学习目标

1. 掌握在 pycharm 中书写 python 程序;
2. 掌握算数运算符的使用;
3. 变量与变量的命名规则;
4. 不同类型变量的转化;
5. input 函数;
6. 变量的格式化输出;

目录

第 1 章 python 基础------认识 python

一、python 的起源

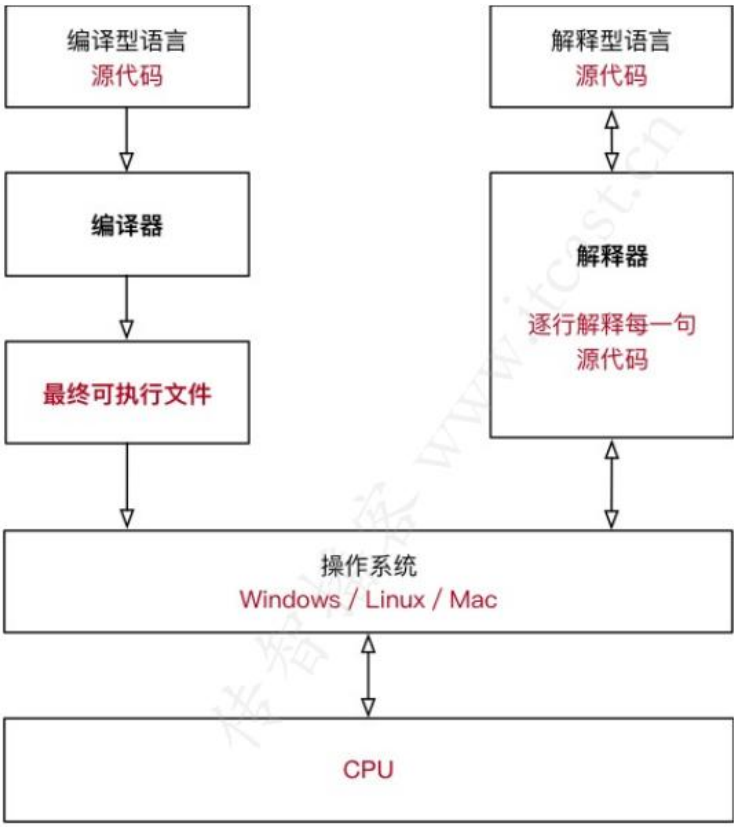
Python 的创始人为吉多·范罗苏姆（Guido van Rossum）。

1991 年，第一个 Python 解释器诞生，它是用 C 语言实现的，并能够调用 C 语言的库文件 。

二、解释器（科普）

计算机不能直接理解任何除机器语言以外的语言，所以必须要把程序员所写的程序语言翻译成机器语言，计算机才能执行程序。将其他语言翻译成机器语言的工具，被称为编译器 。

编译器翻译的方式有两种：一个是编译，另外一个是解释。两种方式之间的区别在于翻译时间点的不同。当编译器以解释方式运行的时候，也称之为解释器。



编译型语言：程序在执行之前需要一个专门的编译过程，把程序编译成为机器语言的文件，运行时不需要重新翻译，直接使用编译的结果就行了。程序执行效率高，依赖编译器，跨平台性差些。如 C、C++ 。

解释型语言：解释型语言编写的程序不进行预先编译，以文本方式存储程序代码，会将代码一句一句直接运行。在发布程序时，看起来省了道编译工序，但是在运行程序的时候，必须先解释再运行 。

三、编译型语言和解释型语言对比

- 执行速度：编译型语言比解释型语言执行速度快 ；
- 跨平台性：解释性语言更容易跨平台，如 java，python；

第 2 章 python 基础------第一个 Python 程序

一、第一个 HelloPython 程序

1. Python 源程序的基本概念

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辑软件做 Python 的开发 。

2. Python 程序的文件扩展名通常都是 .py 。

2. 演练步骤

- 在桌面下，新建 python 目录 ；
- 在 Python 目录下新建 hello.py 文件 ；
- 使用文本编辑器 hello.py 编辑如下内容:

```
print("hello world")
print("hello python")
```

- 在命令终端中输入以下命令执行 hello.py

```
python hello.py
```

print 是 python 中我们学习的第一个函数

print 函数的作用，可以把引号包裹的的内容，输出到屏幕上

3. 演练扩展 —— 认识错误（BUG）

- 关于错误

编写的程序不能正常执行，或者执行的结果不是我们期望的俗称 BUG ，是程序员在开发时非常常见的，初学者常见错误的原因包括：

1. 手误 ；

2. 对已经学习过的知识理解还存在不足 ；

3. 对语言还有需要学习和提升的内容 。

在学习语言时，不仅要学会语言的语法，而且还要学会如何认识错误和解决错误的方法 。

每一个程序员都是在不断地修改错误中成长的 。

- 第一个演练中的常见错误

1> 手误，例如使用 pirnt("Hello world")

NameError: name 'pirnt' is not defined

名称错误：'pirnt' 名字没有定义

2> 将多条 print 写在一行 。

SyntaxError: invalid syntax

语法错误：语法无效

每行代码负责完成一个动作 。

3> 缩进错误

IndentationError: unexpected indent

缩进错误：不期望出现的缩进

Python 是一个格式非常严格的程序设计语言 ；

目前而言，大家记住每行代码前面都不要增加空格。

二、执行 Python 程序的两种方式

1. 命令行运行 python 程序

python 文件.py

2. PyCharm 运行 python 程序

通过集成开发环境 pycharm 编写并运行 python 代码。

第 3 章 python 基础------PyCharm

一、集成开发环境

集成开发环境（ IDE ， Integrated Development Environment）—— 集成了开发软件需要的所有工具，一般包括以下工具：

- 图形用户界面 ；
- 代码编辑器（支持 代码补全 / 自动缩进）；
- 编译器 / 解释器 ；
- 调试器（断点 / 单步执行。

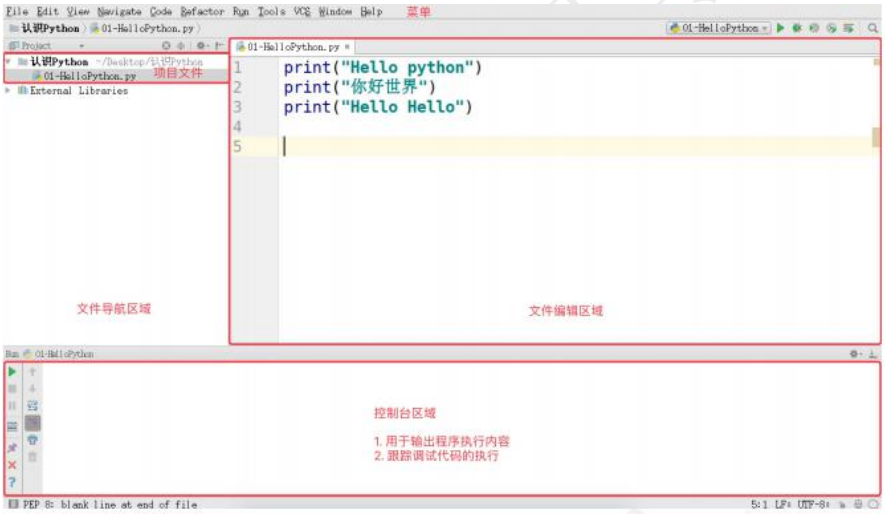
二、PyCharm 介绍

PyCharm 是 Python 的一款非常优秀的集成开发环境 。

PyCharm 除了具有一般 IDE 所必备功能外，还可以在 Windows 、 Linux 、 macOS 下使用 。

PyCharm 适合开发大型项目，一个项目通常会包含很多源文件 ，每个源文件的代码行数是有限的，通常在几百行之内 ，每个源文件各司其职，共同完成复杂的业务功能 。

三、PyCharm 快速体验



文件导航区域能够浏览 / 定位 / 打开 项目文件

文件编辑区域能够编辑当前打开的文件

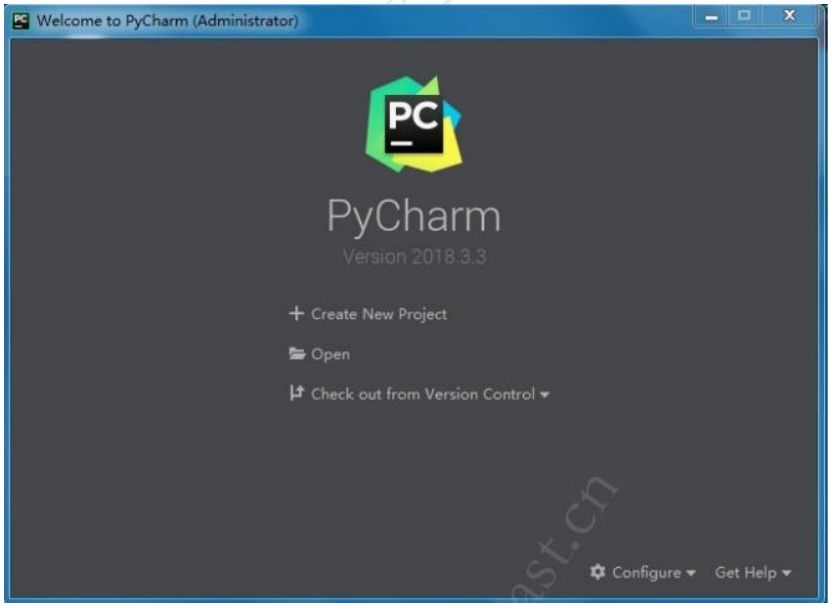
控制台区域能够:

输出程序执行内容

跟踪调试代码的执行

右上角的工具栏能够执行(SHIFT + F10) / 调试(SHIFT + F9)代码

通过控制台上方的单步执行按钮(F8)，可以单步执行代码



四、新建/打开一个 Python 项目

1. pycharm 项目简介

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 。

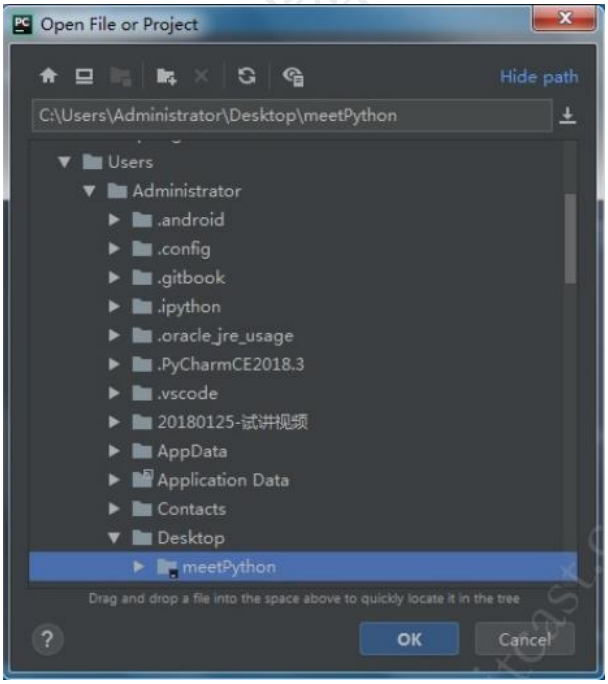
通常每一个项目就具有一个独立专属的目录，用于保存所有和项目相关的文件 。

一个项目通常会包含很多源文件 。

2. 打开 Python 项目

直接点击 Open 按钮，然后浏览到之前保存 Python 文件的目录，既可以打开项目 。

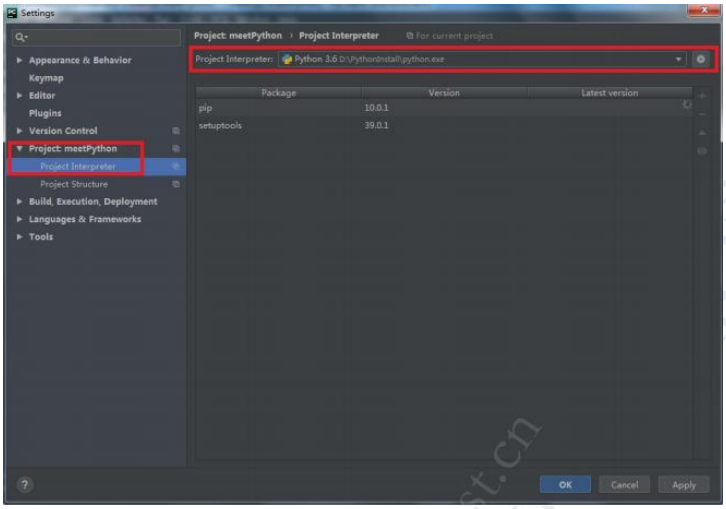
第一次打开项目，需要耐心等待 PyCharm 对项目进行初始设置



3. 设置项目使用的解释器版本

打开的目录如果不是由 PyCharm 建立的项目目录，有的时候 使用的解释器版本是 Python 2.x 的，需要单独设置解释器的版本 。

通过 File / Settings... 可以打开设置窗口，如下图所示



4. 新建项目

- 命名文件名时建议只使用小写字母、数字和下划线 ；
- 文件名不能以数字开始

通过 欢迎界面 或者菜单 File / New Project 可以新建项目 。

五、pycharm 新建项目演练

- 新建 3-code 项目；
  - 注意:
  - 教学中的文件名不规范，目的是为了方便同学学习和课后整理代码;
  - 实际工作中文件名只能使用字母、数字和下划线，文件名不能以数字开始;

- 在项目下新建 hello.py 文件 。
- 编写 print("Hello Python") 代码。

六、多文件项目的演练

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 。

1. 在 3-code 项目中新建一个文件 world.py
2. 在 world.py 文件中添加一句 print("hello world")
3. 点击右键选择 Run ‘world’

- 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。

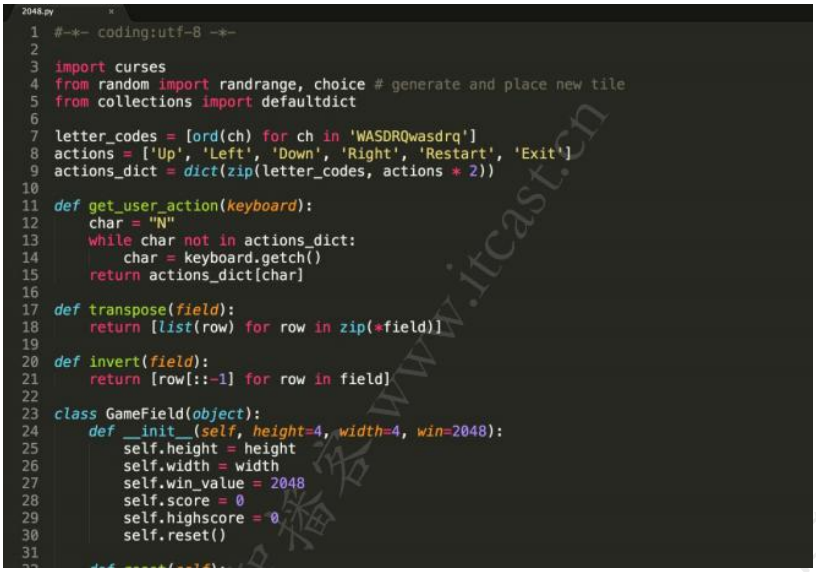
对于商业项目而言，通常在一个项目中，只有一个可以直接执行的 Python 源文件 。

第 4 章 python 基础------程序基本构成

一、注释

1. 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。



```
#-*- coding:utf-8 -*-

import curses
from random import randrange, choice # generate and place new tile
from collections import defaultdict

letter_codes = [ord(ch) for ch in 'WASDRQwasdrq']
actions = ['Up', 'Left', 'Down', 'Right', 'Restart', 'Exit']
actions_dict = dict(zip(letter_codes, actions * 2))

def get_user_action(keyboard):
    char = "N"
    while char not in actions_dict:
        char = keyboard.getch()
    return actions_dict[char]

def transpose(field):
    return [list(row) for row in zip(*field)]

def invert(field):
    return [row[::-1] for row in field]

class GameField(object):
    def __init__(self, height=4, width=4, win=2048):
        self.height = height
        self.width = width
        self.win_value = 2048
        self.score = 0
        self.highscore = 0
        self.reset()
```

2. 单行注释

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，只起到辅助说明作用 。

示例代码如下：

```
# 这是单行注释
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。

3. 多行注释

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：

```
'''
这是多行注释
多行用三个引号来注释
'''
print("hello python")
```

4. 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 。

二、算数运算符

1. 算数运算符定义

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
//	取整除	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。

2. 算数运算符的优先级

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

例如:

2 + 3 * 5 = 17

(2 + 3) * 5 = 25

2 * 3 + 5 = 11

2 * (3 + 5) = 16

三、变量简介

1. 变量的定义与赋值

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。

2. 变量演练

python 中字符串用单引号或者双引号引起来，数字不需要引号。

在程序中，如果要输出变量的内容，需要使用 print 函数

```
name = "妲己"
age = 25
print(name)
print(age)
```

3. 变量结合运算符演练 —— 超市买苹果

- 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

```
# 定义苹果单价变量
price = 8.5
# 定义购买重量变量
weight = 7.5
# 计算金额
money = price * weight
# 显示结果
print(money)
```

- 思考题

如果买 10 斤苹果，就返 5 块钱

请重新计算购买金额

```
# 定义苹果单价变量
price = 8.5
# 定义购买重量变量
weight = 10
# 计算金额
money = price * weight
# 减5块钱
money = money - 5;
# 显示结果
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。

4. 课堂练习一：算数运算符演练

两个变量

```
a = 12
b = 3
```

- 求 a 加 b 的结果；
- 求 a 减 b 的结果；
- 求 a 乘以 b 的结果；
- 求 a 除以 b 的结果；
- 求 a 除以 b 取余数；
- 求 a 除以 b 的取整数；
- 求 a 的 b 次方。

计算结果通过 print 函数显示出来。

5. 课堂练习二：两个变量的值交换

a = 10

b = 20

如何让变量 a 和 b 的值交换?

四、变量的命名

1. 标识符

标示符就是程序员定义的 变量名、函数名 ；

名字需要有见名知意的效果，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 ；
- 不能是括号以及各种特殊符号;
- 不能与**关键字**重名 ；

思考：下面的标示符哪些是正确的，哪些不正确为什么?

fromNo12
from(12)

my_Boolean
my-Boolean
Obj2
2ndObj
myInt

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 。

通过以下代码可以查看 Python 中的关键字 ：

```
import keyword
print(keyword.kwlist)
```

提示：关键字的学习及使用，会在后面的课程中不断介绍。

3. 变量的命名规则

命名规则可以被视为一种惯例，并无绝对与强制目的是为了增加代码的识别和可读性 。

注意 Python 中的标识符是区分大小写的

1. 在定义变量时，为了保证代码格式， = 的左右应该各保留一个空格；

2. 在 Python 中，如果变量名需要由两个或多个单词组成时，可以按照以下

方式命名 ：

i. 每个单词都使用小写字母

ii. 单词与单词之间使用 _ 下划线 连接

iii. 例如： first_name 、 last_name 、 qq_number 、 qq_password

- 驼峰命名法

  当变量是由两个或多个单词组成时，还可以利用驼峰命名法来命名

- 小驼峰式命名法

  第一个单词以小写字母开始，后续单词的首字母大写 。

  例如： firstName 、 lastName

- 大驼峰式命名法

  每一个单词的首字母都采用大写字母 。

  例如： FirstName 、 LastName 、 camelCase

如： userName userLoginFlag

五、变量的类型

数据类型可以分为数字型和非数字型。

1. 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

2. 非数字型

- 字符串 (str);
- 列表(list);

- 元组(tuple)；
- 集合(set)；
- 字典 (dictionary)。

3. None

None 代表具体类型待定，或者说不确定具体类型。

```
# 变量 a 为空类型
a = None
```

六、变量类型的演练 —— 个人信息

- 定义变量保存小明的个人信息

  姓名：小明 ；
  年龄：18 岁 ；
  性别：男；
  身高：1.75 米 ；
  是否为学生：是（True）

```
name = "小明"
age = 18
sex = "男"
height = 1.75
is_student = True
```

- 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，自动推导出变量存储数据的类型

七、不同类型变量之间的计算

1. 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

使用上述三个变量相互进行算术运算

2. 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

```
first_name = "张"
last_name = "三"
name = first_name + last_name
print(name)
```

3. 字符串变量可以和整数使用 * 重复拼接相同的字符串

```
str1 = "张"
str2 = str1 * 5
print(str2)
```

4. 数字型变量和字符串之间不能进行其他计算

```
first_name = "张"
age = 20
abc = first_name + age
# TypeError: can only concatenate str (not "int") to str
```

5. 课堂练习----

不同类型变量的计算

```
# 不同类型变量的计算结果
a = True
b = False
c = "你好"
d = 3.14
e = 5
# 计算 a + b = ?
# 计算 c + d = ?
# 计算 c * e = ?
# 计算 d * b = ?
```

八、不同类型变量的转化

1. 数字类型转化为字符串类型

语法：str(数字)

```
a = 20
#把 a 转化为字符串
str(a)
```

2. 字符串类型转化为整型

语法：int(字符串)

```
a = "123"
# 转化为 int 型
int(a)
```

3. 字符串类型转化为浮点型

语法：float(字符串)

```
a = "3.5"
# 转化为 float 型
float(a)
```

4. 课堂练习----

类型转化

```
a = "123"
b = 456
# 把 b 转化为字符串，与 a 拼接成一个字符串"123456"，并用 print 显示结果
```

```
# 把a转化为数字，结果与b相加，并用print显示结果
```

5. 课堂练习----

四舍五入

```
# a = 任意数字
# b = 任意数字
# 求a / b的结果，要求结果只保留整数，并且四舍五入
```

九、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。

1. input 函数实现键盘输入变量的值

在 Python 中可以使用 input 函数从键盘等待用户的输入

语法如下：

变量 = input("提示信息：")

之前演练的变量值都是程序中写死的，通过 input 函数就可以在程序运行过程中动态的给变量赋值了。

```
# 通过input函数输入变量name的值，
# 通过print函数把name的值通过屏幕打印出来
name = input("请输入姓名")
print(name)
```

- 用户输入的任何内容 Python 都认为是一个字符串

2. 课堂练习----

input 函数

```
a = input("请输入任意数字")
b = input("请输入任意数字")
print(a + b)
# 检查a + b 的结果是什么？
```

- 如果用户需要输入数字，那么需要通过类型转换函数将类型转化为数字

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

3. 变量输入演练 —— 超市买苹果增强版

- 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

```
# 输入苹果的价格
price = input("请输入苹果单价")
# 输入要买的斤数
weight = input("请输入要买的斤数")
# 把输入结果转化为小数并计算和显示总价
money = float(price) * float(weight)
```

```
print(money)
```

- 提问

1. 演练中，针对价格定义了几个变量?

三个

price 记录用户输入的价格字符串 ；

weight 记录重量字符串；

money 记录转换后的价格数值 ；

十、 变量的格式化输出

1. 格式化字符

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，一起输出数字，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	说明
%s	字符串
%d	有符号十进制整数，%06d 表示输出 6 位整数，不足用 0 补全
%f	浮点数，%.2f 表示只显示小数点后两位
%%	输出%

2. 语法格式

print("格式化字符串" % 变量 1)

print("格式化字符串" % (变量 1，变量 2...))

3. 格式化输出 —— 个人名片

在控制台依次提示用户输入：公司、姓名、电话、邮箱

输出结果：

```
**********
公司名称:传智播客
姓名:王老师
电话：12345678
邮箱：12345678@itcast.cn
**********
```

4. 课堂练习---

格式化输出

1. 定义字符串变量 name = “小明”，输出：**我的名字叫小明，请多多关照！**

2. 定义整数变量 num = 1，输出：**我的学号是 000001**

3. 定义小数 price = 8.5、 weight = 5 ，输出：**苹果单价 8.5 元／斤，购买了 5.00 斤，需要支付 42.50 元**

4. 定义一个小数 scale = 10.01 ，输出：**数据是 10.01%**

十一、字符串中的转义字符

1. print 函数的输出默认是回车结尾

```
# 两个print 函数会输出两行内容
print("hello world")
print("hello python")
```

2. 多个 print 函数的输出结果打印到一行

```
# 多个print 函数会输出结果打印到一行
print("hello world", end="")
print("hello python")
```

3. 转义字符

\t 在控制台输出一个制表符(tab)，制表符的功能是在不使用表格的情况下在垂直方向对齐， 这样通过 print 函数输出文本时可以保持垂直方向对齐；

\n 在控制台输出一个换行符；

转义字符	描述
\\	反斜杠符号
\'	单引号
\"	双引号
\n	换行
\t	横向制表符

4. 禁止转义字符串

如果需要字符串输出”\n”或者”\t”，而不是转义，那么就需要在字符串前面加 r。

- 分析如下两行代码输出时的差异：

```
print("hello\tworld\nhello\\world")
print(r"hello\tworld\nhello\\world")
```