Python基础入门

1. pycharm常用快捷键

```
"""
    汇率转换器
"""
# 1. 获取数据 - 美元
usd = input("请输入美元：")
# 2. 逻辑处理 - 美元 * 6.99
cny = int(usd) * 6.99
# 3. 显示结果 - xx美元是xx人民币
print(usd + "美元是" + str(cny) + "人民币")
```

(1) 移动到本行开头：home键

(2) 移动到本行末尾：end键盘

(3) 注释代码：ctrl + /

(4) 复制行：ctrl +d

(5) 删除行：shift + delete

(6) 选择行：ctrl + shift + 左右箭头

(7) 移动行：ctrl + shift + 上下箭头

(8) 代码格式化：ctrl+alt+l

2. 注释

2.1 注释的作用

- 没有注释的代码

```
class Orders(models.Model):
    order_id = models.UUIDField(primary_key=True, auto_created=True, default=uuid.uu
    user = models.ForeignKey(User, on_delete=models.CASCADE, verbose_name='客户')
    order_state = models.SmallIntegerField(verbose_name='订单状态', choices=ORDER_STAT
    addressinfo = models.CharField(verbose_name='收货信息', max_length=255)
    order_price = models.DecimalField(verbose_name='订单金额', max_digits=7, decimal_p
    order_note = models.CharField(verbose_name='订单备注', max_length=100, default='',
    created_time = models.DateTimeField(verbose_name='创建时间', default=timezone.now)

    def get_user_first_name(self):
        return self.user.last_name + self.user.first_name
    get_user_first_name.short_description = format_html('''<span style="color:#428bca

    paymethod = models.SmallIntegerField(verbose_name='支付方式', choices=PAYMETHOD, d
    payinfo = models.CharField(verbose_name='支付信息', max_length=255, default='', bl
    paytime = models.DateTimeField(verbose_name='支付时间', default=None, blank=True,
```

- 添加注释的代码

```
def goods_list(request):
    # 拿到类别
    sort_type = request.GET.get('sort_type')
    # 如果类别存在
    if sort_type:
        # 获得指定类型的商品信息
        goods = models.Goods.objects.filter(goods_type=sort_type)
    else:
        goods = goods = models.Goods.objects.all()

    # sorts = SortType.objects.filter(show_flag=1).order_by('-priority')
    context = {
        'goods': goods,
    }
    # 返回渲染的页面
    return render(request, 'myshop/shop.html', context=context)
```

> 通过用自己熟悉的语言，在程序中对某些代码进行标注说明，这就是注释的作用，能够大大增强程序的可读性。

2.2 注释分类及语法

注释分为两类：**单行注释** 和 **多行注释**。

- 单行注释

只能注释一行内容，语法如下：

```
# 注释内容
```

- 多行注释

可以注释多行内容，一般用在注释一段代码的情况， 语法如下：

```
"""
    第一行注释
    第二行注释
    第三行注释
"""

'''
    注释1
    注释2
    注释3
'''
```

> 快捷键： **ctrl + /**

2.3 快速体验

- 单行注释

```
# 输出hello briup
print('hello briup')

print('hello World')  # 输出(简单的说明可以放到一行代码的后面，一般习惯代码后面添加两个空格再书写注释文字)
```

- 多行注释

```
<!--更换 字符串itcast itheima -->
"""
    下面三行都是输出的作用，输出内容分别是：
    hello Python
    hello World
    hello briup
"""
print('hello Python')
print('hello World')
print('hello briup')


'''
    下面三行都是输出的作用，输出内容分别是：
    hello Python
    hello World
    hello briup
'''
print('hello Python')
print('hello World')
print('hello briup')
```

> 注意：解释器不执行任何的注释内容。

> 小结：
>
> - 注释的作用
>
> 用人类熟悉的语言对代码进行解释说明，方便后期维护。

- 注释的分类
  - 单行：`# 注释内容`，快捷键ctrl+/
  - 多行：`""" 注释内容 """` 或 `''' 注释内容 '''`
- 解释器不执行注释内容

3. 输出

作用：程序输出内容给用户

```
print('hello briup')

age = 18
# 需求：输出“今年我的年龄是18岁”
print("今年我的年龄事" + str(age) +"岁")
```

3.1 格式化输出

所谓的格式化输出即按照一定的格式输出内容。

格式化符号

格式符号	转换
%s	字符串
%d	有符号的十进制整数
%f	浮点数
%c	字符
%u	无符号十进制整数
%o	八进制整数
%x	十六进制整数（小写ox）
%X	十六进制整数（大写OX）
%e	科学计数法（小写'e'）
%E	科学计数法（大写'E'）
%g	%f和%e的简写
%G	%f和%E的简写

实例代码：

格式化字符串除了%s，还可以写为`f'{表达式}'`

```
age = 18
name = 'TOM'
weight = 75.5
```

```python
student_id = 1

# 我的名字是TOM
print('我的名字是%s' % name)

# 我的学号是0001
print('我的学号是%4d' % student_id)

# 我的体重是75.50公斤
print('我的体重是%.2f公斤' % weight)

# 我的名字是TOM，今年18岁了
print('我的名字是%s，今年%d岁了' % (name, age))

# 我的名字是TOM，明年19岁了
print('我的名字是%s，明年%d岁了' % (name, age + 1))

# 我的名字是TOM，明年19岁了
print(f'我的名字是{name}，明年{age + 1}岁了')
```

> f-格式化字符串是Python3.6中新增的格式化方法，该方法更简单易读。

3.2 转义字符

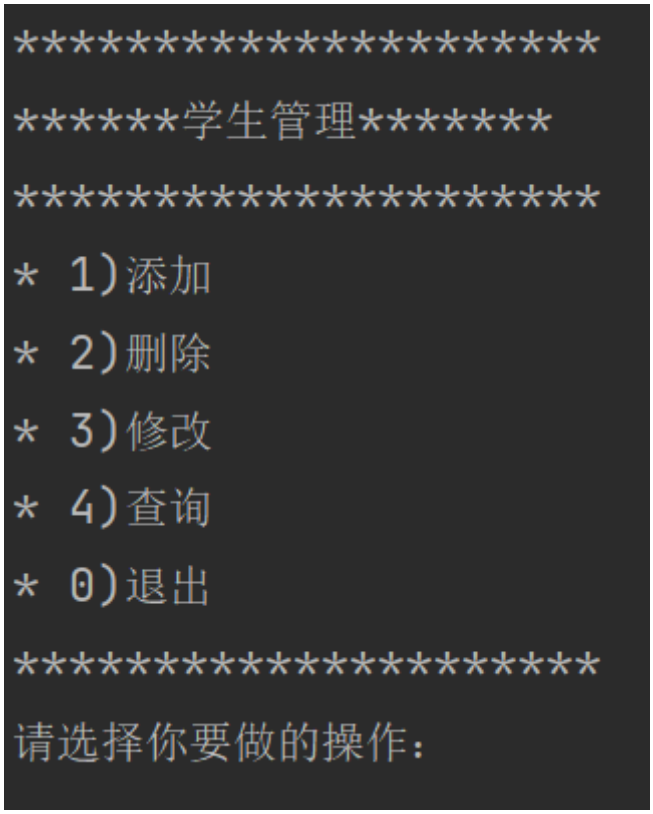
- `\n`：换行。
- `\t`：制表符，一个tab键（4个空格）的距离。

3.3 结束符

> 想一想，为什么两个print会换行输出？

```python
print('输出的内容', end="\n")
```

> 在Python中，print()， 默认自带`end="\n"`这个换行结束符，所以导致每两个`print`直接会换行展示，用户可以按需求更改结束符。




```
print("**********")
print("******学生管理*******")
print("**********")
print("* 1)添加")
print("* 2)删除")
print("* 3)修改")
print("* 4)查询")
print("* 0)退出")
print("**********")
op = input("请选择你要做的操作：")
print("你的选择是：" + op)
```

4. 输入

在Python中，程序接收用户输入的数据的功能即是输入。

4.1 简单语法

```
input("我输入啦")
```

4.2 输入的特点

- 当程序执行到`input`，等待用户输入，输入完成之后才继续向下执行。
- 在Python中，`input`接收用户输入后，一般存储到变量，方便使用。
- 在Python中，`input`会把接收到的任意用户输入的数据都当做**字符串**处理。

```
password = input('请输入您的密码：')

print(f'您输入的密码是{password}')
# <class 'str'>
print(type(password))
```

控制台输出结果如下：

```
C:\Users\Ybin\AppData\Local\
请输入您的密码：123
您输入的密码是123
<class 'str'>
```

> 小结：
>
> - 输入功能
>   - input('提示文字')

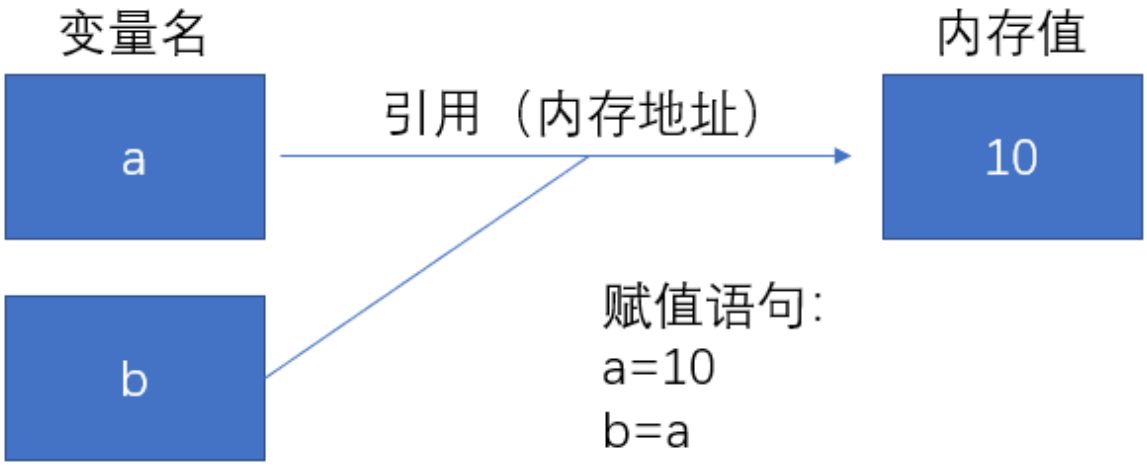
> - 输入的特点
>   - 一般将input接收的数据存储到变量
>   - input接收的任何数据默认都是字符串数据类型

5. 变量

5.1 变量的作用

我们想一想，当我们在计算机上寻找一个文件的时候，是不是首先知道这个文件在什么地方，也就是他的存储路径，是按照一定的组织形式组织起来的数据，每一个文件都是有自己特有的存储位置，当我们照这个文件的时候就可以根据路径快速的找到。

程序中，数据都是临时存储在内存中，为了更快速的查找或使用这个数据，通常我们把这个数据在内存中存储之后定义一个名称，这个名称就是变量。



> 变量就是一个存储数据的的时候当前数据所在的内存地址的名字而已。

5.2 定义变量

```
变量名 = 值
```

> 变量名自定义，要满足**标识符**命名规则。

标识符

标识符命名规则是Python中定义各种名字的时候的统一规范，具体如下：

- 以字母或下划线开头的变量名、函数名、类名、模块名均是标识符
- 支持任意长度，大小写敏感
- 不能与关键字同名
- 不要使用内建标识符
- 带'_'得标识符有特殊含义
- 不能使用数字开头

> 查看系统关键字
>
> import keyword
>
> keyword.kwlist # 查看所有的关键字

```
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', '
def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', '
is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'
]
```

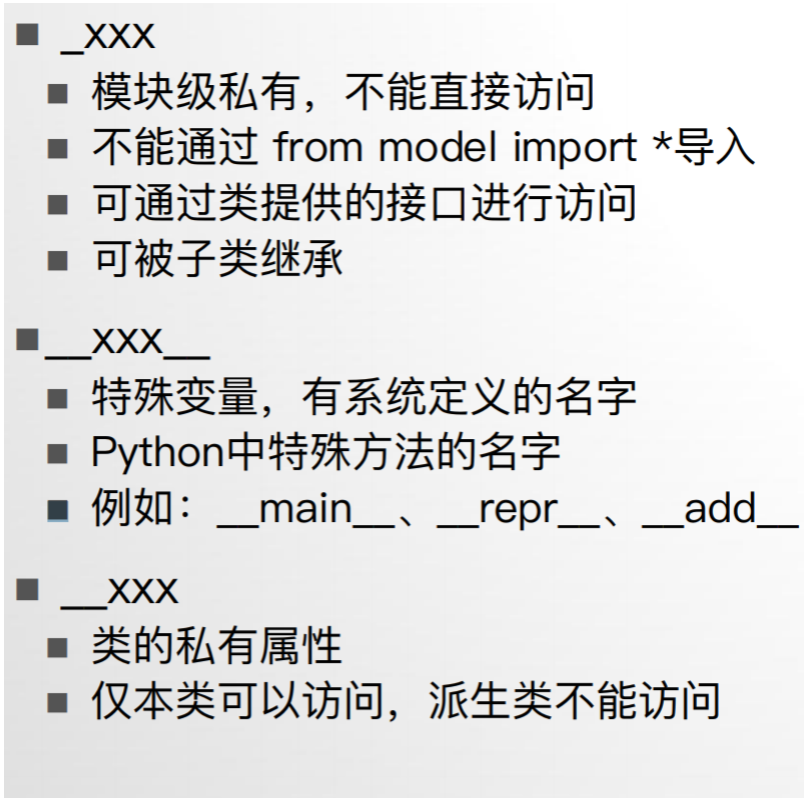
> 内建标识符：
>
> - 非关键字，但不推荐作为标识符使用
> - 在Python解释器中有特殊含义
> - Python程序启动前由解释器自动导入
> - 作为全局变量使用
> - 例如：list、dir、id、dict、str、...

5.3 命名习惯

- 见名知义。
- 大驼峰：即每个单词首字母都大写，例如：`MyName`。
- 小驼峰：第二个（含）以后的单词首字母大写，例如：`myName`。
- 下划线：例如：`my_name`。

5.4 基础拓展

在变量名或者函数名前加下划线一般都会有特殊的含义。



5.5 语法

变量名 = 数据

变量名1 = 变量名2 = 数据

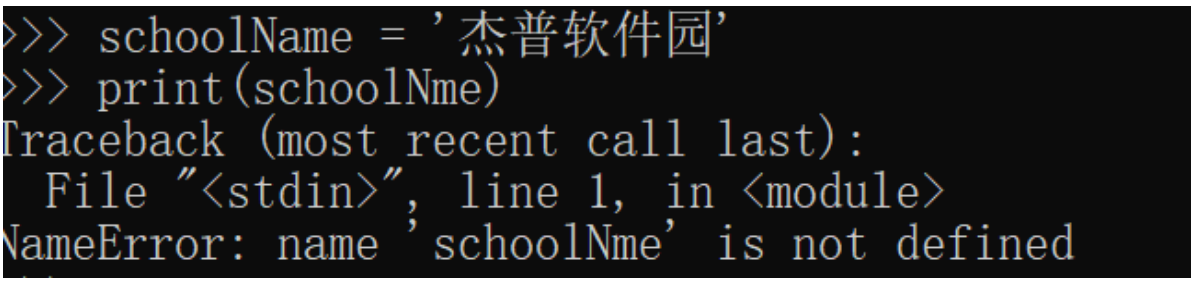
变量名1, 变量名2 = 数据1, 数据2

```python
# 创建变量
name01 = "孙悟空"
name02 = name01
name03, name04 = "唐僧", "八戒"
print(name03 + "," + name04)

# 修改变量
name01 = "悟空"
```

5.6 认识bug

所谓bug，就是程序中的错误。如果程序有错误，需要程序员排查问题，纠正错误。



6.删除语句

6.1 语法:

```python
name01 = "悟空"
name02 = name01
del name01, name02
```

6.2 作用:

删除变量，同时解除与对象的关联。

如果可能则释放对象。

(3) 自动化内存管理的引用计数:

每个对象记录被变量绑定(引用)的数量，当为0时被销毁。

7. 数据

7.1 整形int

(1) 表示整数，包含正数、负数、0。

如： -5, 100, 0

(2) 字面值:

十进制：每位用十种状态计数，逢十进一，写法是0~9。

```python
num01 = 10
```

二进制：每位用二种状态计数，逢二进一，写法是0b开头，后跟0或者1。

```
num02 = 0b10
```

八进制：每位用八种状态计数，逢八进一，写法是0o开头，后跟0~7。

```
num03 = 0o10
```

十六进制：每位用十六种状态计数，逢十六进一，写法是0x开头，后跟0 ~ 9, A ~ F, a ~ f

```
num04 = 0x10
```

7.2 浮点型float

(1) 表示小数，包含正数、负数，0.0。

(2) 字面值：

小数：1.0 2.5

科学计数法：

e/E (正负号) 指数

1.23e-2 (等同于0.0123)

1.23456e5(等同于123456.0)

```
# 小数
num01 = 1.23

# 科学计数法
num02 = 1e-5
print(0.00001)
```

7.3 字符串str

(1) 用来记录文本信息(文字信息)

(2) 字面值：双引号或单引号

7.4 布尔bool

(1) 用来表示真和假的类型

(2) 只有两个值：

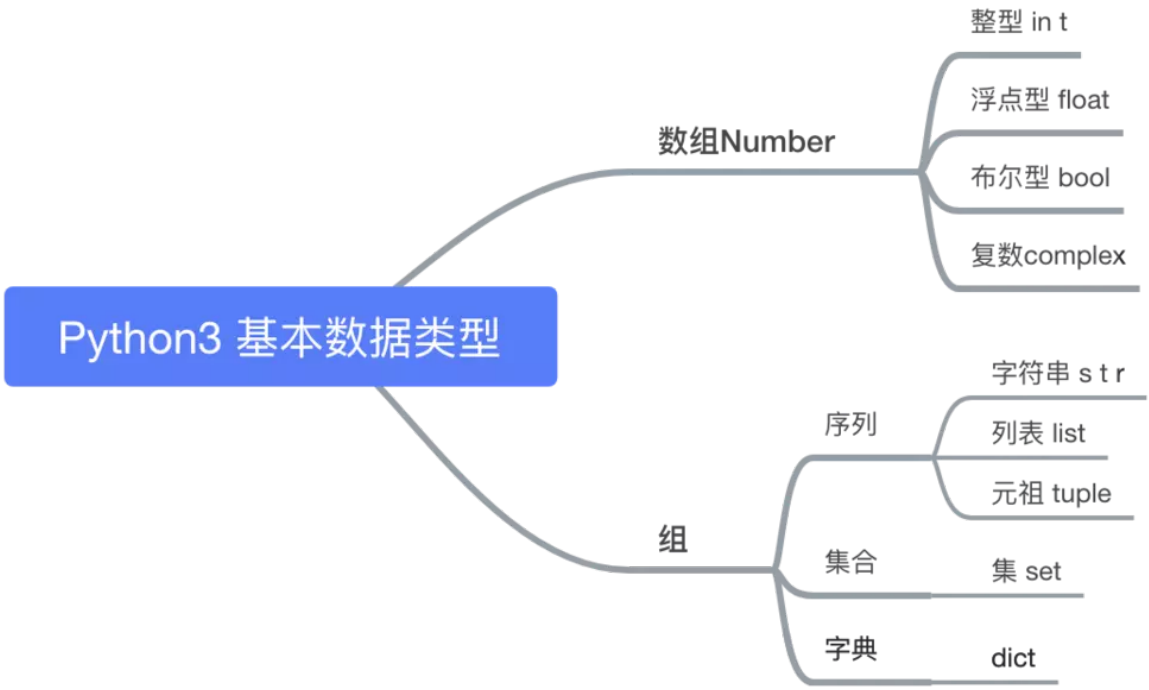
True 表示真(条件满足或成立)，本质是1

False 表示假(条件不满足或不成立)，本质是0

```
result = input("请输入您的职业: ") == "老师"
print(result) # 输入老师,结果True;输入其他,结果False
```

7.5 数据类型

在 Python 里为了应对不同的业务需求，也把数据分为不同的类型。



> 检测数据类型的方法：`type()`

```python
a = 1
print(type(a))  # <class 'int'> -- 整型

b = 1.1
print(type(b))  # <class 'float'> -- 浮点型

c = True
print(type(c))  # <class 'bool'> -- 布尔型

d = '12345'
print(type(d))  # <class 'str'> -- 字符串

e = [10, 20, 30]
print(type(e))  # <class 'list'> -- 列表

f = (10, 20, 30)
print(type(f))  # <class 'tuple'> -- 元组

h = {10, 20, 30}
print(type(h))  # <class 'set'> -- 集合

g = {'name': 'TOM', 'age': 20}
print(type(g))  # <class 'dict'> -- 字典
```

> 小结：
>
> - 定义变量的语法
>
> 变量名 = 值

- 标识符
  - 由数字、字母、下划线组成
  - 不能数字开头
  - 不能使用内置关键字
  - 严格区分大小写
- 数据类型
  - 整型：int
  - 浮点型：float
  - 字符串：str
  - 布尔型：bool
  - 元组：tuple
  - 集合：set
  - 字典：dict

8. 模块基础

模块是Python程序架构的核心概念

以.py结尾的Python源代码均可看作是一个模块

Import关键字可以导入模块，只需输入文件名不必加.py后缀，例如：

- import first

模块不能重复import

模块导入时将会被编译成字节码并执行

重复导入并执行模块需使用reload方法

模块被修改后要通过reload重新载入

9. 数据类型转换

9.1 目标

- 数据类型转换的必要性
- 数据类型转换常用方法

9.2 转换数据类型的作用

问：input()接收用户输入的数据都是字符串类型，如果用户输入1，想得到整型该如何操作?

答：转换数据类型即可，即将字符串类型转换成整型。

9.3 转换数据类型的函数

函数	说明
int(x [,base])	将x转换为一个整数
float(x)	将x转换为一个浮点数

函数	说明
complex(real [,imag])	创建一个复数，real为实部，imag为虚部
str(x)	将对象 x 转换为字符串
repr(x)	将对象 x 转换为表达式字符串
eval(str)	用来计算在字符串中的有效Python表达式,并返回一个对象
tuple(s)	将序列 s 转换为一个元组
list(s)	将序列 s 转换为一个列表
chr(x)	将一个整数转换为一个Unicode字符
ord(x)	将一个字符转换为它的ASCII整数值
hex(x)	将一个整数转换为一个十六进制字符串
oct(x)	将一个整数转换为一个八进制字符串
bin(x)	将一个整数转换为一个二进制字符串

9.4 实际使用

需求：input接收用户输入，用户输入“1”，将这个数据1转换成整型。

```
# str -> int
data01 = int("3")
# int -> str
data02 = str(5)

# str -> float
data03 = float("1.2")
# float -> str
data04 = str(1.2)

# int -> float
data05 = float(250)
# float -> int
data06 = int(1.9)
print(data06)  # 向下取整(截断删除)

# 注意：字符串转换为其他类型时，
# 必须是目标类型的字符串表达形式
# print(int("10.5"))  # 报错
# print(float("abc"))# 报错
```

练习：在终端中输入商品单价、购买的数量和支付金额，计算应该找回多少钱。

效果：

请输入商品单价：5

请输入购买数量：3

请输入支付金额：20

应找回：5.0

10. 运算符

10.1 算术运算符

运算符	描述	实例
+	加	1 + 1 输出结果为 2
-	减	1-1 输出结果为 0
*	乘	2 * 2 输出结果为 4
/	除	10 / 2 输出结果为 5
//	整除	9 // 4 输出结果为2
%	取余	9 % 4 输出结果为 1
**	指数	2 ** 4 输出结果为 16，即 2 * 2 * 2 * 2
()	小括号	小括号用来提高运算优先级，即 (1 + 2) * 3 输出结果为 9

> 注意：
>
> 混合运算优先级顺序：`()` 高于 `**` 高于 `*` `/` `//` `%` 高于 `+` `-`

练习1：在终端中输入疫情确诊人数、治愈人数，计算治愈比例。

格式：治愈比例为xx%

效果：

请输入确诊人数：500

请输入治愈人数：495

治愈比例为99.0%

练习2：古代的秤，一斤十六两，在终端中获取两，计算几斤零几两。

效果：

请输入总两数：100

结果为：6斤4两

练习3：

匀变速直线运动的速度与位移公式：

位移 = 初速度 × 时间 + 加速度 * 时间的平方 / 2

已知(在终端中输入)：位移、时间、初速度

计算：加速度

效果：

请输入位移：100

请输入初速度：6

请输入时间：10

加速度是：0.8

10.2 赋值运算符

运算符	描述	实例
=	赋值	将 = 右侧的结果赋值给等号左侧的变量

- 单个变量赋值

```
num = 1
print(num)
```

- 多个变量赋值

```
num1, float1, str1 = 10, 0.5, 'hello world'
print(num1)
print(float1)
print(str1)
```

结果如下：

```
10
0.5
hello world
```

- 多变量赋相同值

```
a = b = 10
print('a:',a)
print('b:',b)
```

10.3 复合赋值运算符

运算符	描述	实例
+=	加法赋值运算符	c += a 等价于 c = c + a
-=	减法赋值运算符	c -= a 等价于 c = c- a
*=	乘法赋值运算符	c *= a 等价于 c = c * a
/=	除法赋值运算符	c /= a 等价于 c = c / a
//=	整除赋值运算符	c //= a 等价于 c = c // a
%=	取余赋值运算符	c %= a 等价于 c = c % a
**=	幂赋值运算符	c ** = a 等价于 c = c ** a

```
a = 10
a += 20
# 输出30  a = a + 20,最终a = 10 + 20
print(a)

b = 4
b *= 3
# 输出12  b = b * 3,最终b = 4 * 3
print(b)

c = 10
c *= 1 + 2
# 思考：输出是多少？
print(c)

# 输出30，先算运算符右侧1 + 2 = 3，  c *= 3 ，推导出c = 10 * 3
```

练习：在终端中输入一个四位整数，计算每位相加和。

例如：输入1234，打印1+2+3+4结果10

效果：

请输入四位整数：1234

结果是：10

10.4 比较运算符

比较运算符也叫关系运算符， 通常用来判断。

算符	描述	实例

算符	描述	实例
==	判断相等。如果两个操作数的结果相等，则条件结果为真(True)，否则条件结果为假(False)	如a=3,b=3，则（a == b) 为 True
!=	不等于 。如果两个操作数的结果不相等，则条件为真(True)，否则条件结果为假(False)	如a=3,b=3，则（a == b) 为 True如a=1,b=3，则(a != b) 为 True
>	运算符左侧操作数结果是否大于右侧操作数结果，如果大于，则条件为真，否则为假	如a=7,b=3，则(a > b) 为 True
<	运算符左侧操作数结果是否小于右侧操作数结果，如果小于，则条件为真，否则为假	如a=7,b=3，则(a < b) 为 False
>=	运算符左侧操作数结果是否大于等于右侧操作数结果，如果大于，则条件为真，否则为假	如a=7,b=3，则(a < b) 为 False如a=3,b=3，则(a >= b) 为 True
<=	运算符左侧操作数结果是否小于等于右侧操作数结果，如果小于，则条件为真，否则为假	如a=3,b=3，则(a <= b) 为 True

```
a = 100
b = 2
print(a == b)  # False
print(a != b)  # True
print(a < b)   # False
print(a > b)   # True
print(a <= b)  # False
print(a >= b)  # True
```

练习1：写出下列代码表达的命题含义

练习2： 根据命题写出代码

输入的是正数

输入的是月份

输入的不是偶数

10.5 逻辑运算符

运算符	逻辑表达式	描述	实例	
and	x and y	布尔"与"：如果 x 为 False，x and y 返回 False，否则它返回 y 的值。	True and False， 返回 False。	一假俱假

运算符	逻辑表达式	描述	实例	
or	x or y	布尔"或"：如果 x 是 True，它返回 True，否则它返回 y 的值。	False or True， 返回 True。	一真俱真
not	not x	布尔"非"：如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not True 返回 False, not False 返回 True	取反

10.6 身份运算符

(1) 语法：

x is y

x is not y

(2) 作用：

is 用于判断两个对象是否是同一个对象，是时返回True，否则返回False。

is not 的作用与is相反。

10.7 优先级

高到低：

算数运算符

比较运算符

复合赋值运算符

身份运算符

逻辑运算符