

第一章 Processing 开发环境准备

本章内容

1. 安装Java、Python、Py5、VS Code、Jupyter Notebook
2. 学习使用Jupyter Notebook交互式学习环境
3. 学习使用VS Code开发环境 (IDE)
4. 运行第一个Python程序
5. 运行第一个Py5程序

1. Processing介绍

你是不是也曾经想过：如果你现在已经会写代码，真的还有必要学 Processing 吗？“我已经掌握了 Python、Java、甚至 C++，还需要动手学这个给艺术家和设计师准备的玩具？”但你有没有注意到，我们很多编程时的“有趣想法”，在传统开发环境下往往会被实现细节拖慢？你是否遇到过，仅仅是想把数据变成生动的动态图像，或者快速做个算法原型，就要纠结窗口管理、底层绘图库、兼容性适配问题？Processing 的存在正好能让你跳出繁杂的限制，让有创意的想法用极少的代码、最直接的方式变成可以“眼见为实”的作品——不想感受一下“写代码像画速写”的快感吗？

如果你是一位传统艺术家，又有没有质疑过：“我从来没写过程序，数字创作离我会不会太远？”其实，Processing 的设计初心就是让艺术家和设计师不用考虑晦涩难懂的计算机底层细节，也能轻松用“代码”这支画笔，把静态的创意变成会动、能互动、可以被观众参与的数字作品。你有没有幻想过，让你的作品“动态起来”，甚至能和观众实时互动、在线传播，真正打破画布和舞台的边界？Processing，就是为你准备的数字艺术“万能工具箱”。

如果你已经是交互艺术领域的创作人，你肯定也在思考：“怎样才能用更短的时间、学更少的复杂技能，把我的想法又快又稳地做出来，和各种硬件设备、传感器打交道，还能充分发挥我的视觉美学？”Processing 不仅允许你轻松捕捉观众每一个动作、声音或触碰，甚至能和 Arduino 等硬件打通，实现软硬一体的交互现场。至于投影mapping、传感器互动、甚至实时数据可视化，这些曾经需要大团队和复杂工程实现的东西，现在你一个人“十分钟就能玩起来”。难道你不想自己试试吗？

在现代数字艺术与交互装置设计领域，Processing 已经成为极具影响力且不可或缺的核心工具。作为一个开源、免费的图形库和集成开发环境 (IDE)，Processing 最初源于麻省理工学院媒体实验室，由 Casey Reas 和 Ben Fry 于 2001 年共同开发，其目标即是为电子艺术、新媒体艺术和视觉设计等领域的创作者提供一款易用的编程工具。它通过极大程度地简化编程语法和接口，让完全没有计算机科学背景的设计师、艺术家与教育者也能轻松入门数字创作，在短时间内完成视觉效果、动画、交互装置等创新实践。与诞生之初动辄必须掌握 C++ 或 Java 等复杂底层技术的时代相比，Processing 采用了“草图”模式并集成了许多经过高度抽象的功能如绘图、数据处理等，从而显著降低了数字艺术创作的门槛，使非专业编程人员亦可投身数字媒介艺术之中。

IDE（集成开发环境）是一种用于软件开发的应用程序，结合了代码编辑器、调试工具、构建自动化和版本控制等功能，旨在提高程序员的开发效率。常见的IDE有Visual Studio、Eclipse和PyCharm等，它们提供了友好的开发体验，使程序员能够更集中地进行代码编写和调试。

2004 年以来，随着其社区的开放与全球化，以及 Processing 基金会的设立，Processing 不断迭代和壮大，推动了 p5.js、Wiring 等多平台项目的兴起，实现了面向硬件、网络、物联网、实时表演等多样应用场景的跨领域创新。而在技术层面，Processing 兼具简化且强大的开发语言特性，允许用户以最少的代码实现丰富的视觉表现和动态交互，同时兼容主流操作系统，极易与 Arduino 等开源硬件集成，为新媒体艺术与交互设计开拓了新的疆界。

除了Processing，还有许多著名的交互设计程序，包括OpenFrameworks（一个开源C++工具包，适合创意编码和互动艺术）、p5.js（基于JavaScript的库，设计网页中的交互式图形）、TouchDesigner（用于实时视觉表现的程序）、Max/MSP（专注于音频和视觉互动设计的可视化编程语言）、Unity（一个强大的游戏引擎，广泛用于交互体验和虚拟现实项目）以及Adobe XD（用于用户体验设计的工具，适合创建交互式原型）。这些程序各具特色，适用于不同类型的交互设计和艺术创作。

Processing 为什么一开始要基于 Java？这其实是团队经过深思熟虑后的决定。在当时，Java 的最大优势就是跨平台、安全、生态庞大？Processing 利用 Java “写一次到处跑”的特性，让你无论用什么系统都可以无障碍创作，而且继承 Java 强大的图形和网络能力。而且，Processing 也屏蔽了 Java 冗杂难学的部分，通过简约的语法和 API，降低了初学者的上手门槛。是不是既兼顾技术强大、又不用“被代码绑架”？

那为什么本书又特别强调用 Python？你或许好奇，“是不是多此一举，或者只是图个新鲜？”其实恰恰相反——Python 的简单易学几乎是公认的，无论你有没有编程基础，入门都无压力。再者，无论是 Processing 还是 Arduino，如今都为 Python 提供了官方支持，这意味着你不仅能做可视化交互，还能一步到位连接现实世界的各种智能硬件。更关键的是，如果你以后想进阶人工智能、机器学习领域，Python 现在已经成了最主流的开发语言。也就是说，学会了 Processing 和 Python，不仅满足了创意表达，未来无论走工程、艺术还是 AI，你都有了足够多的选择和竞争力。还觉得这是一条特别划算的学习路径吗？

通过本书学习，你会发现你不仅能用 Python 模式写出属于自己的视觉交互作品，还能洞悉游戏编程的基本原理，亲手实践到底什么是交互装置、物联网，以及这些技术如何和日常生活乃至前沿艺术深度融合。你是不是已经在想，未来自己的第一个作品会是什么样？现在，就让我们一起在代码与创意之间，开启一段看得见成果、玩得了技术、讲得出故事的数字创作旅程吧！

以下是一些有价值的Processing学习资源和阅读网站，适合各个水平的学习者，帮助更好地掌握Processing编程和交互设计：

- 菜鸟教程：<https://www.runoob.com>
- MOOC：<https://www.mooc.cn/>
- Processing官网：<https://processing.org/>
- Coursera：<https://www.coursera.org/>

2. 安装基本的运行环境

2.1 安装JAVA运行时

JAVA是一种面向对象的高级编程语言和计算平台，由Sun Microsystems公司于1995年首次发布。它最大的特点之一是“跨平台性”，即所谓的“Write Once, Run Anywhere”（一次编写，到处运行）。这种跨平台特性源于JAVA程序会被编译成字节码（bytecode），运行时由JAVA虚拟机（JVM）执行，因此无论在Windows、macOS还是Linux等不同操作系统，只要安装了JVM，JAVA程序都能顺利运行。

上一节提到过，Processing是一款基于JAVA语言开发的开源可视化编程环境，广泛应用于视觉艺术、图形设计与编程教育等领域。由于Processing本身是基于JAVA开发的，并且所有Processing编写的项目最终也会在JAVA环境中执行，因此在使用Processing之前，必须要确保计算机已经正确安装了JAVA运行环境，也就是通常所说的JAVA运行时（Runtime Environment, 简称JRE）或JAVA开发工具包（JDK）。

JVM作为Java程序运行的核心，将JAVA代码转换为特定平台上的机器码。JRE包含了JVM及Java核心类库，对于仅需运行JAVA程序的用户，安装JRE即可。对于开发者来说，JDK不仅提供了运行时所需的JRE，还附带了编译器及调试工具等开发工具。

推荐从JAVA的官方网站（Oracle官网）下载最新版本的JAVA。本文使用的Processing（Python, Py5）要求至少安装JAVA 17版本，而本书撰写时JAVA的最新版本为JAVA 24。下载对应平台的安装包后，双击运行安装程序，根据提示完成安装即可。对于不同系统，JAVA的安装目录会有所不同，记下该路径以便后续配置环境变量。配置环境变量可确保系统及开发工具（如IDEA、Eclipse、VS Code或Processing）能正确调用Java工具。

在安装结束后，还需要配置系统变量 `JAVA_HOME`，这样当你需要使用JAVA时，系统就会根据这个系统变量保存的值找到JAVA正确的安装位置

- Windows具体步骤如下：

第一步：打开系统属性 在“此电脑”或“我的电脑”图标上右击，选择“属性”，进入后选择“高级系统设置”，然后点击“环境变量”。

第二步：新建JAVA_HOME变量

在系统变量中点击“新建”，变量名填写为 `JAVA_HOME`，变量值填写为JDK安装目录（例如：`D:\Program Files\Java\jdk-24`）。

第三步：打开命令行 使用快捷键Win+R，输入“cmd”，打开命令提示符窗口。

第四步：检查Java版本

在命令行中输入命令：`java -version`

若安装正确，将显示类似以下信息：

```
java version "24.0.1" 2025-04-15
```

```
java(TM) SE Runtime Environment (build 24.0.1+9-30)
```

```
java HotSpot(TM) 64-Bit Server VM (build 24.0.1+9-30, mixed mode, sharing)
```

命令行的历史可以追溯到20世纪60年代，当时的计算机主要通过键盘操作。随着计算机的发展，命令行工具逐渐演变，形成了现代的命令提示符（Windows）、Linux终端和macOS终端。

Mac、Linux和树莓派用户请自行检索，并设定JAVA_HOME值。提示：首先确认本机使用的终端类型，才能选择合适的指令进行配置。

2.2 安装Processing（Python, Py5）开发环境

Python作为一种高级编程语言，以其简洁的语法和庞大的库生态系统，已成为数据科学、人工智能（AI）和快速原型开发领域中的首选语言。同时，Processing历来以其强大的实时图形渲染能力和交互式设计而闻名，尤其在视觉艺术和创意编程方面占据重要地位。随着科技的发展，尤其是在人工智能和数据科学领域的快速进步，Python已经超越JAVA，成为目前全球范围内最受欢迎的编程语言之一。考虑到未来学习和技术应用的广阔前景，本书决定采用Python作为主要开发语言，以便读者能够更好地掌握现代编程工具并具备更高的竞争力。

Processing项目很早就支持了Python语言。在官方的Processing IDE中，用户可以直接切换到Python模式。此外，也有像Py5这样独立与Processing IDE的Python库，完美复刻和扩展了Processing的功能，不仅可以用于创作可视化交互项目，还能够在任何Python程序和开发环境中灵活集成，而无需运行Processing IDE。这使得Python与Processing的结合变得更为高效和强大。

Py5是一个开源Python库，它的出现不仅突破了传统Processing依赖于特定IDE的局限，还将Processing的创意编程能力与Python灵活的开发生态系统相结合。

- 具体安装步骤如下： 第一步：安装Python
下载安装Python。Processing要求Python的最低版本为3.9，而本书编写时的最新稳定版本为Python 3.13。下载后双击安装，一路选择默认选项即可，在安装过程中，**务必勾选"Add Python to PATH"**。

第二步：验证Python安装

在命令行中输入 `python -version` 指令，如果系统显示出类似 `Python 3.13.0` 这样的版本号输出，即说明Python已经成功安装。

第三步：安装Py5库

使用Python自带的包（插件）管理工具pip。在命令行中输入： `pip install py5`，等待命令执行完成。

第四步：验证Py5库安装

在命令行中输入 `python` 进入Python的交互式编辑模式（提示符变为 `>>>`）继续输入： `import py5`，

如果直接回车后没有出现任何错误提示（如ModuleNotFoundError等），说明Py5安装成功。

3. 安装集成开发环境（IDE）

再好的作家也不会只用一支笔写完一本书。现代科技为作家们提供了各种工具，比如电子笔、文字识别软件、资料管理工具等，这些都极大提升了创作效率。同样地，对于程序开发人员来说，仅靠最基础的代码编辑工具早已无法满足日益复杂的项目需求，这时就需要引入更强大、更贴合实际开发流程的工具——IDE。

IDE，全称为“集成开发环境（Integrated Development Environment）”，是专为编程设计的软件工作平台，集代码编辑、自动补全、项目管理、调试、运行、版本控制等多种功能于一体。它能够帮助开发者高效编写、管理和调试代码，大幅提升学习和开发的体验。

Processing 官方虽然自带了一个轻量级的默认IDE，适合初学者快速上手和进行简单实验，但其功能相对有限。考虑到很多读者未来可能会涉及更复杂的项目，或者希望进一步进入如科学计算、数据分析、Web开发等更广泛、主流的技术栈，掌握主流且专业的IDE将极具优势。因此，本文选择介绍目前业界应用广泛、兼容性良好的 Jupyter Notebook 和 Visual Studio Code（VS Code）作为开发环境。

Jupyter Notebook 是一款交互式的编程笔记本，非常适合数据分析、科学计算、教学和探索性编程。它可以在网页界面中编写、运行与可视化代码，并便于文档化、讲解和分享。

VS Code 则是一款轻量但功能强大的现代编辑器，支持众多编程语言、可高度扩展、跨平台、内置Git版本控制，适合专业开发、多人协作及大型项目管理。

两者都拥有庞大的用户社区和丰富的插件生态，能极大提高学习和开发效率。

3.1 使用VS Code进行多语言跨平台开发

为了完成本书的学习，你需要先简单了解市面上的几款主流IDE：

- Processing IDE：Processing的官方自带编辑器，界面极简，上手快，适合初学者可视化和交互实验，但功能略显单薄，项目和插件扩展能力有限。
- PyCharm：专业Python开发利器，专为Python打造，带有完整的项目管理、调试、测试、自动补全和环境配置功能，适合中大型Python项目，但体积偏大，对新手可能有点“压顶”。
- Eclipse：JAVA及其他主流语言的老牌IDE，开源免费、功能强大，插件体系庞大，适合管理大型复杂项目，但上手曲线略陡、UI略显“复古”。
- Arduino IDE：专为Arduino单片机开发设计的编辑器，界面友好，上手简单，适合硬件初学者，但与主流桌面开发环境割裂，日常多领域开发略显捉襟见肘。

看着这些“各自为政”的IDE，天哪，你是不是已经在内心默默许愿：“求求了，让我只用一个软件吧！”

想象这样一个令人头大的画面：你一边要用Processing IDE写交互效果，一边要跑Python处理点AI数据，隔壁同学又喊你帮忙用Arduino IDE制作一个交互装置，结果你的电脑桌面上堆满了各种IDE和窗口。左边是Processing IDE（Processing），右边是PyCharm（Python），一会还得分心切到Eclipse（JAVA），甚至Arduino IDE

(Arduino) 也来掺一脚。此情此景，天哪，你是不是已经开始怀疑人生，搞不清到底该往哪个窗口里敲代码？

如果有一种工具能把所有开发环境揉到一起，不就世界大同了吗？好消息：VS Code就是你的超级救星！

VS Code (Visual Studio Code) 是一款由微软推出的现代、开源、免费的编辑器。它以轻量级著称，启动秒开不拖沓，却又“身轻体壮”——通过丰富的扩展 (Extension) 系统，几乎可以支持市面上你想得到的所有主流编程语言、框架与工具。更棒的是，VS Code 跨平台支持 Windows、macOS 与 Linux，凭借高扩展性、极速响应、优雅界面与巨大社区，被全球开发者誉为“开发界的瑞士军刀”，一跃成为编程必备工具。

默认情况下，VS Code其实只是一个“骨感”的代码编辑器，但通过各种扩展插件 (Extension)，它可以化身为超级 IDE，支持调试、文档、AI 助手、远程开发等各种功能。无论你要搞 Processing (JAVA)、传统 JAVA 工程、Python、Py5 还是 Arduino 单片机开发，都可以通过安装相应扩展，一键集成到 VS Code 里！这样，无论课堂学习还是实际项目，只需要专注一个软件即可，大大降低了环境切换带来的混乱，节省了宝贵的摸鱼时间，学习和开发一次到位。

- VS Code 安装与必备扩展

第一步：下载安装 VS Code 访问 VS Code 官网，选择对应系统下载安装包，双击按照向导安装即可。

第二步：（可选）登录并启用 Copilot AI 辅助开发

启动 VS Code 后，点击左侧“扩展”图标，搜索并安装 GitHub Copilot。安装后会引导你用 GitHub 账号登录，开启智能自动补全与 AI 代码助手。

注：最新版本的 VS Code 已经默认包含 Copilot，只要打开右侧侧边栏即可

第三步：安装 IntelliCode 扩展

在扩展市场搜索 IntelliCode，安装后，享受微软自家 AI 辅助的智能补全和代码建议，提升开发效率。

注：扩展市场在 VS Code 左侧侧边栏，由四个小方块构成： 

第四步：安装 Python 扩展 在扩展市场搜索 Python，选择官方 Microsoft 出品的 Python 扩展，安装后可高亮语法、跳转定义、调试、虚拟环境管理。

第五步：安装 Jupyter 扩展

在扩展市场搜索 Jupyter 并安装后，可直接在 VS Code 里新建和编辑 .ipynb 文件，实现交互编程与可视化。

安装完以上扩展后，就可以用 VS Code 一站式管理和开发 Processing (JAVA模式)、Python、Py5、Arduino 甚至未来更多项目类型，统一开发体验，从此告别“窗口切换地狱”！

3.2 使用Jupyter Notebook进行交互式程序设计学习

***如果你此刻阅读的不是本教材的Jupyter Notebook版本，那么可以跳过这一节，所有代码全部在Code文件夹中*

在传统的程序开发教学中，教师通常会将完整的代码打包发给学生，要求大家直接运行和学习。这种做法虽然保证了代码的完整性，但也有很大局限性：一方面，如果只给代码片段，学生往往无法理解整体逻辑和运行过程；另一方面，完整项目往往体积庞大、结构复杂，初学者很容易感到迷茫和畏惧，难以快速抓住重点、循序渐进地学习。

Jupyter Notebook 的出现很好地解决了这一难题。Jupyter Notebook 是一种基于网页的交互式编程环境，最初由 IPython 项目组发起和开发。Jupyter Notebook 的最大特点在于“交互式细粒度编程”：代码和文档可以按单元格（Cell）逐步组织和运行，用户不仅可以随时运行某个片段，还能即时看到输出、调试结果和可视化图表。这样一来，编程学习过程便可以完全模块化、结构化，阅读和操作更加直观灵活，极大提升了学习和教学效率。

Jupyter Notebook 的文件以 `.ipynb` 作为后缀。其内容通常由两类单元格组成：Markdown Cell 和 Code Cell。

- Markdown Cell 用于书写文字说明、公式、图片、标题及注释，支持 Markdown 语法，方便教师或作者进行详细讲解。
- Code Cell 则专门用于编写和运行 Python 代码，每个代码单元格都能独立执行并展示输出。这样的设计使得讲解与实验紧密结合，理论和实践无缝衔接。

本书默认使用 Jupyter Notebook 进行写作与演示。不过，每个 Code Cell 内部的 Python 代码都是完整且独立可运行的，没有前后文依赖。

如果你正在阅读本书的其他版本，只需将对应的 Python 代码复制到任意 Python IDE 里，按照书中的分步讲解逐个粘贴运行，即可获得相同的学习体验和结果。

- 具体安装步骤如下：
第一步：首先，确保电脑中已经正确安装了 Python 环境（推荐 3.9 及以上版本）。
第二步：在命令行输入以下命令安装 Jupyter Notebook：`pip install notebook`
第三步：在命令行输入：`jupyter notebook`。浏览器会自动打开一个新的窗口，说明已经安装成功。

4. 第一个Python程序

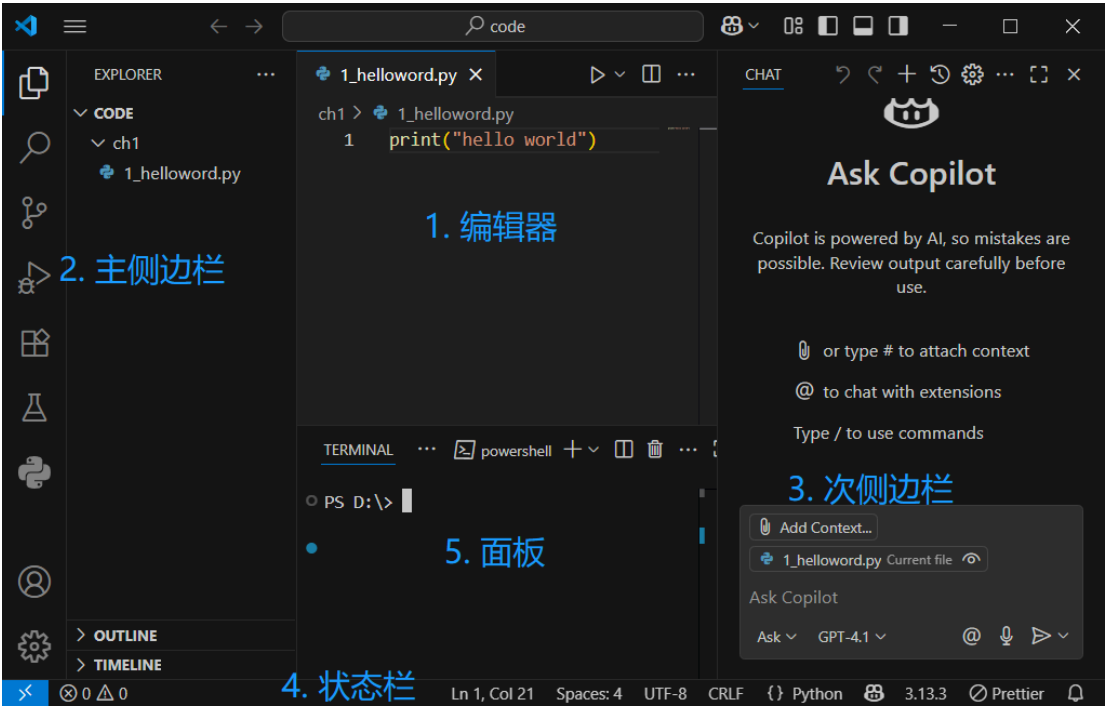
目前为止，如果一切顺利的话，你应当完成了以下内容：

- 安装JAVA运行时，从而运行Processing应用程序
- 安装Python和Py5，使你能够用Python进行Processing开发
- （可选）安装Jupyter Notebook，以便阅读本文档
- 安装VS Code，帮你高效的输入代码

再一次，如果一切顺利话，我们现在可以开始你的第一个Python程序，我将通过这个程序向你简单介绍VS Code的使用和Python应用程序的基本代码结构。


4.1 运行第一个Python程序



启动 VS Code 后，你会看到整洁而现代的主界面。让我们先来简单介绍各个区域的功能：



界面区域	位置	主要功能
编辑器	中央	是你编写和浏览代码的主要窗口，可以同时打开多个文件标签页
主侧边栏	左侧竖排图标	文件管理、搜索、版本控制、扩展（插件）、运行与调试视图等，点击图标可以快速切换各类面板
次侧边栏	右侧扩展视图	当前为Copilot AI Chat交互界面
状态栏	下方横条	显示当前文件信息（如行号、编码格式、Python环境）、Git分支、错误提示等实用状态，实时反馈工作状态和系统信息
面板	下方扩展视图	包括“终端Terminal”（运行命令和代码）、“输出Output”、“调试Console”、“问题Problems”等，当你运行代码或调试程序时，都可以在这里看到相应的输出结果或错误

以上这些面板都可以在 VS Code 右上方进行快捷打开或关闭，也可以在 View（视图）菜单中打开或关闭。

- 创建并运行第一个 Python 文件
 - 通过菜单 `File - New File` 创建一个新的文件， `ch1.py`
 - 在VS Code中打开这个文件，并输入 `print('hello world')`
 - 在代码文件的右上角，你会看到一个运行 
 - 程序运行后，在下方的“终端”面板中就能看到输出结果
 - 这段代码在 `Code/ch1/1_helloworld.py` 中
- 如果你阅读的是Jupyter Notebook版本，可以在本文档中直接运行代码
 - 使用VS Code打开本文件
 - 点击右上角的Jupyter Notebook Kernel选择按钮

- 选择 `python environment - python 3.13`，也就是你之前安装的Python
- 成功启动Kernel的话，会在右上角看到Python版本号  Python 3.13.3
- 选中下面的 Code Cell，在 Cell 左侧外面一点会出现运行 

```
In [ ]: print("hello world") # 这是一个简单的 Python 程序将打印 "hello world" 到控制台
```

现在你试着用同样的操作，创建一个新的Python文件，叫做 `py5test.py`，并将下面的代码拷贝进去然后运行。正确的话，会弹出一个对话框，并绘制了一个蓝色的矩形。这段代码在 `Code/ch1/2_py5test.py` 中

```
In [ ]: # 这是一个使用 Py5 库绘制矩形的 Processing 程序
import py5 # 引入第三方库，Py5，使用Py5包装好的Processing功能

def setup(): # def 表示设计了一个函数（功能模块），setup()是Processing的一个标准函数
    py5.size(400, 400) # 设置画布大小为400x400像素
    py5.background(255) # 设置背景颜色为白色
    py5.stroke(0) # 设置描边颜色为黑色
    py5.stroke_weight(2) # 设置描边宽度为2像素
    py5.fill(100, 150, 250) # 设置填充颜色为RGB(100, 150, 250)
    py5.rect(50, 50, 300, 300) # 绘制一个矩形，位置在(50, 50)，宽度和高度均为300

def draw():
    ...

    draw()函数在setup()之后循环执行，通常用于绘制动画或动态效果
    在这个例子中，draw()函数为空，因为我们只需要绘制一次静态矩形
    ...

    pass # pass语句表示什么都不做，留空函数体

py5.run_sketch() # 函数调用，运行Py5程序，开始执行setup()和draw()函数
```

4.2 注释

你需要掌握的第一个Python知识点就是“注释”与代码结构。

注释是程序代码中用于解释、注解或说明代码目的和功能的文字部分。注释不会程序执行，因此它们主要用于帮助程序员和其他人理解代码，提供上下文信息或记录开发过程中的思路。注释对于提高代码的可读性和可维护性非常重要，尤其在多人协作或长时间后的维护工作中。

- 单行注释：用 `#`，其后内容不会被程序运行，用于解释说明每行代码的作用。
- 多行注释/文档字符串：用三个单引号 `'''` 或三个双引号 `"""` 包裹。

```
# 这是单行注释
'''
```

```
这是多行注释
如果你使用的是IDE默认值
那么你的注释应当是绿色字体
'''
```

```
print('hello')
```

合理的使用注释，不仅能够帮助别人看懂你写的代码，更重要的帮助你自己理解多年前写的乱麻**

4.3 Python程序结构

Python语句是构成Python程序的基本单位，用于执行特定的操作或指令。每个语句都可以由一个或多个词法元素（如标识符、关键字、操作符等）组成。Python的语句可以执行计算、控制程序流程、定义函数等多种功能。

```
# 这是两行简单的Python语句
# 第一行创建了一个被称为变量的Python程序基本元素，并赋值为5
# 第二行将这个变量输出在屏幕中
a = 5
print(a)
```

当然，对于大型应用程序而言，不可能只有这些简单语句构成。但无论程序多么复杂，都可以由以下三个大模块构成，你可以对比前面Py5的例子来看：

- import语句：导入模块或库，如果不使用第三方库则没有这个部分。
- 由 def 开始的函数定义：函数就像“功能块”或“工具”，比如 def setup(): 就封装了初始化Processing的功能；每个函数实现一个相对独立的功能。
- 函数调用：现在有了许多厨具（函数）和食材（数据），你可以像大厨一样“操刀上阵”，用这些工具和食材组合成一道成品（应用程序）。比如前面你调用的 print 函数。

最后，也是最重要的一点，就像在写文章时需要标点符号或者段落分隔符一样，Python也需要对代码格式化，才能让计算机读懂。对于像Java或者C++这样的语言，他们通过识别每行语句末尾的 ; 来对代码格式化，Python使用 Tab 退格键 来区分不同层级的代码块，许多同学在刚开始写程序时，总是忽略这一点，导致程序无法正确运行

4.4 程序调试基本方法

在实际的开发过程中，一定会出现各种代码错误，比如字母写错了，数据输入错误，或者更严重的逻辑错误，所有的开发环境都提供的错误反馈机制，帮助开发者解决这些问题。

运行下面这段错误的代码。在 Code/ch1/3_error.py

```
In [ ]: import pY5 # 这里的pY5是错误的，应该是py5，Python区分大小写

def setup():
    py5.size(400, 400)

def draw():
    pass

py5.run_sktech() # 这里的函数名拼写错误，应该是run_sketch()
```

会发现下方终端面板里出现了红色的 错误提示（error）：

```
ModuleNotFoundError: No module named 'pY5'
```

以及

```
AttributeError: py5 has no field or function named "run_sktech".
Did you mean "run_sketch"?
```

这就是 错误调试 的过程。错误信息通常包括：

- 错误类型：（如 `NameError`、`SyntaxError`、`TypeError` 等），说明出了什么问题；
- 行号：通常提示错误出现在代码的哪一行；
- 错误描述：如这里的 `No module named 'pY5'`，意思是“没有名为pY5的库”。

这段错误信息就是“程序为你敲响的警钟”。通过分析这些提示，你可以快速定位代码问题，例如此例就是“函数名拼写错误”，只需改正即可。学会解读并利用错误信息，是每个程序员走向高手的必经之路——它能帮助你快速发现、修正问题，写出更稳健的代码。

本章总结

1. Processing：开源编程环境，旨在简化创意编程，尤其适合艺术和设计领域的可视化和交互应用。
2. Python：高级编程语言，以简洁的语法和广泛的库生态成为数据科学和人工智能领域的首选。
3. IDE（集成开发环境）：集成多种编程功能的软件，提供代码编辑、调试、项目管理等以提高开发效率。
4. Jupyter Notebook：一种交互式编程环境，适合边写边运行代码，支持数据可视化和文档分享。
5. VS Code：一款功能强大的代码编辑器，支持多种编程语言和插件，适合开发复杂项目。
6. 注释：通过 `#`（单行）或 `''' / '''`（多行）来增强代码可读性，说明代码功能。
7. 函数：一段可重复使用的代码，像厨房中的微波炉，能够接受输入、执行操作并返回结果。
8. 错误信息：程序在运行中产生的问题反馈，包括错误类型、行号和描述，帮助开发者快速定位问题。

课后练习

1. （可选）如果你阅读的是本书的PDF版本，请使用Jupyter Notebook打开本文档。
2. 请把你最拿手的一道菜教给我！把每一步过程都按照 输入、执行方法、结果 的方式写出来。
3. 说明从学校到你家的路线，除了把每一步写清楚，还会在某些地方加上一些小提示（比如“路口这有个大广告牌，很显眼”“马路对面有条狗，别怕它”）。请用 输入、执行方法、结果 的方式，把这件事拆开，并指出哪些内容相当于“注释”。
4. 想象你要教别人用微波炉热饭。请将整个过程分解为 输入、执行方法、结果，并说明 微波炉 在这里相当于什么。
5. 请用 网购 这件事，按 输入、执行方法、结果 来拆解一次完整的购物流程，并说明 函数调用 在这个场景中对应的是什么。
6. 假如你肚子饿了，让家人帮你做三明治。请把这个过程分成 输入、执行方法、结果 三步，并指出谁是 函数，你是怎样 调用 这个函数的。

练习题提示

以5为例，其他雷同：

- 输入：三明治的要求（比如要不要加蛋）
- 执行方法：你把需求告诉家人，家人操作，完成三明治
- 结果：你拿到三明治
- 家人就是三明治 函数，你告诉他们怎么做（参数），他们帮你完成（函数调用）