第 1 章

初识智能体

智能体（Agent）是人工智能（Artificial Intelligence，AI）领域的核心概念之一，也是构建智能化系统的重要基础。本章主要介绍智能体的基本概念，阐述智能体的定义、特征及其在人工智能中的应用，探讨智能体的分类与典型结构，并概述智能体技术的发展历程及其在多领域中的实际应用。此外，本章还将简要介绍多智能体系统的基本概念及其协作机制。

本章重点和难点：理解智能体的核心特性（如自主性、反应性、社会性等），掌握智能体的基本结构和工作原理，了解智能体与大模型的关系，并能够分析多智能体系统的应用场景协作与通信机制。

1.1 智能体是什么

AI智能体（AI Agent）是什么？在科技飞速发展的当下，人工智能正以前所未有的态势深刻变革着我们的生活。而在人工智能的庞大体系中，AI智能体宛如一颗璀璨新星，逐渐崭露头角，成为推动科技进步与产业变革的关键力量。它不仅悄然改变着我们与机器的交互方式，更在诸多领域展现出巨大的应用潜力与创新活力。那么，AI智能体究竟是什么？它又具备哪些独特的功能和优势呢？接下来，让我们一同深入探寻AI智能体的神秘世界。

在人工智能领域，“智能体（Agent）”这个词越来越火。无论是AutoGPT、AgentGPT、OpenAI的GPTs，还是DeepSeek、豆包的智能助手，它们都在推动AI从“工具”向“自主智能体”进化。

1. 定义与概念

什么是智能体？简单来说，智能体就是“能自主执行任务的AI”。传统AI（如ChatGPT）主要依靠用户输入指令，而智能体可以自主思考、决策，并执行复杂任务，就像一个AI助手，能够独立完成多步操作。用一句话总结就是：智能体=AI+目标驱动+自主行动。

以日常生活中的智能语音助手为例，当你向它发出“查询明天天气”的指令时，它能迅速感知你的语音信息，通过对语音的识别和语义理解，在庞大的天气数据中进行搜索，然后作出决策，最后以清晰的语音告知你明天的天气状况。这一系列过程完美诠释了AI智能体的自主感知、决策与执行能力。

2. 工作原理

AI智能体的运作基于PEAS模型，即性能（Performance）、环境（Environment）、执行器（Actuators）和传感器（Sensors）。传感器负责收集来自环境的各种信息，像视觉图像、语音信号、温度数据等，将这些信息传递给智能体的“大脑”。智能体依据机器学习、深度学习、自然语言处理等技术，对这些信息进行深入分析与理解，从而作出决策。最后，通过执行器将决策转换为实际行动，对环境产生影响。

在自动驾驶领域，汽车上的摄像头、雷达等传感器时刻感知路况信息，包括道路标志、车辆位置、行人动态等。智能体根据这些信息，运用复杂的算法规划行驶路线、控制车速和方向，通过执行器操控方向盘、刹车和油门等部件，确保车辆安全、高效地行驶。

1.2　AI智能体的类型

AI智能体可以根据其功能、架构和应用场景分为多种类型。本节将介绍常见的AI智能体分类。

1. 按功能与能力分类

（1）简单反射型（Reflex Agents）：基于“条件–动作”规则（if-then），直接根据当前输入作出反应，无记忆能力。示例：自动化客服关键词回复、温控系统。

（2）基于模型的反射型（Model-Based Reflex Agents）：通过内部模型跟踪环境状态，能处理部分不确定性。示例：自动驾驶车辆实时感知路况。

（3）目标驱动型（Goal-Based Agents）：通过规划实现特定目标，需评估不同行动的后果。示例：物流路径规划、游戏AI（如AlphaGo）。

（4）效用驱动型（Utility-Based Agents）：在多个目标中权衡，选择最大化“效用”（收益/成本）的行动。示例：金融交易AI、资源分配系统。

（5）学习型（Learning Agents）：通过机器学习（如强化学习）动态优化行为。示例：推荐系统（Netflix/Spotify）、机器人自适应控制。

2. 按自主性分类

（1）自主智能体（Autonomous Agents）：独立决策，无须持续人工干预。示例：火星探测车、工业巡检机器人。

（2）半自主智能体（Semi-Autonomous Agents）：部分依赖人类输入。示例：医疗诊断AI（医生辅助决策）。

（3）从属智能体（Subordinate Agents）：完全受控于用户或其他系统。示例：语音助手（如Siri执行用户指令）。

3. 按协作方式分类

（1）单智能体（Single Agent）：独立完成任务。示例：个人语音助手。

（2）多智能体系统（Multi-Agent Systems，MAS）：多个智能体协作或竞争。示例：交通信号协同控制、多机器人协作。

4. 按应用领域分类

（1）物理智能体（Physical Agents）：通过传感器/执行器与环境交互。示例：扫地机器人（如Roomba）、无人机。

（2）虚拟智能体（Virtual Agents）：以纯软件形式存在。示例：聊天机器人（如ChatGPT）、游戏NPC。

（3）混合智能体（Hybrid Agents）：结合物理与虚拟能力。示例：数字孪生（Digital Twin）系统。

5. 按决策方式分类

（1）反应式（Reactive）：实时响应，无长期规划。示例：工业流水线分拣机器人。

（2）慎思式（Deliberative）：基于逻辑推理或符号AI进行规划。示例：IBM Watson（医疗知识推理）。

（3）混合式（Hybrid）：结合反应速度与长期规划。示例：人形机器人（如波士顿动力Atlas）。

6. 其他特殊类型

（1）生成式智能体（Generative Agents）：能生成新内容（文本、图像、代码等）。示例：GPT-4、MidJourney。

（2）认知智能体（Cognitive Agents）：模拟人类思维（如情感、意识）。示例：强AI/通用人工智能（AGI）。

随着技术进步，AI智能体的类型和边界不断扩展（例如具身智能体、AI Agent+大语言模型结合等），而在实际系统中常常混合多种类型以满足复杂的业务需求。

1.3　AI智能体的功能

AI智能体是一种能够感知环境并通过行动来实现特定目标的人工智能系统。它具有多种功能，本节将介绍AI Agent的一些常见功能。

1. 感知功能

- 环境感知：AI智能体可以通过各种传感器（如摄像头、麦克风、温度传感器等）来感知周围环境的信息。例如，自动驾驶汽车的AI智能体通过车载摄像头感知道路状况、交通标志和周围车辆的位置，通过雷达感知车辆与障碍物的距离。
- 信息提取与理解：它能够对感知到的原始数据进行处理和分析，提取有用的信息，并理解其含义。例如，语音助手的AI智能体可以将用户的声音信号转换为文字，并理解用户的意图。

2. 决策功能

- 基于规则的决策：AI智能体可以根据预设的规则和逻辑来作出决策。例如，在一个简单的智能客服系统中，当用户询问产品的价格时，AI智能体会根据预设的规则直接给出产品的价格信息。

- 基于学习的决策：通过机器学习算法，AI智能体可以从大量数据中学习规律和模式，并据此作出决策。例如，推荐系统的AI智能体会根据用户的历史行为数据（如购买记录、浏览历史等）来学习用户的偏好，从而为用户推荐可能感兴趣的商品或内容。
- 基于优化的决策：对于一些需要在多个选项中选择最优解的问题，AI智能体可以利用优化算法来作出决策。例如，在物流配送中，AI智能体可以通过优化算法规划最优的配送路线，以减少配送时间和成本。

3. 行动功能

- 物理行动：一些AI智能体可以控制物理设备来执行行动。例如，工业机器人可以根据AI智能体的指令进行零件的抓取、组装等操作；无人机的AI智能体可以控制无人机的飞行姿态、速度和方向，完成航拍、巡检等任务。
- 信息交互行动：AI智能体可以通过网络与用户或其他系统进行信息交互。例如，聊天机器人可以与用户进行对话，解答用户的问题；智能家居系统中的AI智能体可以与用户的手机或其他设备进行通信，接收用户的指令并反馈设备的状态。

4. 学习功能

- 监督学习：AI智能体通过学习标注好的数据来建立模型，从而能够对新的数据进行预测或分类。例如，在图像识别中，通过使用大量标注了类别（如猫、狗、汽车等）的图像数据来训练模型，使模型能够识别新的图像中的物体。
- 无监督学习：AI智能体在没有标注数据的情况下，通过发现数据中的结构和规律来进行学习。例如，在聚类分析中，AI智能体可以将相似的数据点聚集在一起，从而发现数据中的类别或模式。
- 强化学习：AI智能体通过与环境的交互来学习，根据环境的反馈（奖励或惩罚）来调整自己的行为策略，以达到最大化累积奖励的目的。例如，在游戏AI中，AI智能体通过不断尝试不同的游戏策略，并根据游戏的得分来调整策略，从而学会如何在游戏中获胜。

5. 交互功能

- 人机交互：AI智能体能够与人类用户进行自然、有效的交互。例如，语音助手可以通过语音识别和语音合成技术与用户进行对话，理解用户的指令并提供相应的服务；智能办公软件中的AI智能体可以通过图形用户界面与用户进行交互，帮助用户完成文档编辑、数据分析等任务。
- 智能体间交互：多个AI智能体之间也可以进行交互和协作。例如，在多智能体机器人系统中，各个机器人可以通过通信和协调来完成复杂的任务，如协同搬运货物、搜索救援等；在分布式智能系统中，不同的AI智能体可以共享信息、协同工作，以提高系统的整体性能。

6. 自适应功能

- 环境自适应：AI智能体能够根据环境的变化自动调整自己的行为和策略。例如，在智能交通系统中，当交通流量发生变化时，AI智能体可以自动调整交通信号灯的时长，以缓解交通拥堵；在智能电网中，AI智能体可以根据电力负荷的变化自动调整发电设备的输出功率。
- 用户自适应：AI智能体可以根据用户的个性化需求和偏好进行自适应调整。例如，智能推荐系统会根据用户的行为变化和反馈信息，不断更新用户的兴趣模型，以提供更符合用户当前需求的个性化推荐内容。

1.4　智能体核心组件

智能体（Agent）是指能够在特定环境中自主感知、决策并执行动作的实体，其核心组件通常包括以下几个部分，不同类型的智能体（如软件智能体、物理智能体、AI智能体等）在组件细节上可能有所差异，但整体框架具有共通性。本节讲解智能体的核心组件及其功能说明。

1. 感知模块

1）功能

感知模块（Perception Module）负责从环境中获取信息，将物理世界的信号（如视觉、听觉、触觉等）或数字环境的数据（如传感器数据、网络信息、用户输入等）转换为智能体可处理的内部表示。

2）关键技术

- 传感器技术（如摄像头、麦克风、红外传感器等）。
- 数据采集与预处理（如滤波、降噪、特征提取）。
- 感知算法（如图像识别、语音识别、自然语言处理）。

3）示例

- 自动驾驶汽车通过激光雷达（LiDAR）和摄像头感知路况。
- 聊天机器人通过文本输入感知用户需求。

2. 认知模块

1）功能

认知模块（Cognition Module）负责对感知到的信息进行分析、推理、学习和记忆，形成对环境的理解，并生成决策所需的知识。

2）核心组件

- 知识库（Knowledge Base）：存储智能体的先验知识、规则、模型参数等（如逻辑规则、语义网络、神经网络权重）。
- 推理引擎（Reasoning Engine）：基于知识库进行逻辑推理、概率推断或决策规划（如贝叶斯网络、强化学习算法、决策树）。
- 学习模块（Learning Module）：通过数据或经验更新知识库，提升智能体性能（如监督学习、无监督学习、强化学习）。

3）关键技术

- 知识表示（逻辑表示、语义向量、图结构）。
- 机器学习算法（深度学习、强化学习）。
- 符号推理与数值计算的融合。

3. 决策模块

1）功能

决策模块（Decision Module）根据认知模块的输出，结合智能体的目标和约束，生成具体的动作指令或行动计划。

2）核心逻辑

- 目标驱动：基于预设目标（如最大化收益、最小化风险）选择最优动作。
- 实时响应：处理突发事件（如避障、异常报警）。
- 多目标协调：在多个冲突目标间权衡（如效率与安全）。

3）关键技术

- 优化算法（如贪心算法、动态规划）。
- 决策树、马尔可夫决策过程（MDP）。
- 博弈论（用于多智能体交互场景）。

4. 执行模块

1）功能

执行模块（Execution Module）将决策模块生成的动作指令转换为实际操作，作用于环境或执行器。

2）关键组件

- 执行器（Actuators）：物理智能体的机械臂、电机、扬声器等，软件智能体的API调用、数据库操作、界面交互等。
- 动作协调：确保多个动作的同步或顺序执行（如机器人的多关节协调）。

3）示例

- 工业机械臂根据决策指令抓取零件。
- 智能客服通过API调用查询数据库并回复用户。

5. 记忆模块

1）功能

记忆模块（Memory Module）存储智能体的历史感知数据、决策过程、学习经验等，用于支持长期推理和学习。

2）分类

- 短期记忆（Working Memory）：存储当前任务的临时信息（如对话上下文、实时传感器数据）。
- 长期记忆（Long-Term Memory）：存储持久化知识（如训练好的模型、历史案例库）。

3）技术实现

- 数据库（SQL/NoSQL）、缓存系统。
- 神经科学中的记忆模型（如循环神经网络模拟时序记忆）。

6. 通信模块

1）功能

通信模块（Communication Module）实现智能体与其他智能体、用户或系统的信息交互（如协作、协商、数据共享）。

2）关键能力

- 自然语言交互（语音/文本对话）。
- 多智能体通信协议（如FIPA、ACL）。
- 跨系统接口（API、消息队列）。

3）示例

- 物联网设备通过MQTT协议与云端智能体通信。
- 人类通过语音指令与智能音箱交互。

7. 目标与动机模块

1）功能

目标与动机模块（Goal/Motivation Module）定义智能体的核心目标、优先级和行为动机，驱动决策过程。

2）组成

- 目标集合：显式目标（如“导航至终点”）或隐式目标（如“保持电量充足”）。
- 动机机制：基于奖励函数（强化学习）或价值函数引导行为。

3）作用

确保智能体行为的一致性和目的性，避免无意义的动作。

总之，智能体的核心组件通过“感知–认知–决策–执行”的闭环实现自主行为，记忆与通信模块为其提供历史经验和交互能力，目标模块则赋予其行为方向性。不同应用场景（如自动驾驶、智能客服、工业机器人）的智能体可能在组件实现上各有侧重，但图1.1所示的框架是其功能实现的基础。

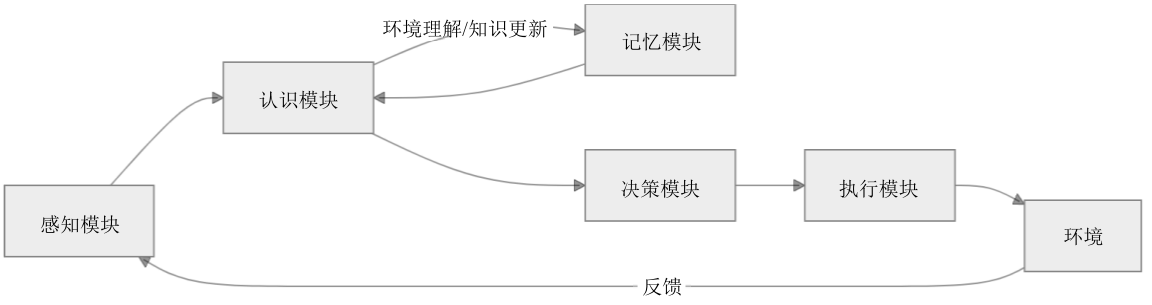

图1.1　智能体核心组件的协作流程

1.5　智能体的发展历程

智能体概念的起源可追溯到20世纪70年代。当时，人工智能领域正处于蓬勃发展阶段，研究者们开始尝试构建能够模拟人类智能行为的系统。早期的智能体模型相对简单，主要侧重于解决特定领域的问题，例如在专家系统中，智能体利用预先设定的规则和知识库来处理特定领域的知识，为用户提供决策支持。本节讲解智能体的发展历程。

1. 早期探索：规则与符号主义（20世纪50年代－20世纪80年代）

1950年，图灵测试叩响机器智能大门，为智能体概念奠定基础。

1986年，马文·明斯基提出智能体的概念，强调多智能体互动协作，但此时AI系统主要依赖预定义规则和符号逻辑，灵活性不足。

2. 互联网与自动化工具兴起（20世纪90年代－21世纪00年代）

互联网普及为智能体发展开辟了新方向，使其承担自动抓取网页信息、邮件过滤、日程管理等基础服务。

2002年，智能体被定义为“能独立行动的计算机系统”，但仍局限于基于规则的简单任务，如早期语音助手只能执行固定指令。

3. 深度学习与强化学习突破（21世纪10年代）

深度学习和强化学习技术取得重大突破，智能体从被动响应迈向主动决策。

2016年，AlphaGo凭借强化学习战胜人类围棋冠军，展示了智能体在复杂环境中的规划能力，智能体在机器人控制、游戏AI等领域崭露头角，但对大量标注数据和固定训练环境有所依赖。

4. 大语言模型与多模态智能体（21世纪20年代）

2022年，GPT-3/4等大语言模型诞生，标志着智能体实现了质的飞跃，拥有强大的自然语言理解和任务规划能力。

2023年，AutoGPT能按自然语言指令自主拆解任务并执行，标志着智能体进入“主动思考”阶段，且融合多模态感知能力，能调用外部工具。

2024－2025年，众多企业推出商业级智能体产品，应用于电商、金融、医疗等多个场景。

5. 未来方向：AGI与生态融合

当下，一些公司正在探索多模态推理和世界模型，用于提升智能体的自主与泛化能力。智能体从单纯工具向“数字员工”转变，未来将更深入融入工作与生活，同时需关注伦理与法律规范等问题。

1.6　智能体与大模型的关系

智能体是一个能够感知环境并通过行为改变环境的实体。它可以是物理实体，如机器人；也可以是软件实体，如软件代理。智能体具有自主性、社会能力、反应性和主动性等特点。

大模型（Large Language Model，LLM）通常是指参数量非常庞大的语言模型，像GPT-3、GPT-4等。这些模型通过在海量文本数据上进行训练，学习语言的模式、语法结构、语义等知识。它们能够生成自然语言文本，完成诸如文本生成、问答、翻译等多种语言任务。

1. 智能体与大模型的联系

1）大模型作为智能体的感知和决策辅助工具

大模型可以为智能体提供强大的语言理解和生成能力。例如，在一个智能客服的场景中，大模型可以帮助智能体理解客户用自然语言提出的复杂问题。当客户询问关于产品技术细节的问题时，大模型能够解析问题的语义，为智能体提供准确的问题理解结果。智能体再根据这个理解结果，结合自身的业务知识库作出相应的回答。同时，大模型还可以帮助智能体生成更加自然、流畅的回答文本，提升用户体验。

对于一些需要自然语言交互的智能体，如智能语音助手，大模型可以增强其对话能力。它能够帮助智能体更好地处理多轮对话，理解上下文信息。例如，在用户和智能语音助手进行连续对话时，大模型可以记住前面的对话内容，使智能体能够给出连贯、合理的回答。

2）为智能体提供知识扩展

大模型经过大量数据训练，拥有广泛的知识。智能体可以利用大模型的知识来弥补自身知识库的不足。例如，一个智能医疗诊断智能体可能主要存储了常见疾病的诊断知识。当遇到一些罕见疾病或者新的医疗研究成果相关的问题时，大模型可以提供相关的知识信息，帮助智能体作出更全面的判断。大模型就像一个知识仓库，智能体可以从中检索有用的信息来丰富自己的知识体系。

2. 智能体与大模型的区别

1）功能侧重点不同

智能体更侧重于与环境的交互，它需要根据环境感知信息作出决策，并通过行动改变环境。例如，一个自动驾驶智能体，它需要感知道路状况（如交通信号、其他车辆和行人的位置等），然后作出驾驶决策（如加速、减速、转向等）来安全地行驶。而大模型主要是处理语言相关的任务，它的核心功能是理解和生成语言，不涉及直接与物理环境的交互。

2）运行机制不同

智能体的运行机制通常包括“感知–决策–行动”的循环。它通过传感器感知环境，然后根据自身的策略或算法作出决策，最后通过执行器来执行决策。大模型的运行机制主要是基于深度学习算法，通过大量的文本数据训练神经网络，学习语言的规律，然后在给定的输入文本基础上进行语言生成或理解等操作。

智能体和大模型可以相互配合，在很多应用场景中发挥各自的优势，共同实现更智能的系统功能。

1.7　AI智能体的应用场景

AI智能体作为具备自主感知、决策和执行能力的智能实体，正在多个领域快速落地，其应用场景覆盖了个人生活、企业服务和社会治理等多个维度。本节介绍当前和未来潜力较大的应用场景分类及具体案例。

1. 个人生活场景

1）虚拟个人助理

案例：ChatGPT、Copilot、Siri的升级版（如Apple Intelligence）、亚马逊的Alexa Guard（家庭安全监控）。

功能：日程管理、邮件自动回复、智能家居控制、个性化推荐（如新闻、购物）。

趋势：多模态交互（语音+视觉）+ 长期记忆，实现更自然的个性化服务。

2）健康管理

案例：AI医生助手（如Ada Health）、心理健康聊天机器人（Woebot）、老年人陪护机器人（如ElliQ）。

功能：症状自查、用药提醒、情感陪伴、紧急情况报警。

2. 企业服务场景

1）客户服务与销售

案例：电商客服（如阿里小蜜）、AI销售顾问（如Gong.io分析销售对话）。

功能：7×24小时自动应答、客户需求分析、销售话术优化。

2）流程自动化（RPA+AI）

案例：UiPath+AI智能体处理财务对账、法律合同审查（如Harvey AI）。

功能：自动处理重复性任务（如发票录入、数据填报），错误率低于人工。

3）知识管理与决策

案例：企业级知识库智能体（如微软365 Copilot）、金融投研助手（如BloombergGPT）。

功能：快速检索内部文档、生成分析报告、辅助战略决策。

3. 垂直行业场景

1）医疗

诊断辅助：AI影像识别（如腾讯觅影）、手术机器人（如达·芬奇系统）。

新药研发：AlphaFold预测蛋白质结构，加速药物发现。

2）金融

风控与投顾：反欺诈系统（如蚂蚁蚁盾）、个性化理财建议（如Betterment）。

高频交易：AI算法自动执行交易策略（如Rebellion Research）。

3）制造业

智能运维：预测设备故障（如西门子MindSphere）。

柔性生产：AI调度系统动态优化生产线（如特斯拉的“无人车间”）。

4）农业

精准农业：无人机巡检+AI分析作物病虫害（如极飞科技）。

自动化养殖：智能饲喂系统（如睿畜科技）。

4. 社会治理与公共领域

1）智慧城市

交通管理：AI优化红绿灯（如阿里云城市大脑）、自动驾驶（Waymo）。
应急响应：AI分析卫星图像预测自然灾害（如Google Flood Forecasting）。

2）教育

个性化学习：自适应学习平台（如Knewton）、AI口语陪练（Duolingo Max）。
教育公平：AI教师覆盖偏远地区（如印度UNESCO项目）。

3）政务与法律

智能政务：自动处理市民咨询（如新加坡“虚拟助手”Jem）。
法律辅助：合同审查（LawGeex）、法律咨询（DoNotPay）。

5. 前沿探索场景

1）元宇宙与虚拟世界

NPC智能化：游戏中的AI角色具备情感交互能力（如Inworld AI）。
数字人：虚拟主播（如央视AI手语主播）、数字员工（如韩国SK Telecom的A.X）。

2）科学发现

自主实验：AI化学家（如利物浦大学的机器人实验室）。
气候建模：AI加速气候模拟（如NVIDIA Earth-2）。

3）太空探索

自主探测器：NASA的AI火星车（如Perseverance的自动导航系统）。

6. 技术驱动趋势

- 多智能体协作：多个AI分工合作（如自动驾驶中感知、规划、控制智能体协同）。
- 具身智能（Embodied AI）：机器人+AI实现物理世界交互（如Figure 01人形机器人）。
- 边缘计算：轻量化智能体部署在终端设备（如手机、IoT设备）。

AI智能体的核心价值在于将被动响应变为主动服务，未来随着大模型、强化学习等技术的进步，其应用场景将从“替代重复劳动”向“创造新价值”演进（如AI科学家、AI艺术家）。

1.8　本 章 小 结

本章系统地介绍了AI智能体的基本概念、类型、功能、核心组件、发展历程、与大模型的关系以及应用场景，为后续深入探讨智能体技术奠定基础。

首先（1.1节），明确了智能体的定义：它是一种能够感知环境、自主决策并执行行动的智能实体，具备目标驱动性、适应性和交互能力，区别于传统程序的被动响应。

其次（1.2节、1.3节），从类型和功能维度展开：智能体可分为简单规则型（如自动化脚本）、基于模型的决策型（如游戏AI）、学习型（如强化学习智能体）和多模态交互型（如ChatGPT）。其核心功能覆盖感知、推理、决策与执行，例如客服场景中的意图识别与问题解决。

接着（1.4节、1.5节），剖析了智能体的核心组件（传感器、知识库、决策引擎、执行器）和发展历程，从早期的符号主义智能体（如Eliza）到现代数据驱动的深度强化学习智能体（如AlphaGo），技术演进依赖算力提升与算法突破。

进一步（1.6节），探讨了智能体与大模型的共生关系：大模型（如GPT-4）为智能体提供通用认知能力，而智能体通过任务规划与工具调用扩展大模型的落地边界，例如Copilot将GPT转换为代码生成执行者。

最后（1.7节），列举了丰富的应用场景，涵盖个人生活（虚拟助手）、企业服务（自动化流程）、垂直行业（医疗诊断）和前沿领域（科学发现），凸显智能体技术对效率提升与范式创新的价值。

综上所述，本章从理论到实践勾勒出AI智能体的全貌，其核心价值在于将被动程序升级为主动服务者，而未来技术发展需解决可解释性、安全伦理等挑战，以推动其应用的规模化落地。

第 2 章

Agent开发环境配置

任何希望将大模型Agent落实到具体项目的开发者，首先要解决“用什么写”和“怎么写”这两个问题。本书统一采用Python 3作为实现语言，原因有以下三点。

（1）生态成熟：Python拥有活跃的开源社区和庞大的第三方科学计算库，涵盖数据获取、预处理、建模、部署、可视化等完整链路。

（2）框架统一：当前主流的大模型训练与推理框架（PyTorch、JAX、TensorFlow 2.x）均将Python视为一等公民，官方示例与文档均以Python为主。

（3）教学成本低：Python语法简洁，新手可快速上手；高级开发者也能通过元编程、Cython等手段获得接近C/C++的性能。

然而，原生Python存在版本碎片化、二进制依赖复杂、环境隔离困难等问题。为解决这些痛点，我们推荐使用Anaconda3（或体积更小的Miniconda）作为发行版。Anaconda3带来以下优势。

- “开箱即用”：一次性安装即可获得NumPy、SciPy、Pandas、JupyterLab等常用科学计算库。
- 环境隔离：通过conda env创建独立虚拟环境，避免库版本冲突。
- 跨平台一致：Windows、macOS、Linux均可通过同一套命令完成安装与升级。

PyTorch的核心特点是动态计算图，这意味着计算图可在运行过程中动态调整，极大地提升了模型的灵活性与运行效率。同时，PyTorch提供了丰富的API，不仅支持多种深度学习模型与算法，还能轻松与NumPy、SciPy等其他Python库进行交互。

在本章中，我们将指导读者完成Anaconda3的完整安装流程，随后通过运行一个程序来验证PyTorch框架的安装情况。通过这些内容的讲解，读者将能够初步掌握Qwen3的环境配置方法。

2.1 智能体开发环境安装

Python与AI Agent（人工智能智能体）之间有着密不可分的关系。作为当前人工智能领域主流的编程语言，Python凭借其简洁的语法、丰富的科学计算库（如NumPy、Pandas）以及强大的AI框架支持（如TensorFlow、PyTorch、Hugging Face），成为开发AI Agent的首选工具。AI Agent是能够感知环境、进行决策并采取行动的智能实体，广泛应用于智能对话系统、自动化决策、机器人控制和自主学

习系统中。借助Python强大的生态系统，开发者可以轻松实现自然语言处理、机器学习模型训练、环境交互逻辑构建等核心功能，从而快速搭建和迭代智能体系统。此外，Python丰富的异步编程和网络库也便于智能体与外部系统（如API、数据库、传感器）进行高效通信。因此，掌握Python不仅是进入人工智能领域的钥匙，更是构建功能强大、灵活智能的AI Agent的基石。

2.1.1　Anaconda的下载与安装

1. 下载Anaconda

（1）访问Anaconda官方网址https://www.anaconda.com，登录官方网址后，在Anaconda官方网址打开下载页面，如图2.1所示。

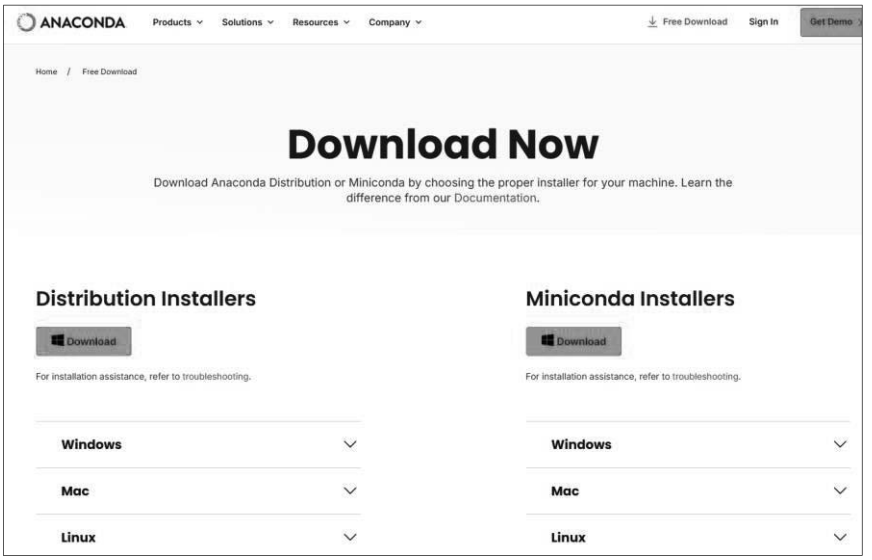

图2.1　Anaconda下载页面

（2）根据读者自己的计算机系统下载相应的安装包，作者下载的是Windows 64位Python 3.13版本的安装包，如图2.2所示。

图2.2　下载Anaconda3 Windows安装包

（3）安装包下载页面如图2.3所示，下载的安装文件名为Anaconda3-2025.06-Windows-x86_64.exe。

2. 安装过程

双击下载好的Anaconda3-2025.06-Windows-x86_64.exe文件，按安装向导操作，建议：

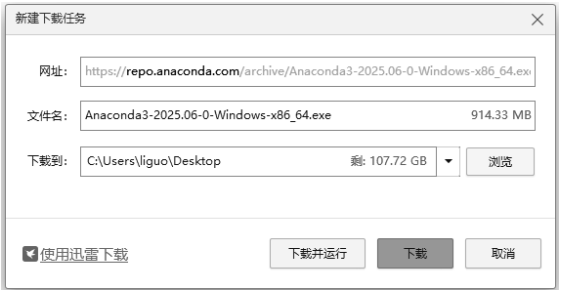

图2.3　Anaconda3-2025.06-Windows-x86_64.exe安装包下载页面

- 为所有用户安装（需要管理员权限）。
- 安装路径不要包含空格或特殊字符（如C:\Anaconda3）。
- 勾选Add Anaconda3 to my PATH environment variable（虽然不推荐，但开发常用）。
- 勾选Register Anaconda3 as my default Python。

3. 验证安装

打开新终端/命令提示符，执行以下命令验证安装是否成功：

```
conda --version
python --version
```

2.1.2　PyTorch的下载与安装

在深度学习领域，PyTorch凭借其卓越的灵活性与强大的功能，已成为业界广泛采用的核心框架之一。其最显著的优势在于支持动态计算图（Dynamic Computation Graph），这一特性使得模型的构建、调试和迭代过程更加直观高效。开发者可以在运行时灵活调整网络结构，实时查看中间结果，极大地提升了研发效率，特别适用于研究探索、模型快速原型设计以及复杂任务的实现。

从实践角度来看，PyTorch提供了简洁清晰的API接口，具有良好的可读性和易用性，无论是初学者还是资深研究人员都能迅速上手。同时，它与Python生态无缝集成，支持主流数据处理和可视化工具，便于构建完整的深度学习工作流。更重要的是，PyTorch背后拥有一个活跃且庞大的开源社区，持续贡献高质量的模型库、教程资源和技术支持，为实际项目落地提供了强有力的保障。

在本书重点讲解的Qwen3、DeepSeek等大语言模型的应用实践中，PyTorch作为其核心后端框架，承担着模型加载、推理执行、参数微调以及分布式训练等关键任务。LLM模型的高效部署与精细化微调，高度依赖于PyTorch所提供的张量计算能力、自动微分机制以及对多设备（CPU/GPU）的统一支持。因此，正确配置PyTorch环境是开展后续所有工作的前提和基础。

在安装层面，PyTorch提供了CPU版本和GPU版本两种选择，以适配不同的硬件条件与性能需求。

- CPU版本：适用于开发测试、轻量级推理或缺乏GPU资源的场景，安装简单，兼容性强。
- GPU版本：能充分利用NVIDIA CUDA加速能力，显著提升训练与推理速度，尤其适合大规模模型（如Qwen3）的实际部署与调优。

以Windows平台为例，根据读者工作计算机的资源情况，若选择安装仅支持GPU的PyTorch版本（以PyTorch 2.8.0（CUDA12.6）为例），可使用如下命令：

```
# CPU only
pip3 install torch torchvision --index-url https://download.pytorch.org/whl/cu126
```

相比之下，GPU版本的安装涉及CUDA驱动、cuDNN等底层依赖的匹配，配置更为复杂。考虑到Qwen3模型对计算资源的高要求，推荐在具备NVIDIA显卡的环境中安装GPU版本PyTorch，以充分发挥其并行计算优势，加速模型训练与推理进程。接下来，我们将重点介绍GPU版本的安装步骤与常见问题解决方案。

1. 核心概念理解

在开始之前，理解这三者之间的关系至关重要。

（1）CUDA：由NVIDIA推出的并行计算平台和编程模型。它允许软件开发者和软件工程师使用支持CUDA的GPU进行通用处理（而不仅仅是图形处理）。简而言之，PyTorch需要CUDA来在NVIDIA GPU上执行计算。

（2）cuDNN：CUDA Deep Neural Network library的缩写。它是NVIDIA提供的针对深度神经网络的GPU加速库。它提供了高度优化的常见深度学习操作（如卷积、池化、归一化等）的实现。cuDNN是运行在CUDA之上的一个专用加速库，PyTorch等框架会调用它来极致地提升训练和推理速度。

（3）PyTorch：一个开源的机器学习框架。它本身不直接与GPU通信，而是通过CUDA接口来利用GPU进行计算。当你安装了PyTorch的CUDA版本（即GPU版本）后，PyTorch就可以调用CUDA和cuDNN来在NVIDIA GPU上高效运行。

调用关系总结：PyTorch→cuDNN→CUDA→NVIDIA GPU Driver→NVIDIA GPU。

因此，安装顺序应该是：先安装GPU驱动，再安装CUDA和cuDNN，最后安装对应版本的PyTorch。

2. 安装前的准备工作

1）检查你的 NVIDIA GPU 是否支持 CUDA

确保你的计算机拥有NVIDIA显卡，并且它支持CUDA。几乎所有现代的NVIDIA GPU（GeForce、RTX、Quadro、Tesla等）都支持CUDA。

官方支持列表请查看：https://developer.nvidia.com/cuda-gpus。

2）查看你的显卡驱动版本

这一步是为了确定你需要安装哪个版本的CUDA。

（1）在键盘上按Win+R键，输入cmd并按Enter键，打开命令提示符。

（2）输入命令：nvidia-smi。

（3）查看右上角显示的驱动版本和最高支持的CUDA版本。

例如，输出中有一行CUDA Version:12.4，这表示你的当前驱动最高可以支持CUDA 12.4。你需要安装不高于12.4的CUDA版本。

3. CUDA安装

CUDA是NVIDIA的并行计算平台，PyTorch的GPU加速需要依赖它。

1）检查 GPU 是否支持 CUDA

访问NVIDIA官方网址查看你的GPU型号是否在支持列表中，或在命令行窗口输入命令：wmic path win32_VideoController get name来检查。

2）下载 CUDA Toolkit

- 访问NVIDIA CUDA下载页面：https://developer.nvidia.com/cuda-toolkit-archive。
- 选择适合你系统的版本，建议选择PyTorch支持的版本，目前常用11.7或11.8。
- 按照系统类型选择相应的安装包。

3）安装 CUDA

运行下载的.exe文件，建议使用默认安装路径。安装过程中确保勾选CUDA Toolkit。

4）验证安装

命令行输入：nvcc -V，若显示版本信息，则安装成功。

4. cuDNN安装

cuDNN是NVIDIA的深度神经网络加速库，是PyTorch GPU加速的必要组件。

1）下载 cuDNN

（1）访问：https://developer.nvidia.com/rdp/cudnn-download。

（2）需要注册NVIDIA开发者账号。

（3）选择与已安装CUDA版本匹配的cuDNN版本。

2）安装 cuDNN

解压下载的ZIP文件。将bin、include、lib文件夹复制到CUDA安装目录（默认为：C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\vxx.x）。

3）验证安装

检查cuDNN版本：查看CUDA安装目录下的cudnn_version.h文件。

5. PyTorch安装

1）推荐使用 Anaconda 环境（可选，但推荐）

（1）安装Anaconda：https://www.anaconda.com/products/distribution。

（2）打开Anaconda Prompt窗口，创建虚拟环境：

```
conda create -n pytorch_env python=3.12
conda activate pytorch_env
```

2）安装 PyTorch

（1）访问PyTorch官方网址：https://pytorch.org/。

（2）根据系统、CUDA版本选择合适的安装命令。

（3）常用命令（以PyTorch 2.6.0 + CUDA 11.8为例）：

```
# CUDA 11.8
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url
https://download.pytorch.org/whl/cu118
```

3）验证 PyTorch 安装

打开Python终端，输入以下代码：

```
import torch
print(torch.__version__)
print(torch.cuda.is_available())            #输出True表示GPU可用
print(torch.backends.cudnn.version())       #输出cuDNN版本
```

如果没有报错且torch.cuda.is_available()返回True，则安装成功。

6. 常见问题解决

- CUDA版本不匹配：确保PyTorch版本与CUDA版本兼容。
- 权限问题：Linux安装时可能需要使用sudo。
- PATH环境变量：确保CUDA安装路径已添加到系统PATH中。
- 驱动问题：确保NVIDIA显卡驱动是最新版本。

如果你的计算机没有NVIDIA显卡，可以安装CPU版本的PyTorch，只需在PyTorch官方网址选择CPU对应的安装命令进行安装即可。

2.1.3　PyCharm的安装与使用

和其他语言类似，Python程序的编写可以使用Windows自带的编辑器。但是这种方式对于较为复杂的程序工程来说，容易混淆相互之间的层级和交互文件，因此在编写程序工程时，我们可以使用专用的Python编译器PyCharm。

由于PyCharm的安装比较简单，本小节就不展开讲解了。读者上网搜索PyCharm官方网址，进入Download页面，选择适合自己操作系统的版本安装即可。PyCharm下载页面如图2.4所示。

图2.4　下载PyCharm

完成PyCharm安装后，可以在PyCharm中打开本书配套代码，并在Anaconda Prompt窗口中搭建虚拟环境。本书配套代码可以运行在虚拟环境中，如图2.5所示。

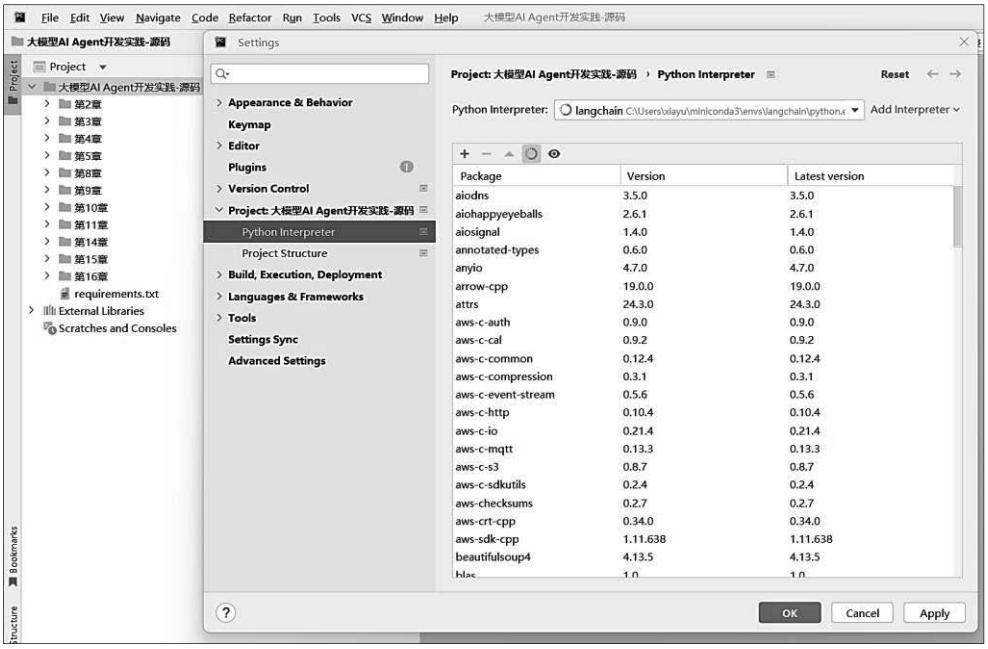

图2.5　在PyCharm中设置代码虚拟运行环境

2.2　LLM的调用与使用

本节将以ModelScope（魔搭社区）中的Qwen3大模型为例介绍LLM的调用与使用。

2.2.1　ModelScope（魔搭社区）

ModelScope（类似于Hugging Face）是由阿里巴巴集团牵头，联合多家中国顶尖科研机构和高校共同推出的一个下一代“模型即服务”（Model-as-a-Service，MaaS）共享平台。它的核心目标是降低人工智能模型的应用门槛，促进开源模型的共享、协作与创新。

1. 核心定位与愿景

ModelScope的愿景是成为AI模型领域的GitHub。就像GitHub托管代码一样，ModelScope致力于托管、分享和运行AI模型。它不仅仅是一个模型仓库，更是一个提供从模型探索、体验、微调到部署的一站式服务的生态系统。

其核心用户包括：

- AI研究人员：发布和验证自己的模型。
- 应用开发者：快速找到并集成现成的模型到自己的应用中，无须从零开始训练。
- 学生和爱好者：学习最前沿的AI技术，动手实践。
- 企业用户：寻找商业化的解决方案，降低AI研发成本。

2. 主要功能与核心组成部分

ModelScope社区提供了从模型探索、体验、部署到再开发的全链路服务。

1）庞大的模型库

（1）ModelScope汇聚了来自阿里巴巴、清华大学、北京大学、浙江大学、商汤科技、澜舟科技等众多顶尖AI实验室和高校的开源模型。

（2）覆盖领域广泛：包括自然语言处理（NLP）、计算机视觉（CV）、语音识别与合成、多模态、科学计算等。

（3）模型类型多样：从基础的图像分类、文本Embedding，到大型语言模型（如Qwen、Baichuan）、文生图模型（如Taiyi、SD）、语音合成模型等。

2）在线体验与 Demo

几乎每一个模型都提供了在线试玩（Playground）功能。用户无须安装任何环境，直接在网页上上传图片、输入文本或录音，即可立即看到模型的推理效果，这极大地降低了模型体验的门槛。

3）Notebook 开发环境

平台提供了集成好的、云端免费的Jupyter Notebook开发环境（基于PAI-DSW），预装了ModelScope SDK和常用依赖。用户可以直接在浏览器中打开Notebook，编写几行代码即可加载和运行模型，进行原型开发和实验。

4）ModelScope Library（Python SDK）

这是ModelScope的核心组件，一个开源Python库。通过它，开发者可以用极其简洁的代码（通常只需2~3行）在自己的本地或云端环境中下载、加载和推理模型。

示例代码如下：

```
    from modelscope.pipelines import pipeline
    from modelscope.utils.constant import Tasks
    # 创建文本摘要 pipeline
    pipe = pipeline(task=Tasks.text_summarization,
model='damo/nlp_bert_document-segmentation_chinese-base')
    # 输入文本并获取结果
    result = pipe('这里是需要摘要的长篇文章内容...')
    print(result)
```

5）ModelScope Studio

类似于Hugging Face的Spaces，ModelScope Studio（创空间）是一个低代码/无代码的应用构建平台。用户可以利用Gradio、Streamlit等框架，快速为自己的模型构建一个功能丰富、界面友好的演示应用，并一键部署和分享给他人。

6）数据集与学习资源

（1）除了模型外，平台还提供了与模型配套使用的训练和评测数据集。

（2）包含丰富的教程、文档、技术博客和视频，帮助用户从入门到精通。

2.2.2　Qwen3的本地调用

如果要使用大模型进行应用开发，我们可以通过ModelScope（魔塔）官方网址在模型库中查找，比如要使用Qwen3大模型，可以查找Qwen3关键字，结果如图2.6所示。

从图2.6中可以看到，这里找到了多个不同种类的Qwen3模型。我们单击“通义千问3-32B”，进入模型介绍界面，如图2.7所示。

图2.6　查找Qwen3

图2.7　“通义千问3-32B”标签界面

这个页面对“通义千问3-32B”进行介绍，在页面下拉左侧介绍框，可以看到基于ModelScope本地模型的读取与处理示例。

需要注意的是，Qwen3的代码已包含在最新的Hugging Face Transformers中，建议读者使用Transformers的最新版本。如果使用Transformers低于4.51.0的版本，可能会将遇到以下错误：KeyError: 'qwen3'。

【示例2.1】基于ModelScope本地模型的读取与处理，本示例展示如何根据给定输入使用模型生成内容，代码如下：

```
from modelscope import AutoModelForCausalLM, AutoTokenizer

model_name = "Qwen/Qwen3-1.7B"

# load the tokenizer and the model
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype="auto",
    device_map="auto"
)

# prepare the model input
prompt = "Give me a short introduction to large language model."
messages = [
    {"role": "user", "content": prompt}
]
```

```
    text = tokenizer.apply_chat_template(
        messages,
        tokenize=False,
        add_generation_prompt=True,
        enable_thinking=True # Switches between thinking and non-thinking modes. Default is
True.
    )
    model_inputs = tokenizer([text], return_tensors="pt").to(model.device)

    # conduct text completion
    generated_ids = model.generate(
        **model_inputs,
        max_new_tokens=32768
    )
    output_ids = generated_ids[0][len(model_inputs.input_ids[0]):].tolist()

    # parsing thinking content
    try:
        # rindex finding 151668 (</think>)
        index = len(output_ids) - output_ids[::-1].index(151668)
    except ValueError:
        index = 0

    thinking_content = tokenizer.decode(output_ids[:index], skip_special_tokens= True).strip("\n")
    content = tokenizer.decode(output_ids[index:], skip_special_tokens=True).strip("\n")

    print("thinking content:", thinking_content)
    print("content:", content)
    response = tokenizer.decode(output_ids, skip_special_tokens=True)
    print(response)
```

```
2025-10-04 11:00:54,127 - modelscope - INFO - PyTorch version 2.8.0 Found.
2025-10-04 11:00:54,138 - modelscope - INFO - Loading ast index from C:\Users\liguo\.cache\modelscope\ast_indexer
2025-10-04 11:00:54,472 - modelscope - INFO - Loading done! Current index file version is 1.9.5, with md5 f85e691e041285c47726759e4603abbb and a total n
umber of 945 components indexed
Downloading: 100%|████████████████████████████████████████████████| 3.69G/3.69G [09:28<00:00, 6.96MB/s]
Downloading: 100%|████████████████████████████████████████████████| 3.63G/3.63G [07:33<00:00, 8.59MB/s]
Downloading: 100%|████████████████████████████████████████████████| 3.63G/3.63G [10:11<00:00, 6.38MB/s]
Downloading:   0%|                                                  | 0.00/3.63G [00:00<?, ?B/s]
```

资源调用太耗费内存和算力，结果暂不展示。对于部署，可以使用sglang>=0.4.6.post1或vllm>=0.8.5来创建一个与OpenAI兼容的API端点。

- SGLang:

```
SGLANG_USE_MODELSCOPE=true python -m sglang.launch_server --model-path Qwen/
Qwen3-1.7B --reasoning-parser qwen3
```

- vLLM:

```
VLLM_USE_MODELSCOPE=true vllm serve Qwen/Qwen3-1.7B --enable-reasoning --
reasoning-parser deepseek_r1
```

代码是用于加载Qwen3-1.7B模型并进行推理的完整流程，使用了ModelScope的AutoModelForCausalLM和AutoTokenizer。如果使用的模型较大，比如Qwen3-32B，则实际运行需要更高性能的GPU，如A100、H100等。

对于本地使用，Ollama、LMStudio、MLX-LM、llama.cpp和KTransformers等应用程序也支持Qwen3，读者可以根据自己的需要选择在本地运行大模型。

2.2.3　Qwen3的在线调用

首先我们需要登录Qwen3官方网站“阿里云百炼”，页面菜单如图2.8所示。

图2.8　“阿里云百炼”菜单

注册并登录后进入百炼控制台，在页面上单击“大模型服务平台百炼”，打开“阿里云百炼”主页面，再单击“模型服务”，页面左下角有个“密钥管理”，如图2.9所示。

图2.9　获取API Key

单击“密钥管理”后，进入“密钥管理”页面，在这里既可以创建新的API-Key，也可以查询以前所创建的API-Key，如图2.10所示。

图2.10　已创建的API Key

接下来，单击“模型广场”，在“模型广场”页面上选择想要开通的模型，如图2.11所示。

图2.11　单击通义千问目录

对于不同的任务需求，Qwen3给我们准备了不同的API调用示例。例如，在“模型广场”上单击“通义千问3-Max”进入模型页面，其中提供了“API代码示例”，如图2.12所示。

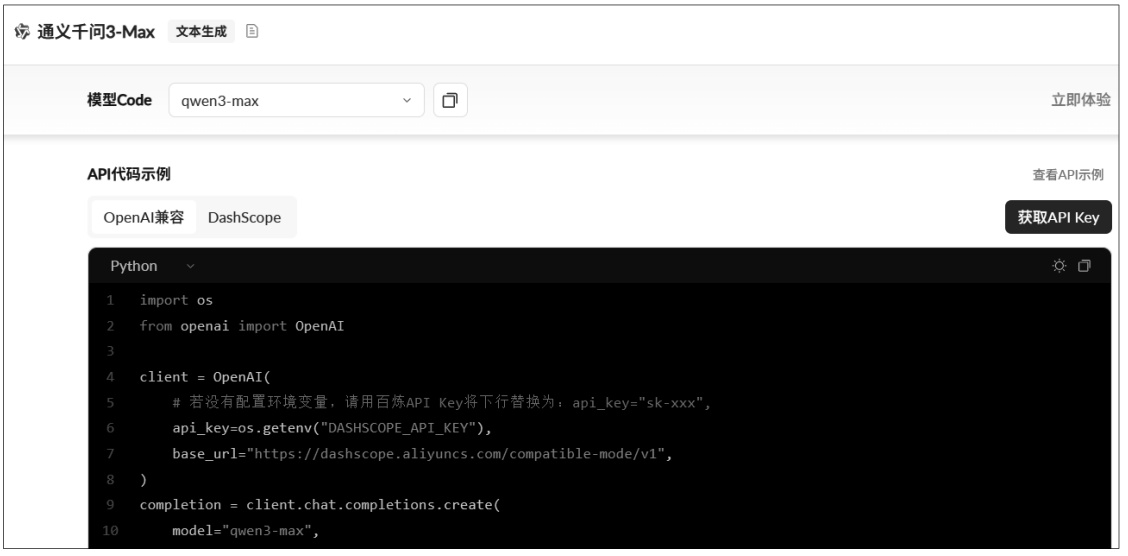

图2.12　Qwen3在线API调用示例

【示例2.2】“模型广场”提供的Qwen3在线API调用示例。

```
import os
from openai import OpenAI

client = OpenAI(
    # 若没有配置环境变量，请用百炼API Key将下一行替换为：api_key="sk-xxx"
    api_key=("DASHSCOPE_API_KEY"),
    base_url="https://dashscope.aliyuncs.com/compatible-mode/v1",
)

completion = client.chat.completions.create(
    # 模型列表：https://help.aliyun.com/zh/model-studio/getting-started/models
    model="qwen-plus",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "你是谁？"},
    ],
    # Qwen3模型通过enable_thinking参数控制思考过程（开源版默认为True，商业版默认为False）
    # 使用Qwen3开源版模型时，若未启用流式输出，请将下一行取消注释，否则会报错
    # extra_body={"enable_thinking": False},
)
print(completion.model_dump_json())
```

运行代码，输出如下：

```
{"id":"chatcmpl-efc22742-e6b8-99c1-bb83-4129b2a379d1","choices":[{"finish_reason":"sto
p","index":0,"logprobs":null,"message":{"content":"我是通义千问，阿里巴巴集团旗下的通义实验室自主研
发的超大规模语言模型。我可以帮助你回答问题、创作文字，比如写故事、写公文、写邮件、写剧本、逻辑推理、编程等，
还能表达观点、玩游戏等。如果你有任何问题或需要帮助，欢迎随时告诉我！","refusal":null,"role":"assistant",
"annotations":null,"audio":null,"function_call":null,"tool_calls":null}}],"created":175662
```

5977,"model":"qwen-plus","object":"chat.completion","service_tier":null,"system_fingerprint":null,"usage":{"completion_tokens":66,"prompt_tokens":26,"total_tokens":92,"completion_tokens_details":null,"prompt_tokens_details":{"audio_tokens":null,"cached_tokens":0}}}

读者可以将上面代码中的API Key替换成自己的。由于Qwen3模型包含思考过程，读者同时还可以自定义思考过程的输出方式。下面是作者重写的参数说明：

（1）model：字符串，必选，指定模型名称。支持通义千问大语言模型（商业版、开源版、Qwen-Long）、通义千问VL、通义千问Omni、数学模型、代码模型。需要注意的是，通义千问Audio暂不支持OpenAI兼容模式，仅支持DashScope方式。具体模型名称及计费规则详见模型列表文档。

（2）messages：数组，必选，对话组成的消息列表。其包含以下消息类型。

- SystemMessage：对象，可选，定义模型目标或角色。若设置，则需置于messages列表首位。但QwQ模型不建议设置，QVQ模型设置后不生效。
- UserMessage：对象，必选，表示用户发送给模型的消息内容。
- AssistantMessage：对象，可选，记录模型对用户消息的回复。
- ToolMessage：对象，可选，承载工具的输出信息。

（3）stream：布尔值，可选，默认为false，控制是否流式输出回复。

- false：模型生成完整内容后一次性返回。
- true：逐片段输出（chunk），需实时读取以获取完整结果。仅Qwen3商业版（思考模式）、Qwen3开源版、QwQ、QVQ支持流式输出。

（4）stream_options：当stream=true时，可通过设置{"include_usage": true}在输出末行显示token使用量，设为false则隐藏该信息。

（5）modalities：仅Qwen-Omni模型支持指定输出模态。

- ["text","audio"]：同时输出文本与音频。
- ["text"]：仅输出文本。

（6）temperature：浮点数，可选，控制生成文本多样性。值越高，结果越随机（范围为[0,2)）。建议与top_p二选一设置，QVQ模型默认值不建议修改。

（7）top_p：浮点数，可选，采样阈值。值越高，结果越随机（范围为(0,1.0]）。建议与temperature二选一设置，QVQ模型默认值不建议修改。

（8）top_k：整数，可选，采样候选集大小（≥0）。值越大，随机性越高。设为None或大于100时仅top_p生效。QVQ模型默认值不建议修改，Python SDK需通过extra_body配置。

（9）presence_penalty：控制内容重复度（范围为[−2.0,2.0]）。

- 正值减少重复（适合创意场景）。
- 负值增加重复（适合专业文档）。

示例：qwen-vl-plus系列模型文字提取建议设置为1.5，QVQ模型默认值不建议修改。

（10）response_format：指定返回格式。

- {"type":"text"}：纯文本。
- {"type":"json_object"}：结构化JSON（需在消息中提示模型输出JSON格式）。

（11）max_tokens：限制返回最大Token数，超限内容将被截断。默认值及上限参考模型列表，qwen-vl-ocr系列默认为2048，最大为8192。QwQ/QVQ模型仅限制回复长度，不限制思考内容。

（12）n：integer（可选）默认值为1，生成响应的数量，取值范围是1~4，适用于多结果场景（如创意写作）。仅qwen-plus及非思考模式Qwen3支持，且传入tools时固定为1。

（13）enable_thinking：控制Qwen3模型思考模式。需通过extra_body配置。商业版默认为false，开源版默认为true。

（14）thinking_budget：设置思考过程最大长度。仅enable_thinking=true时生效，适用于Qwen3商业版及开源版。

（15）seed：设置随机种子（0~$2^{31}-1$）以获得确定性结果，相同seed和其他参数将生成相同内容。

（16）stop：指定终止生成字符串或token_id，可用于敏感词过滤。数组类型不支持混合token_id与字符串。

（17）tools：定义可调用工具列表，当前不支持通义千问VL/Audio及数学/代码模型。每个工具对象包含：

- tool_choice：字符串/对象，可选，默认为"auto"。控制工具调用策略，可选"auto" "none"或指定工具名称。
- parallel_tool_calls：boolean（可选），默认值为false，表示是否开启并行工具调用。相关文档：并行工具调用可选值：true：开启；false：不开启。

（18）translation_options：翻译模型专用配置参数，需通过extra_body传递。

（19）enable_search：控制是否启用互联网搜索。

- true：允许模型参考搜索结果（可能增加Token消耗）。
- false：禁用搜索，支持强制搜索配置。

（20）search_options：对象，可选，联网搜索策略配置，仅enable_search=true时生效。

不同参数定义了用户在使用Qwen3在线API时能够进行的操作。除了基本的文本输入输出外，对于更多的使用示例和样式，读者可以自行验证。

2.3　本 章 小 结

本章系统地介绍了大模型Agent开发的环境配置，为解决“用什么写”和“怎么写”奠定了基础。本书选用Python3作为核心语言，因其具备成熟的科学计算生态、与主流深度学习框架的深度集成以及低教学成本三大优势。为克服原生Python的环境管理难题，本章推荐使用Anaconda3发行版，其开箱即用的科学计算库、强大的环境隔离能力和跨平台一致性为开发提供了极大便利。随后，本章重点阐述了PyTorch框架因其动态计算图带来的灵活性与高效性。通过指导读者完成从Anaconda3安装、PyTorch环境验证，到Qwen3模型在魔搭社区的本地化部署及在线API调用等全流程，本章旨在让读者初步掌握搭建一个功能完备且高效的Agent开发环境的具体方法，为后续的实践开发铺平道路。