# Python自动化运维开发基础-02

## 前言

Python自动化运维基础是在IT运维领域必备技能之一，运维领域中众多自动化脚本或软件都是基于python应用或开发。

## 第二章 Python自动化运维开发基础

学习目标：

1、了解python起源

2、了解python应用场景

3、了解python版本

4、掌握python脚本或程序执行方式

5、掌握python程序中注释方法

6、掌握python程序中变量定义方法

7、掌握python程序中数据类型

8、掌握python程序中输出方法

# 认识python

**人生苦短，我用python -----life is short ,you need python**



目标

python起源

为什么要用python

python特点

python优缺点

### python起源

python的创始人为Guido von Rossum（吉多·范·罗苏姆）



1 1989年圣诞节期间，吉多.范罗苏姆为了在阿姆斯特丹 打发时间，绝对开发一个新的解释程序，作为ABC语言的一种继承（感觉下什么是牛人）

2 ABC 是有吉多参与设计的一种教学语言，**专门为非专业程序员设计**的，但是ABC语言并没有成功，渐渐消失了人们的视野，吉多总结为ABC是**封闭**,吉多决定在python中避免了这一错误，并且获得非常好的效果

4 **1991 年第一个python解释器诞生**了，发布到网上，python解释器所有的代码都是**开放**的，python相当的开放，又容易拓展，在开发的过程中，如果对功能进行了拓展和改进， 这些程序员也可以将改动的代码发给龟叔，龟叔进行代码的审核，和归并。有人把他称之为“终身的仁慈独裁者”注意：python**解释器**语言是用**C实现**的，可以调用C语言的**库文件**

### python的优点

* **易学**：就如同你即将看到的一样，Python极其容易上手。前面已经提到了，Python有极其**简单的语法。**
* **免费、开源**：Python是FLOSS（自由/开放源码软件）之一。简单地说，你可以自由地发布这个软件的拷贝、阅读它的源代码、对它做改动、把它的一部分用于新的自由软件中
* **可移植性：**由于它的开源本质，Python已经被移植在许多平台上（经过改动使它能够工作在不同平台上）。如果你小心地避免使用依赖于系统的特性，那么你的所有Python程序无需修改就可以在下述任何平台上面运行。这些平台包括Linux、Windows mac
* **可扩展性：**如果你需要你的一段关键代码运行得更快或者希望某些算法不公开，你可以把你的部分程序用C或C++编写，然后在你的Python程序中使用它们。
* **丰富的库：**Python标准库确实很庞大。它可以帮助你处理各种工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV文件、密码系统、
* **规范的代码**：Python采用强制缩进的方式使得代码具有极佳的可读性

python缺点：库多 “执行效率慢”

执行效率慢，这是解释型语言的所通有的，同时这个缺点也被计算机越来越强性能所弥补

# Python应用场景

**Web应用开发**

Python经常被用于Web开发。比如，通过mod\_wsgi模块，Apache可以运行用Python编写的Web程序。Python定义了WSGI标准应用接口来协调Http服务器与基于Python的Web程序之间的通信。一些Web框架，如Django,TurboGears,web2py,Zope等，可以让程序员轻松地开发和管理复杂的Web程序。

**操作系统管理、服务器运维的自动化脚本**

一般说来，Python编写的系统管理脚本在可读性、性能、代码重用度、扩展性几方面都优于普通的shell脚本。

**科学计算**

NumPy,SciPy,Matplotlib可以让Python程序员编写科学计算程序。

**服务器软件（网络软件）**

Python对于各种网络协议的支持很完善，因此经常被用于编写服务器软件、网络爬虫。第三方库Twisted支持异步网络编程和多数标准的网络协议(包含客户端和服务器)，并且提供了多种工具，被广泛用于编写高性能的服务器软件。

**游戏**

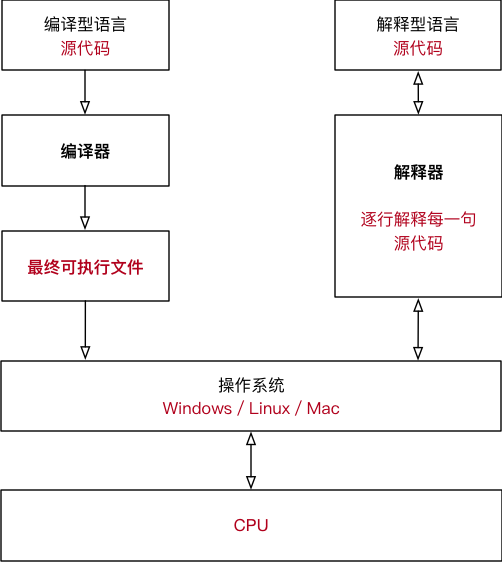
很多游戏使用C++编写图形显示等高性能模块，而使用Python或者Lua编写游戏的逻辑、服务器。相较于Python，Lua的功能更简单、体积更小；而Python则支持更多的特性和数据类型

## 解释器（科普）

计算机不能直接理解任何除**机器语言（0 1）**以外的语言，必须把程序员所编写的程序语言**编译**成**机器语言**，计算机才能执行程序，将其他语言**翻译成机器语言**的工具，称之为**编译器**

例如：有一个人：中国人 ---（翻译）----英国人

编译器翻译的方式有两种，一种是**编译**，一种是**解释**，两种方式之间的区别在于翻译时间点不同



# python版本

<https://www.python.org/getit/> 官网去下载python

python3.x



python3.X 现在和未来的主流版本

解释器名称是python3

我们推荐使用使用时python3.x版本

python 3.0版本发布于 2008年 ，目前 python3.0版本已经非常的稳定了

* Python 3.3 发布于 2012
* Python 3.4 发布于 2014
* Python 3.5 发布于 2015
* Python 3.6 发布于 2016

# 三、第一个python程序

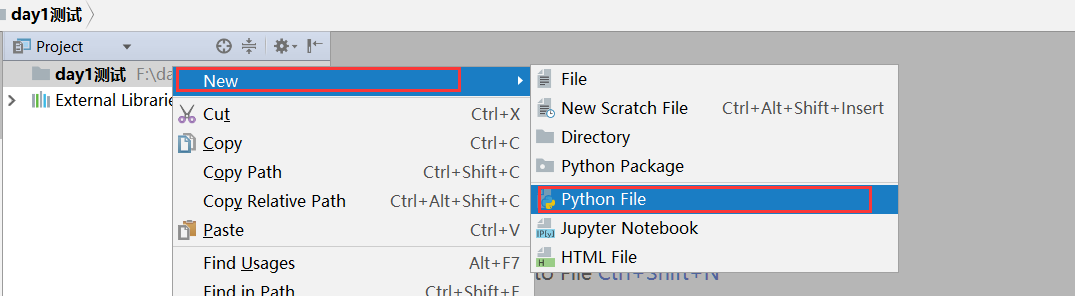
#### python源程序的概念

1 python源程序就是一个特殊格式的文本文件，可以使用任意文本编译软件做python开发

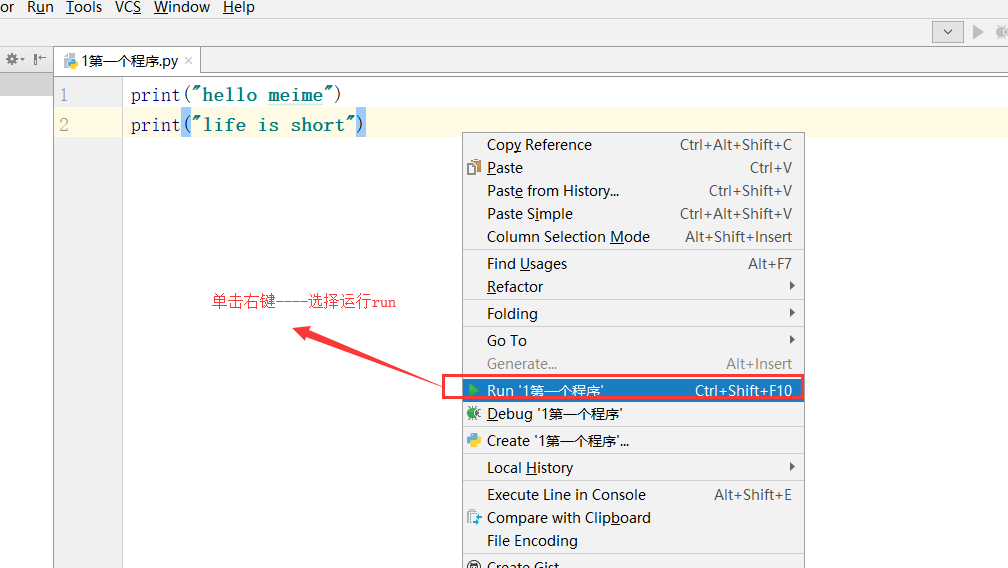
例如 sublime text 编译器 visual studio code 编译器

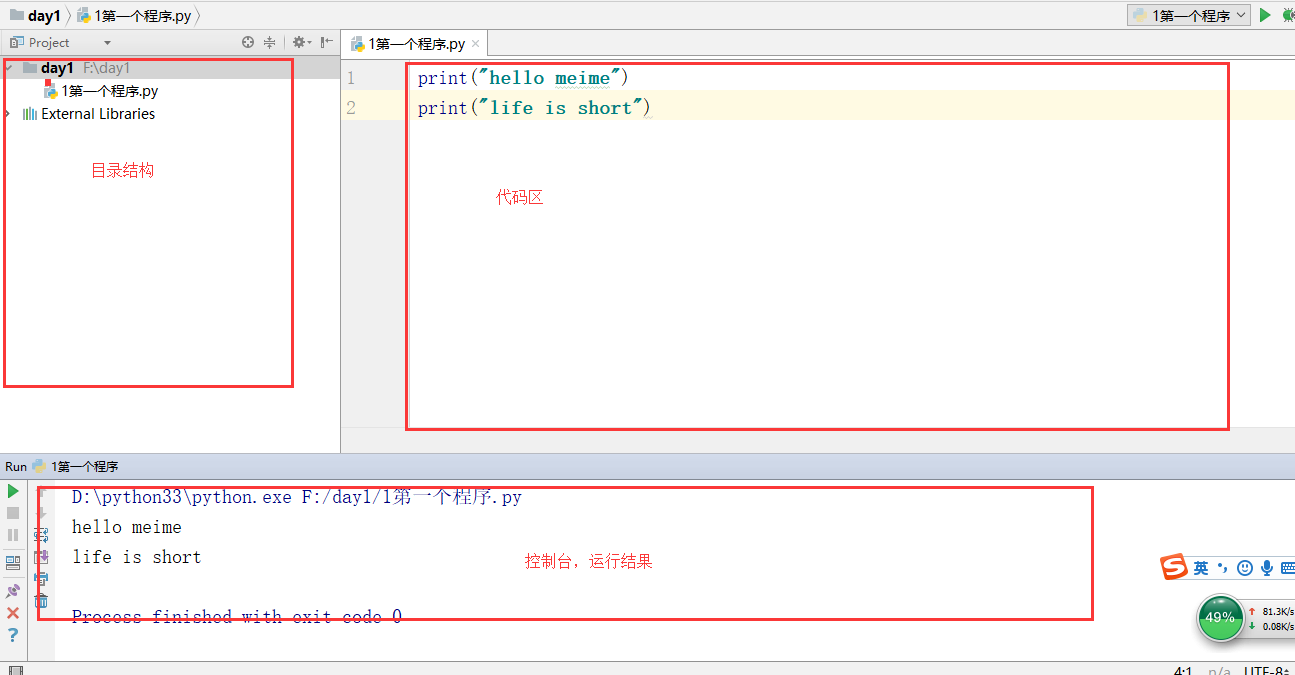
pycharm python集成开发环境 ....

2 python程序是以 文件扩展名 通常以**.py结尾的**



运行





#### 演练的步骤

* 在桌面上，新建一个python目录 day1
* 在day1 目录下新建要给 01第一个python程序.py文件

print(“hello python”)

print(“人生苦短，我用python”)

print(“python是世界上最好的语言”)

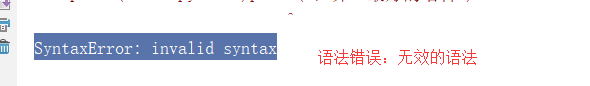
#### 常见错误

##### 手误 例如使用pirnt(“hello python”)

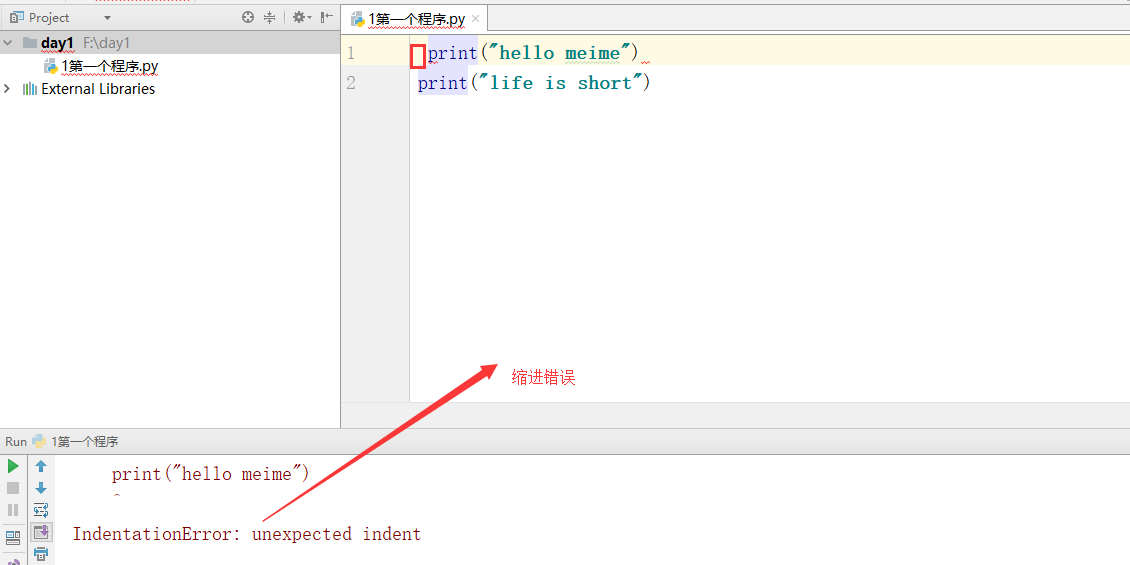


##### 多条语句写在一行





##### 缩进的错误



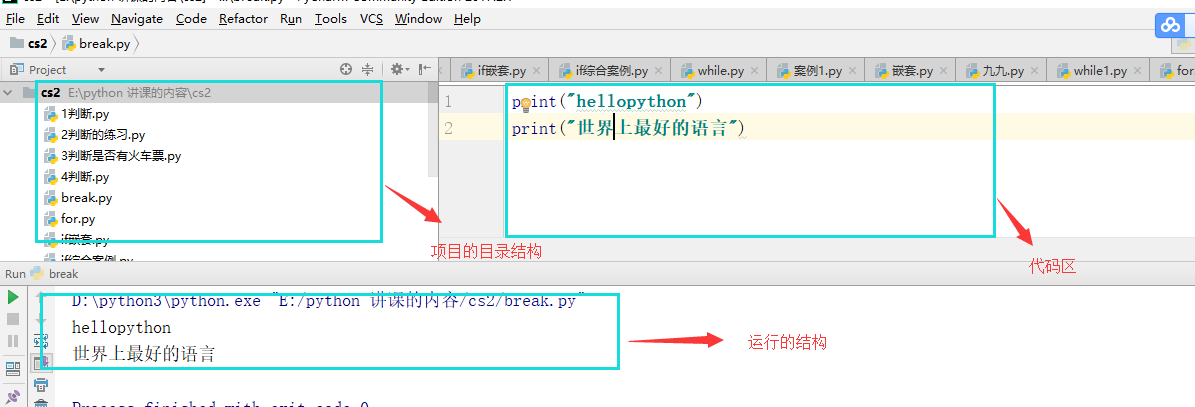
同学们咱们在演练的过程中使用的是pycharm ，除了使用pycharm 可以运行程序，使用其他的方式也可以运行python程序

# 四、执行python程序的方式

### 集成开发环境（IDE）

集成开发环境（IDE Integrated Development Environment）----集成了开发软件所需的所有的工具

* 编辑代码和运行代码在一起，交互模式是编辑代码是一个文件，运行又是一个文件
* 代码编辑器（支持 代码补全/自动缩进）
* 调试器（断点/单步执行）



# 注释

### 注释的作用

注释的作用：在程序中对某些代码进行标注说明 ，增强程序的可读性

# 注释的内容 单行

例如：

# 输出文本

print(“hello python”)

注释的快捷键：快捷键是 ctrl+/

注意： # 后面建议添加一个空格 ,然后在写注释的内容

print(“hellopython”) # 这是一个输出语句

在代码的后面添加注释 ：注释和代码之间要至少有两个空格

多行注释

“””

注释的内容多行

。。。

”””

### 关于代码规范 PEP

Python 官方提供有一系列 PEP（Python Enhancement Proposals） 文档

其中第 8 篇文档专门针对 Python 的代码格式 给出了建议，也就是俗称的 PEP 8

文档地址：<https://www.python.org/dev/peps/pep-0008/>

谷歌有对应的中文文档：<http://zh-google-styleguide.readthedocs.io/en/latest/google-python-styleguide/python_style_rules/>

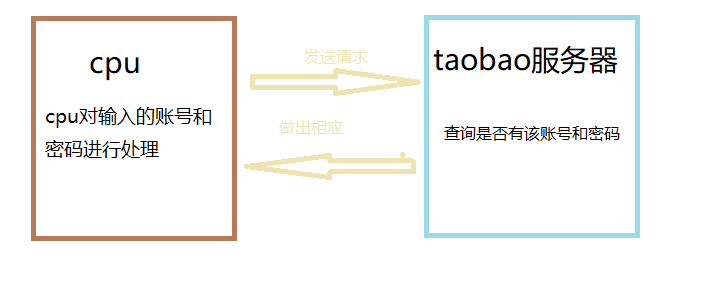
# 六、变量（重点）

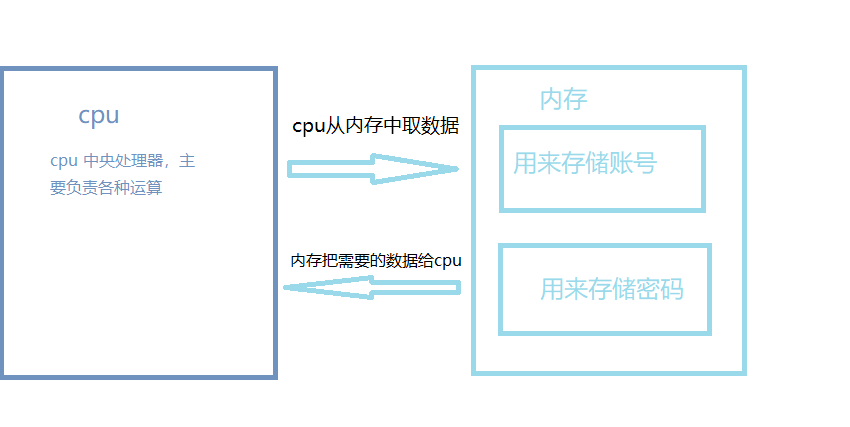
思考taobao登录的过程



1 输入用户名和密码 ，单击登录

淘宝 如何验证 用户名和密码的呢

cpu如何获取用户名和密码的呢



在内存中分别开辟一个空间用了存储用户名和密码,临时把数据保存到内存中

在内存中使用 **变量作为存储空间的别名 来存储和访问内存空间的数据**

### 变量（用了存储数据，找到数据）

**变量：在内存中开辟一块空间，起一个别名，用了访问和存储空间中的数据**

**变量的特点：**

可以反复存储数据

可以反复取出数据

可以反复更改数据

**变量的命名规则：**

* 开头是字母 或下划线 后面可以是字母 数字 下划线

\_ 可以

Name\_user

11abc 错误

Pass$word 特殊字符

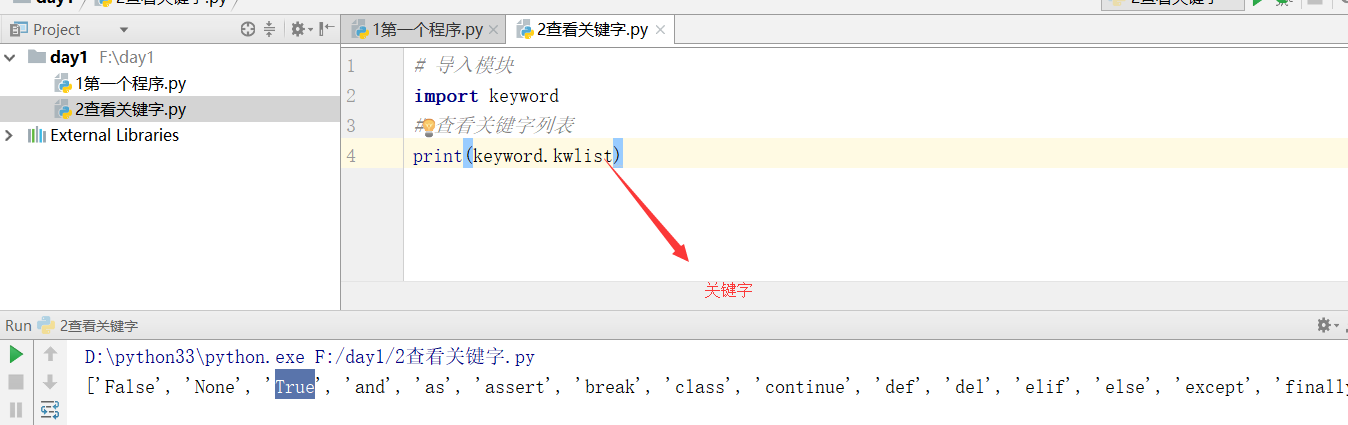
范冰冰 bingbing bb 2b

username 区分大小写

UserName

* 不能以关键字名（关键字在python中有特殊的含有 def if else）

查看关键字用:



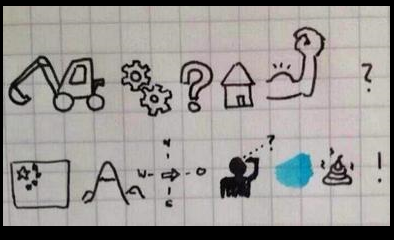
## 标识符和关键字

### 标识符

标识符就是程序员定义的 **变量名**、**函数名**

**名字** 需要有 **见名知义** 的效果，见下图：

* 标示符可以由 **字母**、**下划线** 和 **数字** 组成
* **不能以数字开头**
* **不能与关键字重名**
* 变量名要见名知义的效果



一般在工作中命名规范 :

可以使用驼峰命名法 :小驼峰

第一个单词小写，后面的单词首字母大写

userName passWord

大驼峰 : 每个单词的首字母大写

UserName PassWord

使用下划线命名法: 单词和单词之间用\_

user\_name pass\_word

练习变量的命名

开始是字母 或下划线 后面是数字 字母 下划线

\_1 正确

username1 正确

pwd$abc 特殊$

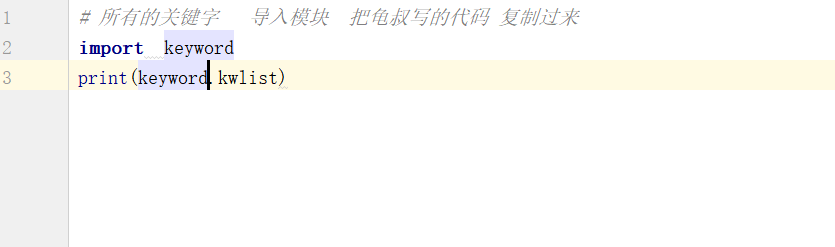
1pass 错误

user\_name\_true 正确

### 1.2 关键字

* **关键字** 就是在 Python 内部已经使用的标识符
* **关键字** 具有特殊的功能和含义
* 开发者 **不允许定义和关键字相同的名字的标示符**

通过以下命令可以查看 Python 中的关键字



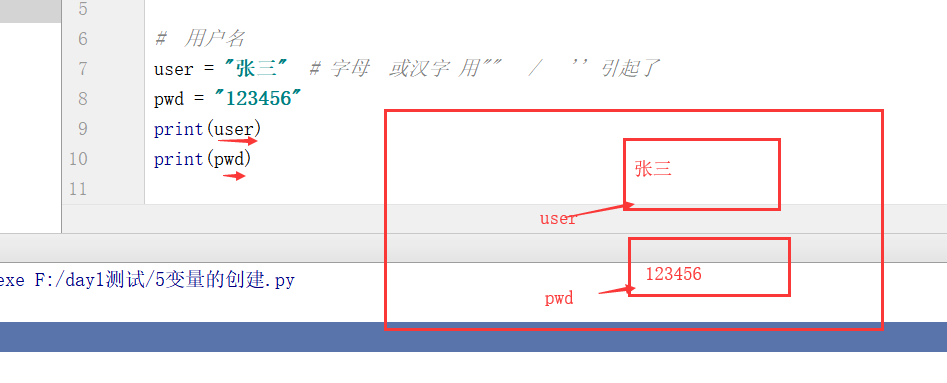
注意：变量名是区分大小写

### 变量的创建(重点)

在python中，每一个变量在使用前都必须**赋值**，变量赋值后，变量就创建了

变量名 = 值

* 等号(=) 用了给变量赋值
* = 左边是变量名
* = 右边的是用来存储变量的值



#### 变量演练1

定义两个变量，分别用来存储用户名和密码，

并且输出用户名和密码

# 定义一个淘宝的账号  
tb\_user = "gongfuxong"  
# 定义一个淘宝的密码  
tb\_password = "123345"  
# 输出变量的值用 print()  
print(tb\_user)  
print(tb\_password)

#### 变量的演练2（学员）

超市买苹果

需求

苹果的价格是 8.5 元/斤 price

买了 7.5 斤 苹果 weight

计算付款金额

price 价格

weight 重量

money 金额

# 定义苹果的价格  
price = 5.5  
# 定义苹果的重量  
weight = 2  
# 计算苹果总共的金额  
money = price \* weight  
# 输出结果  
print(money)

### 变量的类型

生活中的类型例子



在程序中，为了更好的区分变量的功能和更有效的管理内存，变量也分为不同的类型

**变量的数据类型是由值决定的**

**例如**

**定义一个变量保存鹿晗的个人信息**

姓名：鹿晗

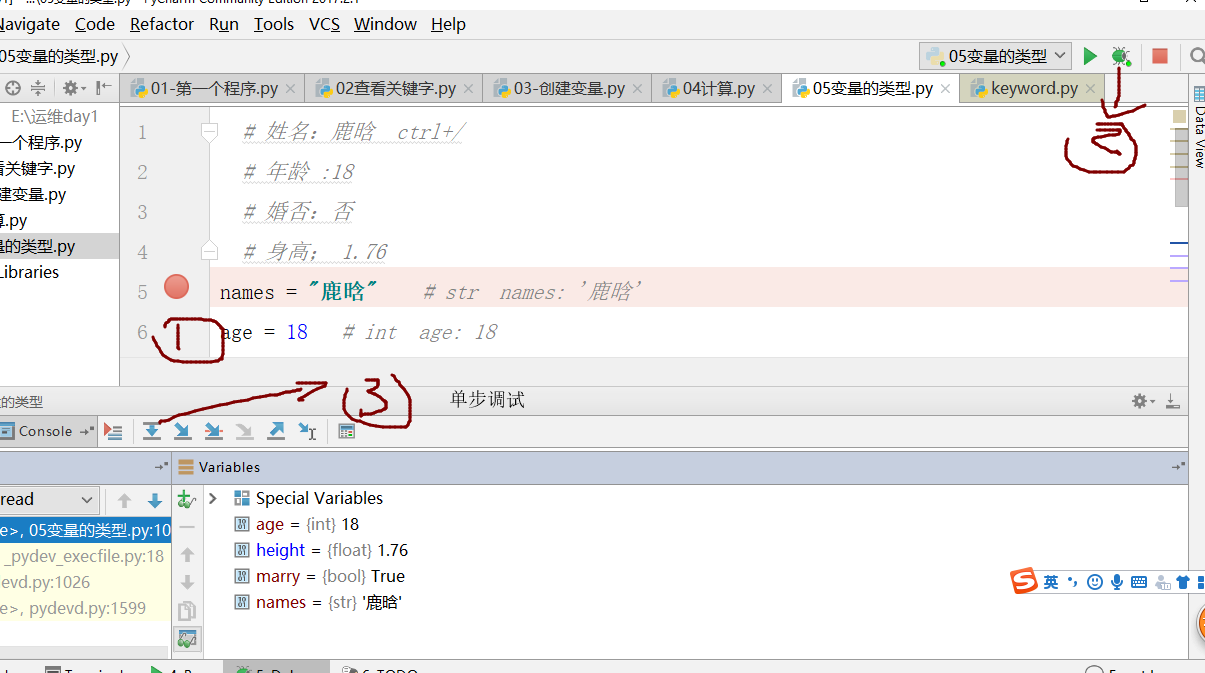
年龄 :18

婚否：否

身高； 1.76

# 变量的的数据类型  
name = "鹿晗"  
age = 18  
height = 1.76  
marry = True  
# 输出变量的值 用print结果  
print(name)  
print(age)  
print(height)  
print(marry)

使用单步调试 查看数据的类型



变量的数据类型，根据 =右边的值推导出来的

### 变量的数据类型的分类

变量的数据类型分为 数字型 和非数字型

1. **数字型**

整型（int） :输入的是整数 例如: age =20

浮点型(float) : 输入的是 小数 例如 height= 1.75

布尔型（bool） 有两个值 True(真) 是1 ,False（假） 是0 例如 marry = True 或 result = 5>6

“5”+”6”

1. **非数字型**

字符串str：用双引号或单引号括起来的是字符串

例如:str=”鹿晗” str1 = “123456” 字符串的类型

列表list

元组tuple

字典dict

集合set

### 测试变量的数据类型

用**type(变量)** 测试变量的数据类型

a = 10 int

b =10.33 float

c = True bool

d= “张三丰” str

print(type(d))

# 七、数据类型的计算

### 数字型 的运算

整型(int) 进行计算 + - \* / %

|  |  |
| --- | --- |
| **a= 10** | |
| **a += 1** | **a = a+1** |
| **a -= 1** | **a=a-1** |
| **a \*= 1** | **a=a\*1** |
| **a /=1** | **a= a/1** |
|  |  |

% 去余数

|  |  |
| --- | --- |
| 10 % 2 | 0 |
| 5 % 3 | 2 |
| 9 % 5 | 4 |
| 6 %5 | 1 |

浮点型(float) ：进行运算 + - \* / 通常不用他进行取余

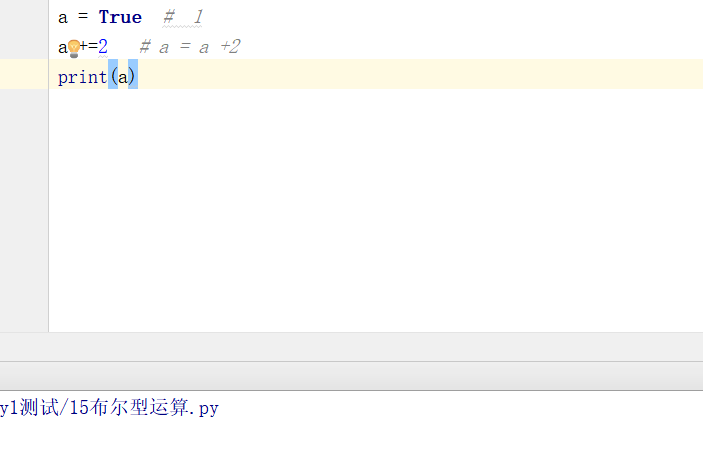
布尔型（bool）:布尔型就有两个值

True（真）在计算的时候是1

False(假) 在计算的时候是0

两个数的比较返回值都是布尔型

例如：



如果是数字型的可以直接进行运算

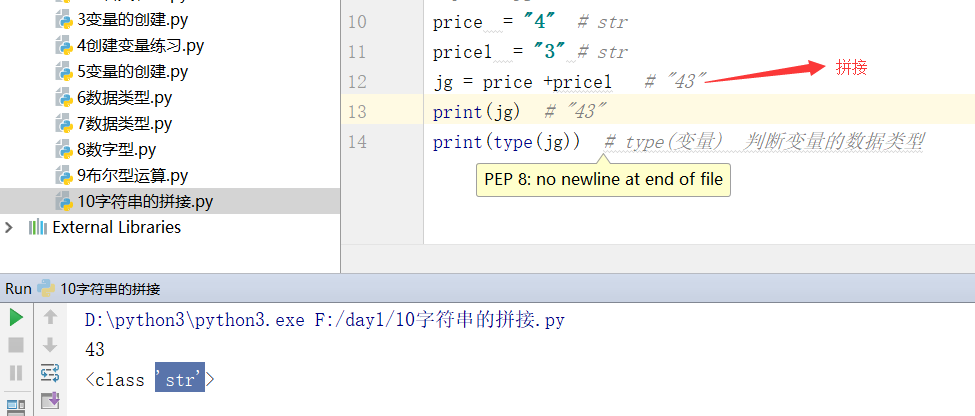
# 数字型的数据可以直接进行计算  
result = 5 \* False # False 相当于0 参与运算  
print(result) # 0  
result1 = 10 \* True # True 相对于0 参与运算  
print(result1) # 10

### 非数字类型的计算

#### 字符串变量之间使用 +拼接字符串

python中字符串变量之间使用+进行拼接运算

name = "姓名鹿晗"  
edu = "学历本科"  
info = name + edu # 两个字符串做连接运算 姓名鹿晗学历本科print(info)



##### 演练 字符串的拼接

效果:

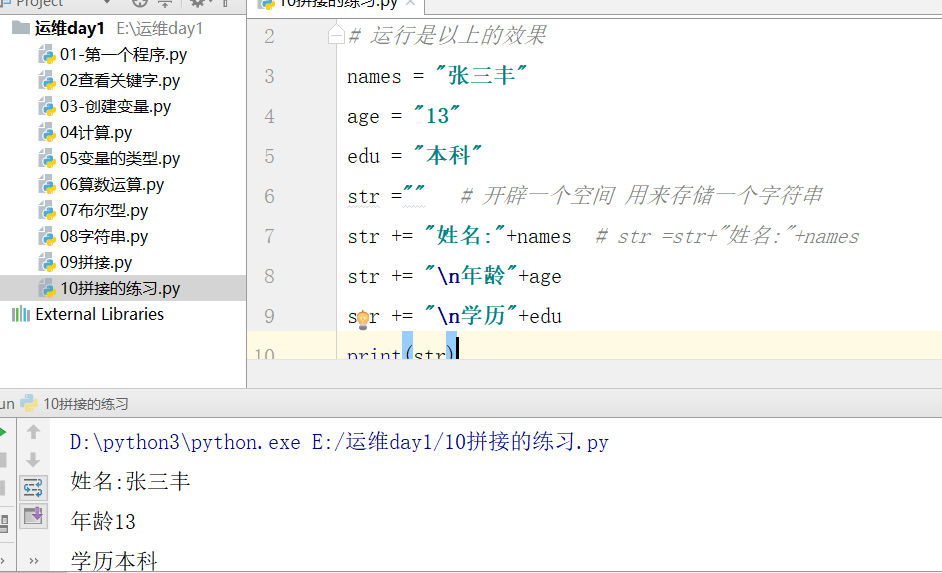
字符串的连接 用+

需求：

names 存储 张三丰 age 存储 13 edu 存储 本科

运行是以上的效果

输出的效果如下:



效果

#### 字符串变量可以和整数使用\* 重复拼接重复的内容

“3”+”4” =”34”

3+4= 7

“你好”\* 4 重复字符串的次数

# 字符串和整数可以进行\*运算，重复该字符串  
print("\*"\*20)  
print("爱你"\*20)  
# 字符串和整型之间不能拼接  
print("你好"+10) #报错

注意：字符串和整型之间不能用+拼接

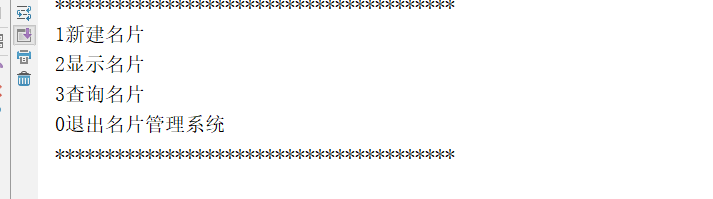
# 输出

#### 普通的输出

在python中将内容输出到控制台

python中输出用print()

用输入完成下面的效果:



print("\*"\*40)  
print("1新建名片")  
print("2显示名片")  
print("3查询名片")  
print("0退出名片管理系统")  
print("\*"\*40)

#### 格式化输出

如果想输出文字的同时，还想输出数字 ，就的格式化输出

例如: 总的价格是45.00元 用格式化输出

用 % 表示格式化操作符

|  |  |
| --- | --- |
| %d | 十进制的整数 %06d 表示输出的整数显示六位 ,不足的补0 |
| %f | 浮点型 %.3f 小数的位数显示3位 |
| %s | 字符串 |
| %% | 输出 % |

#### 格式化输出的演练

* 定义字符串变量 name，输出 我的名字叫 小明，请多多关照！
* 定义整数变量 student\_no，输出 我的学号是 000001
* 定义小数 price、weight、money，输出 苹果单价 9.00 元／斤，购买了 5.0 斤，需要支付 45.000 元
* 定义一个小数 scale，输出 数据比例是 10.00%

**做为作业**

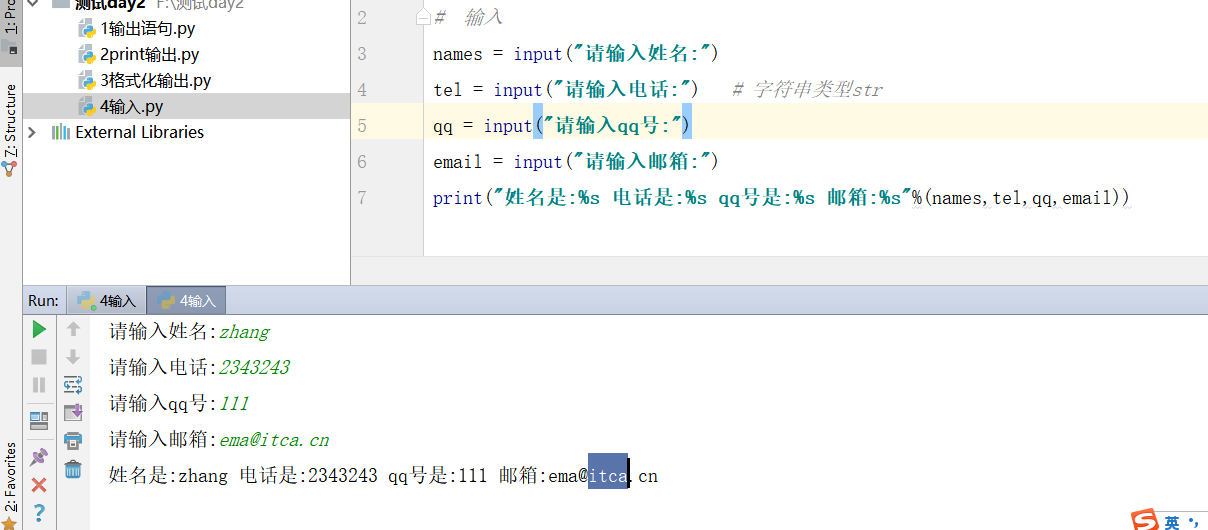
### 输入

用python中可以使用input 函数从键盘等待用户的输入

字符串的变量 = input(“提示信息”)

注意：用户输入的任何内容，python中都是字符串

raw\_input() python 2



作业：自我介绍

姓名

年龄

地址

email

qq号

电话号码

身高

体重

性别

#### 类型的转换

|  |  |
| --- | --- |
| int(变量) | 将xxxx转换为int 整数 |
| float(变量) | 将xxxx转换为浮点型 |
| str(xxxx) | 将XXXX转换为字符串 |
| 了解推理 |  |
| 转换列表 list()  转换元组tuple()  转换字典dict() |  |

price = input(“请输入价格”) “10”

weight = input(“请输入重量”) “12”

求 价格

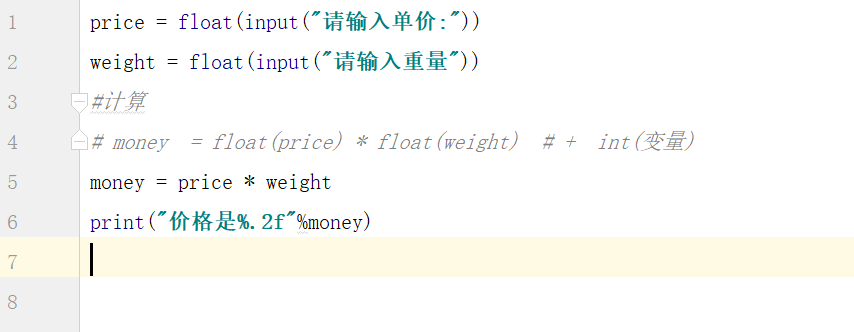
#### 变量的输入演练

需求

收银员输入 苹果的价格，单位：元／斤

收银员输入 用户购买苹果的重量，单位：斤

计算并且 输出 付款金额

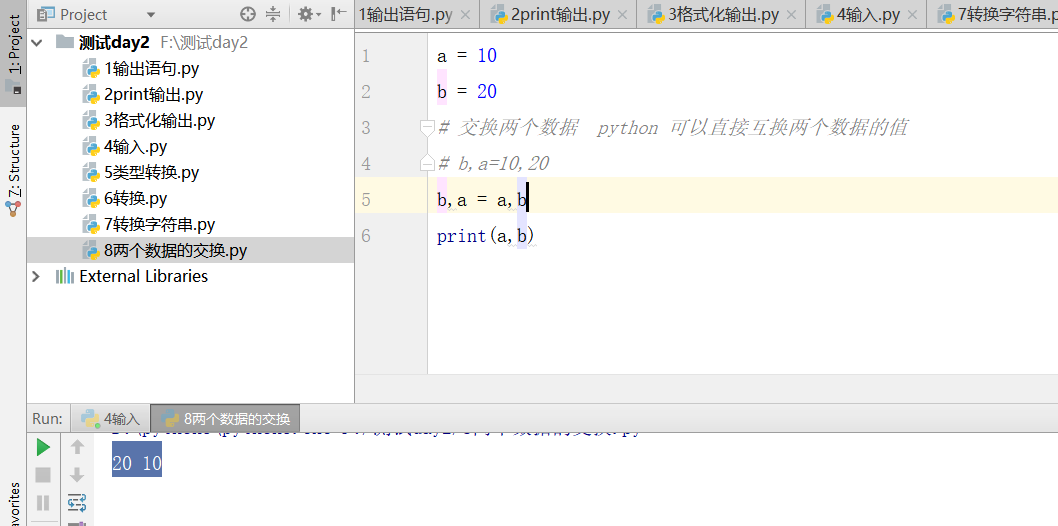


## 两个变量的交换

其它语言中可以借助于第三个变量

交换变量 a 和b 的值

python中可以直接交换，两个变量的值



## 运算符

### 算数运算符

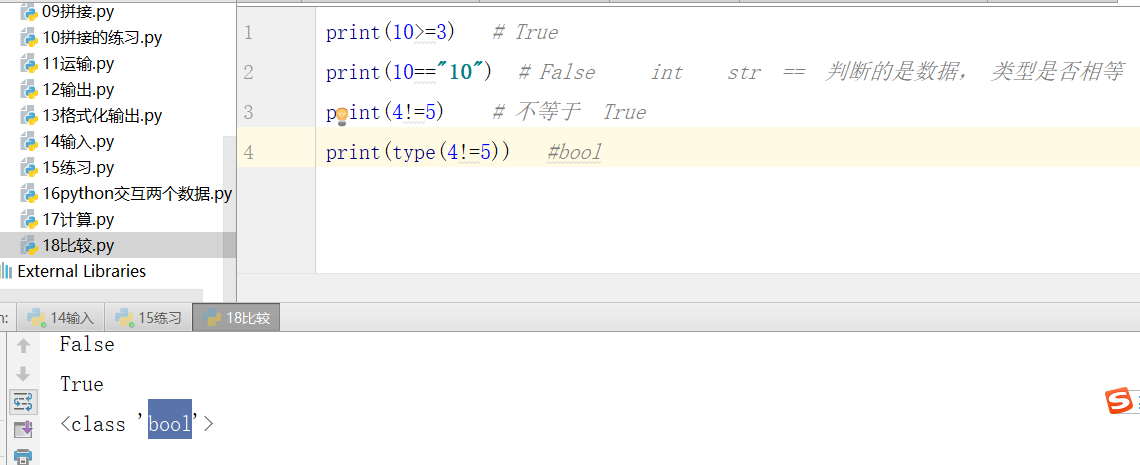


### 比较运算符

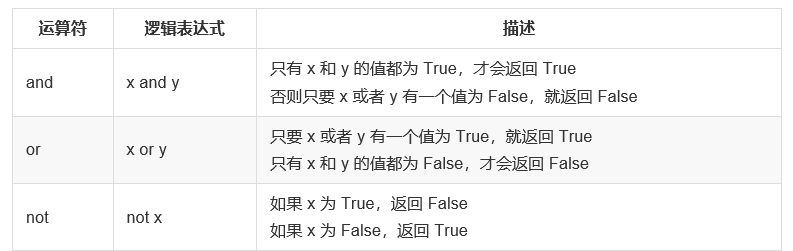
返回的结果只有 True False 是布尔型bool

> >= < <= != 不等 ==





### 逻辑运算符



### 柳岩找对象

and 左 右 必须是True 才是True

要求： 小鲜肉 年龄 小于 28 并且 and 薪资 年薪 大于 1000万

1鹿晗 26 年薪 2000万

True and True === True 处对象

2 小沈阳 35 年薪 5万

False and False =====False 不处对象

3 马云 40 年薪 1000000000万

False and True ===False 不处对象

Or 左右 有一个是True 结果是True

凤姐 找对象

老腊肉 年龄 大于 30 或者 薪资 年薪 大于 1000万

1 宝宝 35 薪资 大于 2000万

True or True ===True

2 宋小宝 40 薪资 1万

True or False ====True

3 谢霆锋 18 薪资 150万

False or False ===False

not 非 ----取反

not True ===False

not False ===True

### 赋值运算符

= 赋值

== 判断 True False



+= -= \*= /= //= %= 复合赋值运算符

a = 10

## 运算符的优先级

算数 > 比较 > 逻辑 >赋值

result = 3-3 >=2 and 4\*6>5 # 算数运算

result = 0>=2 and 24>5 False and True 比较运算

result = False and True False 逻辑

result = False 赋值

# 判断语句的介绍

生活中的判断的场景

火车站安检



密码判断



## if语句的基本格式（单分支）

if语句用来进行判断的，格式

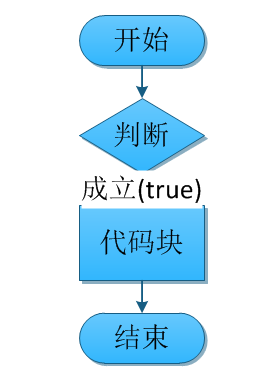
不推荐 :两个数比较 不用浮点型float

### If语句的格式：

if 要判断条件:

条件成立，执行该代码段

执行的流程



### 案例：判断年龄是否条件

注意：如果条件不满足，代码一条都不会执行

提示：代码段缩进为一个tab键，就是四个空格

### 练习

要求：从键盘获取自己的年龄，

判断是否大于或者等于18岁，

如果满足输出 可以找对象了”

age =int( input("请输入你的年龄")) # 转换为整型  
# 判断条件  
if age >= 18:  
 print("恭喜你可以找对象了")

注意：用户输入的内容是字符串

## If.... else编程条件判断（双分支）

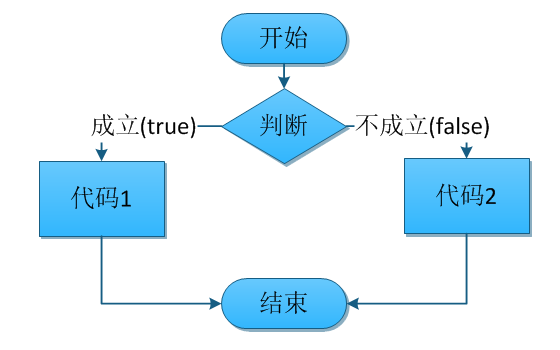
### 格式：

if 判断条件:

True满足条件执行该代码段1

else:

False不满足条件执行该代码段2



### 例如：判断是否有对象

从键盘获取是否有对象 1有/0没有

obj =int（ input("是否有对象 1有 /0没有 请输入: ")）  
if obj ==1:  
 print("你好牛有对象")  
else:  
 print("给你介绍柳岩")

## 

## if ...elif ...else编程判断语句（多分支）

### 语法：

if 判断条件1:

如果判断条件1满足，执行代码段1

elif 判断条件2：

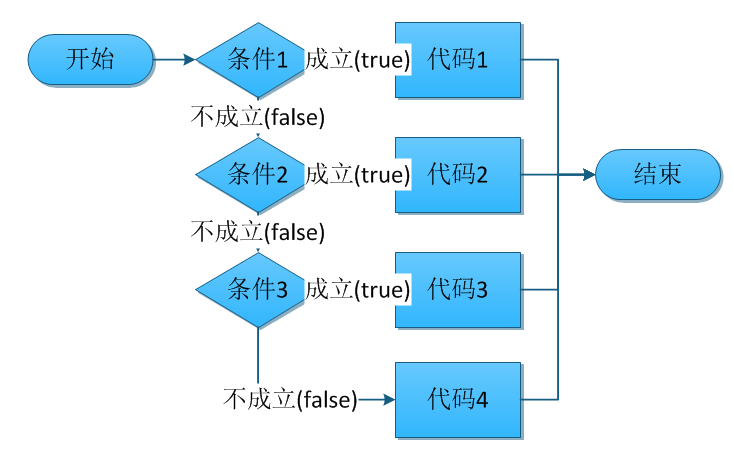
如果判断条件2满足，执行代码段2

elif 判断条件3:

如果判断条件3满足，执行代码段3

else:

上面的条件都不满足执行该语句



### 案例:判断

*定义 day 字符串变量记录节日名称  
如果是 情人节 应该 买玫瑰／看电影  
如果是 平安夜 应该 买苹果／吃大餐  
如果是 生日 应该 买蛋糕  
其他的日子每天都是节日啊……*

day =input("请输入节日的名称:")  
if day == "情人节":  
 print("买玫瑰／看电影")  
elif day == "平安夜":  
 print("买苹果／吃大餐")  
elif day == "生日":  
 print("买蛋糕")  
else:  
 print("其他的日子每天都是节日")

注意：通常在后面加else ，如果上面的条件都不满足执行该代码

Input语句输入明星

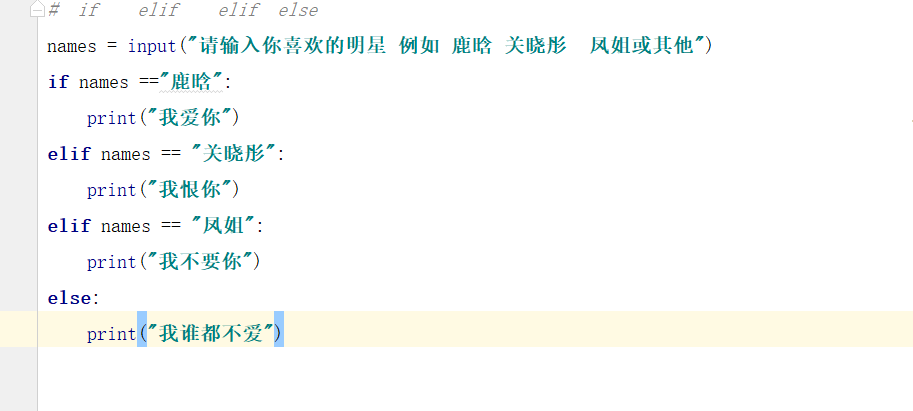
判断 如果姓名 是 鹿晗 输出我爱你

如果 名字 是关晓彤 输出 我恨你

如果 名字 是凤姐 输出 我不要你

否则

我谁都不爱



## if嵌套

想一想：

坐火车,的经过

安检

### if嵌套 执行的过程

if 判断条件1:  
 满足条件1执行该语句

If 判断条件2:

条件2满足 执行该代码  
else:

不满足条件1执行该代码

案例： 判断是否有票*(1 有 0 没有)*

ticket，判断携带物品刀子的长度length

ticket = 1  
length = 5  
if ticket ==1:  
 print("有票 欢迎进站安检")  
 # 判断刀子的长度  
 if length >=10:  
 print("携带了危险品进站")  
 else:  
 print("欢迎上车")  
else:  
 print("回家了")

### 能够用randint获取一个随机数

Import random # 模块导入

random.randint(0,10) #0 到10之间的随机数 int

## 案例：猜拳游戏

玩家输入 剪刀(0) 石头(1) 布（2）

电脑 产生一个随机数 0,1,2中的某一个

判断 用户和电脑 输赢

玩家赢:

电脑 剪刀(0) 玩家 石头(1) or

电脑 石头(1) 玩家 布（2） or

电脑 布（2） 玩家 剪刀(0)

() or () or()

平局

电脑==玩家

电脑赢

else:

玩家赢

电脑 0 玩家1

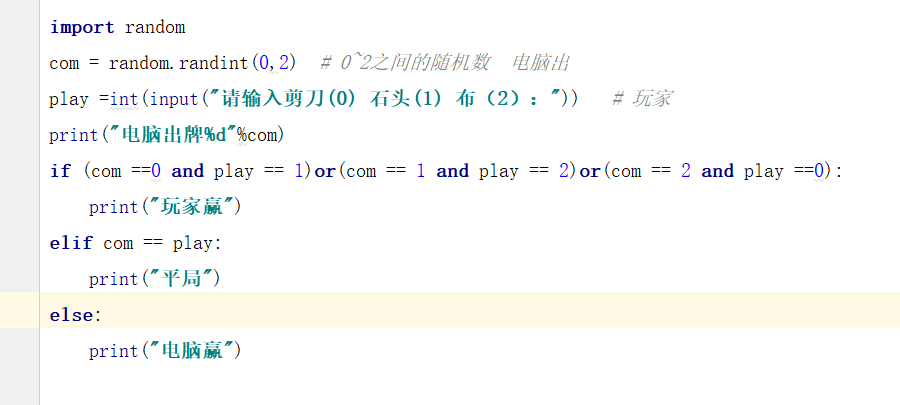
电脑 1 玩家 2

电脑 2 玩家 0

平局 玩家 ==电脑

否则

电脑赢



import random  
# 让玩家自己输入  
play = int(input("请输入:剪刀（0） 石头（1） 布（2）："))  
# 电脑 随机数 0 1 2  
com = random.randint(0,2)  
print(play,com)  
if (play==0 and com==1) or (play==1 and play==2 )or (play==2 and play==0):  
 print("玩家胜 牛")  
elif play == com:  
 print("平局")  
else:  
 print("输了，不要走，洗洗手在来")

# 循环(难点)

## 生活中的循环场景

跑道

****

风扇



## 软件开发中循环的使用场景

跟媳妇承认错误，说一万遍"媳妇儿，我错了"

print("媳妇儿，我错了")

print("媳妇儿，我错了")

print("媳妇儿，我错了")

...(还有99997遍)...

一般情况下，需要多次重复执行的代码，都可以用循环的方式来完成

## while循环

### while 循环的基本格式

1变量的初始化

while 条件2:

条件满足时候:执行该代码

条件满足时候:执行该代码

**3变量的更新**