Python基础语法一

基本语法

注释

注释是任何存在于#号右侧的文字，其主要用作写给程序读者看的笔记。
举个例子：

```
print('hello world')   #注意到print是一个函数
#注意到print是一个函数
print('hello   world')
```

你应该在你的程序中尽可能多地使用有用的注释：

- 解释假设
- 说明重要的决定
- 解释重要的细节
- 说明你想要解决的问题
- 说明你想要在程序中克服的问题，等等。

单引号

你可以使用单引号来指定字符串，例如’将我这样框进来’或’Quote me on this’。
所有引号内的空间，诸如空格与制表符，都将按原样保留。

双引号

被双引号包括的字符串和被单引号括起的字符串其工作机制完全相同。例如 "你的名字是？"或 “What’s your name?”

三引号

你可以通过使用三个引号——“”"或“”‘来指定多行字符串。你可以在三引号之间自由地 使用单引号与双引号。来看看这个例子：

```
'''这是一段多行字符串。这是它的第一行。
This is the second line.
"What's your name?," I asked.
He     said "Bond, James Bond." '''
```

格式化方法

有时候我们会想要从其他信息中构建字符串。这正是format() 方法大有用武之地的地方。
将以下内容保存为文件str_format.py：

```
age     =20
name='Swaroop'
print('{0} was {1} years old when he wrote this book'.format(name,      age))
print('Why is {} playing with that python?'.format(name))
```

结果：

```
$      python  str_format.py
Swaroop was 20 years old when he wrote this book
Why is Swaroop playing with that python?
```

标识命名符

变量是标识符的一个例子。标识符（Identifiers)是为某些东西提供的给定名称。在你命名标识符时，你需要遵守以下规则：

- 第一个字符必须是字母表中的字母（大写 ASCII 字符或小写 ASCII 字符或 Unicode 字 符）或下划线（_）。
- 标识符的其它部分可以由字符（大写 ASCII 字符或小写 ASCII 字符或 Unicode 字符）、 下划线（_）、数字（0~9）组成。
- 标识符名称区分大小写。例如，myname和myName并不等同。要注意到前者是小写字 母n而后者是大写字母N 。
- 有效 的标识符名称可以是i或name_2_3,无效的标识符名称可能是 2things ,this is spaced out ，my-name 和>a1b2_c3。

运算符

- +(加）
  两个对象相加。3+5则输出8。
  ‘a’ + ‘b’ 则输出‘ab’ 。
- -（减）
  从一个数中减去另一个数，如果第一个操作数不存在，则假定为零。
  -5.2将输出一个负数，50-24输出26。
- *（乘）
  给出两个数的乘积，或返回字符串重复指定次数后的结果。
  2*3输出6 。
  ‘la’ *3输出 ‘lalala’ 。
- **（乘方）
  返回x的y次方。3**4输出81（即3*3*3*3）。
- /（除）
  x 除以y 13/3输出 4.333333333333333。
- //（整除）
  x除以y并对结果向下取整至最接近的整数。
  13//3输出4 。-13//3输出-5 。
- %（取模）
  返回除法运算后的余数。
  13%3输出1 。 -25.5 % 2.25 输出 1.5 。
- <<（左移）
  将数字的位向左移动指定的位数。（每个数字在内存中以二进制数表示，即 0 和1）
  2 << 2输出 8。2用二进制数表示为 10 。
  向左移2位会得到1000这一结果，表示十进制中的8 。
- >>(右移）
  将数字的位向右移动指定的位数。
  11 >> 1输出5 。
  11在二进制中表示为1011，右移一位后输出101 这一结果，表示十进制中的 5。
- &（按位与）
  对数字进行按位与操作。
  5 & 3输出1 。
- |（按位或）
  对数字进行按位或操作。
  5 | 3 输出 7 。
- ^（按位异或）
  对数字进行按位异或操作。
  5 ^ 3 输出 6 。
- ~（按位取反）
  x 的按位取反结果为 -(x+1)。
  ~5 输出-6 。
- <（小于）
  返回 x 是否小于 y。所有的比较运算符返回的结果均为 True 或 False。请注意这 些名称之中的大写字母。
  5 < 3输出False ，3 < 6 输出True。
- >（大于）
  返回 x 是否大于y。

5 > 3 返回 True 。如果两个操作数均为数字，它们首先将会被转换至一种共同的 类型。否则，它将总是返回 False 。

- <=（小于等于）
  返回 x 是否小于或等于 y。
  x = 3; y = 6; x<=y 返回 True。
- >=（大于等于）
  返回 x 是否大于或等于 y。x = 4; y = 3; x>=3返 回 True 。
- ==（等于）
  比较两个对象是否相等。
  x = 2; y = 2; x == y 返回True。x = ‘str’; y = ‘stR’;
  x == y 返回 False。 x = ‘str’; y = ‘str’; x == y 返回True 。
- !=（不等于）
  比较两个对象是否不相等。
  x = 2; y = 3; x != y返回True。
- not（布尔“非”）
  如果 x 是 True ，则返回 False。
  如果 x 是 False ，则返回 True。
  x = True, not x 返回 False。
- and（布尔“与”）
  如果 x 是 False ，则 x and y 返回 False ，否则返回 y 的计算值。
  当 x 是 False 时， x = False; y = True; x and y 将返回False 。在这一情境中， Python 将不会计算 y，因为它已经了解 and 表达式的左侧是 False ，这意味着整个 表达式都将是 False 而不会是别的值。这种情况被称作短路计算（Short-circuit Evaluation）。
- or（布尔“或”）
  如果 x 是 True ，则返回 True ，否则它将返回 y 的计算值。
  x = Ture; y = False; x or y 将返回 Ture 。在这里短路计算同样适用。

> if 语句

```
number = 23
guess = int(input('Enter an integer :'))
if      guess == number:
        #新块从这里开始
        print('Congratulations,you      guessed it.')
    print('(but you     do      not     win     any     prizes!)')
        #新块在这里结束
elif guess < number :
    #另一代码块
    print('No,it is     a little higher than that')
    #你可以在此做任何你希望在该代码块内进行的事情
else:
    print('No,  it      is      a       little  lower   than    that')
    #你必须通过猜测一个大于（>）设置数的数字来到达这里。
print('Done')
#这最后一句语句将在
#if      语句执行完毕后执行。
```

while 语句

```
number = 23
running =True
while running:
    guess = int(input('Enter an integer : '))
    if guess == number:
        print('Congratulations, you     guessed it.')
        #这将导致 while循环中止
        running = False
    elif guess  < number:
        print('No, it is a little higher than that.')
    else:
        print('No,      it      is      a       little  lower   than    that.')
else:
    print('The while loop is over.')
    #在这里你可以做你想做的任何事
print('Done')
```

for循环

```
for     i in    range(1,        5):
    print(i)
else:
    print('The  for     loop    is      over')
```

break 语句

```
while True:
    s = input('Enter something:')
    if s == 'quit':
        break
    print('Length       of      the     string  is', len(s))
print('Done')
```

continue 语句

```
while True:
    s = input('Enter something:')
    if s == 'quit':
        break
    if len(s) < 3:
        print('Too small')
        continue
    print('Input is of sufficient length')
    #自此处起继续进行其它任何处理
```