

第 1 章 AI 时代的编程革新

在过去的几十年中，编程技术经历了从简单的机器语言到复杂的高级语言的发展。编程已经不仅仅是计算机科学家的专利，而是进入越来越多的普通人的生活。无论是软件开发、数据分析，还是科学研究、艺术创作，编程都扮演着重要的角色。进入 21 世纪后，随着人工智能（AI）技术的快速发展，编程迎来了全新的时代。

AI 技术的引入，为编程世界带来了革命性的变化。在传统编程方式中，开发者需要花费大量时间编写和调试代码，处理各种复杂的问题。而在 AI 的帮助下，编程变得更加智能化和高效。AI 不仅能自动生成代码，还能优化现有代码，进行错误检测和修复，使得编程过程大大简化，开发者可以将更多精力投入到创意和设计中。

本章将带领读者深入了解 AI 是如何融入编程的，从根本上改变开发流程和思维模式，并重点介绍 AI 在编程中的应用，展示如何通过 AI 生成代码并进行代码优化和错误检测，提升编程效率和代码质量。

AI 不仅改变了编程方式，还重新定义了编程学习和教育方式。传统的编程教育依赖于课堂教学和教材，而 AI 则为个性化学习提供了可能。基于 AI 的学习平台可以根据学习者的进度和表现动态调整教学内容，提供即时反馈，帮助学习者更快地掌握编程技能。

下面我们简单介绍一下 AI 知识，先对其有一个基本认识，后面的学习更容易理解。

1.1 什么是 GPT

GPT（Generative Pre-trained Transformer，生成预训练模型）是一种属于深度学习范畴的大型生成模型。它由以下三部分组成：

- ❑ **G**: 即 **Generative** (生成式)。GPT 通过深度学习算法对已有数据进行学习, 根据输入的指令生成新的内容, 具备一定的原创能力。同一个问题每次提问时可能会获得不同的答案。在生成答案时, GPT 采用自回归语言模型, 基于上下文预测单词的概率分布生成下一个单词, 并将其添加到已有答案中, 这个过程表现为逐字生成的打字机效果。
- ❑ **P**: 即 **Pre-trained** (预训练)。GPT 利用海量语料数据进行预训练, 通过深度学习掌握自然语言的语法、语义和相关知识, 从而构建一个包含大量参数的知识数据库。预训练阶段是关键, 其使模型具备了理解和生成自然语言的能力。
- ❑ **T**: 即 **Transformer** (转换模型)。GPT 的核心是基于 Transformer 架构的机器学习系统, 由 Google 设计的这个大语言模型通过神经网络模拟人脑的学习方式, 实现对复杂数据的高效分析和学习, 从而准确地理解语义并生成新的内容。Transformer 的多头自注意力机制和并行处理能力, 使得 GPT 在自然语言处理任务中表现出色。

通过以上三部分的结合, GPT 展现出了强大的生成能力和理解能力, 使其在文本生成、翻译、问答等领域都有广泛的应用。

1.2 什么是 RAG

RAG (Retrieval-Augmented Generation, 检索增强生成) 是一种结合信息检索和生成式 AI 技术的自然语言处理方法。它的主要目标是利用外部知识库提高生成文本的准确性和丰富性。这种方法不仅依赖于大型语言模型 (Large Language Model, LLM) 的生成能力, 还能够从知识库或搜索引擎中检索相关信息, 从而生成更为精确和上下文相关的回答。

1. RAG的工作流程

RAG 的工作一般分为两个阶段: 检索阶段和生成阶段。

- ❑ **检索阶段**: 在接收到用户的查询后, RAG 首先从外部数据库、文档集合或者知识库中检索出相关的信息。这一步通常使用传统的搜索技术或特

定的向量数据库进行相似性匹配。检索出的信息作为上下文数据，用于增强生成模型的输入。这些信息可以是文本段落、数据表或其他类型的相关数据。

- ❑ **生成阶段：**在获取了检索信息后，RAG 会将这些信息与用户的初始查询相结合并输入生成式语言模型（如 GPT-4）中。生成式语言模型会根据检索出的上下文数据生成回答，确保生成的内容不仅基于模型的内部知识，而且参考了最新的检索结果。

2. RAG的优势

- ❑ **知识更新：**传统的语言模型通常无法及时更新其内部知识，而 RAG 通过检索最新信息可以弥补这个不足。
- ❑ **增强的准确性：**通过引入外部信息（特别是领域知识或时间敏感的信息），RAG 能生成更具准确性和可靠性的回答。
- ❑ **多样化的应用：**RAG 可以应用于多个领域，如客户服务、医疗健康、教育等，其中，准确的信息检索和生成尤为关键。

3. RAG的应用实例

RAG 一个常见的应用场景是问答系统。假设用户询问一个近期事件的详细信息，传统的生成模型可能由于知识库的滞后性而无法提供准确的答案，而 RAG 可以通过检索新闻数据库来获取最新信息，再结合生成模型，为用户提供准确且详细的答案。

RAG 作为一种新兴的自然语言处理技术，能够有效结合检索和生成的优势，为用户提供更具实用性和准确性的文本生成服务。这种方法在不断发展的 AI 技术中扮演着重要角色，为未来的智能系统提供了新的发展方向。

未来，随着 AI 技术的不断发展，编程方式将进一步演变。开发者将更多地依赖 AI 工具，从而提升生产力和创造力。同时，AI 也将推动编程教育的革新，培养出更多适应新时代需求的技术人才。在后面的内容中将通过实例来展示 AI 在编程中的具体应用场景，帮助读者全面理解和掌握这种新兴技术。

1.3 编程的演变与 AI 的融入

编程技术自诞生以来经历了多个重要的发展阶段。从早期的机器语言到现代高级编程语言，编程方式不断演变。早期的编程需要直接操作机器指令，这种方式既烦琐又容易出错。随着计算机技术的发展，汇编语言应运而生，它用助记符取代了复杂的机器指令，使编程变得简单一些。然而，汇编语言仍然要求编程者了解底层硬件结构，对开发者的要求很高。

为了解决这些问题，高级编程语言如 Fortran、C 语言等相继出现。这些语言更接近人类的自然语言，使编程变得更加直观和容易理解。进入 21 世纪后，Python、JavaScript 等更为简洁和功能强大的高级语言开始流行，极大地提高了开发人员的开发效率和代码的可读性。同时，各种集成开发环境（Integrated Development Environment, IDE）和版本控制系统的出现，进一步提升了开发人员的开发效率和代码管理能力。

AI 技术的融入使得编程过程更加智能和高效。在传统的编程模式下，开发者需要花费大量时间编写和调试代码，而在 AI 的帮助下，这一过程变得更加自动化和高效。AI 可以根据开发者提供的需求描述，自动生成符合要求的代码片段，显著减少了手动编写代码的工作量而且还能优化代码，在错误检测和修复方面也发挥着重要作用。修复错误的过程相当于一次学习机会和实时反馈。

随着 AI 技术的不断进步，编程方式和编程思维将进一步发展，开发者将会更多地依赖 AI 工具来提升生产力和创造力。同时，AI 技术也将推动编程教育的革新，培养出更多适应新时代需求的技术人才。未来，AI 将继续引领编程变革，开发者需要不断学习和适应新技术，迎接新的挑战 and 机遇。

1.4 为何思维方式比技术更重要

早期，开发者必须深入了解编程语言的语法规则，掌握各种函数的使用，才能手工编写每一行代码。这种编程方式要求开发者不仅要有扎实的技术基础，还

需要具备极强的逻辑思维能力。开发者需要仔细思考每个步骤，以确保代码逻辑的正确性。然而，当遇到超出自己知识范围的问题时，开发者往往感到无从下手，以至于影响项目的进度。

如今，编程方式和要求发生了根本性的变化。AI 技术能够根据开发者的需求自动生成代码，并优化代码性能，甚至能为复杂的问题提供解决方案。开发者只需要提出想要实现的功能，AI 就能提供详细的实现步骤和方法，这大大简化了编程步骤。这项变革让开发者不需要亲自编写每行代码，而是将更多的精力放在更高层次的设计和架构上。

在 AI 时代，思维方式比具体的编程技术更重要。AI 能够处理大量的基础编码工作，让开发者的角色从传统的代码编写者转变为系统架构师和需求分析师。开发者需要具备宏观的思维方式，从整体上把握项目的架构和设计，而不是只关注具体的代码实现细节。

这种转变带来了巨大的优势，开发者可以将更多的时间和精力投入需求分析和系统设计中。理解用户需求并设计出合理的系统架构成为开发者的主要任务，AI 技术则负责具体的代码生成和优化工作，从而确保代码的高效和可靠。通过这种分工，编程效率大大提高，项目的开发周期也明显缩短。

以往，编程者被烦琐的编码工作所束缚，难以有足够的时间和精力去探索新的创意。如今，AI 承担了大量的基础工作，开发者可以将更多的注意力集中在创新设计和功能实现上。

在 AI 时代，开发者的思维方式需要从具体的编码细节转向整体的架构设计。这种思维方式的转变不仅适应技术发展的趋势，还提高了开发的效率和质量。理解和掌握这种新的思维方式，会让开发者在 AI 时代占据更重要的地位。

1.5 传统开发方式与 AI 开发方式

在软件开发的历史长河中，传统开发方式与 AI 开发方式各有特点和优势。AI 技术快速发展，开发方式正在发生根本性的变化。通过对比传统开发方式与 AI 开发方式，可以更清晰地了解这种变革的深远影响。

1.5.1 传统开发方式与 AI 开发方式对比

为了更好地理解传统开发方式与 AI 开发方式的区别，可以通过具体的例子进行详细分析。下面以开发一个用户管理系统为例，比较两种开发方式在不同阶段的差异。

1. 需求分析

- ❑ **传统开发：**在传统开发方式中，需求分析通常由业务分析师和开发人员组成的开发团队共同完成。开发团队需要经过多次会议，详细讨论系统的功能需求、用户角色、权限设置等方面的内容。需求分析文档需要明确每个功能模块的细节，确保开发人员在编码时有清晰的指导。
- ❑ **AI 开发：**在 AI 开发方式中，需求分析依然重要，但 AI 工具可以辅助这个过程。例如，使用自然语言处理技术，AI 能够根据开发者描述的需求自动生成需求分析文档。这不仅提高了需求分析的效率，还减少了人为错误。

2. 系统设计

- ❑ **传统开发：**传统开发的系统设计阶段需要资深架构师进行系统架构设计，包括数据库设计、接口设计和模块划分等。架构师需要考虑系统的可扩展性、性能和安全性等多个方面。系统设计文档通常非常详细，以指导后续的开发工作。
- ❑ **AI 开发：**在 AI 开发方式中，AI 工具可以辅助系统设计。例如，AI 可以根据需求自动生成数据库模型和接口设计。开发者可以利用 AI 工具快速创建系统架构并进行模拟和测试，从而优化设计方案。这种方式大大缩短了系统设计的时间，提高了设计的质量。

3. 代码编写

- ❑ **传统开发：**在传统开发中，代码编写是最耗时的阶段。开发者需要根据需求分析和系统设计文档，手工编写每个功能模块的代码。这个过程需要大量的编码工作，同时需要不断进行单元测试和调试，确保代码的正确性和

性能。

- ❑ **AI 开发：**AI 开发在代码编写阶段展现出了显著优势。通过描述需要实现的功能，AI 能够自动生成相应的代码。例如，开发者只需要输入“创建一个用户注册功能”，AI 便能生成包含用户输入验证、数据存储、错误处理等功能的完整代码。这样，开发者可以节省大量的编码时间，将更多精力放在功能设计和优化上。

4. 测试与调试

- ❑ **传统开发：**测试与调试是传统开发中的一个关键环节。开发者需要编写单元测试代码，并通过手动测试和自动化测试工具来检测代码中的错误。发现错误后，需要进行逐步调试和修正，确保代码的可靠性和稳定性。
- ❑ **AI 开发：**AI 开发在测试与调试方面也有显著优势。AI 工具可以自动生成测试用例并进行自动化测试。对于发现的错误，AI 能够提供详细的错误分析和修复建议，甚至直接修正代码中的问题。这种方式不仅提高了测试的覆盖率和效率，还大大缩短了调试的时间。

5. 用户管理系统

1) 传统开发的用户管理系统

- ❑ **需求分析：**开发团队通过多次会议讨论，编写详细的需求文档。
- ❑ **系统设计：**架构师设计数据库结构、用户接口和各个功能模块。
- ❑ **代码编写：**开发者逐行编写用户注册、登录和信息更新等功能的代码，并进行单元测试和调试。
- ❑ **测试与调试：**编写测试用例，进行手动和自动化测试，发现并修正错误。

2) AI 开发的用户管理系统

- ❑ **需求分析：**开发者输入需求描述，AI 生成需求分析文档。
- ❑ **系统设计：**AI 工具自动生成数据库模型和接口设计，开发者进行优化。
- ❑ **代码编写：**开发者描述要实现的功能，AI 生成完整的代码，包括用户注册、登录和信息更新等功能。
- ❑ **测试与调试：**AI 自动生成测试用例，然后进行自动化测试并提供错误修复建议。

1.5.2 时间与资源消耗对比

时间与资源的消耗是衡量开发效率的重要指标。通过对比可以发现，AI 开发对每个阶段的工作效率和开发质量都有所提升，减少了开发与资源消耗。在传统开发中，项目的时间与资源消耗主要集中在代码编写、测试和调试上。开发者需要花费大量时间来理解项目需求、设计系统和编写代码，并通过反复测试来修正错误。尤其是对于大型项目，开发周期往往较长，资源消耗也非常巨大。

在 AI 开发方式中，时间与资源的消耗大大减少。AI 工具可以自动生成代码，进行优化和测试，显著缩短了开发周期。开发者可以将更多的时间和资源投入到需求分析和系统设计上，从而提高项目的整体效率和质量。

1.5.3 AI 时代开发人员的新技能要求

在传统开发方式中，编程技能是开发人员的核心竞争力，要求开发人员具备扎实的编程基础和丰富的项目经验。在 AI 时代，编程技能要求相对降低，但开发人员需要具备以下新技能。

- ❑ **熟练使用 AI 工具：**熟练使用各种 AI 开发工具，如代码生成器、自动化测试工具等。
- ❑ **数据分析能力：**能够利用 AI 进行数据分析，从中提取有价值的信息，以指导系统设计和优化工作。
- ❑ **系统设计能力：**具备宏观思维，能够从整体上设计系统架构，确保系统的灵活性和可扩展性。
- ❑ **跨领域知识：**了解 AI 技术在不同领域的应用，如自然语言处理和机器学习等，以便更好地应用这些技术。

1.5.4 从传统项目到 AI 项目的转变

从传统开发模式向 AI 开发模式转变，不仅涉及技术和工具的更新，更需要

开发思维和流程的全面革新。下面详细探讨这个转变过程，并通过具体案例说明如何有效地实现这个转变。

1. 识别适用场景和选取AI工具

- ❑ 识别适用场景：开发团队需要识别哪些项目或工作可以引入 AI 技术。例如，在一个大型电子商务平台项目中，用户推荐系统、客服聊天机器人等功能非常适合使用 AI 技术。
- ❑ 选取合适的 AI 工具：选择适合项目需求的 AI 工具是转变的关键。常见的 AI 工具包括自然语言处理（NLP）工具、机器学习平台、自动代码生成工具等。

2. 引入AI工具进行试点

在大型项目中选择一个小型或独立的模块进行 AI 技术试点。例如，电子商务平台中，客服聊天机器人功能可以作为试点。开发团队可以通过 AI 工具生成聊天机器人相关的对话逻辑代码，并使用机器学习算法训练聊天模型。例如现在要做一个电子商务平台的客服聊天机器人，步骤如下：

(1) 需求分析：开发团队识别出客服聊天机器人是一个适合 AI 技术的模块，确定需求，如处理常见的客户问题、引导用户完成购买等。

(2) 工具选择：选择 Dialogflow 工具进行自然语言处理，结合 GPT 模型生成对话逻辑代码。

(3) 系统设计：设计聊天机器人的系统架构，包括用户接口、对话管理、数据库连接等。

(4) 生成代码：通过 AI 工具生成基础对话代码并进行人工调整和优化。

(5) 测试与优化：使用自动化测试工具进行对话测试，优化响应时间和准确性。

3. 调整开发流程和团队角色

- ❑ 调整开发流程：在试点成功的基础上，开发团队需要调整整体开发流程，将 AI 工具融入需求分析、系统设计、代码编写、测试和部署的各个环节。例如，在需求分析阶段可以使用 AI 协助提取并分析用户需求。
- ❑ 重构团队角色：引入 AI 技术后，团队角色和职责也需要相应调整。传统

的开发人员角色可能转变为 AI 工具操作员、数据分析师、系统架构师等。

4. 全面推广和持续优化

- ❑ **全面推广**：在经过试点和流程调整后，开发团队可以将 AI 技术全面推广到所有项目中。例如，电子商务平台可以将 AI 技术应用到用户推荐系统、动态定价系统、库存管理系统等多个模块。
- ❑ **持续优化**：AI 工具迭代速度很快，开发团队应定期评估 AI 工具的效果并进行更新优化。

从传统项目到 AI 项目的转变是一个渐进且系统的过程，通过识别适用场景、引入 AI 工具进行试点、调整开发流程和团队角色、全面推广和持续优化，开发团队可以有效地实现这一转变，并且不断学习和适应新的 AI 技术，在快速变化的技术环境中保持竞争力。

1.6 AI 时代的开发思维和工具

在 AI 时代，编程不仅仅是编写代码，而是思考如何高效地利用 AI 工具和框架来实现目标。这种转变不仅改变了开发者工作方式，也重新定义了编程的思维模式。

1.6.1 AI 编程的基本思维方式

传统编程强调对编程语言的掌握，而 AI 编程则更强调对 AI 工具和技术的理解与应用。以下是几种 AI 编程的基本思维方式。

- ❑ **需求驱动**：与手工编写代码相比，AI 编程开发者更多地是通过描述需求让 AI 生成代码。这要求开发者具备明确的需求表达能力，能够清晰地描述要实现的功能。
- ❑ **数据导向**：AI 编程依赖于大量的数据，开发者需要善于收集、清洗和处理数据，以便训练 AI 模型。这种数据导向思维要求开发者具备一定的数据分析能力。
- ❑ **反馈迭代**：AI 模型需要不断训练和优化，开发者需要通过持续的反馈和迭

代来提高模型的准确性和效率。这种思维方式强调快速试验和持续改进。

1.6.2 什么是框架思维

框架思维是一种系统化的开发思维方式，强调从整体上统筹和设计系统，同时充分利用已有的框架和工具，实现高效、模块化和可扩展的解决方案。框架思维不仅仅指使用现有的框架进行程序设计，更是一种全局视野和统筹能力的体现，注重规划和架构的合理性，确保系统的灵活性和可维护性。

1. 统筹能力

框架思维的一个重要方面是统筹能力。开发者需要从全局出发，统筹系统的各个部分，确保各模块之间的协调与统一。这种思维方式要求开发者具备宏观视野，能够从整体上把握系统的结构和功能。

例如在开发一个大型企业管理系统时，需要统筹考虑用户管理、权限控制、数据存储、业务逻辑等各个方面，确保每个模块都能无缝集成。开发者需要设计出合理的系统架构，使得各个模块既可以独立开发，又能相互协作。

2. 思维方式

框架思维包括一种特定的思维方式，强调模块化设计、组件重用和配置驱动。这种思维方式使开发者在设计和实现系统时更加注重结构的合理性与功能的可扩展性。

- 模块化设计：将系统拆分为多个独立的模块，每个模块完成特定的功能。这种方式不仅便于开发和维护，还可以提高系统的灵活性。
- 组件重用：尽可能重用已有的组件和工具，避免重复劳动。这不仅提高了开发效率，还减少了代码中的错误。
- 配置驱动：通过配置文件和参数化设计，简化系统的定制和扩展，使得系统在不修改代码的情况下，也能轻松适应不同的需求。

3. 统筹能力与思维方式相结合

框架思维的核心在于将统筹能力与模块化思维方式结合起来，形成一个高效的开发流程。这种结合使得开发者能够在复杂的系统开发中保持清晰的思路，从

而提高开发效率和系统质量。

案例：构建一个电子商务平台

以下是一个利用框架思维构建电子商务平台的案例。

1) 统筹规划

- ❑ 需求分析：确定系统需要实现的核心功能，如用户注册与登录、商品管理、订单处理、支付系统等。
- ❑ 系统架构设计：设计系统的总体架构，包括前端、后端、数据库和第三方服务的集成。确保各模块的接口和数据流一致。

2) 模块化设计

- ❑ 用户管理模块：处理用户的注册、登录、权限控制等功能。
- ❑ 商品管理模块：管理商品的添加、删除、修改、分类等功能。
- ❑ 订单处理模块：处理用户的订单生成、支付、发货等功能。

3) 组件重用

- ❑ 前端框架：使用 React 或 Vue.js 构建响应式用户界面。
- ❑ 后端框架：使用 Django 或 Spring Boot 处理服务器端逻辑。

□ 支付集成：利用 Stripe 或 PayPal 的 API 进行支付集成。

4) 配置驱动

□ 配置管理：通过配置文件管理系统不同环境（如开发、测试、生产）下的数据库连接、API 密钥等信息。

□ 参数化设置：使用配置文件调整系统的参数，如分页大小、缓存策略等，无须修改代码。

框架思维不仅是使用现有框架进行程序设计，更是一种宏观的统筹能力和系统化的思维方式。它强调在开发过程中，从整体上把握系统的结构和功能，利用模块化设计、组件重用和配置驱动，实现高效、灵活和可扩展的开发解决方案。这种思维方式使开发者能够用更低的时间成本和更高的效率去应对各种需求变化。

1.6.3 框架思维与 AI 开发的关系

在 AI 时代，开发的重心已经从手动编写函数和代码等基础工作，转向更高层次的系统设计和架构规划。这一变化使得框架思维在 AI 开发中显得尤为重要，框架思维能帮助开发者更好地利用 AI 技术高效地进行软件开发和设计。

AI 可以当成开发者的一名员工，开发者的任务不再是亲自编写每一行代码，而是发号施令，指导 AI 完成具体的工作。在传统的编程模式中，开发者需要花费大量时间编写和调试代码，关注每个函数的实现细节，现在，许多基础的编程工作都可以由 AI 自动完成。开发者可以将精力放在系统的整体设计和架构规划上，基础劳动力被解放的同时就需要一种全新的思维方式，即框架思维。

框架思维强调模块化设计、组件重用和配置驱动，使开发者可以从宏观上把握

系统的整体结构，而不是拘泥于具体的代码实现。这种思维方式不仅提高了开发效率，还使系统更加灵活和易于维护。可以将 AI 时代的编程变革类比工业革命时期的生产方式变革。在工业革命之前，制造业依靠手工劳动，每件产品都需要工匠“从头到尾”亲自制作，效率低下且质量难以保证。而工业革命引入了流水线生产模式，通过模块化的分工和标准化的组件，大大提高了生产效率和产品质量。

同样，在 AI 时代，编程从手工编码模式转向利用 AI 工具进行自动化开发模式。开发者不需要亲自编写每行代码，而是通过设计模块化的系统架构，利用 AI 生成和优化代码。这种转变使得开发过程更高效、更可靠，同时也促使开发者在更高层次上思考和设计系统。这种转变使开发者从基础的编码工作中解放出来，专注于更高层次的系统设计和优化。通过模块化设计和配置驱动，开发者可以更加高效地构建和部署 AI 系统，充分利用 AI 技术的优势，实现智能化和自动化的开发流程。在 AI 时代，掌握和应用框架思维，是开发者保持竞争力的关键。

1.6.4 如何构建一个简单的 Web 应用框架

下面通过一个实际案例来展示如何构建一个简单的 Web 应用框架，并通过与 ChatGPT 沟通完成代码的编写。

案例目标

构建一个简单的博客系统，包括以下功能：

- ❑ 用户注册和登录；
- ❑ 发布文章；
- ❑ 查看文章列表；
- ❑ 查看文章详情。

操作步骤如下。

(1) 规划项目结构。这是框架思维的体现，通过清晰的结构，使得项目的各个模块分工明确，便于后续开发和维护。

```
blog/  
├── app.py  
├── config.py
```

```

├── requirements.txt
├── static/
├── templates/
└── models.py

```

(2) 与 ChatGPT 沟通，以便配置 Flask 应用。使用 Flask 框架搭建 Web 应用。在 `app.py` 中配置 Flask 应用并设置数据库链接。可以通过与 ChatGPT 沟通生成代码。

提示词：

请帮我编写一个 Flask 应用的基础代码，包括配置数据库链接。

ChatGPT 回复：

```

from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///app.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

if __name__ == '__main__':
    app.run(debug=True)

```

(3) 定义数据模型。在 `models.py` 中定义用户和文章的数据模型。通过与 ChatGPT 沟通生成代码。

提示词：

请编写 Flask SQLAlchemy 模型代码，包括用户模型和文章模型。

ChatGPT 回复：

```

from app import db

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), unique=True,
        nullable=False)
    password = db.Column(db.String(128), nullable=False)

class Post(db.Model):

```

```
id = db.Column(db.Integer, primary_key=True)
title = db.Column(db.String(128), nullable=False)
content = db.Column(db.Text, nullable=False)
user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
user = db.relationship('User', backref=db.backref('posts',
lazy=True))
```

(4) 在 `app.py` 中实现用户注册和登录、发布文章、查看文章列表和详情的路由。通过与 ChatGPT 沟通生成代码。

提示词：

请帮我实现 Flask 应用的基本路由，包括用户注册、登录，发布文章、查看文章详情。

ChatGPT 回复：

```
from flask import render_template, redirect, url_for, request
from models import User, Post
from app import app, db

@app.route('/')
def index():
    posts = Post.query.all()
    return render_template('index.html', posts=posts)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User(username=username, password=password)
        db.session.add(user)
        db.session.commit()
        return redirect(url_for('index'))
    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
```

```
        user = User.query.filter_by(username=username, password=
password).first()
        if user:
            return redirect(url_for('index'))
        else:
            return "Login Failed"
    return render_template('login.html')

@app.route('/post', methods=['GET', 'POST'])
def post():
    if request.method == 'POST':
        title = request.form['title']
        content = request.form['content']
        # 假设用户 ID 为 1
        post = Post(title=title, content=content, user_id=1)
        db.session.add(post)
        db.session.commit()
        return redirect(url_for('index'))
    return render_template('post.html')

@app.route('/post/<int:post_id>')
def post_detail(post_id):
    post = Post.query.get(post_id)
    return render_template('post_detail.html', post=post)
```

(5) 创建数据库。在项目的根目录下创建一个 Python 脚本 `create_db.py`，用于创建 SQLite 数据库。

提示词：

请帮我编写一个 Python 脚本，用于创建 SQLite 数据库并初始化表结构。

ChatGPT 回复：

```
from app import db
from models import User, Post
# 创建所有表
db.create_all()
# 添加一些初始数据
admin = User(username='admin', password='admin')
db.session.add(admin)
db.session.commit()
```

```
print("Database and tables created successfully.")
```

运行上面的脚本创建数据库并初始化表结构：

```
python create_db.py
```

通过上述步骤，一个简单的 Web 应用框架就构建完成了。这个案例展示了如何通过框架思维与 ChatGPT 沟通，快速实现一个功能齐全的 Web 应用。在这个过程中，开发者的主要任务是规划和设计，而具体的编码工作由 AI 辅助完成，大大提高了开发效率。

1.7 AI 工具在 Python 中的应用

市场上有多款 AI 编程工具，这些工具在提高开发效率、优化代码质量和简化复杂任务方面发挥了重要作用。Python 作为一种广泛应用于 AI 开发的语言，因其简洁、易用和强大的库支持，成为 AI 开发者的首选。利用 AI 工具编写 Python 代码，不仅能提高开发速度，还能帮助开发者更好地理解和维护代码。

AI 编程工具一般具备以下功能。

- ❑ 代码续写：基于上下文智能补全代码，减少手动编写的工作量，提高开发效率。
- ❑ 代码生成：根据描述自动生成完整的函数或类，实现从需求到代码的快速转换。
- ❑ 代码优化：提供代码优化建议，包括性能提升、代码简化和最佳实践等。
- ❑ 错误检测和修复：自动检测代码中的错误和潜在问题并给出修复建议。
- ❑ 文档生成：根据代码自动生成文档，提高代码的可维护性和可读性。
- ❑ 测试生成：自动生成单元测试和集成测试，确保代码的稳定性和可靠性。
- ❑ 重构支持：提供安全的代码重构工具，帮助开发者重构代码而不会引入新错误。

有些工具已经集成到流行的开发环境中，如 PyCharm 和 Visual Studio Code (简称为 VSCode)。后面会以这两个开发工具为例，展示如何利用它们内置的 AI

功能进行 Python 开发。

1.7.1 AI 编程工具简介与使用场景

本节介绍市面上一些 AI 编程工具，并提供它们的官网链接。

1. 国内的 AI 编程工具

1) 腾讯云 AI 代码助手

□ 简介：腾讯云 AI 代码助手基于腾讯混元代码模型，支持多种编程语言和主流 IDE，提供代码补全、Bug 诊断、生成测试等功能，帮助提升研发效率。

□ 官网地址为 <https://cloud.tencent.com/product/acc>。

2) Fitten Code

□ 简介：Fitten Code 是一款 AI 编程助手，提供代码自动补全、生成、翻译、注释等功能，支持多种编程语言和 IDE。

□ 官网地址为 <https://code.fittentech.com/>。

3) 代码小浣熊 (Raccoon)。

□ 简介：代码小浣熊由商汤科技推出，支持代码生成、编辑、解释等功能，覆盖软件研发的多个环节。

□ 官网地址为 <https://raccoon.sensetime.com/>。

4) DeepSeek

□ 简介：DeepSeek 是一款基于 Transformer 结构的语言模型，支持自然语言处理任务，广泛应用于机器翻译、智能客服等领域。

□ 官网地址为 <https://www.deepseek.com/zh>。

5) 豆包 MarsCode

□ 简介：豆包 MarsCode 是由字节跳动推出的 AI 编程助手，具备智能代码补全和代码生成等功能，支持多种编程语言和主流 IDE。

□ 官网地址为 <https://www.marscode.cn/>。

6) 通义灵码

□ 简介：通义灵码是阿里巴巴团队推出的智能编程辅助工具，基于通义大模

型，提供代码续写和自然语言生成代码等功能。

❑ 官网地址为 <https://tongyi.aliyun.com/lingma>。

2. 国外的AI编程工具

1) GitHub Copilot

❑ 简介: GitHub Copilot 是 GitHub 推出的 AI 编程助手,支持多种语言和 IDE,提供代码自动生成和补全等功能。

❑ 官网地址为 <https://copilot.github.com/>。

2) CodeWhisperer

❑ 简介: CodeWhisperer 是亚马逊 (AWS) 团队推出的 AI 编程软件,提供代码生成器和实时代码建议。

❑ 官网地址为 <https://aws.amazon.com/codewhisperer/>。

3) Project IDX

❑ 简介: Project IDX 是 Google 推出的由 AI 驱动的云端全栈开发平台和代码编辑器。

❑ 官网地址为 <https://cloud.google.com/project-idx>。

4) Codeium

❑ 简介: Codeium 是一款由 AI 驱动的编程助手工具,提供代码建议、重构提示和代码解释等功能。

❑ 官网地址为 <https://www.codeium.com/>。

5) TabNine

❑ 简介: TabNine 是一款代码自动补全工具,支持多种编程语言和 IDE,通过深度学习模型为开发者提供智能代码建议。

❑ 官网地址为 <https://www.tabnine.com/>。

6) Kite

❑ 简介: Kite 是一款由 AI 驱动的代码补全工具,支持 Python 等多种语言,提供实时代码建议和文档提示,帮助开发者提高编程效率。

❑ 官网地址为 <https://www.kite.com/>。

7) DeepCode

❑ 简介: DeepCode 利用机器学习和代码分析技术,为开发者提供智能的代

码审查和错误检测功能，支持多种语言和 IDE。

❑ 官网地址为 <https://www.deepcode.ai/>。

8) Codota

❑ 简介：Codota 是一款 AI 驱动的代码补全工具，支持 Java 和 Python 等多种语言，通过学习开源代码库，为开发者提供智能代码建议。

❑ 官网地址为 <https://www.codota.com/>。

还有很多类似的工具，这里就不一一介绍了。这些 AI 编程工具有一些集成到了 IDE 中，有的没有，在后面的章节中将使用开发工具 PyCharm 或 VSCode 以及 ChatGPT 和 Fitten (集成) 进行介绍，这些工具各有优势，一通百通。ChatGPT 编写代码和理解需求的能力较强，虽然没有集成 IDE，但是将其作为辅助编程工具是不错的选择。

1.7.2 如何选择 AI 代码生成器

在选择 AI 代码生成器时，开发者需要考虑多方面的因素，以确保所选工具能够真正提高开发效率和代码质量。以下是一些选择建议。

- ❑ 功能全面性：选择 AI 代码生成器时首先要看其功能是否全面。一个好的代码生成器应该具有代码自动补全、代码生成、错误检测与修复、代码优化、文档生成、提升代码可读性等功能。
- ❑ 集成能力：选择与常用开发环境集成良好的工具，可以大大提升开发体验。当前，PyCharm 和 VSCode 是两款非常流行的 IDE，许多 AI 代码生成器都支持它们。因此应确保选择的工具能够与自己的开发环境无缝集成，以便更高效地进行编码工作。
- ❑ 性能与效率：选择一个能够快速响应、生成高质量代码的工具是至关重要的。性能不佳的工具不仅会拖慢开发进程，而且出错率高，要花更多的时间调试和修改，时间成本大大增加。目前 ChatGPT 的代码编写能力相比其他模型要强一些。
- ❑ 成本：AI 代码生成器的成本也是选择时需要考虑的一个因素。不同工具的定价策略可能差别很大，有些提供免费版本，有些则需要订阅服务。可以根据自身预算和需求，选择性价比最高的工具。

通过以上标准，开发者可以更好地选择适合自己的 AI 代码生成器，从而在日常开发工作中充分利用 AI 技术提升开发效率和代码质量。

1.7.3 实例 1：让 AI 生成数据分析代码

本节详细介绍如何使用 AI 代码生成器进行 Python 开发工作。通过实例展示 ChatGPT 和 Fitten Code 这两款工具在 AI 开发中的具体应用，并提供提示词模板，帮助开发者更好地利用这些工具。

ChatGPT 是一款由 OpenAI 开发的功能强大的自然语言处理模型，可以通过对话生成高质量的代码。下面通过一个例子来展示如何使用 ChatGPT 生成一个简单的 Python 脚本并进行数据处理工作。

任务描述：生成一个 Python 脚本，从 CSV 文件中读取数据，进行简单的数据统计分析。

CSV 文件的内容如图 1-1 所示。

接下来跟 ChatGPT 对话，提示词模板如下：

帮我写一个 Python 代码，要求：根据健康保险客户的特征数据，预测医疗费报销支出。附件中的 `insurance.csv` 是数据集，格式如下：

- 年龄：主要受益人年龄
- 性别：保险客户的性别，女、男
- BMI：身体质量指数（也称为体重指数），即体重除以身高的平方，理想范围为 18.5~24.9
- 儿童：健康保险覆盖的儿童人数/受抚养者人数
- 吸烟者：吸烟
- 地区：受益人在美国的居住地，东北、东南、西南、西北
- 收费：由健康保险支付的个人医疗费用

#	A	B	C	D	E	F	G
1	age	sex	bmi	children	smoker	region	expenses
2		19 female	27.9	0	yes	southwest	16884.92
3		18 male	33.8	1	no	southeast	1725.55
4		28 male	33	3	no	southeast	4449.46
5		33 male	22.7	0	no	northwest	21984.47
6		32 male	28.9	0	no	northwest	3866.86
7		31 female	25.7	0	no	southeast	3756.62
8		46 female	33.4	1	no	southeast	8240.59
9		37 female	27.7	3	no	northwest	7281.51
10		37 male	29.8	2	no	northeast	6406.41
11		60 female	25.8	0	no	northwest	28923.14
12		25 male	26.2	0	no	northeast	2721.32
13		62 female	26.3	0	yes	southeast	27808.73
14		23 male	34.4	0	no	southwest	1826.84
15		56 female	39.8	0	no	southeast	11090.72
16		27 male	42.1	0	yes	southeast	39611.76
17		19 male	24.6	1	no	southwest	1837.24
18		52 female	30.8	1	no	northeast	10797.34
19		23 male	23.8	0	no	northeast	2395.17
20		56 male	40.3	0	no	southwest	10602.39
21		30 male	35.3	0	yes	southwest	36837.47
22		60 female	36	0	no	northeast	13228.85

图 1-1 CSV 文件内容

ChatGPT 生成的代码如下：

我们将使用 Python 编写一个脚本，利用数据集来预测医疗费报销支出。我们将使用以下步骤：

1. 加载数据集
2. 数据预处理
3. 特征工程
4. 构建并训练模型
5. 评估模型

以下是实现上述步骤的 Python 代码：

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
# 1. 加载数据集
data = pd.read_csv('/mnt/data/insurance.csv')

# 2. 数据预处理
# 将分类变量进行 One-Hot 编码
categorical_features = ['sex', 'smoker', 'region']
numerical_features = ['age', 'bmi', 'children']

# 创建预处理步骤的 Pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline([
            ('imputer', SimpleImputer(strategy='mean')),
            ('scaler', StandardScaler())
        ]), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

# 3. 特征工程
# 特征和目标变量分离
X = data.drop('charges', axis=1)
y = data['charges']

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

# 4. 构建并训练模型
# 创建 Pipeline, 包含预处理和模型
model_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', LinearRegression())
])

# 训练模型
model_pipeline.fit(X_train, y_train)
```

```
# 5. 评估模型
# 预测
y_pred = model_pipeline.predict(X_test)

# 计算均方误差和 R2 分数
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R2 Score: {r2}')
```

这段代码使用了 `pandas` 来处理数据集，使用 `scikit-learn` 进行数据预处理和模型构建，使用线性回归模型作为示例模型。你可以根据需要选择其他模型并进行调整。

以上代码假设你已经安装了相关的 Python 库，如果没有安装，可以使用以下命令进行安装：

```
pip install pandas scikit-learn
```

数据表中的列名是英文，而提示词表述为中文，ChatGPT 同样能够识别。

 **提示：**通过调整提示词，可以生成不同的代码片段。例如，可以增加数据可视

化部分，生成使用 `matplotlib` 绘制图表的代码。

1.7.4 实例 2：使用 Fitten Code 自动补全和生成代码

Fitten Code 是一款 AI 编程助手，提供代码自动补全、生成、翻译和注释等功能，支持多种编程语言和 IDE。下面将 Fitten Code 插件安装到 PyCharm 中，在 PyCharm 的菜单栏中选择“文件”|“设置”命令，如图 1-2 所示。



图 1-2 PyCharm 菜单栏

在弹出的“设置”对话框中查找 Fitten Code 后安装即可，如图 1-3 所示。

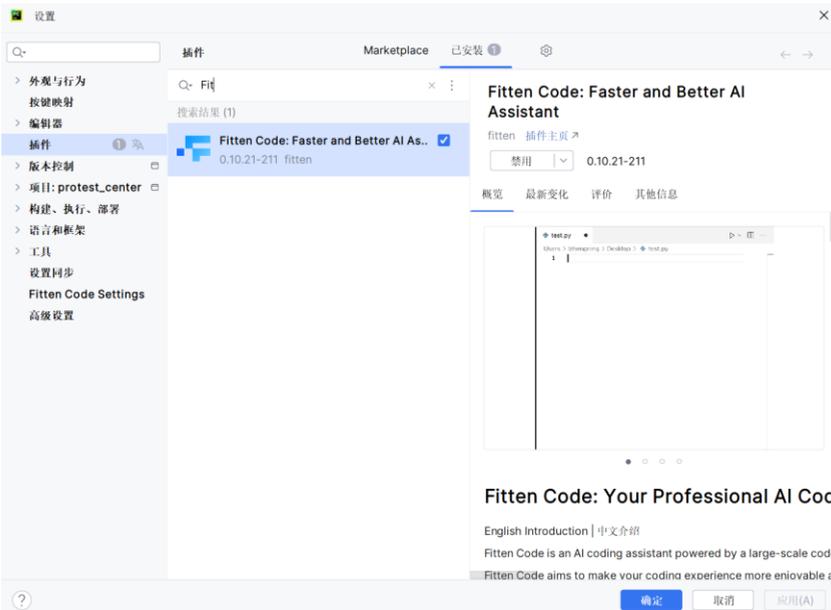


图 1-3 安装 PyCharm 插件

安装完成后重启 PyCharm 即可，在左侧可以看到 Fitten 的小图标，Fitten

界面如图 1-4 所示。

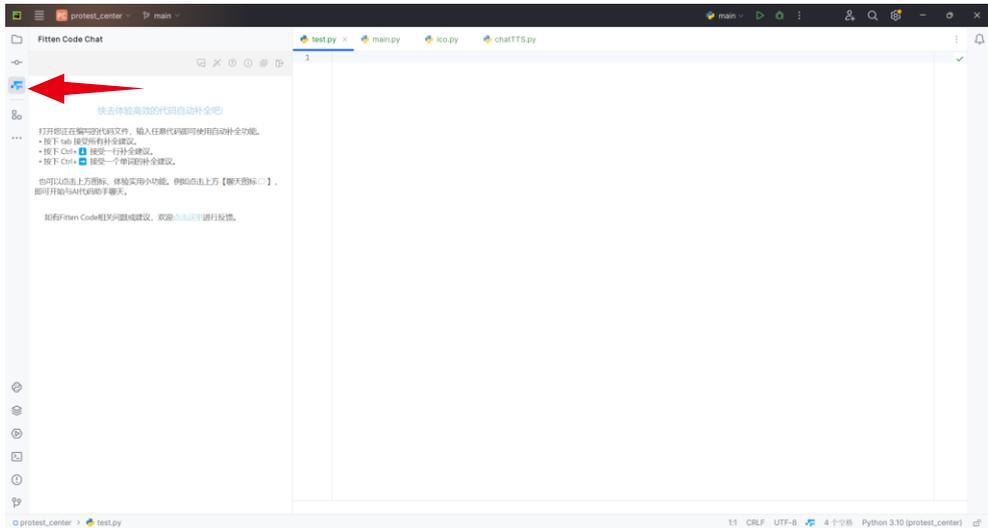


图 1-4 Fitten 插件界面

VSCode 的安装也同理，这里不再赘述。

单击魔法棒按钮，即可开始生成代码，如图 1-5 所示。



图 1-5 Fitten Code 插件界面

现在让 Fitten Code 生成一个实例，提示词如下：

编写一个 Python 代码，使用 Flask 框架初始化一个 Web 应用程序

Fitten Code 就会生成一段代码，单击左下角的 Insert 按钮，代码就会插入右侧的文本框中，如图 1-6 所示。

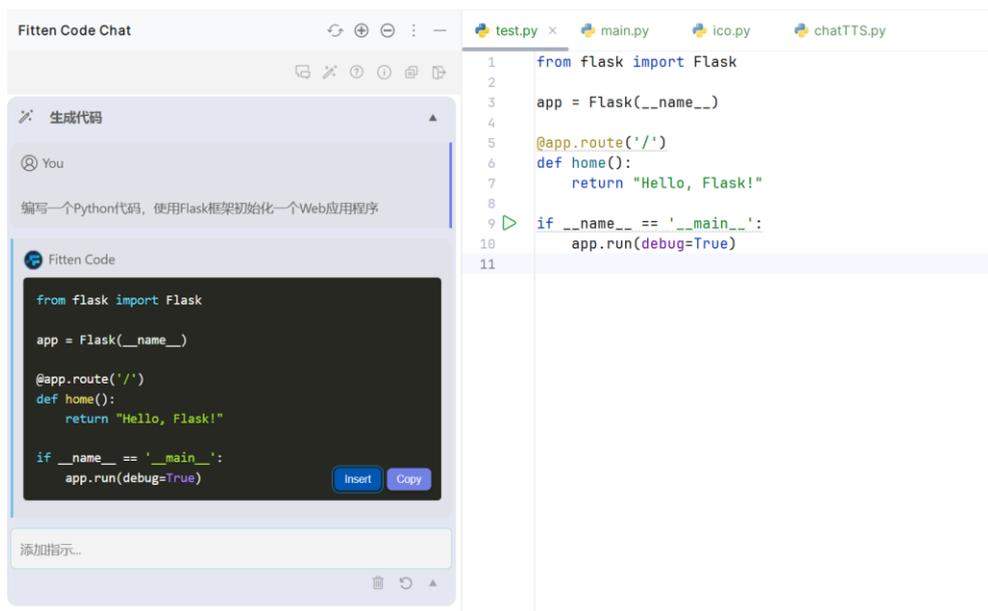


图 1-6 Fitten Code 输入界面

假设现在输入一段代码，输入一半时就会有代码补全的提示，如图 1-7 所示。

Fitten Code 猜测开发者可能会将文本赋值给变量，因此给出补全提示：

```
String = "Hello, Flask!"
```

按 Tab 键，代码即自动补全。

注释也可以补全，不仅如此，还可以根据注释自动生成代码，如图 1-8 所示。

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def home():
7     String = "Hello, Flask!"
8     return "Hello, Flask!"
9
10 if __name__ == '__main__':
11     app.run(debug=True)
12
```

图 1-7 Fitten Code 代码补全提示

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def home():
7     # 循环10000次
8     for i in range(10000):
9         return "Hello, Flask!"
10 if __name__ == '__main__':
11     app.run(debug=True)
12
```

图 1-8 Fitten Code 根据注释自动写代码

从图 1-8 中可以看到，输入注释后，会自动出现 for 的补全代码，按 Tab 键之

后，代码会自动补全。

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    # 循环 10000 次
    for i in range(10000):
        pass
    return "Hello, Flask!"

if __name__ == '__main__':
    app.run(debug=True)
```

如果遇到不懂的问题应该怎么办呢？单击第一个按钮，Fitten Code 就会弹出对话机器人，与其他聊天机器人一样，有技术问题可以向其提问，如图 1-9 所示。



图 1-9 Fitten Code 提问界面

选中代码并右击后会弹出更多的功能选项，如图 1-10 所示。



图 1-10 Fitten Code 菜单

选择“解释代码”命令，Fitten Code 会非常详细地解释该代码的作用，如图 1-11 所示。

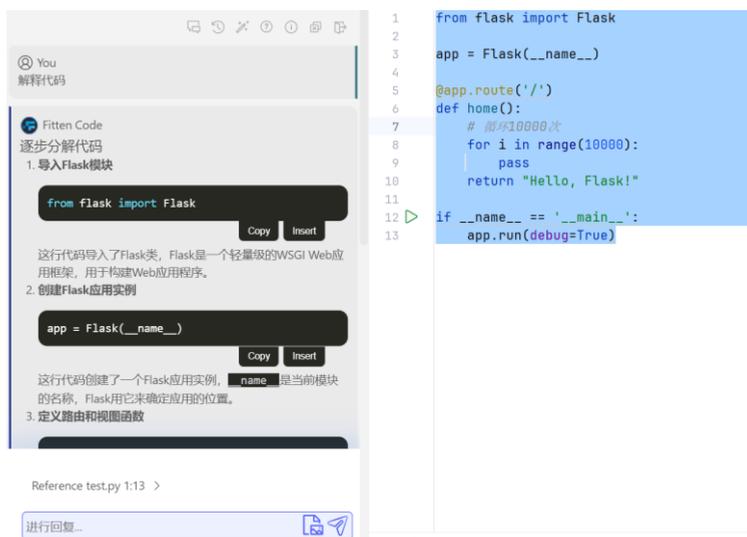


图 1-11 Fitten Code 生成解释代码

选择“查找 Bug”命令，Fitten Code 便能详细告知这段代码存在的风险，如图 1-12 所示。

总结：通过上述两个实例可以看出 AI 代码生成器在 Python 开发中的强大应用。ChatGPT 和 Fitten Code 不仅可以生成高质量的代码，还能通过调整提示词，满足不同的开发需求。开发者可以根据具体任务，灵活使用这些 AI 工具，提高

开发效率和代码质量。

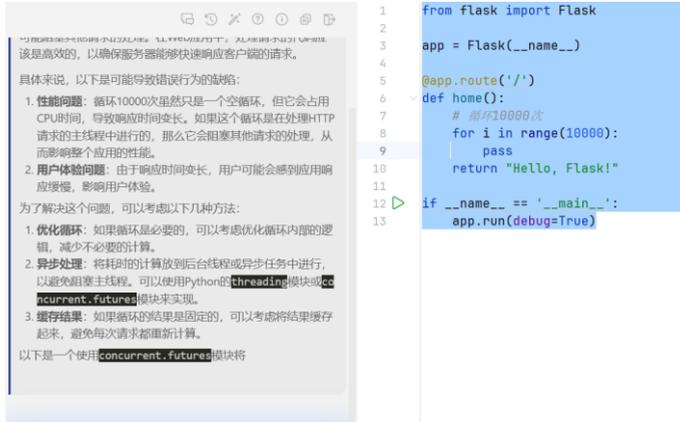


图 1-12 Fitten Code 查找 Bug

1.7.5 实例 3：让 AI 在线运行 Python 代码

本节来体验一下 ChatGPT 在线运行代码的功能，让它运行斐波那契数列函数，如图 1-13 所示。



图 1-13 ChatGPT 运行斐波那契数列函数

用户：

帮我写一个 Python 函数，实现斐波那契数列

ChatGPT 回复：

以下是一个用 Python 实现斐波那契数列的函数。这个函数通过递归方式计算斐波那契数列的第 n 项：

```
def fibonacci_recursive(n):
    if n <= 0:
        return "Input should be a positive integer."
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

# 示例用法
n = 10
print(f"斐波那契数列的第{n}项是：{fibonacci_recursive(n)}")
```

如果要优化性能，可以使用动态规划的方法来实现斐波那契数列，这样可以避免递归带来的大量重复计算：

```
def fibonacci_dynamic(n):
    if n <= 0:
        return "Input should be a positive integer."
    elif n == 1:
        return 0
    elif n == 2:
        return 1

    fib = [0, 1]
    for i in range(2, n):
        fib.append(fib[i-1] + fib[i-2])

    return fib[-1]

# 示例用法
```

```
n = 10
print(f"斐波那契数列的第{n}项是: {fibonacci_dynamic(n)}")
```

上述代码分别展示了递归和动态规划两种实现斐波那契数列的方法。可以根据具体需求选择使用。

用户:

```
帮我执行 fibonacci_dynamic(20)
```

ChatGPT 回复:

```
斐波那契数列的第 20 项是: 4181。
```

接下来再给 ChatGPT 一个难度更高的任务:

```
帮我写一个 Python 函数, 实现圆周率的计算, 开发者可自设置要显示小数点后多少位数
```

ChatGPT 回复:

以下是一个用 Python 实现圆周率计算的函数, 用户可以设置显示小数点后多少位数。这个函数使用了 Nilakantha 级数来计算圆周率:

```
def calculate_pi(precision):
    if precision < 0:
        return "Precision should be a non-negative integer."

    pi = 3
    sign = 1
    i = 2

    while True:
        term = 4 / (i * (i + 1) * (i + 2))
        if round(term, precision + 1) == 0:
            break
        pi += sign * term
        sign *= -1
        i += 2

    return round(pi, precision)
```

```
# 示例用法
```

```
precision = 10
print(f"圆周率显示到小数点后{precision}位数是: {calculate_pi(precision)}")
```

以上代码通过 Nilakantha 级数计算圆周率，并根据用户设置的精度显示结果。用户可以通过修改 `precision` 变量来设置所需的精度。

用户：

```
帮我运行 calculate_pi(100)
```

ChatGPT 回复：

```
def calculate_pi(precision):
    if precision < 0:
        return "Precision should be a non-negative integer."

    pi = 3
    sign = 1
    i = 2

    while True:
        term = 4 / (i * (i + 1) * (i + 2))
        if round(term, precision + 1) == 0:
            break
        pi += sign * term
        sign *= -1
        i += 2

    return round(pi, precision)

# 运行 calculate_pi(100)
pi_100_decimal = calculate_pi(100)
pi_100_decimal
-----
KeyboardInterrupt                                Traceback (most recent call last)
Cell In[6], line 20
     17     return round(pi, precision)
     19 # 运行 calculate_pi(100)
--> 20 pi_100_decimal = calculate_pi(100)
     21 pi_100_decimal
```

```

Cell In[6], line 11, in calculate_pi(precision)
     9 while True:
    10     term = 4 / (i * (i + 1) * (i + 2))
---> 11     if round(term, precision + 1) == 0:
    12         break
    13     pi += sign * term

```

KeyboardInterrupt:

计算圆周率到 100 位小数精度的过程比较复杂，Nilakantha 级数可能需要大量的迭代才能达到这样的精度。为了更高效地计算高精度的圆周率，我们可以使用更快速的算法，比如 Bailey-Borwein-Plouffe (BBP) 公式。

下面是一个使用 BBP 公式计算圆周率到指定精度的小数位的函数：

```

from decimal import Decimal, getcontext

def calculate_pi_bbp(precision):
    # 设置计算精度，加 2 以确保最后结果的精度
    getcontext().prec = precision + 2
    pi = Decimal(0)
    k = 0
    while True:
        new_term = (Decimal(1)/(16k)) * (
            Decimal(4)/(8*k+1) - Decimal(2)/(8*k+4) -
            Decimal(1)/(8*k+5) - Decimal(1)/(8*k+6)
        )
        pi += new_term
        if new_term < Decimal(10) ** (-precision):
            break
        k += 1
    return +pi          # 使用一元加号操作符返回舍入后的 pi 值

# 示例用法
precision = 100
calculate_pi_bbp(precision)

```

这段代码使用 BBP 公式计算圆周率并返回指定精度的小数位。我们可以运行这段代码来

获取 100 位小数精度的圆周率。

这一次它在运行时就出现错误了，不过随即又修复了错误，我们再运行一次程序：

```
运行 calculate_pi_bbp(100)
```

ChatGPT 回复：

圆周率显示到小数点后 100 位数是：

```
3.14159265358979323846264338327950288419716939937510582097494459  
230781640628620899862803482534211706798
```

这次的运行结果是正确的，ChatGPT 自动修复错误并完成了任务。

如果是像类似 Fitten 这种集成在 IDE 里面的，测试错误可以直接查错。