
视频讲解

# 第5章 Python编程基础

在大数据分析与人工智能领域，Python 已成为重要的入门级编程语言。Python 拥有简洁的语法特性、丰富的开源生态以及强大的社区支持，已成为全球增长最快的主流编程语言。掌握 Python 这一支撑现代智能技术发展的核心工具，对于从事大数据分析与人工智能相关工作具有重要意义。

## 5.1 Python 快速入门

Python（得名于英国喜剧团体 Monty Python，国际音标标注为/ˈpaɪθən/，常见中文音译为“派森”）是一种面向对象的解释型计算机程序设计语言，由荷兰人吉多·范罗苏姆（Guido van Rossum）于 1989 年发明，第一个公开发行版于 1991 年发布。经过三十余年的发展，Python 凭借简洁的语法、丰富的生态和强大的社区支持，已成为全球最受欢迎的编程语言之一，在数据分析、人工智能、Web 开发等领域占据主导地位。

### 5.1.1 Python 语言的特点和发展

Python 语言具有以下几个显著特点。

（1）语法简洁清晰。Python 语言采用极简主义设计哲学，通过缩进机制定义代码块，使得程序结构直观易读，大幅降低了初学者的学习门槛。

（2）开发效率卓越。相较于 C、Java 等语言，Python 语言能够用更少的代码量实现相同的功能，显著提升了程序开发、测试和维护的效率。

（3）开源与跨平台特性。Python 也是一种免费开源的编程语言，任何人可以自由地发布这个软件的副本、阅读它的源代码、对它做改进等。由于它的开源本质，Python 已经被移植在许多平台上，所有 Python 程序无须修改就可以在各类主流操作系统上运行。

（4）生态体系丰富。围绕 Python 形成了涵盖数据科学、机器学习、Web 开发等领域的完善工具链，特别是 NumPy、Pandas、TensorFlow 等库的加持，使其成为数据科学与 AI 开发的首选语言。

根据 KDnuggets 网站调查显示，Python 自 2017 年超越 R 语言成为数据科学与机器学习领域使用最广泛的语言后，其领先地位保持至今。根据 TIOBE 编程语言排行榜数据显

示，Python 自 2021 年首次登顶后已连续多年保持榜首地位，这充分体现了其在全球开发者社区中的广泛影响力和受欢迎程度。TIOBE 指数作为衡量编程语言流行度的重要指标，其排名结果客观反映了 Python 在工业界和学术界的普及程度。

在我国，Python 语言的重要性也得到教育部门的充分认可。自 2018 年 3 月起，“Python 语言程序设计”被纳入全国计算机等级考试二级科目。同时，北京、山东、浙江等多个省市已将 Python 编程纳入中小学信息技术课程体系及高考考核范围，这充分体现了 Python 语言在数字时代人才培养中的重要地位。

### 5.1.2 搭建编程环境

工欲善其事，必先利其器。选择合适的开发环境是学习 Python 编程的重要一步。Python 的开发环境种类丰富，既有轻量便捷的原生 IDLE（Integrated Development and Learning Environment，集成开发和学习环境，一般简称集成开发环境），也有功能强大的专业集成开发环境（如 PyCharm、Anaconda），以及开箱即用的云计算平台（如百度 AI Studio）。每种环境各有特色，适合不同的学习阶段和应用场景。本书选用 Python 原生 IDLE 作为学习工具。

Python 原生 IDLE 是 Python 官方随安装包自带的开发工具，也是初学者入门的理想选择。它具有以下几个显著特点。

（1）轻量便捷。IDLE 随 Python 安装包自动配备，无须额外下载或配置，安装完成后即可直接使用，启动速度快，系统资源占用极少。

（2）交互式学习。IDLE 提供交互式命令行窗口（Shell），用户输入一行代码即可立即看到运行结果，非常适合初学者逐步测试代码片段、验证语法规则，有助于快速建立编程直觉。

（3）程序编辑。IDLE 内置代码编辑窗口，可以编写、保存和运行完整的 Python 程序，支持语法高亮和自动缩进，方便代码的阅读与修改。

（4）跨平台一致。IDLE 在 Windows、macOS 和 Linux 系统上的界面与操作方式基本一致，学习经验可以无缝迁移。

大多数 Linux 和 macOS 系统已预装 Python 及其原生 IDLE，用户可直接打开使用；而 Windows 系统默认不包含 Python，需要用户自行安装。

在 Windows 系统中，安装 Python 常用的方式主要有两种：一种是从 Python 官方网站下载安装包进行安装；另一种是通过 Python Install Manager 进行安装。前者操作直观、适合初学者，后者便于下载、安装和管理不同版本的 Python。本书采用下载安装包的方式进行说明，安装完成后即可同时获得 Python 解释器和原生开发环境 IDLE。

具体安装步骤如下。

（1）访问官网下载安装包。

打开 Python 官方网站（网址详见前言二维码），如图 5-1 所示。将鼠标指针指向页面中的 Downloads 菜单，网站会自动显示适用于 Windows 系统的最新版本安装包，单击 Python 3.14.3 按钮（可能会随 Python 版本更新变化）即可开始下载。

（2）运行安装程序。

安装包下载完成后，双击运行安装程序，弹出 Python 的安装界面，如图 5-2 所示。本书

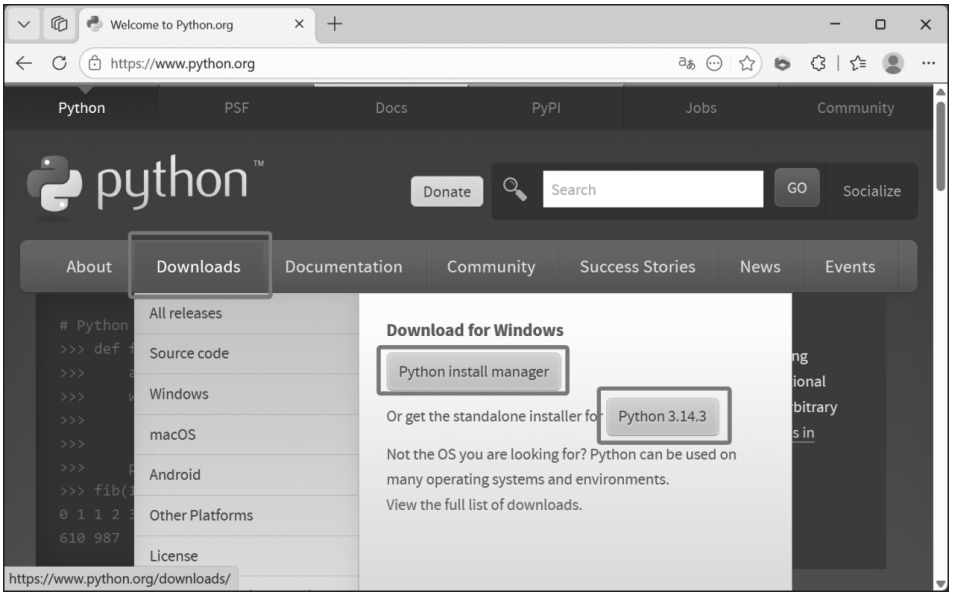


图 5-1　Python 官方网站

使用的是 Python 3.14.3 安装包。在安装界面中，务必选中 Add python.exe to PATH 复选框，这样可自动将 Python 路径配置到系统的环境变量中。同时建议选中 Use admin privileges when installing py.exe，这样可以以管理员权限进行相关安装设置。

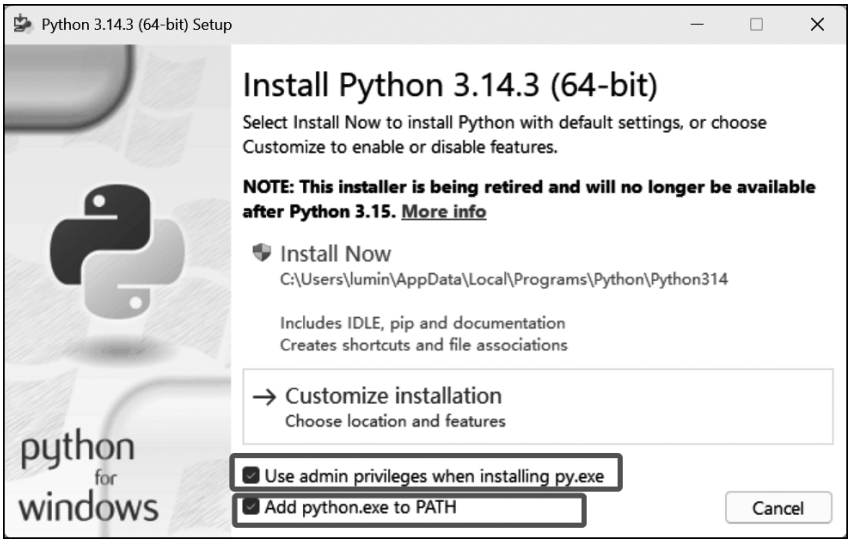


图 5-2　将 Python 路径配置到系统环境变量中

其次，可以通过选择图 5-2 中的 Customize installation 选项，在弹出的设置界面中对默认的 Python 安装路径进行更改，在后续界面中将默认安装路径修改为 C:\Python\Python314，方便后期的使用。

配置 Python 安装路径，如图 5-3 所示。可以在该界面下的文本框中直接输入安装路径，或者单击 Browse 按钮选择相应的安装路径。

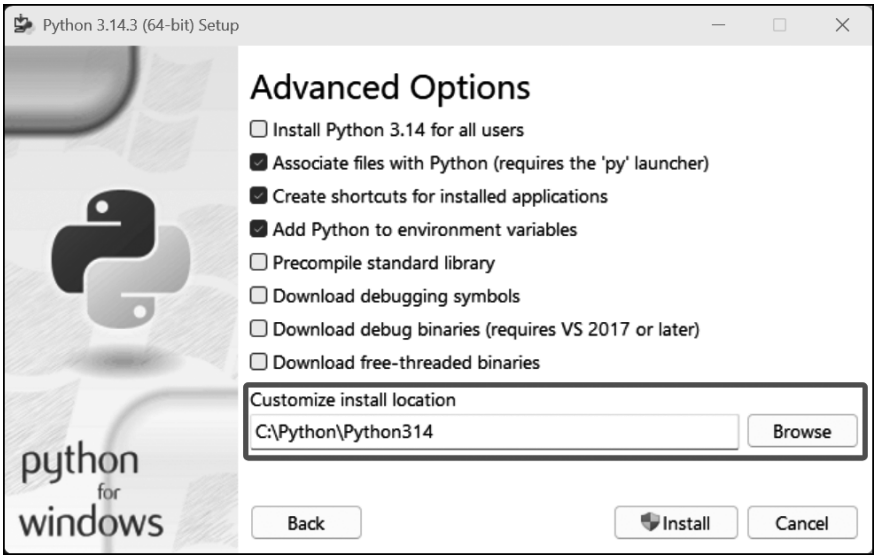


图 5-3　配置 Python 安装路径

（3）验证安装结果。

安装完成后，可以在 Windows 命令窗口中检查 Python 安装是否成功。在“开始”菜单的“搜索”栏中输入 cmd.exe，打开 Windows 命令窗口，输入 Python 后，按 Enter 键，启动 Python 交互式命令执行终端，如图 5-4 所示，该结果表示 Python 安装成功。

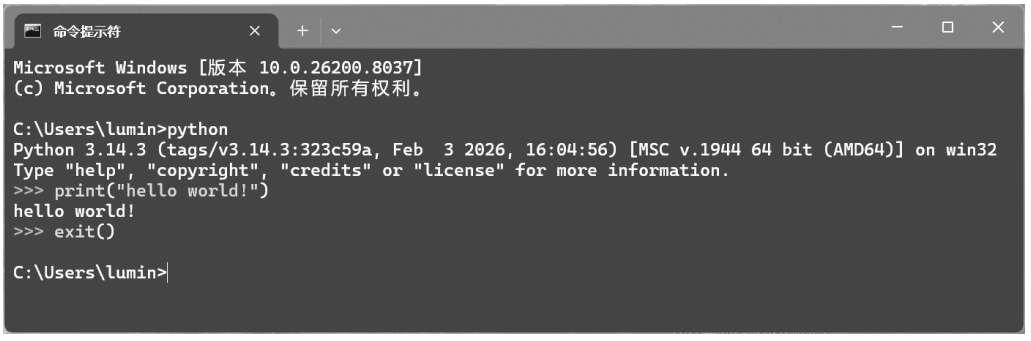


图 5-4　启动 Python 交互式命令执行终端

命令执行终端显示了系统安装的 Python 版本，最后的符号>>>是一个提示符，表示可以输入 Python 命令。在交互式命令窗口中输入代码 print("hello world")后按 Enter 键，系统将执行 Python 语句并输出结果。可以通过 exit()函数退出 Python 环境。

### 5.1.3　运行 Python 程序

Python 安装完成后，可以通过 IDLE 进入 Python 原生开发环境。使用原生 IDLE 运行 Python 程序通常有两种方式：一种是在交互式命令窗口中逐行输入并立即执行代码；另一种是将代码编写为.py 文件后统一运行。

（1）在交互式命令窗口中运行 Python 程序。

在 Windows“开始”菜单的“搜索”栏中输入 IDLE，找到并打开 IDLE 程序，如图 5-5 所

示。启动后，系统会进入 IDLE 的交互式命令窗口，用户可以在其中直接输入并执行简单的 Python 语句。


图 5-5　找到并打开 IDLE 程序

学习一门编程语言时，第一个程序通常是在屏幕上输出一条消息“Hello world!”。在 Python 中，可以直接在交互式命令窗口中使用 print()函数输出这条信息，Python 交互式命令执行窗口如图 5-6 所示。这种方式适合输入和测试简单语句，能够立即看到运行结果，便于初学者理解程序的执行过程。

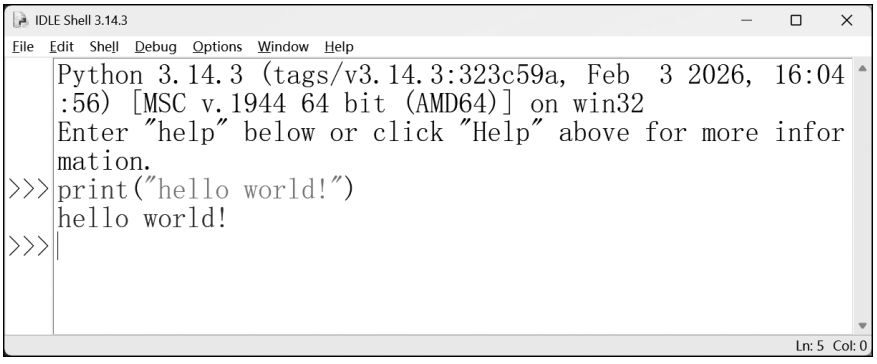


图 5-6　Python 交互式命令执行窗口

(2) 编写并运行.py 程序文件。

当需要编写和执行多行代码时，可以在 IDLE 中创建、编辑并运行 Python 源程序文件。方法如下。

第 1 步，在 IDLE 窗口中选择 File→New File 命令，新建一个空白的编辑窗口。

第 2 步，选择 File→Save 命令，将文件保存为 score_v1.py 文件，选择文件所在的位置为

C:\Python 文件夹。

第 3 步，在 score_v1.py 文件中输入相应代码并保存，如图 5-7 所示。


图 5-7 输入相应代码并保存

第 4 步，选择 Run→Run Module 命令，或者按 F5 键，即可通过 Python IDLE 运行 py 文件，如图 5-8 所示。

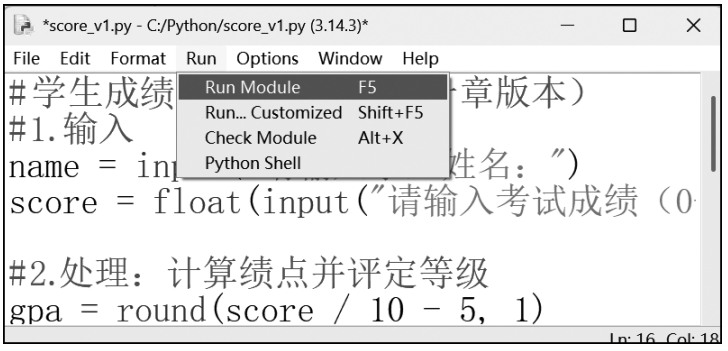


图 5-8 通过 Python IDLE 运行 py 文件

## 5.1.4 Python 程序的语法结构

Python 程序语法的格式框架主要包括两部分：代码缩进和注释。

在编写多行 Python 代码来求解问题时，为了方便程序能够被轻松地读懂，一般要为程序添加注释。注释的形式有两种，一种是用井号（#）来注释一行，# 后面的内容不管写什么都不会执行。示例代码如下：

```
#这是一行 Python 注释
print("Hello Python world!")
```

另一种是用两个三引号(''')用来注释多行,两个三引号(''')之间的部分为注释内容,将不会被执行。例如下面代码所示:

```
'''这是多行注释,用三个单引号
这是多行注释,用三个单引号'''
print("Hello Python world!")
```

注释内容要尽量以简洁清晰的语言把代码的功能讲清楚,方便今后的代码阅读和修改等。

Python 程序运行时,根据缩进来解读代码,不考虑空行,所以空行一般用来增强程序的可读性,对程序的运行没有影响。缩进一般是 4 个空格。

【注意】 并不是所有的代码都可以通过缩进来包含其他的代码,否则会出现缩进错误(IndentationError)。

例如下面代码所示:

```
print("Life is short, I use Python!")
    print("Me too!")
```

运行代码后错误信息如下:

```
  File "C:/Python/untitled1.py", line 8
    print("Me too!")
    ^
IndentationError: unexpected indent
```

此处 IndentationError: unexpected indent 错误的意思是,Python 解释器在应该没有缩进的地方发现了一个多余的缩进。

Python 的设计哲学集中体现为“Python 之禅”(The Zen of Python),由资深开发者蒂姆·彼得斯(Tim Peters)总结。在 Python 交互式命令终端输入 import this 命令,即可查看这组指导 Python 语言设计的格言。代码及运行结果如下:

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
```

```
Although never is often better than *right* now.
If the implementation is hard to explain,it's a bad idea.
If the implementation is easy to explain,it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

“Python 之禅”的核心要义可概括为：优美优于丑陋，明了优于晦涩，简洁优于复杂；代码的可读性至关重要，应当用最直接的方式完成任务，而不应过度追求技巧。正是在这种设计哲学的指引下，Python 逐渐发展为一门简明友好、易于上手且功能强大的语言。读者在后续的编程实践中，可以逐步领悟其中蕴含的设计智慧。

## 5.2 变量与数据类型

Python 程序的核心任务是处理数据。变量（Variable）是程序中用于标识和引用数据的基本单元。Python 是一种面向对象的语言，程序中的任何数据都是对象（Object），每个对象都有其对应的数据类型（Data Type）。理解数据类型与变量的工作方式，是掌握 Python 程序设计的重要基础。

### 5.2.1 变量的使用

在 Python 中，创建变量只需通过赋值语句即可：

```
变量名 = 变量值
```

在编程语言中，“=”与数学意义上的“相等”的含义不同，表示赋值运算符，即将右侧的值赋给左侧的变量。

当执行如下赋值语句时，Python 的实际执行过程为：首先在内存中创建一个值为 85.0 的浮点数对象；然后将变量名 score 像标签一样“贴”在这个对象上；通过这个标签，程序就能在内存中找到并操作对应的数据。

```
score = 85.0
```

Python 中的变量包括以下两个重要特性。

（1）变量本身没有类型，类型属于数据对象。因此，Python 无须像 C 语言那样在使用变量前提前声明其类型，解释器会在程序运行时根据变量所引用的对象自动判断，这种特性称为动态类型（Dynamic Typing）。

（2）在程序中可以随时修改变量的值，而变量只能保存最后修改的值。当对变量重新赋值时，变量名不再指向原来的对象，而是转而引用新的对象。因此，通过变量名访问到的始终是最近一次赋值所对应的数据。示例代码如下：

```
score = 85.0          #score 贴在浮点数对象上
score = "优秀"        #score 改贴在字符串对象上
print(score)          #输出优秀
```

变量的命名需要遵循如下规则。

（1）变量名只能由字母、数字和下画线组成，且不能以数字开头。例如，name、name_1、_name_1 等都是合法的变量名，而 1name、1_name 则不合法。

（2）变量区分大小写，且不能包含空格。例如，Score 与 score 是两个不同的变量。

（3）不能使用 Python 的保留字和内置函数名作为变量名。例如，if、True、False、print 等。

另外，变量名应做到“见名知意”。例如，用 score 表示成绩、name 表示姓名，而不用 a、b 等无意义的字母，这有助于提高代码的可读性与可维护性。

在使用变量的过程中，最容易出现的错误是命名错误（NameError）。例如，变量名定义为 score，却在使用时误写为 Score，由于 Python 区分大小写，解释器无法找到名为 Score 的变量，因此便会报错。相关代码及运行结果如下：

```
>>>score = 95
>>>print(Score)
Traceback(most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print(Score)
NameError: name 'Score' is not defined
```

遇到此类错误时，只需仔细对照变量名的定义与使用是否一致，修正拼写后程序即可正常运行。

### 5.2.2　数据与数据类型

Python 将程序中处理的所有数据统称为对象，每个对象都有其对应的数据类型。数据类型决定了数据的表示方式以及可以对其执行的操作。

Python 中常用的数据类型如表 5-1 所示。其中，整数、浮点数和布尔型用于表示数值类数据，是进行数学运算和逻辑判断的基础；字符串用于表示文本类数据，是程序与用户交互时最常用的类型；列表、元组和字典则用于组织和管理多个数据，适合处理批量信息。

表 5-1　Python 中常用的数据类型

| 类型名称 | 关键字 | 示　例 | 说　明 |
|---|---|---|---|
| 整数 | int | 85、－5、0 | 不含小数部分的数字 |
| 浮点数 | float | 85.5、3.14 | 含小数部分的数字 |
| 布尔型 | bool | True、False | 表示逻辑真或假 |
| 字符串 | str | "张三"、"良好" | 用引号括起来的字符序列 |
| 列表 | list | [85,92,76] | 有序、可修改的数据集合 |
| 元组 | tuple | (85,92,76) | 有序、不可修改的数据集合 |
| 字典 | dict | {"name":"张三"} | 键值对形式的数据集合 |

一般通过 type()函数查看数据或变量的类型。相关代码及运行结果如下：

```
>>> name = "张三"
>>>score = 85.0
>>>gpa = 3.0
>>>is_passed = True
>>>print(type(name))
```

```
<class 'str'>
>>>print(type(score))
<class 'float'>
>>>print(type(gpa))
<class 'float'>
>>>print(type(is_passed))
<class 'bool'>
```

在实际程序中，经常需要在不同数据类型之间进行转换。例如，input()函数获取的用户输入默认为字符串类型，若要进行数值计算，必须先将其转换为数字类型。这也是在示例代码文件 score_v1.py 中使用 float()函数的原因：

```
score = float(input("请输入考试成绩："))          #将字符串转换为浮点数
```

Python 中常用的类型转换函数如表 5-2 所示。

表 5-2 Python 中常用的数据类型转换函数

| 函 数 | 功 能 | 示 例 |
| --- | --- | --- |
| int(x) | 转换为整数 | int("85")→85 |
| float(x) | 转换为浮点数 | float("85")→85.0 |
| str(x) | 转换为字符串 | str(85)→"85" |
| bool(x) | 转换为布尔型 | bool(0)→False |

### 5.2.3 字符串及其操作

字符串(String)是由若干字符组成的序列，是程序中最常用的数据类型之一。在示例代码文件 score_v1.py 中，学生姓名“张三”和评定等级“良好”都是字符串类型的数据。

Python 中的字符串需用单引号或双引号括起来，这两种形式完全等价：

```
name = "张三"
grade = '良好'
```

当字符串内容本身包含引号时，需要交替使用两种引号，即双引号字符串内嵌套单引号，或单引号字符串内嵌套双引号。若使用相同类型的引号嵌套，则会产生语法错误。相关代码及运行结果如下：

```
>>> print("该同学评定等级为'良好'。")        #正确：双引号内嵌套单引号
该同学评定等级为'良好'。
>>> print('该同学评定等级为"良好"。')        #正确：单引号内嵌套双引号
该同学评定等级为"良好"。
>>>print("该同学评定等级为"良好"。")          #错误：双引号内嵌套双引号
SyntaxError: invalid syntax
```

若需要定义多行字符串，则可以使用三引号('''或""")，三引号内部可以自由使用单引号与双引号。相关代码及运行结果如下：

```
>>> report = """学生姓名：张三
考试成绩：85.0
```