetoptError**

在没有找到参数列表，或选项的需要的参数为空时会触发该异常。

异常的参数是一个字符串，表示错误的原因。属性 **msg** 和 **opt** 为相关选项的错误信息。

**实例**

假定我们创建这样一个脚本，可以通过命令行向脚本文件传递两个文件名，同时我们通过另外一个选项查看脚本的使用。脚本使用方法如下：

usage: test.py -i <inputfile> -o <outputfile>

test.py 文件代码如下所示：

#!/usr/bin/python3

import sys, getopt

def main(argv):

inputfile = ''

outputfile = ''

try:

opts, args = getopt.getopt(argv,"hi:o:",["ifile=","ofile="])

except getopt.GetoptError:

print ('test.py -i <inputfile> -o <outputfile>')

sys.exit(2)

for opt, arg in opts:

if opt == '-h':

print ('test.py -i <inputfile> -o <outputfile>')

sys.exit()

elif opt in ("-i", "--ifile"):

inputfile = arg

elif opt in ("-o", "--ofile"):

outputfile = arg

print ('输入的文件为：', inputfile)

print ('输出的文件为：', outputfile)

if \_\_name\_\_ == "\_\_main\_\_":

main(sys.argv[1:])

执行以上代码，输出结果为：

$ python3 test.py -h

usage: test.py -i <inputfile> -o <outputfile>

$ python3 test.py -i inputfile -o outputfile

输入的文件为： inputfile

输出的文件为： outputfile

Python允许你同时为多个变量赋值。例如：

a = b = c = 1

以上实例，创建一个整型对象，值为 1，从后向前赋值，三个变量被赋予相同的数值。

您也可以为多个对象指定多个变量。例如：

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

以上实例，两个整型对象 1 和 2 的分配给变量 a 和 b，字符串对象 "runoob" 分配给变量 c。

Python3 中有六个标准的数据类型：

* Number（数字）
* String（字符串）
* List（列表）
* Tuple（元组）
* Set（集合）
* Dictionary（字典）

Python3 的六个标准数据类型中：

* **不可变数据（3 个）：**Number（数字）、String（字符串）、Tuple（元组）；
* **可变数据（3 个）：**List（列表）、Dictionary（字典）、Set（集合）。

## Number（数字）

Python3 支持 **int、float、bool、complex（复数）**。

在Python 3里，只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。

像大多数语言一样，数值类型的赋值和计算都是很直观的。

内置的 type() 函数可以用来查询变量所指的对象类型。

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

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

<class 'int'> <class 'float'> <class 'bool'> <class 'complex'>

此外还可以用 isinstance 来判断：

## 实例

>>>a = 111 >>> isinstance(a, int) True >>>

isinstance 和 type 的区别在于：

* type()不会认为子类是一种父类类型。
* isinstance()会认为子类是一种父类类型。

>>> class A:

... pass

...

>>> class B(A):

... pass

...

>>> isinstance(A(), A)

True

>>> type(A()) == A

True

>>> isinstance(B(), A)

True

>>> type(B()) == A

False

***注意：****在 Python2 中是没有布尔型的，它用数字 0 表示 False，用 1 表示 True。到 Python3 中，把 True 和 False 定义成关键字了，但它们的值还是 1 和 0，它们可以和数字相加。*

### 数值运算

## 实例

>>>5 + 4 # 加法 9 >>> 4.3 - 2 # 减法 2.3 >>> 3 \* 7 # 乘法 21 >>> 2 / 4 # 除法，得到一个浮点数 0.5 >>> 2 // 4 # 除法，得到一个整数 0 >>> 17 % 3 # 取余 2 >>> 2 \*\* 5 # 乘方 32

**注意：**

* 1、Python可以同时为多个变量赋值，如a, b = 1, 2。
* 2、一个变量可以通过赋值指向不同类型的对象。
* 3、数值的除法包含两个运算符：**/** 返回一个浮点数，**//** 返回一个整数。
* 4、在混合计算时，Python会把整型转换成为浮点数。

### 数值类型实例

|  |  |  |
| --- | --- | --- |
| **int** | **float** | **complex** |
| 10 | 0.0 | 3.14j |
| 100 | 15.20 | 45.j |
| -786 | -21.9 | 9.322e-36j |
| 080 | 32.3e+18 | .876j |
| -0490 | -90. | -.6545+0J |
| -0x260 | -32.54e100 | 3e+26J |
| 0x69 | 70.2E-12 | 4.53e-7j |

Python还支持复数，复数由实数部分和虚数部分构成，可以用a + bj,或者complex(a,b)表示， 复数的实部a和虚部b都是浮点型

与 C 字符串不同的是，Python 字符串不能被改变。向一个索引位置赋值，比如word[0] = 'm'会导致错误。

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）。

inputWords=inputWords[-1::-1]

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。