[笔记][黑马 Python 之 Python 基础 - 2]

Python

---

038. 多文件项目演练

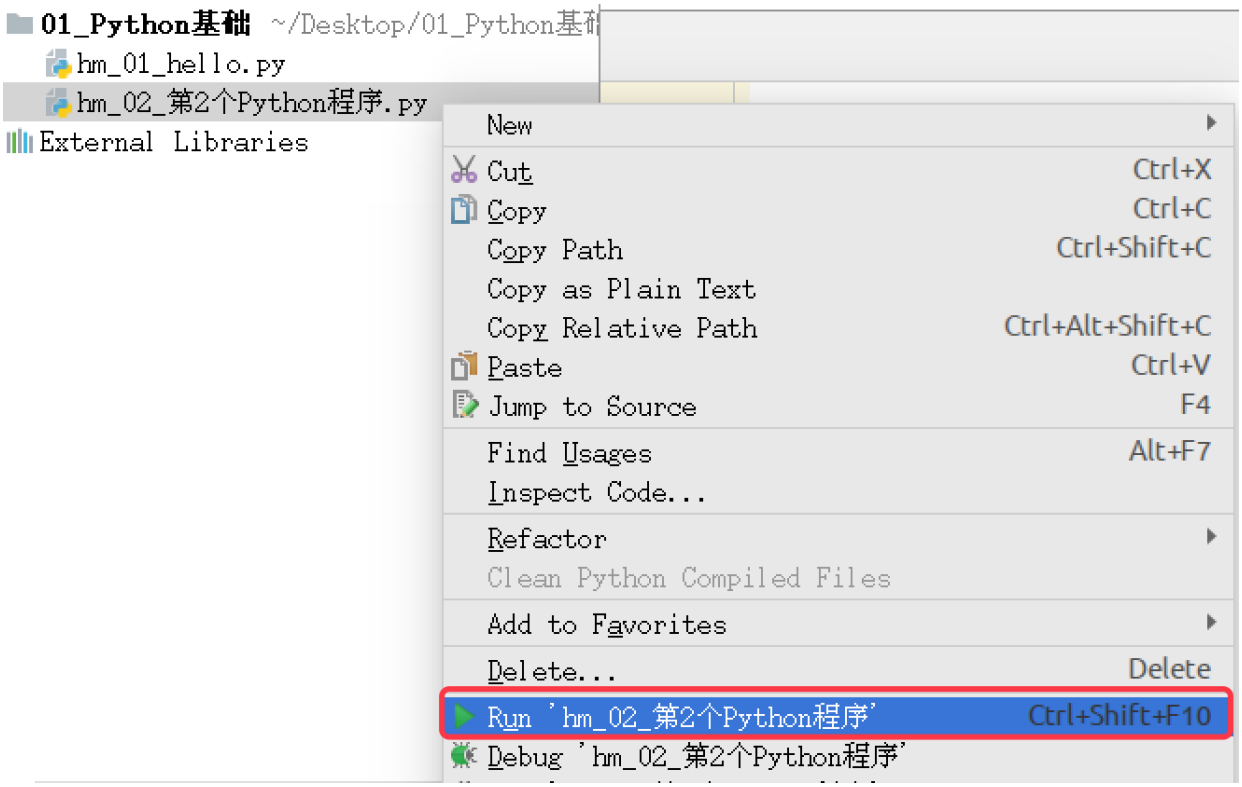
- 开发 **项目** 就是开发一个 **专门解决一个复杂业务功能的软件**
- 通常每 **一个项目** 就具有一个 **独立专属的目录**，用于保存 **所有和项目相关的文件**
    - 一个项目通常会包含 **很多源文件**

目标

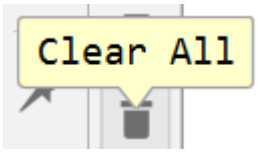
- 在项目中添加多个文件，并且设置文件的执行

多文件项目演练

1. 在 `01_Python基础` 项目中新建一个 `hm_02_第2个Python程序.py`
2. 在 `hm_02_第2个Python程序.py` 文件中添加一句 `print("hello")`
3. **点击右键执行** `hm_02_第2个Python程序.py`



> 【!】这个可以清空控制台所有内容



提示

- 在 `PyCharm` 中，要想让哪一个 `Python` 程序能够执行，必须首先通过 **鼠标右键的方式执行** 一下
- 对于初学者而言，在一个项目中设置多个程序可以执行，是非常方便的，可以方便对不同知识点的练习和测试
- 对于商业项目而言，通常在一个项目中，只有一个 **可以直接执行的 Python 源程序

---

039. 程序的注释-01-注释的作用

目标

- 注释的作用
- 单行注释（行注释）
- 多行注释（块注释）

注释的作用
使用用自己熟悉的语言，在程序中对某些代码进行标注说明，增强程序的可读性

---

040. 程序的注释-02-单行注释

单行注释(行注释)

- 以 `#` 开头，`#` 右边的所有东西都被当做说明文字，而不是真正要执行的程序，只起到辅助说明作
- 示例代码如下：

```
# 这是第一个单行注释
print("hello python")
```

> 为了保证代码的可读性，`#` 后面建议先添加一个空格，然后再编写相应的说明文字

【!】如何让整个文件的 `#` 后面都加一个空格？
虚线上面有黄色小灯泡，点 `Reformat file`

快捷键 `Alt + Enter` 再回车

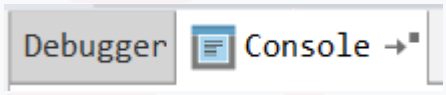
---

041. 程序的注释-03-解释器不会解释#右侧的内容

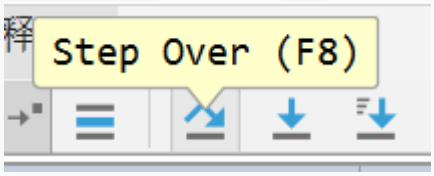
解释器不会解释#右侧的内容

调试 `Shift + F9`

点击 `Console` 查看输出结果



`Step Over` ，快捷键 `F8` ，下面的英文说明是 `step to the next line in this file`



---

042. 程序的注释-04-在代码末尾增加单行注释

在代码后面增加的单行注释

- 在程序开发时，同样可以使用 `#` 在代码的后面（旁边）增加说明性的文字
- 但是，需要注意的是，**为了保证代码的可读性，注释和代码之间** 至少要有 **两个空格**
- 示例代码如下：

```
print("hello python")  # 输出 `hello python`
```

---

043. 程序的注释-05-多行注释

多行注释（块注释）

- 如果希望编写的 **注释信息很多，一行无法显示**，就可以使用多行注释

- 要在 Python 程序中使用多行注释，可以用 **一对 连续的 三个 引号**(单引号和双引号都可以)
- 示例代码如下：

```
"""
这是一个多行注释

在多行注释之间，可以写很多很多的内容……
"""
print("hello python")
```

---

044. 程序的注释-06-注释的使用以及代码规范文档

什么时候需要使用注释？

1. **注释不是越多越好**，对于一目了然的代码，不需要添加注释
2. 对于 **复杂的操作**，应该在操作开始前写上若干行注释
3. 对于 **不是一目了然的代码**，应在其行尾添加注释（为了提高可读性，注释应该至少离开代码 2 个空格）
4. 绝不要描述代码，假设阅读代码的人比你更懂 Python，他只是不知道你的代码要做什么

> 在一些正规的开发团队，通常会有 **代码审核** 的惯例，就是一个团队中彼此阅读对方的代码

关于代码规范

- `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/

> 任何语言的程序员，编写出符合规范的代码，是开始程序生涯的第一步

---

045. 算术运算符

计算机，顾名思义就是负责进行 **数学计算** 并且 **存储计算结果** 的电子设备

目标

- 算术运算符的基本使用

算数运算符

- 算数运算符是 **运算符的一种**
- 是完成基本的算术运算使用的符号，用来处理四则运算

运算符	描述	实例
`+`	加	10 + 20 = 30
`-`	减	10 - 20 = -10
`*`	乘	10 * 20 = 200
`/`	除	10 / 20 = 0.5
`//`	地板除	返回除法的整数部分（商） 9 // 2 输出结果 4
`%`	取余数	返回除法的余数 9 % 2 = 1
`**`	幂	又称次方、乘方，2 ** 3 = 8

- 在 Python 中 `*` 运算符还可以用于字符串，计算结果就是**字符串重复**指定次数的结果
- `+` 用于字符串则是**字符串拼接**

`ipython3` 交互式执行代码

运算符的优先级

- 和数学中的运算符的优先级一致，在 Python 中进行数学计算时，同样也是：
  - **先乘除后加减**
  - 同级运算符是 **从左至右** 计算
  - 可以使用 `()` 调整计算的优先级
- 以下表格的算数优先级由高到最低顺序排列

运算符	描述
`**`	幂 (最高优先级)
`*` `/` `%` `//`	乘、除、取余数、取整除
`+` `-`	加法、减法

046. 程序执行原理-01-明确目标

目标

- 计算机中的 **三大件**
- 程序执行的原理
- 程序的作用

---

047. 程序执行原理-02-计算机中的三大件

计算机中包含有较多的硬件，但是一个程序要运行，有 **三个** 核心的硬件，分别是：

1. **CPU**
    - 中央处理器，是一块超大规模的集成电路
    - 负责 **处理数据 / 计算**
2. **内存**
    - **临时** 存储数据（断电之后，数据会消失）
    - 速度快
    - 空间小（单位价格高）
3. **硬盘**
    - **永久** 存储数据
    - 速度慢
    - 空间大（单位价格低）

---

048. 程序执行原理-03-计算机三大特点的问答

思考题

1. 计算机中哪一个硬件设备负责执行程序？
    - **CPU**
2. **内存** 的速度快还是 **硬盘** 的速度快？
    - **内存**
3. 我们的程序是安装在内存中的，还是安装在硬盘中的？

- 硬盘

4. **我买了一个内存条，有 500G 的空间！！！**，这句话对吗？
   - 不对，内存条通常只有 `4G` / `8G` / `16G` / `32G`

5. **计算机关机之后，内存中的数据都会消失**，这句话对吗？
   - 正确

---

049. 程序执行原理-04-程序执行原理简介

1. 程序 **运行之前**，程序是 **保存在硬盘** 中的
2. 当要运行一个程序时
   - 操作系统会首先让 **CPU** 把程序复制到 **内存** 中
   - **CPU** 执行 **内存** 中的 **程序代码**

> **程序要执行，首先要被加载到内存**

---

050. 程序执行原理-05-Python程序执行原理

1. 操作系统会首先让 **CPU** 把 **Python 解释器** 的程序复制到 **内存** 中
2. **Python 解释器** 根据语法规则，**从上向下** 让 **CPU** 翻译 **Python 程序中的代码**
3. **CPU** 负责执行翻译完成的代码

Python 的解释器有多大？

- 执行以下终端命令可以查看 Python 解释器的大小

```
# 1. 确认解释器所在位置
$ which python

# 2. 查看 python 文件大小(只是一个软链接)
$ ls -lh /usr/bin/python

# 3. 查看具体文件大小
$ ls -lh /usr/bin/python2.7
```

> 提示：建立 **软链接** 的目的，是为了方便使用者不用记住使用的解释器是 **哪一个具体版本**

---

051. 程序执行原理-06-明确程序的作用

> 程序就是 **用来处理数据** 的！

- **新闻软件** 提供的 **新闻内容、评论……** 是数据
- **电商软件** 提供的 **商品信息、配送信息……** 是数据
- **运动类软件** 提供的 **运动数据……** 是数据
- **地图类软件** 提供的 **地图信息、定位信息、车辆信息……** 是数据
- **即时通讯软件** 提供的 **聊天信息、好友信息……** 是数据
- ……

---

052. 程序执行原理-07-明确变量负责保存数据

思考 QQ 程序的启动过程

1. QQ 在**运行之前**，是保存在 **硬盘** 中的
2. **运行之后**，QQ 程序就会被加载到 **内存** 中了

思考 QQ 程序的 **登录** 过程

1. 读取用户输入的 **QQ 号码**
2. 读取用户输入的 **QQ 密码**
3. 将 **QQ 号码** 和 **QQ 密码** 发送给腾讯的服务器，等待服务器确认用户信息

思考 1

> 在 QQ 这个程序将 **QQ 号码** 和 **QQ 密码** 发送给服务器之前，**是否需要先存储一下 QQ 号码 和 密码?**

答案

肯定需要！—— 否则 QQ 这个程序就不知道把什么内容发送给服务器了！

思考 2

> QQ 这个程序把 **QQ 号码** 和 **QQ 密码** 保存在哪里？

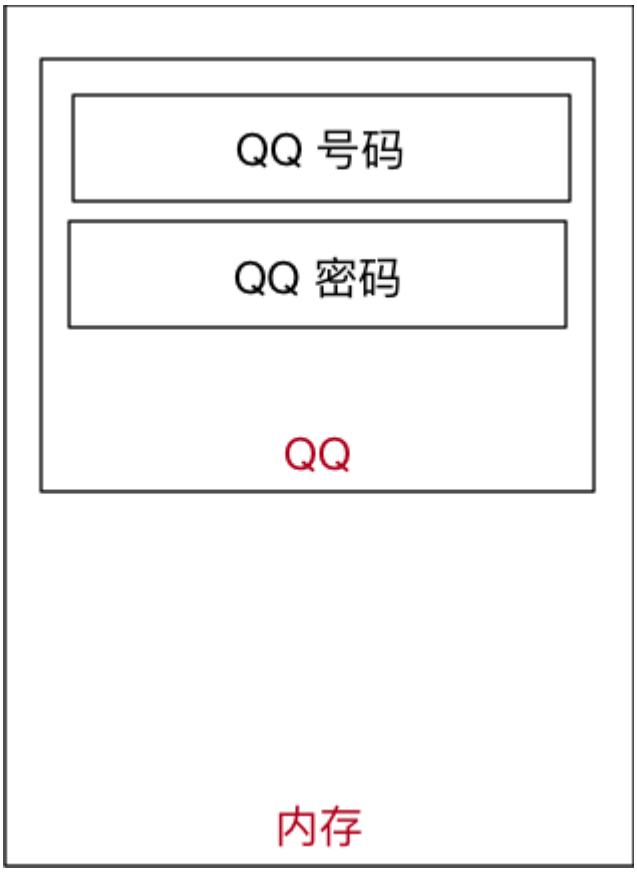
答案

保存在 **内存** 中，因为 QQ 程序自己就在内存中

思考 3

> QQ 这个程序是怎么保存用户的 **QQ 号码** 和 **QQ 密码** 的？

答案



1. 在内存中为 **QQ 号码** 和 **QQ 密码** 各自分配一块空间
    - 在 QQ 程序结束之前，这两块空间是由 QQ 程序负责管理的，其他任何程序都不允许使用

    - 在 QQ 自己使用完成之前，这两块空间始终都只负责保存 **QQ 号码** 和 **QQ 密码**

2. 使用一个 **别名** 标记 **QQ 号码** 和 **QQ 密码** 在内存中的位置

> - 在程序内部，为 **QQ 号码** 和 **QQ 密码** 在内存中分配的空间就叫做 **变量**
> - **程序就是用来处理数据的，而变量就是用来存储数据的**

---

053. 变量的使用-01-明确目标和变量定义

目标

- 变量定义
- 变量的类型
- 变量的命名

变量定义

- 在 Python 中，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**
- 等号（ `=` ）用来给变量赋值
    - `=` 左边是一个变量名
    - `=` 右边是存储在变量中的值

```
变量名 = 值
```

> 变量定义之后，后续就可以直接使用了

变量演练1 —— iPython

```
# 定义 qq_number 的变量用来保存 qq 号码
In [1]: qq_number = "1234567"

# 输出 qq_number 中保存的内容
In [2]: qq_number
Out[2]: '1234567'

# 定义 qq_password 的变量用来保存 qq 密码
In [3]: qq_password = "123"

# 输出 qq_password 中保存的内容
```

```
In [4]: qq_password
Out[4]: '123'

# 如果有多个变量，iPython 会提示
In [5]: qq
qq_nubmer    qq_password
```

> 使用交互式方式，如果要查看变量内容，直接输入变量名即可，不需要使用 `print` 函数

---

054. 变量的使用-02-使用PyCharm定义QQ变量

变量演练 2 —— PyCharm

```
# 定义 qq 号码变量
qq_number = "1234567"

# 定义 qq 密码变量
qq_password = "123"

# 在程序中，如果要输出变量的内容，需要使用 print 函数
print(qq_number)
print(qq_password)
```

> 使用解释器执行，如果要输出变量的内容，必须要要使用 `print` 函数

---

055. 变量的使用-03-超市买苹果

> - 可以用 **其他变量的计算结果** 来定义变量
> - 变量定义之后，后续就可以直接使用了

需求

- 苹果的价格是 **8.5 元/斤**
- 买了 **7.5 斤** 苹果
- 计算付款金额

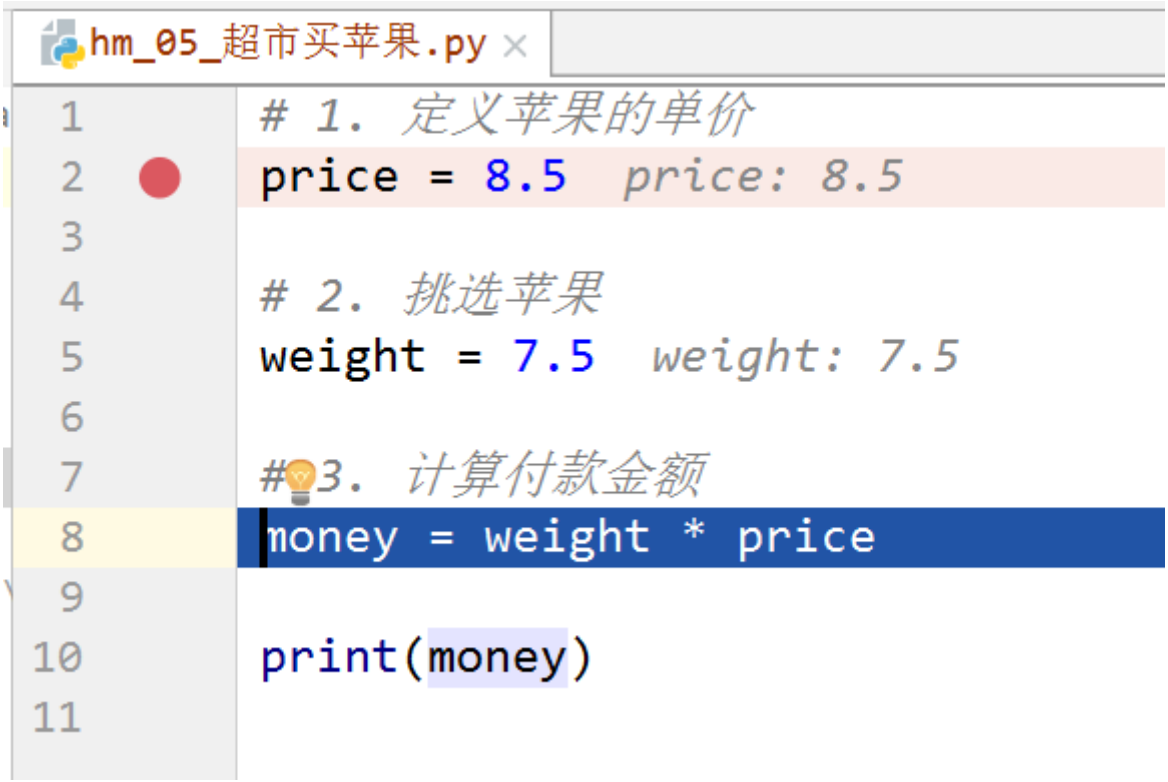
```python
# 定义苹果价格变量
price = 8.5

# 定义购买重量
weight = 7.5

# 计算金额
money = price * weight

print(money)
```

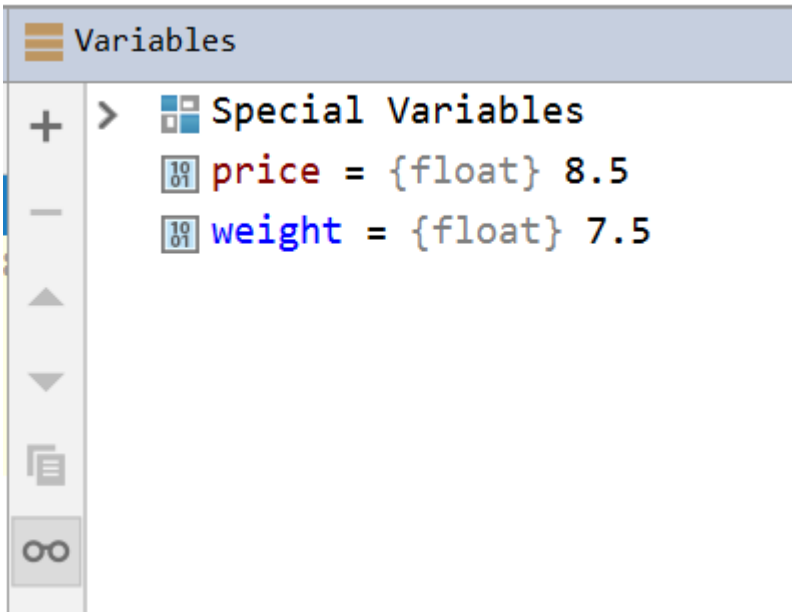
056. 变量的使用-04-PyCharm单步执行查看变量值

如何关闭其他标签页？

第二个 `Close Others`

用调试模式的单步执行查看变量值





程序执行完毕之后，变量被释放

---

057. 变量的使用-05-超市买苹果的定义和使用

- 如果 **只要买苹果，就返 5 块钱**
- 请重新计算购买金额

```
# 定义苹果价格变量
price = 8.5

# 定义购买重量
weight = 7.5

# 计算金额
money = price * weight

# 只要买苹果就返 5 元
money = money - 5
print(money)
```

提问

- 上述代码中，一共定义有几个变量？
  - 三个：`price` / `weight` / `money`
- `money = money - 5` 是在定义新的变量还是在使用变量？
  - 直接使用之前已经定义的变量
  - 变量名 只有在 **第一次出现** 才是 **定义变量**
  - 变量名 再次出现，不是定义变量，而是直接使用之前定义过的变量

- 在程序开发中，可以修改之前定义变量中保存的值吗？
    - 可以
    - 变量中存储的值，就是可以 **变** 的

---

058. 变量的类型-01-明确演练需求/项目文件准备

变量的类型
在内存中创建一个变量，会包括：

1. 变量的名称
2. 变量保存的数据
3. 变量存储数据的类型
4. 变量的地址（标示）

变量类型的演练 —— 个人信息

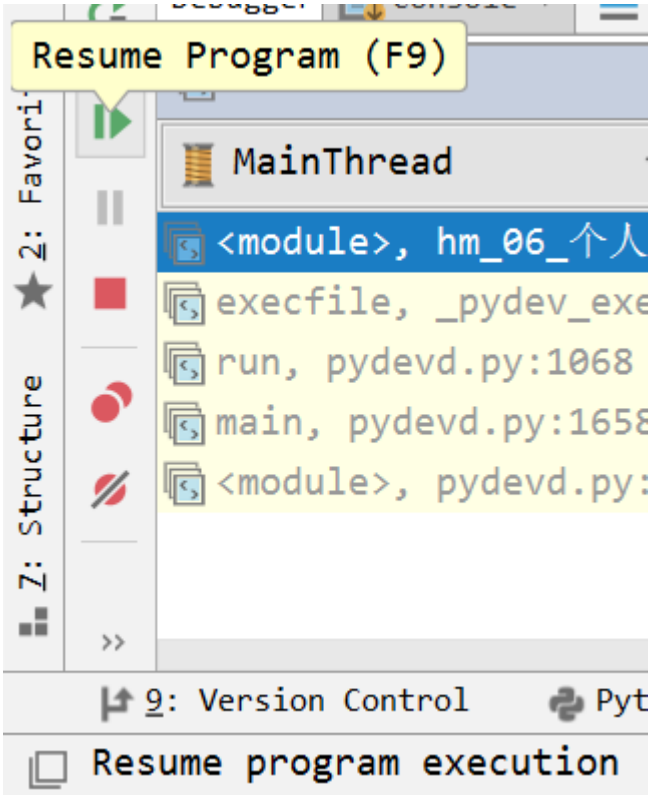
需求

- 定义变量保存小明的个人信息
- 姓名：**小明**
- 年龄：**18** 岁
- 性别：**是**男生
- 身高：**1.75** 米
- 体重：**75.0** 公斤

> 利用 **单步调试** 确认变量中保存数据的类型

---

059. 变量的类型-02-个人信息案例演练



继续执行程序

```
"""
姓名：小明
年龄：18 岁
性别：是男生
身高：1.75 米
体重：75.0 公斤
"""

# 在 Python 中，定义变量时是不需要指定变量的类型的
# 在运行的时候，Python 解释器，会根据赋值语句等号右侧的数据
# 自动推导出变量中保存数据的准确类型
# str 表示是一个字符串类型
name = "小明"
# int 表示是一个整数类型
age = 18
# bool 表示是一个布尔类型，真 True 或者假 False
gender = True  # 是
# float 表示是一个浮点数类型
height = 1.75

weight = 75

print(name)
```

提问

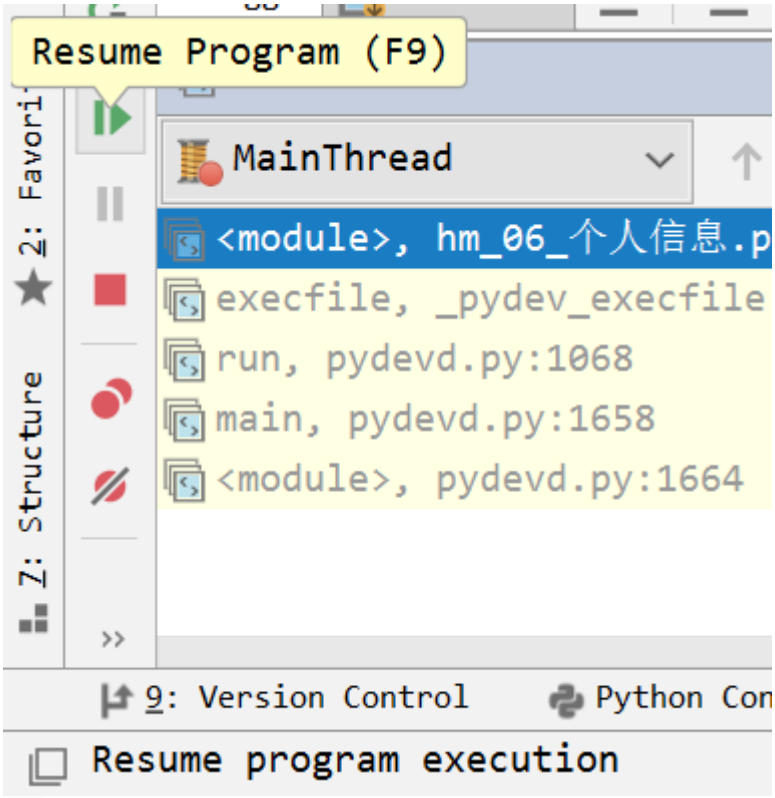
1. 在演练中，一共有几种数据类型？
    - 4 种

- `str` —— 字符串
- `bool` —— 布尔（真假）
- `int` —— 整数
- `float` —— 浮点数（小数）

2. 在 `Python` 中定义变量时需要指定类型吗？
   - 不需要
   - `Python` 可以根据 `=` 等号右侧的值，自动推导出变量中存储数据的类型

---

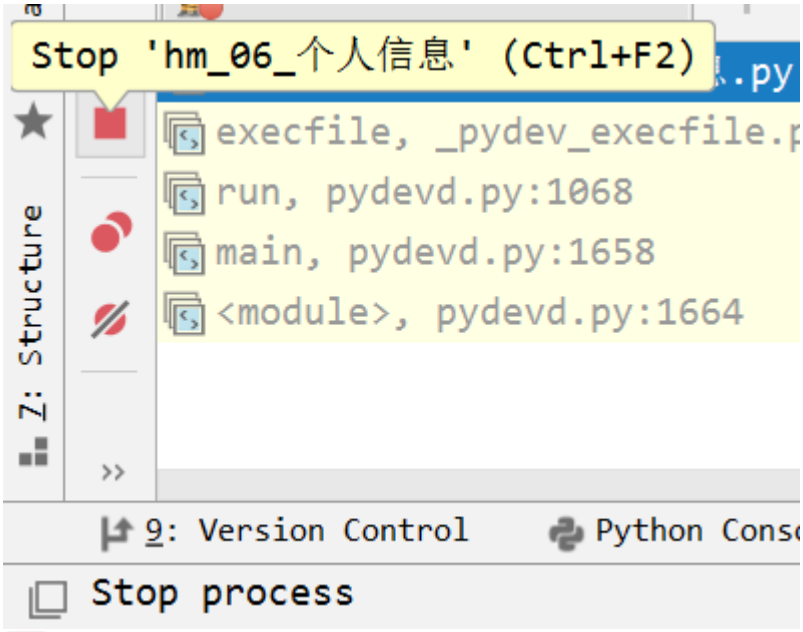
060. 变量的类型-03-[扩展]PyCharm的调试细节-调试之前先继续执行程序



`F9` 一次性执行剩下的所有代码

如何判断程序处于调试状态？
蓝色高亮条

`Shift + F9` 进入调试
`F8` 单步调试 `Step Over`
`Ctrl + F2` 停止调试



`F9` 完成所有调试

注意

如果之前的调试没有结束，如果直接点击调试按钮，会新开启一个调试环境，在新的调试环境中调试代码，之前的调试环境没有释放。这样会浪费内存。

- 所以开启新调试之前，需要按 `F9` 完成调试，或者 `Ctrl + F2` 停止调试。
- 还可以右键标签页，点击 `Close All`

---

061. 变量的类型-04-Python中的变量类型

变量的类型

- 在 `Python` 中定义变量是 **不需要指定类型**（在其他很多高级语言中都需要）
- 数据类型可以分为 **数字型** 和 **非数字型**
- 数字型
  - 整型（`int`）
  - 浮点型（`float`）
  - 布尔型（`bool`）
  - 真 `True` `非 0 数` —— **非零即真**
  - 假 `False` `0`
  - 复数型（`complex`）
  - 主要用于科学计算，例如：平面场问题、波动问题、电感电容等问题

- 非数字型
  - 字符串
  - 列表
  - 元组
  - 字典
  - 集合

062. 变量的类型-05-type函数查看变量类型

- 打开 `ipython3`
- 使用 `type` 函数可以查看一个变量的类型

```
In [1]: type(name)
```

063. 变量的类型-06-Python2.x区分int和long

> 提示：在 Python 2.x 中，**整数** 根据保存数值的长度还分为：
>
> - `int`（整数）
> - `long`（长整数）

Python2 实验
打开 `ipython`
`type(2 ** 32)` 的类型是 `int`
`type(2 ** 64)` 的类型是 `long`
`2 ** 64` 得到的数字后面会有一个 `L`
`exit` 退出解释器

`Python3` 只有一个 `int` 类型（已经包括了 `long`）

064. 变量间的计算-01-数字型变量可以直接计算

数字型变量 之间可以直接计算

- 在 Python 中，两个数字型变量是可以直接进行 算数运算的
- 如果变量是 `bool` 型，在计算时
  - `True` 对应的数字是 `1`
  - `False` 对应的数字是 `0`

演练步骤

1. 定义整数 `i = 10`
2. 定义浮点数 `f = 10.5`
3. 定义布尔型 `b = True`

    - 在 iPython 中，使用上述三个变量相互进行算术运算

---

065. 变量间的计算-02-拼接字符串的两种方式

字符串变量 之间使用 `+` 拼接字符串

- 在 Python 中，字符串之间可以使用 `+` 拼接生成新的字符串

```
In [1]: first_name = "三"

In [2]: last_name = "张"

In [3]: first_name + last_name
Out[3]: '三张'
```

字符串变量 可以和 **整数** 使用 `*` 重复拼接相同的字符串

```
In [1]: "-" * 50
Out[1]: '--------------------------------------------------'
```

数字型变量 和 **字符串** 之间 **不能进行其他计算**

```
In [1]: first_name = "zhang"

In [2]: x = 10

In [3]: x + first_name
---------------------------------------------------------------------------
TypeError: unsupported operand type(s) for +: 'int' and 'str'
类型错误：`+` 不支持的操作类型：`int` 和 `str`
```

---

066. 变量的输入输出-01-输入和函数的概念

变量的输入

- 所谓 **输入**，就是 **用代码 获取** 用户通过 **键盘** 输入的信息
- 例如：去银行取钱，在 ATM 上输入密码
- 在 Python 中，如果要获取用户在 **键盘** 上的输入信息，需要使用到 `input` 函数

关于函数

- 一个 **提前准备好的功能**(别人或者自己写的代码)，**可以直接使用**，而 **不用关心内部的细节**
- 目前已经学习过的函数

函数	说明
`print(x)`	将 `x` 输出到控制台
`type(x)`	查看 `x` 的变量类型

---

067. 变量的输入输出-02-input函数的基本使用

input 函数实现键盘输入

- 在 Python 中可以使用 `input` 函数从键盘等待用户的输入
- 用户输入的 **任何内容** Python 都认为是一个 **字符串**
- 语法如下：

```
字符串变量 = input("提示信息：")
```

> 如果在 `ipython` 里面没有设置变量接受返回结果
> 那么会在 `Out` 自动打印结果

---

068. 变量的输入输出-03-类型转换函数介绍

类型转换函数

函数	说明

`int(x)`	将 `x` 转换为一个整数
`float(x)`	将 `x` 转换到一个浮点数

---

069. 变量的输入输出-04-买苹果增强版演练

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

演练方式 1

```
# 1. 输入苹果单价
price_str = input("请输入苹果价格：")

# 2. 要求苹果重量
weight_str = input("请输入苹果重量：")

# 3. 计算金额
# 1> 将苹果单价转换成小数
price = float(price_str)

# 2> 将苹果重量转换成小数
weight = float(weight_str)

# 3> 计算付款金额
money = price * weight

print(money)
```

---

070. 变量的输入输出-05-提出问题——从控制台输入输出数字需要两个变量处理

提问

1. 演练中，针对 **价格** 定义了几个变量？
    - **两个**
    - `price_str` 记录用户输入的价格字符串
    - `price` 记录转换后的价格数值

2. **思考** —— 如果开发中，需要用户通过控制台 输入 **很多个 数字**，针对每一个数字都要定义两个变量，**方便吗**？

---

071. 变量的输入输出-06-单步执行确认变量数量

调试模式下，如果需要输入内容，需要切换到 `Console`

---

072. 变量的输入输出-07-买苹果案例改进

演练方式 2 —— 买苹果改进版

1. **定义** 一个 **浮点变量** 接收用户输入的同时，就使用 `float` 函数进行转换

```
price = float(input("请输入价格:"))
```

- 改进后的好处：

    1. 节约空间，只需要为一个变量分配空间
    2. 起名字方便，不需要为中间变量起名字

- 改进后的“缺点”：

    1. 初学者需要知道，两个函数能够嵌套使用，稍微有一些难度

提示

- 如果输入的不是一个数字，程序执行时会出错，有关数据转换的高级话题，后续会讲！

---

073. 变量的输入输出-08-格式化输出语法介绍

> 苹果单价 `9.00` 元 / 斤，购买了 `5.00` 斤，需要支付 `45.00` 元

- 在 Python 中可以使用 `print` 函数将信息输出到控制台
- 如果希望输出文字信息的同时，**一起输出 数据**，就需要使用到 **格式化操作符**
- `%` 被称为 **格式化操作符**，专门用于处理字符串中的格式
  - 包含 `%` 的字符串，被称为 **格式化字符串**
  - `%` 和不同的 **字符** 连用，**不同类型的数据** 需要使用 **不同的格式化字符**

格式化字符	含义
`%s`	字符串
`%d`	有符号十进制整数，`%06d` 表示输出的整数显示位数，不足的地方使用 `0` 补全
`%f`	浮点数，`%.2f` 表示小数点后只显示两位
`%%`	输出 `%`

- 语法格式如下：

```
print("格式化字符串" % 变量1)

print("格式化字符串" % (变量1, 变量2...))
```

---

074. 变量的输入输出-09-格式化输出字符串变量

需求

1. 定义字符串变量 `name`，输出 **我的名字叫 小明，请多多关照！**

---

075. 变量的输入输出-10-格式化输出整数变量

2. 定义整数变量 `student_no`，输出 **我的学号是 000001**

---

076. 变量的输入输出-11-格式化输出浮点型变量

3. 定义小数 `price`、 `weight`、 `money` ，输出 **苹果单价 9.00 元 / 斤，购买了 5.00 斤，需要支付 45.00 元**

---

077. 变量的输入输出-12-格式化输出%及小结

4. 定义一个小数 `scale` ，输出 **数据比例是 10.00%**

```
# 4. 定义一个小数 scale，输出 数据比例是 10.00%
scale = 0.25
print('数据比例是 %.2f%%' % scale * 100)
```

这样的话会重复 `100` 遍，需要加括号

之前的演练集合：

```
# 1. 定义字符串变量 name，输出 我的名字叫name，请多多关照！
name = '小明'
print('我的名字叫%s，请多多关照！' % name)

# 2. 定义整数变量 student_no，输出 我的学号是 000001
student_no = 1
print('我的学号是%06d' % student_no)

# 3. 定义小数 price、weight、money，输出
# 苹果单价 9.00 元／斤，购买了 5.00 斤，需要支付 45.00 元
price = 8.5
weight = 7.5
money = price * weight
print('苹果单价 %.2f 元／斤，购买了 %.3f 斤，需要支付 %.4f 元' % (price, weigh
t, money))

# 4. 定义一个小数 scale，输出 数据比例是 10.00%
scale = 0.25
print('数据比例是 %.2f%%' % (scale * 100))
```

完成于 `201809271657`