

# Python Note

在python中缩进是核心，通常推荐类似于C语言中的  
大括号{}；



# 第一章：预备知识



## 目录

Contents

- ◆ 字面量
- ◆ 注释
- ◆ 变量
- ◆ 数据类型
- ◆ 数据类型转换
- ◆ 标识符
- ◆ 运算符
- ◆ 字符串扩展
- ◆ 数据输入

# 预备知识：

**标识符命名规则 - 内容概览**

**规则 1.**  
标识符的名称，只允许出现：  
· 英文  
· 数字  
· 下划线(\_)  
这些组合元素，  
其余任何内容都不被允许。

**注意**  
1. 不推荐使用中文  
2. 数字不可以开头

**但是建议你以数字开头**

**规则 2.**  
标识符命名规则 - 大小写敏感

以定义变量为例：  
Andy = "安迪1"  
andy = "安迪2"  
字母a的大写和小写，是完全能够区分的。

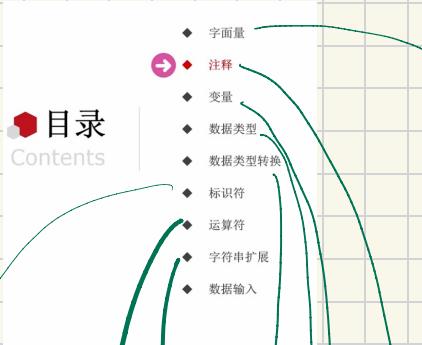
```
Andy = "安迪1"
andy = "安迪2"
print(Andy)
print(andy)

# 定义文件
with open('D:\dev\Python\Python3.10\'
          '元组.py', 'w') as f:
```

**规则 3.**  
标识符命名规则 - 不可使用关键字

**Python 中有一系列单词，称之为关键字**  
关键字在 Python 中都有特定用途，  
我们不可以将其用作标识符

False	True	None	and	or	not	del	if	else	while	for	class	def	lambda	nonlocal	yield
from	global	import	raise	return	try	except	break	continue	in	is	as	assert	del	pass	delattr



## 常用的值类型

Python中常用的有**6**种值（数据）的类型

类型	描述	说明
数字 (Number)	支持 · 整数 (int) · 浮点数 (float) · 复数 (complex) · 布尔 (bool)	整数 (int)，如：10、-10 浮点数 (float)，如：13.14、-13.14 复数 (complex)，如：4+3j，以j结尾表示复数 布尔 (bool) 表达现实生活中的逻辑，即真和假，True表示真，False表示假。 True本质上是一个数字记作1，False记作0
字符串 (String)	描述文本的一种数据类型	字符串 (string) 由任意数量的字符组成
列表 (List)	有序的可变序列	Python中使用最频繁的数据类型，可有序记录一堆数据
元组 (Tuple)	有序的不可变序列	可有序记录一堆不可变的Python数据集合
集合 (Set)	无序不重复集合	可无序记录一堆不重复的Python数据集合
字典 (Dictionary)	无序Key-Value集合	可无序记录一堆Key-Value型的Python数据集合

## 注释的分类

- 单行注释：以 **#** 开头, **#** 右边 的所有文字当作说明，而不是真正要执行的程序，起辅助说明作用

```
1 # 我是单行注释
2 print("Hello World")
```

注意，#号和注释内容一般建议以一个空格隔开

- 多行注释：以 **一对三个双引号** 引起来 (**'''注释内容'''**) 来解释说明一段代码的作用使用方法

```
1 """
2     我是多行注释
3     诗名:悯农
4     作者:李绅
5 """
6 print("锄禾日当午")
7 print("汗滴禾下土")
8 print("谁知盘中餐")
9 print("粒粒皆辛苦")
```

## 算术（数学）运算符

运算符	描述	实例
+	加	两个对象相加 $a + b$ 输出结果 30
-	减	得到负数或是一个数减去另一个数 $a - b$ 输出结果 -10
*	乘	两个数相乘或是返回一个被重复若干次的字符串 $a * b$ 输出结果 200
/	除	$b / a$ 输出结果 2
//	取整除	返回商的整数部分 $9 // 2$ 输出结果 4, $9.0 // 2.0$ 输出结果 4.0
%	取余	返回除法的余数 $b \% a$ 输出结果 0
**	指数	$a ** b$ 为 10 的 20 次方, 输出结果 10000000000000000000000000000000

运算符	描述	实例
=	赋值运算符 把 = 号右边的结果 赋给 左边的变量, 如 $num = 1 + 2 * 3$ , 结果 num 的值是?	

## 复合赋值运算符

运算符	描述	实例
+=	加法赋值运算符	$c += a$ 等效于 $c = c + a$
-=	减法赋值运算符	$c -= a$ 等效于 $c = c - a$
*=	乘法赋值运算符	$c *= a$ 等效于 $c = c * a$
/=	除法赋值运算符	$c /= a$ 等效于 $c = c / a$
%=	取模赋值运算符	$c \%= a$ 等效于 $c = c \% a$
**=	幂赋值运算符	$c **= a$ 等效于 $c = c ** a$
//=	取整除赋值运算符	$c //= a$ 等效于 $c = c // a$

高级知识点

## 字符串的三种定义方式

- 字符串在Python中有多种定义形式：  
 1. 单引号定义法: `name = '黑马程序员'`
- 2. 双引号定义法: `name = "黑马程序员"`
- 3. 三引号定义法: `name = """黑马程序员"""`

三引号定义法, 和多行注释的写法一样, 同样支持使用变量接收它, 它就是字符串

不使用变量接收它, 就可以作为多行注释使用。

## 字符串的引号嵌套

思考: 如果我想要定义的字符串本身, 是包含: 单引号、双引号自身呢? 如何写?



- 单引号定义法, 可以内含双引号
- 双引号定义法, 可以内含单引号
- 可以使用转义字符 (\) 来将引号解除作用, 变成普通字符串

## 什么是变量

变量: 在程序运行时, 能储存计算结果或能表示值的抽象概念。

简单的说, 变量就是在程序运行时, 记录数据用的

### 变量的定义格式

变量名称 = 变量的值

每一个变量都有自己存储的值(内容), 称之为: 变量值

每一个变量都有自己的名称, 称之为: 变量名, 也就是变量本身



变量就像盒子  
可以存放内容

## 数据类型

在学习变量的时候, 我们了解到: 数据是有类型的。

目前在入门阶段, 我们主要接触如下三类数据类型:

类型	描述	说明
string	字符串类型	用引号引起来的数据都是字符串
int	整形(有符号)	数字类型, 存放整数, 如 -1, 10, 0 等
float	浮点型(有符号)	数字类型, 存放小数, 如 -3.14, 6.66

## type()语句的使用方式

1. 在 print 语句中, 直接输出类型信息:

```
print(type('黑马程序员'))
print(type(666))
print(type(11.345))

int
D:\dev\Python\Python3.10
<class 'str'>
<class 'int'>
<class 'float'>
```

str 是 string 的缩写

2. 用变量存储 type() 的结果(返回值):

```
string_type = type("黑马程序员")
int_type = type(666)
float_type = type(11.345)
print(string_type)
print(int_type)
print(float_type)

Run test
D:\dev\Python\Python3.10
<class 'str'>
<class 'int'>
<class 'float'>
```

我们通过 type(变量) 可以输出类型, 这是查看变量的类型还是数据的类型?

查看的是: 变量存储的数据的类型。因为, 变量无类型, 但是它存储的数据有。

## type()语句

那么, 问题来了, 如何验证数据的类型呢?

我们可以通过 type() 语句来得到数据的类型:

语法:  
`type(被查看类型的表达式)`

## 字符串拼接

如果我们有两个字符串（文本）字面量，可以将其拼接成一个字符串，通过+号即可完成，如：

```
print("学IT来黑马" + "月薪过万")
输出结果：学IT来黑马月薪过万
```

不过一般，单纯的2个字符串字面量进行拼接显得很呆，一般，字面量和变量或变量和变量之间会使用拼接，如：

```
name = "黑马程序员"
print("我的名字是：" + name + "，我可以教大家IT技能")
D:\dev\Python\Python3.10.4\python.exe
我的名字是：黑马程序员，我可以教大家IT技能
```

# 字符串字面量之间的拼接

```
print("学IT来黑马" + "月薪过万")
```

## 字面量与变量的拼接

# 字符串字面量和字符串变量的拼接

```
name = "黑马程序员"
address = "建材城东路9号院"
print("我是：" + name + "，我的地址是：" + address)
```

字符串不能与整数的  
数据类型进行拼接。

```
# 字符串字面量之间的拼接
print("学IT来黑马" + "月薪过万")
# 字符串字面量和字符串变量的拼接
name = "黑马程序员"
address = "建材城东路9号院"
print("我是：" + name + "，我的地址是：" + address + "，我的电话是：" + tel)
```

```
D:\dev\Python\Python3.10.4\python.exe D:/python-learn/02_Python/1/进阶/06_Python的字符串拼接.py
学IT来黑马月薪过万
Traceback (most recent call last):
File "D:/python-learn/02_Python/1/进阶/06_Python的字符串拼接.py", line 7, in <module>
    print("我是：" + name + "，我的地址是：" + address + "，我的电话是：" + tel)
TypeError: can only concatenate str (not "int") to str
```

Process finished with exit code 1

## 字符串格式化

我们可以通过如下语法，完成字符串和变量的快速拼接。

```
name = "黑马程序员"
message = "学IT就来 %s" % name
print(message)

# D:\dev\Python\Python3.10.4\python
学IT就来 黑马程序员
```

其中的，%s

- % 表示：我要占位

- s 表示：将变量变成字符串放入占位的地方

所以，综合起来的意思就是：我先占个位置，等会儿有个变量过来，我把它变成字符串放到占位的位置

## 常见的转换语句

语句(函数)	说明
int(x)	将x转换为一个整数
float(x)	将x转换为一个浮点数
str(x)	将对象 x 转换为字符串

同前面学习的type()语句一样，这三个语句，都是带有结果的（返回值）  
我们可以用print直接输出

## 那对于这些带有结果的语句

如果字符串转化为其他数据类型，必须是全  
数字(int, float)，否则失败

```
# 将数字类型转换成字符串
num_str = str(11)
print(type(num_str), num_str)

float_str = str(11.345)
print(type(float_str), float_str)
# 将字符串转换成数字
```

## 字符串格式化

数字类型，也太没有地位了吧，竟然要被转成字符串拼接。  
有没有体面一点的方式，让数字以其原本的面貌拼接进去呢？

安排。

Python中，其实支持非常多的数据类型占位

最常用的是如下三类

格式符号	转化
%s	将内容转换成字符串，放入占位位置
%d	将内容转换成整数，放入占位位置
%f	将内容转换成浮点型，放入占位位置

### 字符串格式化

如下代码，完成字符串、整数、浮点数，三种不同类型变量的占位

```
name = "李雷雷雷"
set_up_year = 2006
stock_price = 19.99
message = "我是: %s, 我成立于: %d, 我今天的股价是: %.2f" % (name, set_up_year, stock_price)
print(message)
```

D:\dev\Python\Python3.10\python.exe D:/python-learn/B1\_Python基础知识/test.py

我足: 李雷雷雷, 我成立于: 2006, 我今天的股价是: 19.990000

### 字符串格式化 - 数字精度控制

我们可以使用辅助符号“%m.n”来控制数据的宽度和精度

- m, 控制宽度，要求是数字（很少使用），设置的宽度小于数字自身，不生效
- n, 控制小数点精度，要求是数字，会进行小数的四舍五入

示例：

- %50: 表示将整数的宽度控制在5位，如数字11，被设置为5d，就会变成：[空格][空格][空格]11，用三个空格补足宽度。
- %.2f: 表示将宽度控制为5，将小数点精度设置为2  
小数点和小数部分也算入宽度计算。如，对1.345设置了%.2f后，结果是：[空格][空格]11.35。2个空格补足宽度，小数部分限制2位精度后，四舍五入为 .35
- %.2: 表示不限制宽度，只设置小数点精度为2，如11.345设置%.2f后，结果是11.35
- %.1: 表示不限制宽度，只设置小数点精度为1，即11.345设置%.1f后，结果是11.3

若m>数本身；  
则m恰生效。

## 字符串格式化 - 快速写法

目前通过%符号占位已经很方便了，还能进行精度控制。

可是追求效率和优雅的Python，是否有更加优雅的方式解决问题呢？



那当然：有

通过语法：“f”内容(变量)“的格式来快速格式化

看如下代码

```
name = "传智播客"  
set_up_year = 2006  
stock_price = 19.99  
print(f"我是{name}，我成立于:{set_up_year}，我今天的股票价格是:{stock_price}")
```

test (1)

D:\dev\Python\Python3.10.4\python.exe D:/python-learn/01\_Python基础知识  
我是传智播客，我成立于：2006。我今天的股票价格是：19.99 不做精度控制，原样输出

比如我们来看一个代码的示例好

## 字符串格式化 - 表达式的格式化

刚刚的演示，都是基于变量的。

可是，我想更加优雅些，少写点代码，直接对“表达式”进行格式化是否可行呢？

那么，我们先了解一下什么是表达式。

表达式：一条具有明确执行结果的代码语句

如：

$1 + 1$ 、 $5 * 2$ ，就是表达式，因为有具体的结果，结果是一个数字

又或者，常见的变量定义：

name = "张三" age = 11 + 11

等号右侧的都是表达式呢，因为它们有具体的结果，结果赋值给了等号左侧的变量。

✓

那么，对于字符串格式化，能否直接格式化一个表达式呢？

可以，上代码：

```
print("1 * 1的结果是: %d" % (1 * 1))  
print(f"1 * 1的结果是: {1 * 1}")  
print("字符串在Python中的类型是: %s" % type('字符串'))
```

test (1) ×

D:\dev\Python\Python3.10.4\python.exe D:/python-

1 \* 1的结果是：1

1 \* 1的结果是：1

字符串在Python中的类型是: <class 'str'>

在无需使用变量进行数据存储的时候，可以直接格式化表达式，简化代码哦

```
print("1 + 1 的结果是: %d" % (1 + 1))  
print(f"1 + 1 的结果是: {1 + 1}")  
print("字符串在Python中的类型名是: %s" % type("字符串"))
```

D:\dev\Python\Python3.10.4\python.exe D:/python-learn/02\_Python入门语法

1 + 1 的结果是：2

1 + 1 的结果是：2

字符串在Python中的类型名是: <class 'str'>

# input 函数用途：

我们前面学习过print语句（函数），可以完成将内容（字面量、变量等）输出到屏幕上。

在Python中，与之对应的还有一个input语句，用来获取键盘输入。

- 数据输出：print
- 数据输入：input

使用上也非常简单：

- 使用input()语句可以从键盘获取输入
- 使用一个变量接收（存储）input语句获取的键盘输入数据即可

```
print("请告诉我你是谁?")
name = input()
print("Get! ! ! 你是: %s" % name)
```

D:\dev\Python\Python3.10\python.exe  
请告诉我你是谁?  
黑马程序员  
Get! ! ! 你是: 黑马程序员

从 input 语句输入的所有  
信息初使化为字符串。

那么你输入内容就有到

## input语句（函数）

在前面的代码中，“输出”“请告诉我你是谁？”的print语句其实是多余的

```
print("请告诉我你是谁?")
name = input()
print("Get! ! ! 你是: %s" % name)
```

input()语句其实是在要求使用者输入内容前，输出提示内容的哦，方式如下：

```
name = input("请告诉我你是谁?")
print("Get! ! ! 你是: %s" % name)
```

test (1) <br>D:\dev\Python\Python3.10.4\py  
请告诉我你是谁?  
黑马程序员  
Get! ! ! 你是: 黑马程序员

总结：

1. input()语句的功能是，获取键盘输入的数据
2. 可以使用：input(提示信息)，用以在使用者输入内容之前显示提示信息。
3. 要注意，无论键盘输入什么类型的数据，获取到的数据永远都是字符串类型

如图，在input的括号内直接填入提示内容即可。

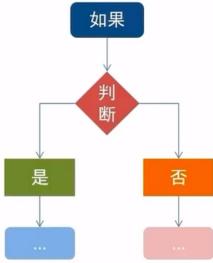
## 第二章: Python 判断语句:

- ◆ 布尔类型和比较运算符 , True & False
- ◆ if语句的基本格式
- ◆ if else 语句
- ◆ if elif else语句
- ◆ 判断语句的嵌套
- ◆ 实战案例

## Python 判断语句:

进行判断，只有2个结果：

- 是
  - 否



### 布尔类型

Python中常用的有**6**种值（数据）的类型

类型	描述	说明
数字 (Number)	支持 整数 (int) 浮点数 (float) 复数 (complex) 布尔 (bool)	整数 (int)，如：10, -10 浮点数 (float)，如：13.14, -13.14 复数 (complex)，如：4+3j，以括号表示复数 布尔 (bool) 表达现实生活中的逻辑，即真假 True表示真 False表示假 True表示1 False表示0 True和False是关键字，False表示假的理由是太长了
字符串 (String)	描述文本的一种数据类型	字符串 (string) 由任意数量的字符组成
列表 (List)	有序的可变序列	Python中使用最频繁的数据类型，可有序记录一堆数据
元组 (Tuple)	有序的不可变序列	可有序记录一堆不变的Python数据集合
集合 (Set)	无序不重复集合	可无序记录一堆不重复的Python数据集合
字典 (Dictionary)	无序Key-Value集合	可无序记录一堆Key-Value型的Python数据集合

布尔类型的定义

布尔类型的字面量：

- True 表示真 (是、肯定)
  - False 表示假 (否、否定)

### 定义变量存储布尔类型数据

变量名称 = 布尔类型字面量

## 布尔类型不仅可以自行定

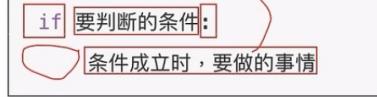
同时也可以通过计算的来

也就是使用比较运算符进行比较运算得到布尔类型的结果。

运算符	描述	示例
<code>==</code>	判断内容是否相等, 满足为True, 不满足为False	<code>如a=3,b=3, 则(a == b) 为 True</code>
<code>!=</code>	判断内容是否不相等, 满足为True, 不满足为False	<code>如a=3,b=3, 则(a != b) 为 False</code>
<code>&gt;</code>	判断运算符左侧内容 是否 大于 右侧 满足为True, 不满足为False	<code>如a=7,b=3, 则(a &gt; b) 为 True</code>
<code>&lt;</code>	判断运算符左侧内容 是否 小于 右侧 满足为True, 不满足为False	<code>如a=7,b=7, 则(a &lt; b) 为 False</code>
<code>&gt;=</code>	判断运算符左侧内容 是否 告于 等于 右侧 满足为True, 不满足为False	<code>如a=3,b=3, 则(a &gt;= b) 为 True</code>
<code>&lt;=</code>	判断运算符左侧内容 是否 小于 等于 右侧 满足为True, 不满足为False	<code>如a=3,b=3, 则(a &lt;= b) 为 True</code>

# if语句:

程序中的判断



```
age = 18  
  
print("今年我已{age}岁了")  
if age >= 18:  
    print("我已经成年了")  
    print("即将步入大学生活")  
  
    print("时间过的真快")
```

判断语句的结果, 必须是布尔类型 True 或 False  
True 会执行 if 内的代码语句  
False 则不会执行

注释: 当 if 判断条件结果为 True 时, 代码会被执行。当为 False 时, 代码不会执行。

# 四格缩进规范:

# if else语句:

else表示条件不成立

```
print("欢迎来到黑马儿童游乐场, 儿童免费, 成人收费。")  
  
age = int(input("请输入你的年龄: "))  
  
if age >= 18:  
    print("您已成年, 游玩需要补票10元。") 条件成立时执行  
  
else:  
    print("您未成年, 可以免费游玩。") 条件不成立时执行  
  
print("祝您游玩愉快。")
```

## 1. if else 语句, 其中

- if 和其代码块, 条件满足时执行
- else 搭配 if 的判断条件, 当不满足的时候执行

## 2. if else 语句的注意事项:

- else 不需要判断条件, 当 if 的条件不满足时, else 执行
- else 的代码块, 同样要 4 个空格作为缩进

# if elif else语句

## 程序中的判断



## elif语句可以写多个

判断是互斥且有顺序的。

- 满足1(如图编号)将不会理会2和3
- 满足2, 将不会理会3
- 1.2.3均不满足, 进入else
- else也可以省略不写, 效果等同3个独立的if判断

```
if height < 120:  
    print("您的身高小于120CM, 可以免费游玩。")  
elif vip_level > 3:  
    print("您的vip级别大于3, 可以免费游玩。")  
elif day == 1:  
    print("今天是1号免费日, 可以免费游玩。")  
else:  
    print("不好意思, 所有条件都不满足, 需要购票10元。")
```

空格缩进同样不可省略

## 1. if elif else语句的作用是?

可以完成多个条件的判断

## 2. 使用if elif else的注意点有:

- elif可以写多个
- 判断是互斥且有序的, 上一个满足后面的就不会判断了
- 可以在条件判断中, 直接写input语句, 节省代码量

# 判断语句的嵌套:

基础语法格式如下：



如上图，第二个if，属于第一个if内，只有第一个if满足条件，才会执行第二个if

嵌套的关键点，在于**空格缩进**

**关键点在于空格缩进**

通过空格缩进，来决定语句之间的：**层次关系**

## 判断语句的嵌套

简单嵌套：

```
print("欢迎来到黑马动物园。")  
if int(input("请输入你的身高:")) > 120:  
    print("你的身高大于120cm, 不可以游玩")  
    print("不过如果你的vip等级高于3, 可以免费游玩")  
        2  
    if int(input("请告诉我你的vip级别:")) > 3:  
        print("恭喜你, 你的vip级别大于3, 可以免费游玩。")  
    else:  
        print("Sorry, 你需要补票, 10元。")  
else:  
    print("欢迎你小朋友, 可以免费游玩。")
```

如图：

- 判断有2层
- 当外层if满足条件（图中编号1）时，才会执行内层if判断（图中编号2）
- 当外层if（编号1）不满足，直接执行外层else

\* python 的关键在于

空格缩进：  
python是通过空格缩进来决定语句间的层次关系。

# 循环语句: while

## 1. while循环的语法格式

```
while 条件:  
    条件满足时，做的事情1  
    条件满足时，做的事情2  
    条件满足时，做的事情3  
    ... (省略) ...
```

## 2. while循环的注意事项

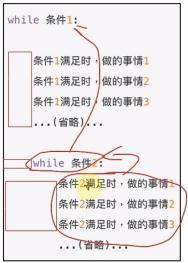
- 条件需提供布尔类型结果，True继续，False停止
- 空格缩进不能忘
- 请规划好循环终止条件，否则将无限循环

## 1. 嵌套循环的语法格式：

- 见右图

## 2. 嵌套循环需要注意的地方：

## 3. 嵌套循环的使用难点：



## while循环注意点

```
i = 0  
while i < 100:  
    print("小美，我喜欢你")  
    i += 1
```

1. while的条件需得到布尔类型，True表示继续循环，False表示结束循环

2. 需要设置循环终止的条件，如i += 1配合i < 100，就能确保100次后停止，否则将无限循环

3. 空格缩进和if判断一样，都需要设置

# 补充知识: Supplement knowledge

No line break:

```
print("Hello", end='')
print("world", end='')

← "end=''" indicate:
      no line break
```

test  
D:\dev\python\python3.10.4\python.exe D:/python-learn/04\_Python循环语句/test.py  
Helloworld  
Process finished with exit code 0

TAB character

补充知识-制表符\t

在字符串中，有一个特殊符号：\t，效果等同于在键盘上按下 tab键。它可以让我们多行字符串进行对齐。

比如：

```
print("Hello World")
print("itheima best") 使用空格 无法对齐

print("Hello\World")
print("itheima\btest") 使用\t后，可以对齐

test
D:\dev\python\python3.10.4\python.exe D:/py
Hello World
itheima best
Hello \World
itheima\btest
```

看这里有一下我们前面的

