1.1 开始

::: details INFO
译者：Mancuoj

来源：1.1 Getting Started

对应：HW 01
:::

计算机科学是一门极其广泛的学科，每年，全球的分布式系统、人工智能、机器人、图形学、安全、科学计算、计算机体系结构以及更多新兴的二级领域都会随着计算机科学的新技术和新发现而扩展。计算机科学的迅猛发展已经深刻地改变了人类生活的各个层面。商业、通信、科学、艺术、娱乐和政治都在计算的领域中得到了新的定义。

之所以计算机科学的巨大生产力能够成为可能，都是因为其建立在一套优雅而强大的基本思想之上。所有计算都始于三点：信息的表示、处理的逻辑、设计抽象来管理逻辑的复杂性，掌握这些基础知识需要我们去精确理解计算机程序的构造和解释。

长期以来，我们都使用 Harold Abelson、Gerald Jay Sussman 和 Julie Sussman 三人编写的经典教科书《计算机程序的构造与解释》即 SICP 来教授这些基本思想，而本书大量借鉴了这本教科书。

1.1.1 Python 编程

> A language isn't something you learn so much as something you join.
>
> — Arika Okrent

为了定义计算过程，我们需要一种被人们广泛使用和各类电脑广泛接受的编程语言，在本文中，我们将主要使用 Python 语言。

Python 是一种被广泛使用的编程语言，其吸引了来自各行各业的爱好者：Web 程序员、游戏工程师、科学家、学者甚至是新语言的设计者。当你学习 Python 时，你就加入到了一个拥有着百万开发人员的社区。开发者社区是非常重要的组织：成员可以互相帮助解决问题，分享他们的项目和经验，共同开发软件和工具。经常会有一些专注的成员因其贡献而获得名誉和广泛的尊重。

Python 语言本身是一个大型志愿者社区的产物，它以其贡献者的 多元化 为傲。该语言由 Guido van Rossum 在 20 世纪 80 年代末构思并实现，他在 Python 3 教程 的第一章中解释了 Python 在当今的众多语言中受欢迎的原因。

Python 作为一种教学语言非常出色，因为在其整个历史中，Python 的开发人员一直在强调 Python 代码的人类可解释性，并在 Python 之禅 的美观、简约和可读的原则下进一步加强。因为它宽泛的特性能够支持各种不同的编程风格，所以十分适合本书，我们将在之后逐一探讨这些风格。从来没有单一的 Python 编程方法，但遵守开发人员社区共享的一组约定会有助于现有程序的阅读、理解和扩展。Python 巨大的灵活性和易学性可以使学生探索许多编程范式，然后将获得的新知识应用到数以千计的 正在进行的项目 中。

这本书极力保留了 SICP 的精神：通过抽象和严格的计算模型逐步介绍 Python 的特性。此外，本书还提供了 Python 的编程实践，包括一些高级语言功能和说明示例。随着阅读的进行，你会自然而然地增加使用 Python 的能力。

开始使用 Python 编程的最佳方法就是直接与解释器进行交互。本节将介绍如何安装 Python 3、如何启动与解释器的交互式会话及如何开始编程。

1.1.2 安装 Python 3

与所有伟大的软件一样，Python 有很多版本，而本文将使用最新稳定版本的 Python 3。许多计算机已经安装了旧版本的 Python，如 Python 2.7，它们与本文要求不符，你需要使用任意安装了 Python 3 的计算机（别担心，Python 是免费的）。

你可以从 Python 下载页面点击以 3 开头的版本下载 Python 3，并按照安装程序的说明完成安装。

如需进一步指导，请观看由 Julia Oh 创建的有关 Python 3 的 Windows 安装 和 Mac 安装 的视频教程。

1.1.3 交互式会话

在与 Python 的交互式会话中，你可以在提示符 `>>>` 后键入一些 Python 代码，Python 解释器会读取并执行你键入的各种命令。

要启动交互式会话，请在终端 (Mac/Unix/Linux) 中键入 `python3` 或在 Windows 中打开 Python 3 应用程序。

如果你看到了 Python 提示符 `>>>`，则已经成功启动交互式会话。我们会使用提示符和一些输入来展示示例。

```
>>> 2 + 2
4
```

交互控制：每个会话都会保留键入内容的历史记录，可以按下 `<Control>-P`（上一个）和 `<Control>-N`（下一个）浏览该历史记录。使用 `<Control>-D` 会退出会话并丢弃此历史记录。在某些系统上，上、下箭头也可以用于循环浏览历史记录。

1.1.4 第一个例子

> And, as imagination bodies forth
>
> The forms of things to unknown, and the poet's pen
>
> Turns them to shapes, and gives to airy nothing
>
> A local habitation and a name.
>
> — William Shakespeare, A Midsummer-Night's Dream

本节将以一个使用“多种语言特性”的示例来介绍 Python，你可以将此部分视为即将到来的功能的预览，在下一节中，我们将从头开始逐步了解整个语言。

Python 内置了一些常见编程功能，例如处理文本、显示图形以及通过互联网进行通信。下面这行 Python 代码

```
>>> from urllib.request import urlopen
```

是一个 `import` 语句，它会导入一个用于“访问互联网数据”的功能，该功能特别提供了一个名为 `urlopen` 的函数，可以访问 URL（也就是访问互联网上的某个网址）上的内容。

语句和表达式：Python 代码由表达式和语句组成，从广义上讲，计算机程序由以下指令组成

1. 计算一些值
2. 执行一些操作

语句通常描述操作，Python 解释器每执行一条语句，计算机就会执行相应的操作。另外，表达式通常用于描述计算，当 Python 计算一个表达式时，它会计算出该式的值。本章会介绍语句和表达式的几种类型。

下面的赋值语句

```
>>> shakespeare = urlopen('https://www.composingprograms.com/shakespeare.txt')
```

将名称 `shakespeare` 与 `=` 后面的表达式的值相连，这个表达式将 `urlopen` 函数应用在了一个包含莎士比亚 37 部戏剧完整文本的 URL 上。

函数：函数封装了操作数据的逻辑。`urlopen` 就是一个函数，而网址是一个数据，莎士比亚的戏剧是另一个数据。从前者到后者的转换过程可能会很复杂，但我们可以将这种复杂性隐藏在一个函数中，从而能够使用一个简单的表达式来跳过该过程。函数是本章的主题。

另一个赋值语句

```
>>> words = set(shakespeare.read().decode().split())
```

将 `words` 与莎士比亚戏剧中出现的共 33,721 个单词的集合相连。其命令链调用了 `read`、`decode` 和 `split`，每个函数都会操作一个中间的计算实体：从 URL 中 `read`（读取）数据，然后将数据 `decode`（解码）为文本，最后将文本 `split`（拆分）为单词放在一个 `set` 中。

对象：`set` 就是一种对象，支持如计算交集和集合关系（membership）等运算。对象无缝整合了数据以及用于操作该数据的逻辑，并隐藏了二者的复杂性。对象是第二章的主题。

最后，这个表达式

```
>>> {w for w in words if len(w) == 6 and w[::-1] in words}
{'redder', 'drawer', 'reward', 'diaper', 'repaid'}
```

是一个复合表达式，它的计算结果是反向拼写同时也为单词的莎士比亚单词集合。神秘符号 `w[::-1]` 表示枚举单词中的每个字母，其中 -1 代表反向枚举。当你在交互式会话中输入表达式时，Python 会在下一行打印它的值。

解释器：复合表达式的求解需要以一个可预测的方式来精确解释代码的过程。实现这样的过程，用于计算复合表达式的程序就称为解释器。解释器的设计和实现是第三章的主题。

与其他计算机程序相比，编程语言的解释器具有独特的通用性。Python 在设计时并不会考虑莎士比亚，但它的高度灵活性使我们能够只用少量的语句和表达式来处理大量的文本。

最后，我们会发现所有这些核心概念都是紧密相关的：函数是对象，对象是函数，解释器是二者的实例。但是，清楚地理解每一个概念及其在组织代码中的作用对于掌握编程艺术至关重要。

1.1.5 错误

Python 正在等待你的命令。即使你可能还不了解其完整的词汇和结构，我们仍鼓励你尝试使用该语言。当然也请你为错误做好准备，因为计算机在极其快速灵活的同时也十分古板。计算机的特性在 [斯坦福的入门课程] 中被描述为

> The fundamental equation of computers is: `computer = powerful + stupid`
>
> Computers are very powerful, looking at volumes of data very quickly. Computers can perform billions of operations per second, where each operation is pretty simple.

> Computers are also shockingly stupid and fragile. The operations that they can do are extremely rigid, simple, and mechanical. The computer lacks anything like real insight ... it's nothing like the HAL 9000 from the movies. If nothing else, you should not be intimidated by the computer as if it's some sort of brain. It's very mechanical underneath it all.
>
> Programming is about a person using their real insight to build something useful, constructed out of these teeny, simple little operations that the computer can do.
>
> — Francisco Cai and Nick Parlante, Stanford CS101

当你尝试使用 Python 解释器时，计算机的古板会立即显现出来：即使是最小的拼写和格式更改也会导致预料之外的输出和错误。

学着解释错误和找到错误的原因称为调试，关于调试的一些指导原则是：

1. **增量测试**：每个编写良好的程序都由可以单独测试的小型模块化组件组成。尽快测试你已经编写的所有内容，以尽早发现问题并获得对组件的信心。
2. **隔离错误**：语句输出中的错误通常可归因于特定的模块化组件。所以在诊断问题时，先追踪错误到最小的代码片段，然后再试着修复问题。 记得打断点
3. **检查你的假设**：解释器会一字不漏地执行你的指示——不多也不少。当某些代码的行为与程序员假设的行为不匹配时，它们的输出就是不符合预期的。明确你的假设，然后将调试的工作集中在验证你的假设上。
4. **咨询别人**：你不是一个人！如果你不理解错误信息，请询问朋友、老师或搜索引擎。如果你已经找出了一个错误，但却不知道如何更正它，可以请其他人查看。在小组解决问题的过程中会分享很多有价值的编程知识。

增量测试、模块化设计、明确的假设和团队合作是贯穿本书的主题，希望它们也将贯穿你的计算机科学职业生涯。