第一篇

Python编程基础

第 1 章　Python 语言概述

1.1　Python 语言发展

你可能已经听说过很多种流行的编程语言，比如 C/C++、Java 等等。Java 占据了世界上绝大部分电商、金融、通信等服务端应用开发，C/C++占据了世界上绝大部分贴近操作系统的硬件编程、嵌入式编程。而目前，由于大数据、人工智能(AI)的流行，python 已经变得非常重要，大有“三分天下”之势。

1.1.1　Python 是一种什么语言？

python 也是一种计算机程序设计语言，由荷兰人 Guido 发明。Guido 希望有一种语言，这种语言能够像 C 语言那样，能够全面调用计算机的功能接口，又可以像 Unix 的 shell 调用那样，可以轻松的编程。

1991 年，第一个 python 编译器诞生。它是用 C 语言实现的。中间陆续进行了多个版本的演进。2014 年的 python2.7 是 2.x 系列的最终版本。截止 2021 年 2 月份，**最新版本是 3.9.2**。注意：2020 年之后不再支持 2.x 版本，因此 python 编程，**请尽量使用 3.x 版本**。本教材使用 3.7 版本，不建议使用最新版本 3.9，因为很多学习资料不是使用最新的。

1.1.2　Python 语言的特点

python 是一种解释型脚本语言。主要特征有：

（1）在众多语言中，python 是易学易懂，功能强大的语言，也是最好的入门语言之一。这是 python 具有巨大吸引力的一大特点。

（2）python 的语法非常清晰，它甚至不是一种格式自由的语言。例如，它要求 if 语句的下一行必须向右缩进，否则不能通过编译。

（3）python 的**可扩展性**体现为它的模块，python 具有脚本语言中最丰富和强大的类库，这些类库覆盖了文件 I/O、GUI、网络编程、数据库访问、文本操作等绝大部分应用场景。

（4）python 作为一门**解释型**的语言，它具有**跨平台**的特征，只要为平台提供了相应的 python 解释器，python 就可以在该平台上运行。

缺点：

（5）**速度慢**：python 程序比 Java、C、C++等程序的运行效率都要慢。

（6）源代码加密困难：不像编译型语言的源程序会被编译成目标程序，python 直接运行源程序，因此对源代码加密比较困难。

关于速度慢的问题，可以从两个方面考虑：目前计算机的硬件速度越来越快，我们往往更关注开发过程的效率和可靠性，而不是软件的运行效率。另外，在速度要求比较高的场合，python 程序员可以深入底层，写 C 程序，编译为 .so 文件引入到 python 中使用。关于加密问题，现在软件行业的大势本来就是开源，就像 Java 程序同样很容易反编译，但丝毫不会影响它的流行。

总之，正是由于 python 语言的简单易用性，使得多数专业技术人员从繁杂的编程工作中解放出来，可以更加专注于专业工作。也正是由于 python 语言的易扩展性和易用性，使得为其提供第三方库提供了更大的可能性，特别是**随着大数据、人工智能（AI）的发展，python 逐渐成为一种流行的编程工具。**

1.2 集成开发环境

1.2.1 Python 的安装

下载地址：https://www.python.org/downloads/，可以选择操作系统和安装的 python 版本。本书以 windows10 操作系统和 python3.7（安装在 C:\Program Files\python37\目录）为例（图 1.1 为默认安装后在资源管理器中的样子）。

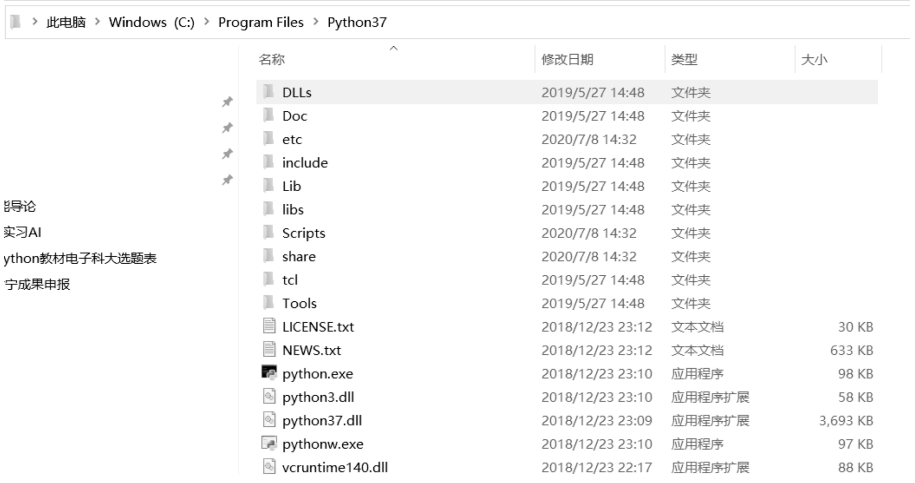

图 1.1 python 安装目录以及“python.exe”命令的位置

安装后在 Windows 的 cmd 窗口中输入“python”命令回车（即执行 python.exe 文件），输出结果如图 1.2 则说明安装正确（在操作系统“开始”菜单上右键点击“运行”输入 cmd 按回车键，就会打开 cmd 命令窗口）。

```
C:\Users\admin>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 1.2　安装 python 后的运行窗口

如果没有弹出如上关于 python 的版本信息，提示“python 不是内部或外部命令，也不是可运行的程序或批处理文件”，这是因为 Windows 不知道你输入的“python”(即 python.exe)命令在哪里。有两种处理方法：

第一，到 python 的安装目录去运行 python 程序，每次都要把安装目录作为当前目录，才能运行(图 1.3)。或者如果当前在其他目录，就要输入完整的 python 路径才能运行。这比较麻烦。

```
C:\Users\admin>cd \program files\python37

C:\Program Files\Python37>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 1.3　使用 cd 命令转到 python 安装目录运行 python.exe

第二，我们先告诉操作系统 python 程序在哪里，操作系统会自动去找我们输入的 python.exe命令。Windows 会根据一个叫 Path 的环境变量设定的路径去查找我们在 cmd 窗口中输入的命令。因此需要我们把安装的 python 路径加入到 path 中去。步骤如下：

小贴士

基本上，所有编程语言工具的安装都需要 path 环境变量的设置。基本原理和过程同上，请务必弄懂。

(1) 右键点击“我的电脑”→属性→高级系统设置→环境变量；

(2) 在系统变量中找到 path，编辑，把“C:\Program Files\python37\”加入进去(这个路径是作者电脑的安装路径，读者应替换为自己的 python 的安装路径)；

(3) 确定，关闭窗口。

其实，**python 在安装过程中有一个是否加入到 path 环境变量的选项，选中后就不需要手动加入 path 了。**前述安装过程之所以没有提到，正是由于设置 path 几乎是很多编程语言工具所需要的安装配置过程的基本操作，必须熟练掌握。

1.2.2　Python 基本开发环境

再次打开 cmd 窗口，运行 python 命令，或者在“开始”菜单找到 python 程序组，也可以点击运行 cmd 下的 python 程序。在“>>>”提示符下，可以运行 python 的代码了。输入语句和运行结果如图 1.4 所示。

小贴士

本节内容基本熟悉即可，我们一般不在这个环境下进行开发。我们会使用较“专业”的开发环境。

```
C:\Users\admin>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a=10
>>> b=20
>>> a+b
30
>>>
```

图 1.4　python 自带的代码工具

可以看到，上述代码的运行是“一条一条”输入运行的(再次印证了 python 是解释型语言)，这就是交互式开发。输入“exit( )”可以退出 python 环境。

在 cmd 窗口下运行 python 代码不是很方便，python 提供了 IDLE 环境，同样在“开始”菜单的 python 程序组下。如果我们要写一段代码后再“一起”运行，可以在 IDLE 环境下新建文件，输入代码，保存，按 F5 运行显示结果。如图 1.5 所示。

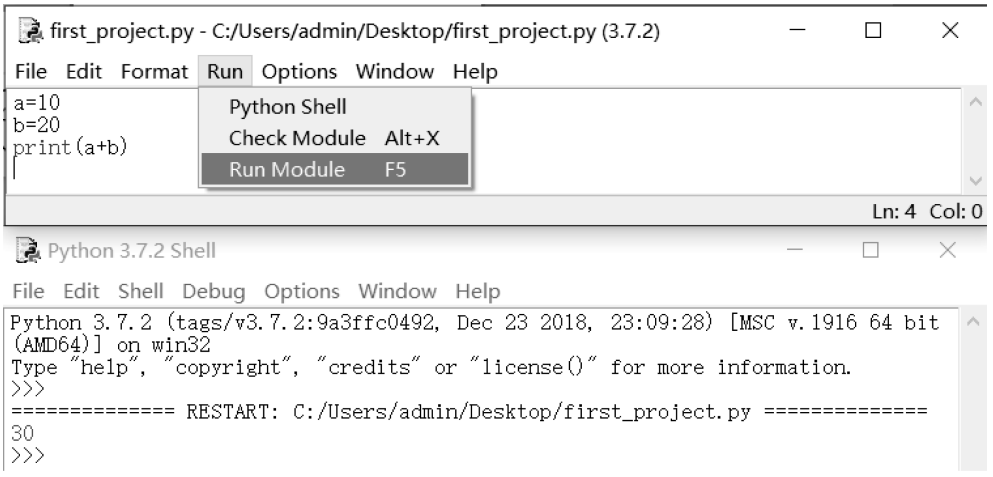

图 1.5　使用 IDLE 运行一段代码

1.2.3　安装、使用 Jupyter 环境

1. 为什么使用 Jupyter?

有很多的 python IDE(Integrated Development Environment-集成开发环境)工具可以用：

(1) PyCharm 是一款比较好用比较流行的 IDE；

(2) Microsoft Visual Studio(简称 VS)，也可以配置 python 的开发 IDE；

(3) Spyder 也是使用 python 编程语言进行科学计算的集成开发环境。

还有其他。上述 IDE 固然是比较专业的，但是作者**推荐初学者使用 jupyter notebook**(新版本名字叫 jupyter lab，都可以用)。理由如下：

(1) jupyter 具备更强的交互能力，代码既可以“一条一条”执行，也可以任意划分为许多“段落”分别运行，已经成功运行的代码不需要再次运行，这使得初学者更容易定位 bug 的位置；

(2) jupyter 很像一个 Web 应用，应用中的代码和结果可视化，初学者可以使用这个工

具快速地展现自己的想法，能够快速的理解、熟悉和掌握代码；

（3）jupyter 类似于写一本书或者一篇文章。初学者学习使用它做学习笔记非常容易。你的代码、代码运行结果和文字、解释、图片、链接等都可以装在里面。

本教材后续所有的代码都是在 jupyter notebook 中实现并运行。

2. python 包的安装原理

先看一下 python 包的安装原理。以管理员权限打开一个 Windows 命令行（cmd 窗口），输入：pip install numpy，回车。就会从 python 的官网安装 numpy 包（官网地址：pypip. org），但是你会发现很慢，因为服务器在国外。国内也有镜像服务器：

（1）http://mirrors. aliyun. com/pypi/simple/

（2）https://pypi. tuna. tsinghua. edu. cn/simple/

因此，安装包的命令为（使用第二个，清华大学服务器。也可以使用第一个）：

pip→install→-i→https://pypi. tuna. tsinghua. edu. cn/simple/→XXX_name（注意→代表此处是空格，XXX_name 处代表的是要安装的包的名字，install 后的 i 字母的前面有个减号）。

python 中的包都可以按照这个过程安装。

3. 安装并启动 jupyter notebook

（1）输入 pip install -i https://pypi. tuna. tsinghua. edu. cn/simple/jupyter 回车，最后几行中出现的信息中包含“successfully installed”表示安装成功。

（2）在命令行输入 jupyter notebook，回车运行。图 1. 6 表示启动 jupyter 服务（服务器 Server），图 1. 7 表示启动 jupyter 客户端（浏览器 Browser）。因此**jupyter 是基于 B/S 模式的。**如果没有正常启动图 1. 7，则继续按照后续步骤操作，如果已经启动则不需要。注意，**不能关闭图 1. 6 窗口**，否则图 1. 7 中的 jupyter 客户端是不能工作的。

```
C:\Users\admin>jupyter notebook
[I 13:37:06.723 NotebookApp] [jupyter_nbextensions_configurator] enabled 0.4.1
[I 13:37:07.107 NotebookApp] JupyterLab extension loaded from c:\program files\python37\lib\site-packages\jupyterlab
[I 13:37:07.107 NotebookApp] JupyterLab application directory is c:\program files\python37\share\jupyter\lab
[I 13:37:07.114 NotebookApp] Serving notebooks from local directory: E:/python
[I 13:37:07.114 NotebookApp] The Jupyter Notebook is running at:
[I 13:37:07.114 NotebookApp] http://localhost:8888/?token=a0b588b8a48b95a08c393dd27e417f3e814dd7cadce4d7ff
[I 13:37:07.115 NotebookApp]  or http://127.0.0.1:8888/?token=a0b588b8a48b95a08c393dd27e417f3e814dd7cadce4d7ff
[I 13:37:07.115 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:37:07.317 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/admin/AppData/Roaming/jupyter/runtime/nbserver-5372-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=a0b588b8a48b95a08c393dd27e417f3e814dd7cadce4d7ff
     or http://127.0.0.1:8888/?token=a0b588b8a48b95a08c393dd27e417f3e814dd7cadce4d7ff
```

图 1. 6　运行 jupyter 后的结果（jupyter 服务器）

（3）在运行结果上选中图 1. 6 小方框内的三个地址区域，按回车键（回车代表复制选中区域）；

（4）打开记事本，粘贴，然后复制三个网址中的一个；

（5）打开浏览器，在地址栏粘贴网址，回车，就可以看到 jupyter 的运行界面，如图 1. 7。

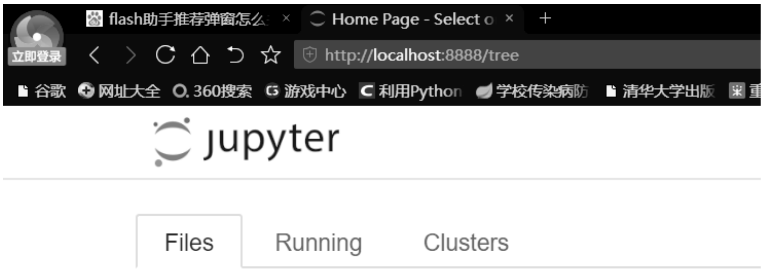

图 1.7　运行 jupyter 后的结果(jupyter 客户端)

1.3　著名的 Hello World 程序

按 1.2.3 节中的方法启动 jupyter。在图 1.8 中点击右侧“new”新建一个“python”应用。在 In[ ]右边的框中输入“print('hello world')”，按 ctrl+enter 键运行可以输出“hello world”字符串。如图 1.8 所示。

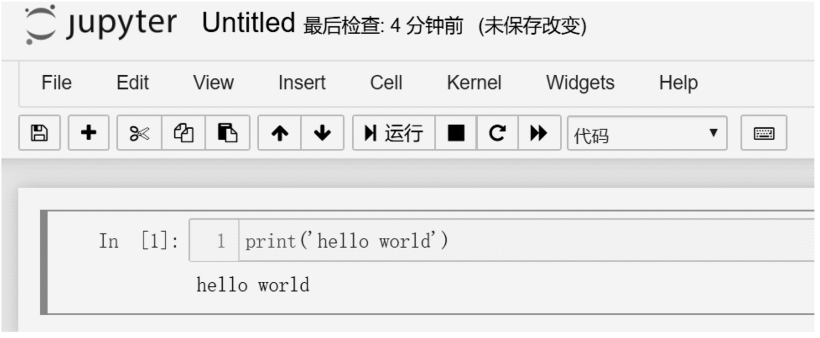

图 1.8　启动 jupyter，新建 jupyter 的 python 文件并运行

1.4　Python 语言的书写规则与执行的一般方法

先看一个实例，代码如下：

例 1.1　语法示意代码

```
'''
  author:  Tim
  date: 2020年3月
'''
#这一行都是注释语句，本程序功能：若a是最大的则输出a
a=\
  10          # a=10语句写在两行上，行尾用续行符斜杠，不建议。
b=50;c=30     # 一行写多条语句用分号分开,不建议。
if a<b:       #表示层次结构的顶层，用冒号表示开始
     pass     #使用pass语句代表什么也不做。如果不写会报错。
else:
     if a>=c:
          print(a) #不同的缩进代表不同的层次
```

（1）注释：python 中的**注释使用符号#**。注释可以是独立的行(第 5 行)，也可以出现在行中(第 7 行等)，#之后的内容为注释。注意 python 没有多行注释，往往使用成对的三单引号(或三双引号)，如第 1 行到 4 行，引号内的是 python 的 raw string(原始字符串)，解释器会忽略，此处相当于多行注释的作用。

（2）python 中的**缩进代表代码的层次**(第 10/12/13 行)。缩进以冒号：开始，接下来的行以多个空格缩进，一般使用四个空格。不同的缩进代表不同的层次，缩进相同的一组语句构成一个代码块。

（3）第 10 行，**空语句，要使用 pass 语句，**否则报错。也可以使用分号，但从程序的易读性考虑，不建议使用分号。

（4）多条语句可以写在同一行，使用分号隔开如第 8 行。不建议。

（5）一条语句可以写在多行，使用符号“\”续行，如第 6/7 行。不建议。语句中包含[]，{}或()括号就不需要使用多行连接符。

1.5　基本输入输出的使用

1.5.1　使用 input 函数输入

例 1.2 代码，基本功能是要分别输入两个整数，输出其和。

第 1 行**调用内置函数 input()进行数据的输入。**函数中的引号内的内容是提示字符串，提示用户。**从键盘输入的任何字符(包含数字)都被当成字符串**赋值给变量 a。

第 2 行输出 a 的类型(使用 type 内置函数

Windows 下 tab 键一般代表四个空格，但是不建议用 tab 键，因为在另一种操作系统下，对 tab 的解释会不一样，这样就可能带来意料之外的错误。

获得类型，使用 print 函数输出到屏幕）。

第 3 行输出 a 的值。

第 5 行输出 a+b，结果是“200300”，再次说明 a 和 b 是字符串，字符串的“+”运算是连接字符串。

第 6、7 行对 a 和 b 进行类型转换，利用 int( ) 函数转为整数。因此，**从键盘输入的数据如果要作为非字符串进行操作，就要进行类型转换或格式转换。**

第 8 行输出 a 和 b 的和。结果为 500。

例 1.2　输入函数的使用

```
1  a=input('请输入第一个加数：')#输入当成字符串处理
2  print(type(a))
3  print(a)
4  b=input('请输入第二个加数：')
5  print(a+b)#字符串的加，是连接作用
6  a=int(a)#类型转换为浮点型好进行加法运算
7  b=int(b)
8  print(a+b)
```

运行结果：

```
请输入第一个加数：200
<class 'str' >
200
请输入第二个加数：300
200300
500
```

1.5.2　使用 print 函数输出

python 使用内置函数 print( ) 进行输出操作。函数原型如下：

print(value, …, sep = ' ', end = '\n' , file = sys. stdout, flush = False)

sep 参数指定输出的多个表达式的值之间用什么符号分开，可不设置，默认是空格。end 参数指定输出的行尾符号，默认是换行符“ \ n”，亦即默认每个 print 函数输出完都要换行。

例 1.3　格式化输出

```
1  a=30
2  b=20
3  print('hello world:', a, '+', b, '=', a+b)
4  #连续输出多个不同类型的值,非格式化输出的时候会用。像这种格式化输出一般不用
5  print('hello world:%d+%d=%d' %(a, b, a+b))
6  # 较早的输出格式,不建议用
7  print('hello world:{0}+{1}={2}'. format(a, b, a+b))
8  # 连续输出多个不同类型的值,使用字符串的格式化 format 操作
9  print(f'hello world:{a}+{b}={a+b}')
10  #连续输出多个不同类型的值。新格式规范,使用 f 引导字符串,{ }引用变量
```

输出结果如下：

hello world：30+20=50

hello world：30+20=50

hello world：30+20=50

hello world：30+20=50

根据上例，基本上，把 print( ) 函数的输出划分为四种形式：

（1）第一种，print( ) 函数可以输出多个表达式的值，每个表达式用逗号分隔，如例 1.3 第 3 行。这种方式仅仅是简单罗列要输出的项，没有格式控制。

（2）第二种，例 1.3 第 5 行，使用%引导占位符(类似于 C 语言的格式控制)。字符串内%后紧跟格式控制符:%d-整数、%f-浮点数、%s-字符串、%x-十六进制整数，等等。在%和 d、f、s 等字符之间还可以加入长度控制(这里不详细介绍，参见例 1.4)。不建议使用。

（3）第三种，例 1.3 第 7 行使用字符串对象的 format( ) 函数。字符串内使用“{索引:格式控制符}”的形式进行控制，format 函数里面的参数是输出的表达式。其中索引代表 format 函数的参数的位置(从 0 开始)，见例 1.4 第 7 行。

（4）第四种，例 1.3 第 9 行，在字符串前**使用格式控制符“f”**(format 的第一个字母)。字符串内的表达式仍然使用大括号，括号内是表达式本身，表达式的后面加冒号进行格式控制，如例 1.4 第 8/10 行。这种 f-string 格式化输出方式 python3.6 版本后才支持，**目前是最容易和方便使用的，强烈建议使用。**

例 1.4　更复杂一点的格式控制输出示例

```
1   a=3.1415926
2   b='重庆理工大学'
3   print("@%f# @%9.2f#, @%.2f#, @%9.2f#, @%10s#"%(a, a, a, a, b))
4   #仔细观察上面输出语句中格式控制的作用。
5   #为了看清每个输出的开始和结束,特意使用@和#,表示每个输出的起止。
6   print("%9.2f"%(a), end=",")
7   print("{0:9.2f}".format(a), end=", ")
8   print(f"{a:9.2f}")
9   #仔细观察以上三条语句,体会哪种更容易理解和使用。
10  print(f"假设 π 的值为{a:.2f}, 那么 2π 的值为{2*a:.2f}。")
```

输出结果为：

@3.141593# @　　3.14#, @3.14#, @　　3.14#, @　　重庆理工大学#

　　3.14,　　3.14,　　3.14

假设 π 的值为 3.14，那么 2π 的值为 6.28。

1.6 Python 文件名

在 python 中，不同的文件扩展名代表不同的文件类型，常见的扩展名主要有以下几种：

- py：python 源文件，由 python 解释器负责解释执行；
- pyw：python 源文件，常用于图形界面程序文件；
- ipynb：全称为 ipython notebook，是由 jupyter 生成的文件；
- pyc：pyc 是一种二进制文件，是由 py 文件经过编译后生成的文件。

pyc 文件是一种字节码文件，py 文件变成 pyc 文件后，加载的速度有所提高。直接执行 python 程序文件时并不生成 pyc 文件，可以使用 py_compile 模块的 compile( ) 函数来对 py 文件进行编译以提高加载和运行速度。

1.7 习题与实验

一、填空题

1. python 程序文件 test. py 编译后的文件名为_____。

2. python 源代码程序编译后生成文件是_____文件，这种文件加载速度更快而且能提高代码的安全性。

3. 在 python 语言中，有_____和_____两种注释。

4. 在 python 语言中，单行注释一般是以_____开头，python 解释器不会解释其右边的内容。

5. 在 python 语言中，多行注释采用连续的_____引号来标记，引号可以是单引号或双引号。

6. 在编写 python 代码遇到需要将一条语句分成多行编写时，一般使用_____，也可以使用圆括号将多行代码括起来。

二、选择题

1. 在下面关于 python 语言的描述中，错误的选项是(　　)。

A. python 不支持中文　　B. python 是解释型编程语言

C. python 是免费开源的编程语言　　D. python 具有良好的跨平台特性

2. 在以下选项中，不属于 python 语言特点的是(　　)。

A. 具有极强的可移植性　　B. 支持面向对象程序设计

C. 语法简洁　　D. 运行效率高

3. python 源程序编译后的文件的扩展名是(　　)。

A. py　　B. pyw

C. pyc　　D. exe

4. Anaconda 自带的 Spyder IDE 中运行选中代码的快捷键是(　　)。

A. F5　　B. F9　　C. Ctrl+Enter　　D. Ctrl+F5

5. python 语言的单行注释以(　　)开头。

A. #　　B. //　　C. \　　D. ’’’

三、编程题

1. 编写程序，使用 print( ) 函数输出学校、院系、学号等信息，以及输出想对自己说的一句话。

2. 编写程序，输入两个整数 m 和 n，例如输入 3 和 5，显示出：‘3+5 的和为：8’。

第 2 章　数据类型与基本运算

2.1　标识符和保留字

2.1.1　标识符

简单地理解，标识符就是一个名字，就好像我们每个人都有属于自己的名字，它的主要作用就是作为变量、函数、类、模块以及其他对象的名称。python 中标识符的命名不是随意的，而是要遵守一定的命令规则：

（1）标识符是由字符（A～Z 和 a～z）、下划线和数字组成，但第一个字符不能是数字。

（2）标识符不能和 python 中的保留字相同。

（3）python 中的标识符中，不能包含空格、@、%以及$等特殊字符。

（4）汉字可以做标识符。

下面所列举的标识符是合法的：UserID、name、mode12、user_age。以下命名的标识符不合法：4word（不能以数字开头）、try（try 是保留字）、$money（不能包含特殊字符，特别是美元符号在很多语言中是合法的）。

特别注意，以下划线开头的标识符是有特殊意义的，如图 2.1 所示。因此，除非特定场景需要，应避免使用以下划线开头的标识符。

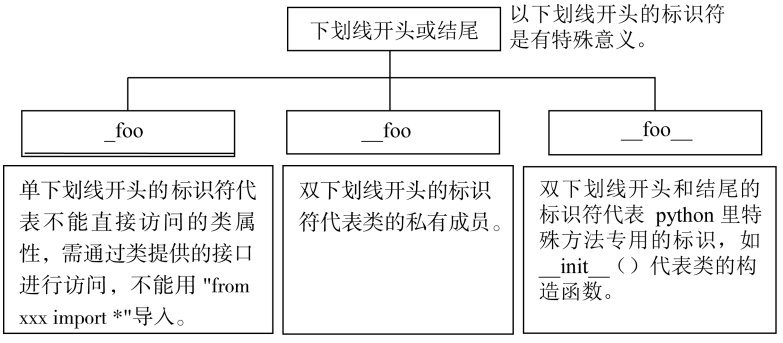

图 2.1　以下划线开头和结尾的特殊标识符有不同含义

标识符的命名，除了要遵守以上这几条规则外，不同场景中的标识符，其名称也有一定的规范可循，例如：当标识符用作类名时，应采用单词首字母大写的形式；函数名、类中的属性名和方法名，应全部使用小写字母；**常量命名应全部使用大写字母。**当命名有多个单词时，单词之间可以用下划线分割，也可以从第二个单词开始首字母大写。

标识符命名的时候，除了表示循环次数的 i、j 等，其他命名应该**顾名思义，使用有意义的名字。**

2.1.2　保留字

保留字指的是语句中的属于 python 的语法部分的具有特殊含义的关键字。表 2-1 显示了 python 中的保留字。这些保留字不能用作常量、变量、函数或任何其他标识符的名称。倘**若使用保留字作为自定义标识符使用，会报错。**

表 2.1　python 的保留字。左边是代码，右边是显示结果

#显示 python 的保留字 import keyword print(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']

python 还包含一系列内置函数，一般也不建议使用它们作为变量名，如果开发者尝试使用内置函数作为变量名，python 解释器不会报错，内置函数被这个变量覆盖了，该内置函数就不能使用了。

2.2　变量的定义和赋值

什么是变量？简单地说，变量是某物的名称，变量一词表示它是变化的。python 不是强制类型的语言，因此变量在使用的时候只要直接赋值就可以使用了，其他语言是必需先声明再赋值、使用。表 2.2 列出了 C 语言和 python 语言在变量使用上的不同。

表 2.2　C 语言和 python 语言在变量使用上的不同

C 语言	python

int a=10;　　//声明变量的时候赋值
int b;　　//先声明
b=20;　　//再赋值
print(a+b);
//变量必修先声明，再赋值后使用。
	a=10　　#直接赋值，a 为整数类型
	b="book" #赋值，b 为字符串类型
	print(a, b);
	#变量直接赋值就相当于定义了。
	#后面直接使用即可。

2.2.1　变量赋值

等号(=)用来给变量赋值。等号(=)运算符左边是一个变量名，等号(=)运算符右边

是存储在变量中的值，如 a=10。也可以通过形如“a=b=c=100”这种形式进行多个变量的相等赋值。

特别注意：

- 如果变量没有赋值就直接使用，会触发“NameError”异常。这是一个常见异常。
- 在 python 中，变量本身没有类型，我们所说的类型是变量所指的内存中对象的类型。

2.2.2 python 变量的内存机制

变量更像是一份对内存的引用，通过这个变量可以访问到内存中的数据。如 a=10，此时，系统为数字 10 这个对象分配内存空间(具有地址)，并让变量 a“指向”此空间。使用**内置函数 id(变量名)可以获得变量引用的对象的内存地址**。先看代码例 2.1。

例 2.1 变量的赋值和对象的地址演示

```
1  a=b=10          #a/b 指向同一个对象,地址应该一样
2  c=10            #c 的地址和 a/b 的一样吗？答案:一样
3  d=e=257         #d/e 指向同一个对象,地址应该一样
4  f=257           #f 的地址和 d/e 的一样吗？答案:不一样
5  m="python"
6  n="python"   #m 的地址和 n 的一样吗？答案:一样
7  print(id(a), id(b), id(c))
8  print(id(d), id(e), id(f))
9  print(id(m), id(n))
```

输出结果(不同机器或不同运行结果不同)：

```
140707420518464 140707420518464 140707420518464
2075124619472 2075124619472 2075124619760
2075093509936 2075093509936
```

python 变量即对象的引用，通俗来说就是指向值的名称。关于 python 的变量内存机制，主要注意下面两点：

(1) python 在执行的时候，为了节约空间，帮我们创建好了小整数对象池，范围为[-5~256]，程序每次启动都分配固定的地址，不管你用不用，都会存在。所以 a/b/c 的地址相等。不在这个范围的整数，值虽然相同，但是分别分配空间，如 d 和 f，因而地址不同。再如，a=1000，b=1000，这时会给 a 一个地址，也会给 b 一个地址，但这两个地址不相等。

(2) 对于不可变数据类型，内存中只能有一个相同值的对象。比如字符串是不可变对象，因此内存中的“python”字符串只有一个，所以 m 和 n 指向同一个“python”对象。

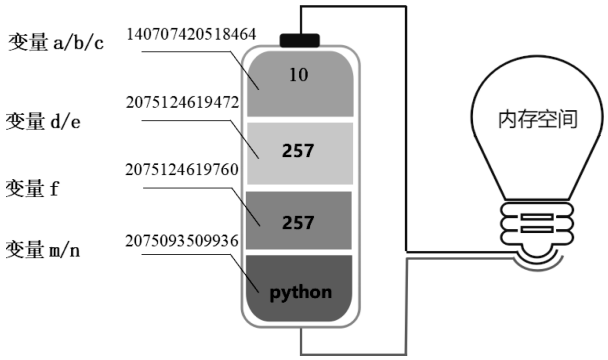

图 2.2　python 变量的内存空间示意图

例 2.2　变量的赋值和对象的“相等”

```
# is 和 is not,身份运算符 id
a='python'
b='python'
if a is b:
    print('a 和 b 指向同一个存储对象,有相同的标识')
else:
    print ('a 和 b 没指向同一个存储对象,没有相同的标识')
if id(a)==id(b):
    print ('a 和 b 指向同一个存储对象,有相同的标识')
else:
    print ('a 和 b 没指向同一个存储对象,没有相同的标识')
b='b'    #修改变量 b 的值
if a is b:
    print ('a 和 b 指向同一个存储对象,有相同的标识')
else:
    print ('a 和 b 没指向同一个存储对象,没有相同的标识')
if a is not b:
    print ('a 和 b 没指向同一个存储对象,没有相同的标识')
else:
    print ('a 和 b 指向同一个存储对象,有相同的标识' )
```

输出结果：

```
a 和 b 指向同一个存储对象，有相同的标识
a 和 b 指向同一个存储对象，有相同的标识
a 和 b 没指向同一个存储对象，没有相同的标识
a 和 b 没指向同一个存储对象，没有相同的标识
```

2.3 数据类型

python 3 中有六个标准的数据类型：数字、字符串、列表、元组、集合和字典，图 2.3 给出了六种数据类型的思维导图。

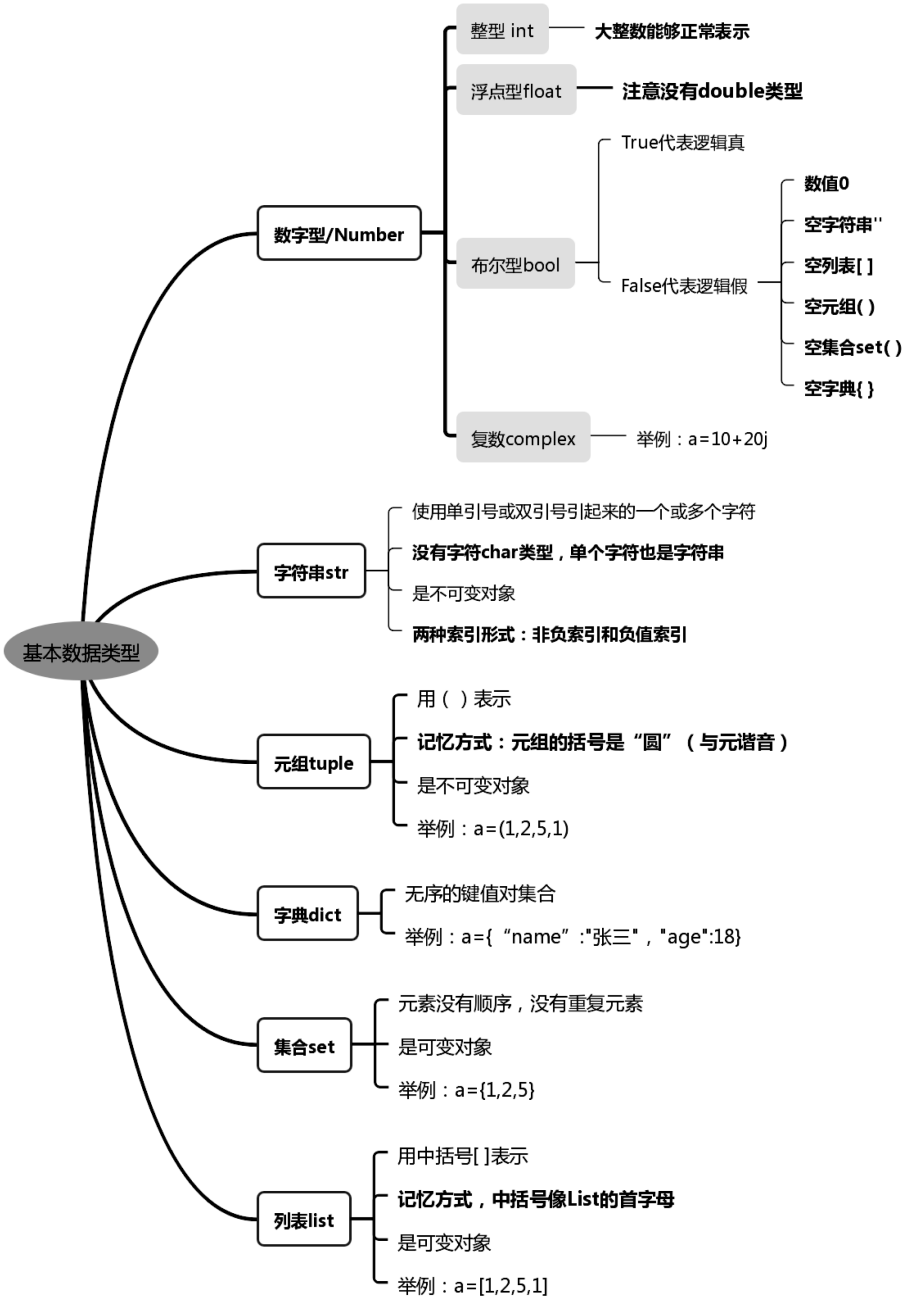

图 2.3 python 基本数据类型

2.3.1　整型、浮点型和复数型

python 可以处理**任意大小**的整数，当然包括负整数。整数的存储是精确的。下面都是正确的整数表示：100、-200、1234567890123456789、0o37(数字 0 和字母 o 开头代表八进制数)、0xA8(十六进制数)。

浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可变的。下面都是正确的浮点数表示：12.35、12E2(12 的 2 次方)、1e-6(10 的-6 次方)。浮点数运算则可能会有四舍五入的误差，例如 print(0.10+0.20)，其结果是 0.30000000000000004。因此，浮点数相等的比较判断，一般不要写为："0.10+0.20==0.30"形式的相等比较，应该写为：abs(0.10+0.20-0.30)<1e-6，即两个浮点数相减的绝对值小于一个很小的数。

python 支持复数，复数由实数部分和虚数部分构成，可以用 a+bj，或者 complex(a,b)表示，复数的实部 a 和虚部 b 都是浮点型，如 3.0+5j、complex(3，5)都是合法的复数。

2.3.2　布尔型

python 中布尔值使用常量 True(逻辑真)和 False(逻辑假)来表示，注意大小写。

含有<、>、==等比较运算符的表达式返回的类型就是 bool 类型，布尔类型通常在 if 和 while 语句中应用。注意的是，bool 是 int 的子类(继承 int)，故 True==1 和 False==0 是会返回 True 的。由于 bool 是 int，故可进行数字计算，例如：print(True+True)。

python 中，**满足如下条件即为 False：数值 0、空字符串''、空列表[]、空元组()、空集合 set()或者空字典{}**。其他情况，进行逻辑判断结果为 True。这个需要注意，与其他的语言有比较大的不同。

2.3.3　字符串

字符串是以成对的单引号' '或成对的双引号""括起来的任意文本，比如'abc'，"Gyz"等等。

请注意，' '或""必须成对使用，是字符串的界定符，不是字符串的一部分。如果'本身也是一个字符，那就可以用""括起来，反之亦然。如：print("I'm a student."，'This book is "mine".')，结果为：I'm a student. This book is "mine."。

python 没有字符 char 类型，单个字符也是字符串。python 字符串是不可变对象，有两种索引形式：非负索引和负值索引。字符串内部可以用转义字符"\"来标识。

2.4　运算符及其优先级

python3 支持以下类型运算符：算术运算符、比较(关系)运算符、赋值运算符、逻辑运算符、位运算符、成员运算符和身份运算符。如图 2.4 所示。表 2.3 给出了详细的示例。

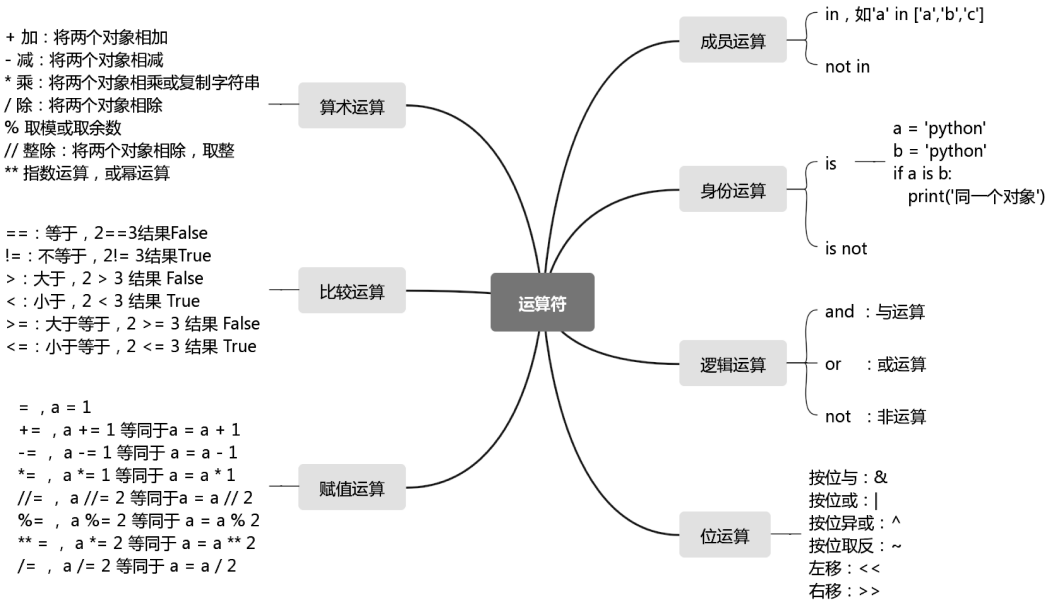

图 2.4　python3 运算符一览

表 2.3(a)　算术运算符及示例

运算符	说明	示例(a=2，b=3)
+	加：将两个对象相加	a+b 结果 5
-	减：将两个对象相减	a-b 结果-1
*	乘：将两个对象相乘或复制字符串	a * b 结果 6
/	除：将两个对象相除	**1/2 结果 0.5**
%	取模或取余数	a%b 结果 2
//	整除：将两个对象相除，取整	a//b 结果 0
**	指数运算，或幂运算	a ** b 结果 8

表 2.3(b)　比较运算符及示例

运算符	说明	示例(a=2，b=3)
==	等于：比较两个对象是否相等	a==b 结果 False
! =	不等于：比较两个对象不相等	a! =b 结果 True
>	大于：比较对象 a 是否大于对象 b	a>b 结果 False
<	小于：比较对象 a 是否小于对象 b	a<b 结果 True
>=	大于等于：比较对象 a 是否大于或等于对象 b	a>=b 结果 False
<=	小于等于：比较对象 a 是否小于或等于对象 b	a<=b 结果 True

表 2.3(c)　赋值运算符及示例

运算符	说明	示例
=	赋值运算	a=1
+=	加法赋值	a+=1 等同于 a=a+1
-=	减法赋值	a-=1 等同于 a=a-1
*=	乘法赋值	a*=1 等同于 a=a*1
//=	整除赋值	a//=2 等同于 a=a//2
%=	取余赋值	a%=2 等同于 a=a%2
**=	幂运算赋值	a*=2 等同于 a=a**2
/=	除法赋值	a/=2 等同于 a=a/2

表 2.3(d)　逻辑运算符及示例

运算符	说明	示例(a=2，b=3)
and	与运算：如果 a 为 False；a and b 返回 a 的值，否则它返回 b 的值	a and b 结果 3
or	或运算：如果 a 为 True，a or b 返回 a 的值，否则返回 b 的值	a or b 结果 2
not	非运算：单目运算符，如果 a 为 True，not a 返回 False；反之则返回 True	not a 结果 False

注意：**逻辑运算的 and 和 or 是短路运算符。**or 只有在第一个操作数为 False 时才会计算第二个操作数的值。and 只有在第一个操作数为 True 时才会计算第二个操作数的值。一些实例如表 2.3(e)所示。

表 2.3(e)　逻辑运算符示例(结合图 2.2 关于 False 的介绍)

表达式	值	表达式	值	表达式	值
2 and 3	3	2 or 3	2	not 3	False
0 and 3	0	0 or 3	3	not 0	True
''and 3	''	''or 3	3	not ( )	True
[ ] and 3	[ ]	''or {}	{}	not [1，2]	False

表 2.4 列出了优先级从高到低的所有运算符。注意最后一行的逻辑运算符，虽然放在同一行，但是 not 最高，or 最低。

表 2.4　运算符优先级(从高到低)

运算符	描述
**	指数(最高优先级)
~，　+，　-	按位翻转，一元加号和减号

续表

运算符	描述
*，　/，　%，　//	乘，除，取模和取整除
+，　-	加法减法
>>，　<<	右移，左移运算符
&	位 'and'
^	位异或运算符
<=，　<，　>，　>=	比较运算符
<>，　==，　！=	(不)等于运算符
=，%=，/=，//=，-=，+=，*=，**=	(复合)赋值运算符
is，　is not	身份运算符
in，　not in	成员运算符
not，　and，　or	逻辑运算符(not 最高，or 最低)

2.5　表达式

表达式是由变量、常量和运算符组成的式子。对应于运算符，表达式有算术运算表达式、赋值运算表达式、关系运算表达式、逻辑运算表达式等。

赋值时，可以连续赋值，如**a=b=c=100**。也可以拆包式赋值，如**a，b，c=100，200，300** 为 a、b、c 分别赋值 100、200、300。

python 允许连续进行比较，形如**a<b<c，**就像数学中一样。

例 2.3　表达式演示

```
1  a=b=c=90   #连续赋值
2  a, b, c=90, 200, 300   #拆包式赋值,表达式分别为 a、b、c 赋值 100、200、300。
3  if a<b<c:   #条件满足,打印下面语句
4      print("python 可以使用连续比较表达式,就像数学中的一样。")
```

2.6　了解内置函数

2.6.1　查看内置函数

查看内置函数只需要执行：dir(_builtins_)，注意 builtins 单词的前后都是双下划线。表 2.5 给出 python 的内置函数名称。

表 2.5　内置函数一览表

abs( )	delattr( )	hash( )	memoryview( )	set( )
all( )	dict( )	help( )	min( )	setattr( )
any( )	dir( )	hex( )	next( )	slice( )
ascii( )	divmod( )	id( )	object( )	**sorted( )**
bin( )	**enumerate( )**	input( )	oct( )	staticmethod( )
bool( )	**eval( )**	int( )	open( )	str( )
breakpoint( )	exec( )	isinstance( )	ord( )	sum( )
bytearray( )	filter( )	issubclass( )	pow( )	super( )
bytes( )	float( )	iter( )	print( )	tuple( )
callable( )	format( )	len( )	property( )	type( )
chr( )	frozenset( )	list( )	**range( )**	vars( )
classmethod( )	getattr( )	locals( )	repr( )	**zip( )**
compile( )	globals( )	**map( )**	reversed( )	_import_( )
complex( )	hasattr( )	max( )	round( )	

2.6.2　内置函数简介

（1）已经用到的输入输出函数 input、print 函数。不赘述。

（2）已经用到的类型判定函数 type、地址函数 id。不赘述。

（3）类型转换函数。前面章节提到的类型 int、float、str、list、set、dict、complex、tuple、bool 等，都对应于一个转换函数。如 a='100'，b=int(a)。

（4）进制转换函数

- bin(x)：将一个整数 x 转变为一个前缀为“0b”的二进制字符串；
- oct(x)：将一个整数 x 转变为一个前缀为“0o”的八进制字符串；
- hex(x)：将整数 x 转换为以“0x”为前缀的小写十六进制字符串。

（5）帮助函数 help([object])。可以利用这个函数查看其他函数的帮助信息。如执行 help(print)，则会显示 print 函数的原型信息。

（6）数学计算类函数 abs、max、min、sum、round、pow

- abs(x)：返回一个数的绝对值。实参可以是整数或浮点数。如果实参是一个复数，返回它的模；
- round(number[，ndigits])：返回 number 舍入到小数点后 ndigits 位精度的值，此函数并非完全是四舍五入，请查阅相关资料；
- pow( )：幂函数，r=pow(2，10)#2 的 10 次方。

（7）len(s)：返回对象的长度(元素个数)。实参可以是序列(如 string、bytes、tuple、list 或 range 等)或集合(如 dictionary、set 或 frozen set 等)。

（8）eval(expression)：python 将 expression 当做一个 python 表达式进行解析和计算，

例如 a=10，eval("a*10+4*5")的结果就是 120。

（9）其他一些重要函数：**enumerate**、**isinstance**、**sorted**、**map**、**zip**、**reversed** 等函数后续会陆陆续续用到。此处不述。

2.6.3 range()内置函数

可创建一个整数列表，一般用在 for 循环中。可用下述三种方式调用 range()：

（1）range(stop)需要一个参数，相当于 range(0，stop，1)。1 代表步长。

（2）range(start，stop)需要两个参数，相当于 range(start，stop，1)。

（3）range(start，stop，step)需要三个参数。返回计数从 start 开始到 stop 结束，但不包括 stop，间隔(步长)为 step 的 range 对象。例如：range(1，7，2)是代表[1，3，5]的 range 对象，不含 7，即左闭右开的区间[start，stop)。

注意：python3 中 range()返回的是一个可迭代对象(类型是对象)，而不是列表类型，所以打印的时候不会打印列表，只有当它参与运算的时候，才会真正显示为一个列表。具有这种特性的对象可以称为“惰性计算对象”。

惰性计算(lazy evaluation)是指仅仅在真正需要执行的时候才计算的表达式的值。由于避免了不必要的计算，节省了计算资源，因此惰性计算不仅在内存层级对空间有着优化的效果，在计算时间上也有着一定的提高。

例 2.4 range 函数的使用

```
1  a=range(10)
2  b=range(10, 20)
3  c=range(-10, 10, 2)
4  print(type(a))   #结果为:<class 'range'>
5  print(a.start, a.stop, a.step)   #结果为:0 10 1
6  print(b)          #结果为:range(10, 20)
7  print(*c)         #结果为: -10 -8 -6 -4 -2 0 2 4 6 8
8  print(list(b))    #结果为: [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
9  print(tuple(c))   #结果为: (-10, -8, -6, -4, -2, 0, 2, 4, 6, 8)
10  #下面举一个求 1 到 100 和的例子(先观察 range 函数的使用,后续讲 for 循环再深入)
11  s=0
12  for i in range(1,101):
13      s+=i
14  print(s)   # 5050
```

注意，第 7 行的“*”是对序列解包，例如 a=[1，2，3]，print(*a)得到 1 2 3 三个独立的数。从例 2.4 可以看出，当把 range 函数返回的对象用于类型转换(8/9 行)、解包运算(7 行)、in 表达式等场合，range 才真正从“对象”转为“数列”。

第 12 到第 14 行给出了 range 函数的常用的一种场合，通常用于“迭代”，“for i in range(1，101)”类似于 C 语言或者 Java 语言的“for(i=1；i<101；i++)”。

2.7　使用数学库 math

上节介绍了最基本的数学运算内置函数。math 数学库补充了更多的函数。math 库(以 python3.5 为例)有 4 个常数和 44 个函数(图 2.5 所示)。

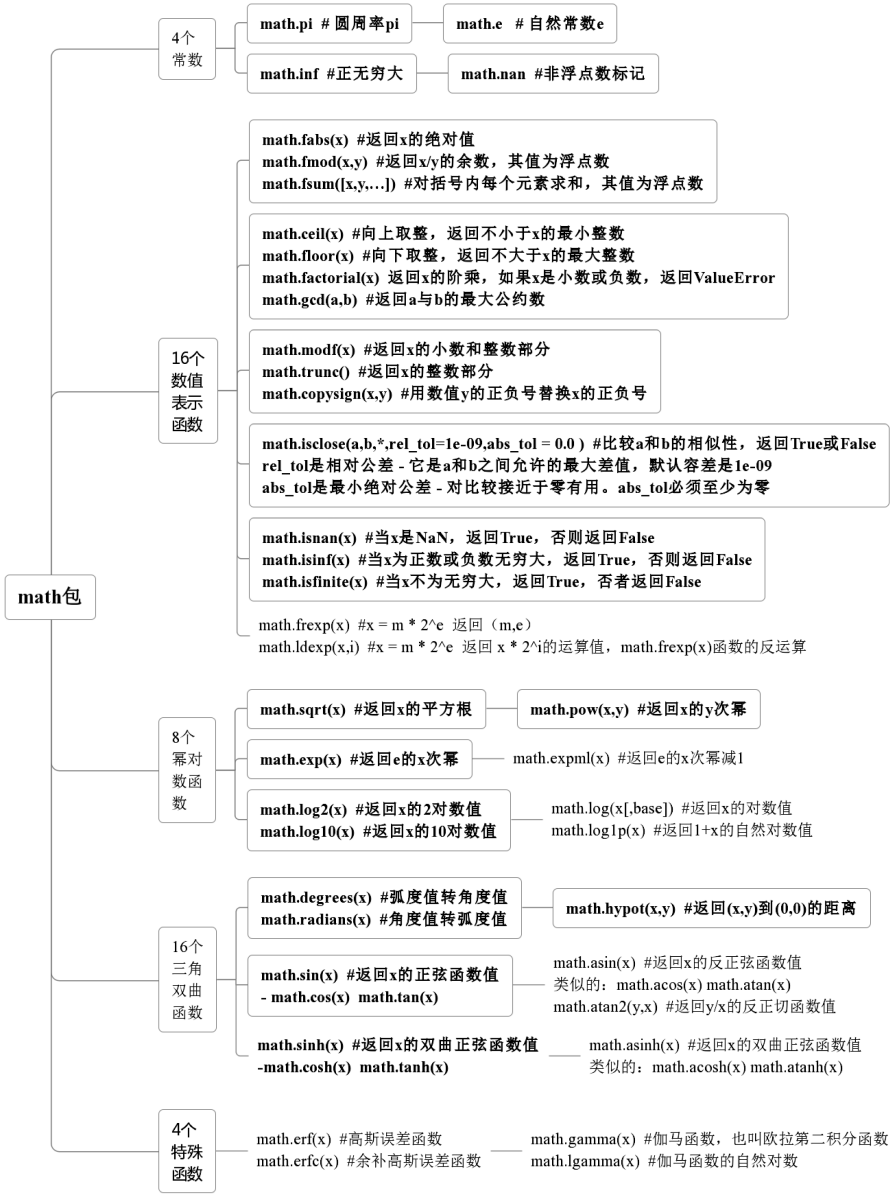

图 2.5　math 包的常数和函数(python3.5)

注意：

（1）math 库不支持复数类型；使用 math 必须首先导入。

（2）使用“import math”和“print(dir(math))”两条语句可以查看当前安装的 python 版本的 math 库支持的函数和常量。不同的版本会有变化，比如 python3.9 版本出现了 math.lcm(*integers)，返回给定的整数参数的最小公倍数；

（3）更加高级的数学功能，可以考虑选择标准库之外的 numpy 和 scipy，它们不但支持数组和矩阵运算，还有丰富的数学和物理方程可供使用；

（4）random 包可以用来生成随机数。

例 2.5　math 包的使用

```
import math   #这种导入方式,后续必须使用"math. 函数名或常量名"形式,否则报错。
print("math. pi", math. pi)                #结果:3. 141592653589793
print("math. floor(3. 2)", math. floor(3. 2))   # 向下取整,结果:3
print("math. ceil(4. 5)", math. ceil(4. 5))     # 向上取整,结果:5
print("math. pow(3, 4)", math. pow(3, 4))   # =3**4,结果:81. 0
```

```
from math import *   #这种导入方式,后续可以直接使用函数名,可以不用"math. "
r=modf(3. 14)
print("math. modf(3. 14)", r)   # 拆分小数和整数,结果:(0. 14000000000000012, 3. 0)
m, n=modf(3. 14)
print(f"math. modf(3. 14)的整数是{m},小数是{n}")
print("math. copysign(3, -5)", copysign(3, -5))   #-5 的符号赋给 3,结果:-3. 0
```

特别提醒，由于浮点数在计算机存储中会有精度损失，因此**对于浮点数的相等比较运算不应该使用“a=b”的形式。**而应该使用“math.fabs(a-b)<1e-6”的形式。但是每次这样写不方便，图 2.5 中的 math.isclose() 函数可以比较两个浮点数是不是非常接近，非常方便。

例 2.6　浮点数的相等比较

```
a, b, c=0. 1, 0. 2, 0. 3   #拆包赋值
print(a+b)          #结果为:0. 30000000000000004
print(a+b==c)               #结果为:False
print(math. fabs(a+b- c)<1e- 6)   #结果为:True
print(math. isclose(a+b,c))       #结果为:True
print(math. isclose(a+b- c,0))    #结果为:False,该函数第二个参数最好不要为 0
```

说明：isclose 函数的原型为 isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)，a 和 b 是两个需要比较的浮点数，本质上如果 abs(a-b)<=max(rel_tol * max(abs(a), abs(b)), abs_tol)，则认为 a 与 b 相等，返回 True。

2.8　习题与实验

一、填空题

1. python 解释器会根据赋值或运算来自动推断变量类型，python 还是一种_____类型语言，变量的类型也是可以随时变化的。

2. 在 python 中，标识符是_____组成，但第一个字符不能是数字。

3. python3 提供 4 种数字类型用于存储数字类型数据，分别是：_____、_____、_____、_____。

4. 执行程序 s=1；s='Python'；print(type(s))后，输出结果是_____。

5. 执行程序 s=1；s='Python'；print(isinstance(s，str))后，输出结果是_____。

6. 执行程序 s=None；print(type(s))后，输出结果是_____。

7. 在 python 中，用于表示空类型的是_____。

8. 在 python 中，_____函数可查看变量的类型。

二、选择题

1. 下面关于 python 标识符的描述中，描述错误的是(　　)。

A. 标识符命名尽量采用有意义的名称

B. 标识符区分大小写

C. 标识符不能包含空格、@ 、%以及$等特殊字符

D. 标识符尽量避免使用 python 的保留字

2. 在下面关于 python 标识符的描述中，不合法的标识符是(　　)。

A. pi　　B. false　　C. pass　　D. _test

3. 在下面变量命名中，符合变量名命名规范的是(　　)。

A. @ gdei　　B. 123abc　　C. 姓名　　D. None

4. 在下面关于变量的描述中，描述错误的是(　　)。

A. 变量在赋值前不需要声明　　B. 变量在赋值前必须先指定类型

C. 变量必须赋值才会被创建　　D. 变量的类型由所赋的值的类型决定

5. 执行下面有关变量引用的程序，关于程序中 print( )语句输出结果描述错误的选项是(　　)。

```
number1 = 1; print("第一个输出结果", id(number1))
number2 = number1; print("第二个输出结果", id(number2))
number3 = 1; print("第三个输出结果", id(number3))
number1 = 3; print("第四个输出结果", id(number1))
del number1; number4 = 1; print("第五个输出结果", id(number4))
```

A. 前面两个输出结果都是一样的　　B. 前面三个输出结果是一样的

C. 有四个输出结果都是一样的　　D. 全部输出结果都是一样的

6. 下面关于数字类型的描述中，描述错误的是(　　)。
   A. 数字类型是不可变类型
   B. 整数类型没有取值限制
   C. 如果改变数字类型变量的值，变量会被重新分配内存空间
   D. 如果两个数字类型变量的值相同，那么它们的 id 值也相同
7. 执行下面有关 int( )函数使用的程序，输出结果不是“1”的选项是(　)。
   A. print(int(True))　　B. print(int(1.9))
   C. print(int("1"))　　D. print(int("1.0"))
8. 执行下面有关 float( )函数使用的程序，输出结果不是“1.0”的选项是(　　)。
   A. print(float(True))　　B. print(float("1"))
   C. print(float("1.0"))　　D. print(float("1.0_Test"))
9. 执行下面有关复数的程序，输出结果不一样的选项是(　　)。
   A. print(3+0j)　　B. print(complex(3))
   C. print(3+complex( ))　　D. print(complex(3).real+complex(3).imag)
10. 执行下面程序，输出结果是 True 的选项是(　　)。
    A. print(1.1-1==0.1)　　B. print(True * 1)
    C. print(1<2<3)　　D. print('A' in "abc")
11. 执行 import math 后，再执行下面程序，输出结果与其他选项不一样的是(　　)。
    A. print(math.ceil(3.1415926)-1)　　B. print(round(3.1415926))
    C. print(round(3.1415926, 0))　　D. print(math.floor(3.1415926))
12. 执行下面有关 bool( )函数的程序，输出结果是 True 的选项是(　　)。
    A. print(bool(0b0)or bool(""))　　B. print(bool(-0x1) and bool(0.0))
    C. print(bool(0j)or bool(None))　　D. print(not bool( ))
13. 执行下面有关 python 运算符的程序，对输出结果描述错误的选项是(　　)。
    print(3//2); print(3%2); print(--1); print(++1 * * 0)
    A. 输出结果都是“1”　　B. 有一个输出结果不是“1”
    C. 前面两个输出结果都是“1”　　D. 后两个输出结果都是“1”
14. 执行下面程序，输出结果不是“1”的选项是(　　)。
    A. x, y=1, 2; x, y=y, x; print(y)
    B. print(1+(1>2))
    C. print(2-1.0)
    D. x=0; x+=1; print(++x)
15. 假设 pi=3.1415926，执行下面程序，输出结果不是“3.14”的选项是(　　)。
    A. print("%.2f" % pi)
    B. print(round(pi, 2))
    C. print("{:.2f}".format(pi))
    D. print(str(pi)[0:3])

三、编程题

1. 请编写程序，实现十进制数转换成二进制数、八进制数或者十六进制数，而且要求运行程序时显示如下提示信息：

(1)二进制　　(2)八进制　　(3)十六进制

请选择1、2或者3：

2. 请编写程序，实现四则运算的计算，例如：当输入四则运算表达式“3+5＊6=”时，能计算并输出表达式的计算结果。

3. 输入一个正整数n，求1+2+3+…+n的累加和。要求显示“1+2+3+…+n的累加和。

4. 输入n(n≥5)，求n!。要求显示：“n! 为：(此处放计算得到的值)”

5. 编写程序，实现角度转换弧度的计算，如：当输入“30”度时，计算出对应的弧度值，并输出“30°转换成弧度的值是：0.52”。

6. 编写程序，实现从键盘分别输入扇形半径与扇形圆心角的度数，计算并输出扇形面积。(注：扇形圆心角的度数angle的范围：0<angle<=360°)