第 1 章

人工智能概论

“如果你问我未来 20 年最重要的技术是什么，我会告诉你是人工智能。人工智能会像一百多年前的电力一样重要。”

——凯文·凯利　美国《连线》杂志创始主编

1.1　人工智能的定义和分类

1. 人工智能的定义

《人工智能标准化白皮书(2018 版)》将人工智能定义为：人工智能(Artificial Intelligence，AI)是利用数字计算机或者数字计算机控制的机器模拟、延伸和扩展人的智能，感知环境、获取知识，并使用知识获得最佳结果的理论、方法、技术及应用系统。

在人工智能的定义中，被广泛接受的定义有几种，在此简要列举比较常用的几种人工智能的定义。

定义 1：人工智能是一种使计算机能够思维，使机器具有智力的激动人心的新尝试(Haugeland，1985 年)。

定义 2：人工智能是研究那些让理解、推理和行为成为可能的计算(Winston，1992 年)。

定义 3：人工智能是与人类思考方式相似的机器和计算机程序。

从根本上讲，这是一种类似仿生学的直观思路，用程序模拟人的思考方式。

定义 4：人工智能是与人类行为相似的机器和计算机程序。

该观点认为，只要该功能与人在类似的环境下的行为相似，就可以说这个计算机程序拥有了该领域的人工智能。

定义 5：人工智能是能够学习的机器和计算机程序。

这一定义认为，具备机器学习、数据挖掘、深度学习能力的机器和计算机程序就可以称为人工智能。

定义 6：人工智能是根据对环境的感知做出合理的行为并获得最大的收益的机器和计算机程序。

为了更形象地解读人工智能的含义，我们从以下三个方面来分析何谓人工智能。

(1) 人工智能研究的目的是延伸和扩展人类功能与人类智慧。

(2) 人工智能的本质是人创造的机器和计算机程序，不是天然存在的。

(3) 人工智能的作用是让机器和计算机程序比人更会看、会听、会说、会读、会写、会思考、会决策、会行动。

人工智能示意图及其延伸人的功能体系分别如图 1.1 和表 1.1 所示。

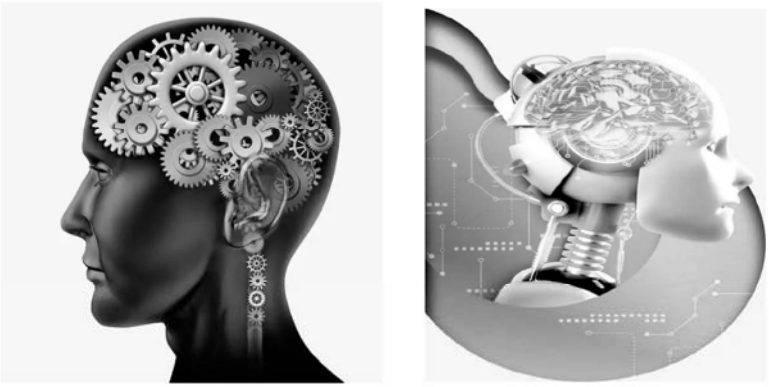

图 1.1 人工智能示意图

表 1.1 人工智能延伸人的功能体系

序 号	人的能力	人工智能对应的识别功能	模拟、扩展人的功能
1	会看	图像识别	模拟、扩展人的视觉功能
2	会听	语音识别	模拟、扩展人的听觉功能
3	会说		模拟、扩展人的视觉智能
4	会读	语义识别	模拟、扩展人的语言功能
5	会写		模拟、扩展人的触觉功能
6	会思考	逻辑思考	模拟、扩展人的大脑思维功能
7	会决策		
8	会行动	步态识别	模拟、扩展人的行动功能
9	综合运用	综合各类识别	模拟、扩展人的综合功能

2. 人工智能的分类

人工智能是一个知识工程领域，是让机器模仿人类利用知识完成一定行为的过程。根据人工智能是否能真正实现推理、思考和决策，可将人工智能分为弱人工智能和强人工智能。

1) 弱人工智能

弱人工智能是指不能真正实现推理和解决问题的智能机器，这些机器从表面上看像是智能的，其实并不算拥有智能，也没有自我分析、自我思考、自我决策、自我执行的能力。

当今，人工智能研究的方向仍然集中在弱人工智能，并在不同领域中取得了显著的成果，如在语音识别、图像分类、物体检测、机器翻译等方面取得了重大突破，甚至可以接

近或超越人类的水平。

2) 强人工智能

强人工智能(又称通用人工智能或类人智能)是指拥有人类思维意识的智能，也就是说，拥有强人工智能的机器能够进行自我分析、自我思考、自我决策、自我执行。这类强人工智能机器可分为类人工智能(机器的分析、思考、推理、决策方式类似人的思维方式)与非类人工智能(这类机器分析、思考和推理的方式不同于人的思维方式)两大类。

从一般意义来说，达到人类水平的、能够自适应地应对外界环境挑战的、具有自我意识的人工智能称为强人工智能。强人工智能不仅在哲学上存在巨大争论(涉及思维与意识等根本问题的讨论)，在技术的研究上也具有极大的挑战性。

1.2 人工智能的特征

根据《人工智能标准化白皮书(2018 版)》可知，人工智能具有以下三大特征。

(1) 由人类设计，为人类服务，本质为计算，基础为数据。

从根本上说，人工智能系统必须以人为本，通过人类设计的算法或软件采用人类发明的硬件设备来工作。人工智能的本质体现为数据计算，通过对图像、视频、语音等各类数据的采集、加工、处理、分析和挖掘，形成有一定价值的信息流和知识模型，为模拟、延伸和扩展人类能力服务，实现人类对机器期望的一些“智能行为”的模拟。人工智能的目的是服务人类，而不是伤害人类。

(2) 能感知环境，能产生反应，能与人交互，能与人互补。

人工智能应能通过传感器等硬件设备了解、感知外界环境(包括人类)信息，可以像人一样通过听觉、视觉、嗅觉、触觉等获得来自环境的各种信息，并对外界输入产生文字、语音、表情、动作(控制执行机构)等必要的反应，甚至影响环境或人类。借助按钮、键盘、鼠标、屏幕、手势、体态、表情、力反馈、虚拟现实/增强现实等方式，人与机器之间可以产生交互，使机器设备越来越“理解”人类，乃至与人类共同协作、优势互补。

(3) 有适应特性，有学习能力，有演化迭代，有连接扩展。

人工智能应具有一定的自适应特性和学习能力，即具有一定的随环境变化而自适应调节参数或更新优化模型的能力；并且能够在此基础上通过与云、端、人、物越来越广泛深入的数字化连接扩展，实现机器客体乃至人类主体的演化迭代，以使系统具有适应性、鲁棒性、灵活性、可扩展性，来应对不断变化的现实环境，从而使人工智能系统在各行各业得到丰富的应用。

1.3 人工智能的主要流派

人工智能在发展过程中涌现出很多流派，主要有符号主义、联结主义、行为主义等流派。这些流派相辅相成，共同推进了人工智能的发展。人工智能的发展历程如图 1.2 所示。

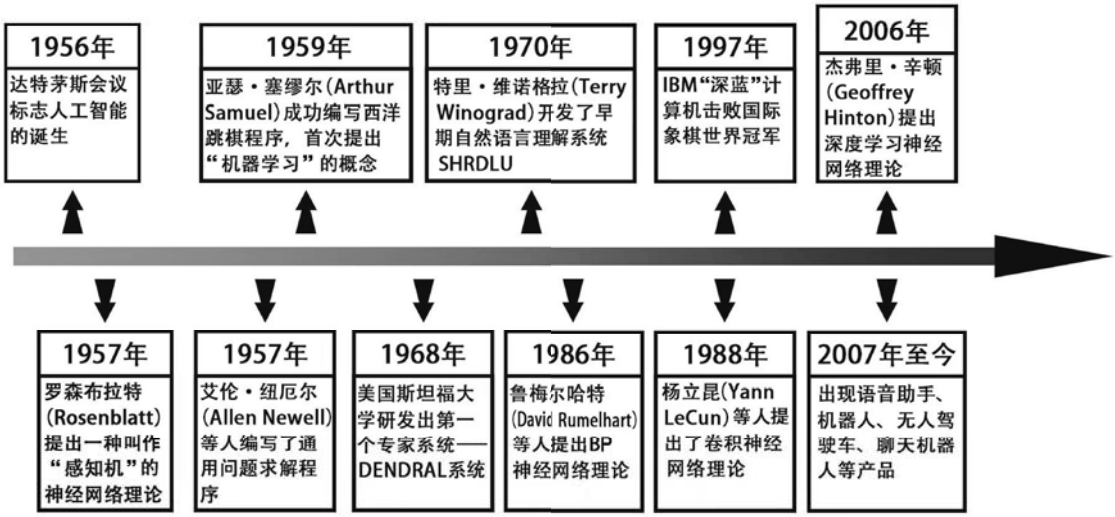

图 1.2 人工智能的发展历程

1.3.1 符号主义学派

符号主义学派又称逻辑主义学派或计算机学派。

符号主义学派认为，人工智能源于数学逻辑，人的认知基元是符号，认知过程即符号操作过程；符号认知过程是一个符号获取、存储、分析、处理的过程，通过分析人类认知系统所具备的功能和机能，然后使用计算机来模拟人的这些功能，从而实现人工智能。

符号主义学派的发展大概经历了以下两个阶段。

第一阶段：推理期(20 世纪 50 至 70 年代)，人们基于符号知识表示，通过演绎推理技术取得了很大的成就。

第二阶段：知识期(20 世纪 70 年代)，人们通过符号表示和领域知识的获取与应用，在构建专家系统方面取得了显著的成就。

1.3.2 联结主义学派

联结主义学派又称仿生学派或生理学派。

联结主义学派认为，人工智能源于仿生学，特别是对人脑模型的研究，人的思维基元是神经元，而不是符号处理过程。连接主义学派经历了三个阶段。

第一阶段：20 世纪 60 至 70 年代，对以感知机(Perceptron)为代表的人脑模型的研究曾出现过热潮。

第二阶段：20 世纪 80 年代，霍普菲尔德教授提出用硬件设备模拟神经网络；1986 年，鲁梅尔哈特等人提出 BP(Back Propagation)神经网络算法来模拟人脑分析问题和决策问题。

第三阶段：进入 21 世纪后，联结主义学派推动了深度学习理论。

1.3.3 行为主义学派

行为主义学派是西方心理学的主要流派之一，其学习理论主要是联结论和刺激反应论。

行为主义学派把目标聚焦在相对低等的生物身上，他们发现昆虫这种比人类简单得多的生物也表现出了非凡的能力，比如，可以灵活地行走，躲避障碍物，快速精准地捕食猎物，等等。从这点出发，行为主义学派模仿动物的行为，在不需要大脑干预的情况下，仅凭四肢和关节的协调来适应环境。行为主义学派大概经历了三个阶段。

第一阶段：20 世纪 40～50 年代，受控制论思想的影响，早期的人工智能专家维纳(Wiener)和麦克洛克(McCulloch)等人提出的控制论和自组织系统，以及钱学森等人提出的工程控制论和生物控制论影响了许多领域。

第二阶段：20 世纪 60～70 年代，播下智能控制和智能机器人的种子，并在 20 世纪 80 年代诞生了智能控制和智能机器人系统。

第三阶段：20 世纪末，行为主义学派以人工智能新学派的面孔出现，引起许多人的兴趣。这一学派在机器人领域成果卓著，例如，美国波士顿动力公司模仿小狗的行为研发出“Big Dog”，模仿人的开门行为研发出“开门机器人”。

1.4 人工智能编程语言——Python

在人工智能研究和应用上，主要有 Python、Java、C++及 R 等编程语言，而 Python 是一种简单易学、功能强大、兼容性强且被广泛应用的编程语言，其被广泛应用于人工智能算法的研究和实现，逐渐成为人工智能领域的主流编程语言之一，深受广大人工智能研究

者的青睐。Python 语言具有如下三大优势。

1) 拥有丰富的数据处理库

Python 语言在科学计算和数据处理方面有着丰富的扩展库，如 NumPy、Pandas、Matplotlib 和 SciPy，这些扩展库提供了丰富的数据处理和数据分析函数，使人工智能算法的实现更加便捷、高效。

NumPy 扩展库提供了丰富的数组和矩阵处理函数，主要包括数组索引、数组切片、数组数学运算、CSV 存取数组数据、矩阵变换、矩阵运算等函数，为数值计算提供了强大的支持。

Pandas 扩展库支持多种格式(如 CSV、Excel 格式)的文件，可以方便地从文件中读写数据，以及进行数据的筛选、排序、聚合、统计与数据可视化等操作，为数据处理提供了丰富的工具。

Matplotlib 扩展库提供了各种画图函数(比如画折线图、柱形图等)，可以使数据以图表的形式呈现给用户，大大增强了数据的表达效果。

2) 拥有强大的学习库

Python 语言拥有丰富的机器学习和人工智能扩展库，如机器学习库 Scikit-learn、神经网络库 TensorFlow、高级神经网络库 Keras 和开源深度学习库 PyTorch，提供了丰富的机器学习和深度学习算法与模型。

其中，Scikit-learn 扩展库拥有强大的机器学习算法集，比如线性回归算法、逻辑回归算法、k 均值算法、决策树算法、支持向量机算法、朴素贝叶斯算法、随机森林算法等；同时，Scikit-learn 扩展库还提供了丰富的试验样本数据集，分别为通用数据集和真实世界中的数据集，这些数据集对应的函数信息可参考本书附录。

(1) 通用数据集。

通用数据集包含波士顿房价数据集、鸢尾花数据集、糖尿病数据集、手写数字集、体能训练数据集、红酒数据集、威斯康星辛州乳腺肿瘤细胞数据集等七个数据集。

(2) 真实世界中的数据集。

真实世界中的数据集主要包括 Olivetti 脸部图像数据集、做好类别标签的人脸数据集、森林植被类型数据集、路透社新闻语料数据集、网络入侵检测数据集、物种分布数据集等。

机器学习算法一般会把数据集按一定的比例(比如 7∶3 或 7.5∶2.5 或 8∶2)分成两部分，一部分是训练数据，用于训练和构建数据模型；另一部分是测试数据，用于评估模型和测试数据。

TensorFlow 和 PyTorch 库提供了丰富的图像识别、语音识别、自然语言处理等人工智能函数库，使开发人员能够更轻松地研究和开发人工智能的相关功能。

3) 拥有丰富的计算机视觉库

Python 语言还拥有丰富的自然语言处理和计算机视觉库，如 NLTK 和 OpenCV，这些扩展库提供了文本数据和图像数据处理的算法及工具，使人工智能系统可以更好地理解和处理自然语言与图像。

1.5 人工智能的关键技术

人工智能以计算机视觉、机器学习、感知技术等技术为基础进行感知环境、自我学习、自我分析、自我决策、自我执行等人工智能行为。人工智能涉及的关键技术主要包括计算机视觉、知识图谱、自然语言处理、机器学习、感知技术等。

1.5.1 计算机视觉

计算机视觉是通过摄像头或计算机应用程序对生物视觉的一种模拟，从而实现对事物的形状、大小等特征进行感知、识别、分类。确切地说，计算机视觉就是让摄像头或计算机代替人眼对事物进行检测、识别、跟踪等操作。

根据计算机视觉的目标任务，可以把其分为图像检测、图像分割、图像分类等任务。

1) 图像检测

图像检测也称目标检测，是检测图像中含有特定物体目标，并获取这个目标的类别、大小、位置、姿态等信息的过程。图像检测广泛应用在人脸检测盒、车牌检测、文字检测、商品检测方面。比如人脸检测，即采用一定的方法在一幅图像中检测出含有人脸的区域、大小、姿态等信息。

2) 图像分割

图像分割可以形象地理解为精细化的目标检测，是把图像分割成一个或多个特定的、具有独特性质的像素区域并获取特定目标的过程。

3) 图像分类

图像分类也叫图像识别，是将检测到的图像目标和文件或数据库中的某个样例对应起来，并将此图像归为文件或数据库中的某个类别，或数据库中的某个物体或某个人，从而完成图像的识别功能。

1.5.2 知识图谱

知识图谱是结构化的语义知识库，是一种由节点和边组成的用图表示的数据结构，即用直线、椭圆形等图形符号描述物理世界中的实体及其相互关系。知识图谱基本组成单元包括“实体—关系—实体”三元组、实体及其相关“属性—值”组。

在知识图谱中，每个节点表示现实世界的“实体”，每条边为实体与实体之间的“关系”，不同实体之间通过关系构成网状的知识结构。

知识图谱被广泛用在打击团伙诈骗、不一致性验证、搜索引擎、可视化展示和精准营销等方面。但是，知识图谱的发展面临很大的挑战，比如，数据本身有错误或者数据存在冗余，将会严重影响知识图谱的准确表示。

1.5.3 自然语言处理

自然语言处理(Natural Language Processing，NLP)是人工智能的一项关键技术，是指利用计算机对自然语言的字形、发音、含义等信息进行处理，即对文字、词语、句子、段落进行输入、输出、识别、分析、理解、生成等操作。自然语言处理分成两部分：自然语言理解(Natural Language Understanding，NLU)和自然语言生成(Natural Language Generation，NLG)。

(1) 自然语言理解旨在理解和分析人类语言，重点关注对文本数据的理解，通过对其分析和处理来提取相关信息，理解文本的真实意图。

(2) 自然语言生成是用自然语言文本来表达给定的意图、思想等，简单地说，就是生成包含意义和结构的短语、句子及段落的过程，是自然语言处理的一个不负责理解文本的领域。为了生成自然语言，自然语言生成的方法需要利用相关数据。

自然语言处理可以对词、句子、篇章进行分析，还可以对内容里面的人物、时间、地点等进行理解，并在此基础上支持一系列核心技术(如跨语言的翻译、问答系统、阅读理解、知识图谱等)。基于这些技术，又可以把它应用到其他领域，如搜索引擎、客服、金融、新闻等。自然语言处理技术不是一项独立的技术，它需要云计算、大数据、机器学习、知识图谱等各个方面的支撑。

1.5.4 机器学习

机器学习(Machine Learning，ML)是人工智能的一大关键技术，是一门涉及数理统计、

神经网络、计算机科学、心理学、生物科学等诸多学科的交叉学科，也是研究计算机模拟或学习人类行为，以获取知识或技能，不断改善自身性能的学科。根据不同的学习方法，我们可以把机器学习分为传统机器学习、深度学习及强化学习等机器学习方法。

1) 传统机器学习

传统机器学习方法是指试图从一些训练样本出发，挖掘样本数据本身真实存在的特点或规律，从而实现对测试数据行为或趋势的准确预测。

传统机器学习方法主要包括线性回归、隐马尔科夫模型、支持向量机、k 近邻算法、人工神经网络、Adaboost 算法、贝叶斯方法以及决策树等方法。

传统机器学习方法应用广泛，主要应用在自然语言处理、语音识别、图像检测、图像分类、信息检索等领域。

2) 深度学习

深度学习又称深度神经网络(指层数超过三层的神经网络)，经过多年的摸索和研究，已经产生了诸多深度神经网络的模型。其中，卷积神经网络、循环神经网络是两类典型的模型。

目前，流行的深度学习方法包括深度置信网络、卷积神经网络、受限玻尔兹曼机和循环神经网络等。主流的深度学习框架有 TensorFlow、Caffe/Caffe2、CNTK、Torch/PyTorch、百度飞桨深度学习平台(PaddlePaddle)等。

3) 强化学习

强化学习(Reinforcement Learning，RL)又称评价学习或增强学习，是机器学习中的一种方法，主要用于描述和解决智能体在与环境的交互过程中通过某个学习策略以获得回报最大化或实现某个目标的问题的方法。

强化学习理论受到行为主义心理学理论的启发，侧重在线学习，并试图在探索—利用间保持平衡。不同于监督学习算法，强化学习不要求预先给定任何带类别标签的数据，而是通过环境对动作的反馈进行知识学习和行动决策。

强化学习方法在围棋、网络游戏、自动控制等领域中得到广泛应用，被用于解释有限理性条件下的平衡态，设计出最理想的推荐系统和完美的机器人交互系统。

1.5.5 感知技术

感知技术是人工智能硬件方面的关键技术，它包括传感器技术、射频识别技术和卫星定位技术等。

(1) 传感器技术扮演着对外部环境信息进行采集、处理、存储、传递的角色，有效地为

人工智能的应用提供有用的信息。常用的传感器有温湿度传感器、加速度传感器、视觉传感器、红外热释电传感器、毫米波雷达等。

(2) 射频识别(Radio Frequency Identification，RFID)是一种非接触的自动识别技术，通过射频信号自动识别目标对象并获取目标对象的相关信息的技术。射频识别技术无须人工干预，可工作在任何环境中，即使在严寒或炎热的环境中也能正常工作。RFID 系统一般由电子标签、阅读器、天线等部件构成。

(3) 卫星定位技术是一种使用卫星进行位置准确定位的技术，由最初的定位精度低、不能实时定位的卫星定位系统，发展成如今高精度、实时性强的全球卫星定位系统，可以在任何地方、任何时刻进行实时精确定位和准确导航。

迄今为止，全球知名的卫星定位系统共有四个，分别为美国的全球定位系统(GPS)、俄罗斯的格洛纳斯卫星导航系统(GLONASS)、欧盟的伽利略卫星导航系统(GALILEO)、中国的北斗卫星导航系统(BDS)。

1.6 人工智能的应用场景

人工智能时代已经到来，人工智能应用场景越来越多，主要运用在图像识别、语音识别、无人驾驶、聊天机器人、智能家居等场景中。

1.6.1 图像识别

图像识别是人工智能的一大应用场景，它是利用计算机对物体图像进行分类的一种技术，确切地说，通过图像识别技术可以识别出该图像真实表达的物体。比如人脸识别，通过识别出不同人脸信息就可以进行身份认证、安全检查与移动支付。又如商场的机器收银员，通过识别出顾客选购的商品的二维码对选购的商品进行结算等处理。

一个完整的图像识别系统需要经历五个阶段，包括图像数据采集阶段、图像预处理阶段、特征及选择提取阶段、分类器设计阶段、分类决策阶段。

1.6.2 语音识别

语音识别是以语音、声音、声波为对象，通过声波信号处理和模式识别知识让机器识别和理解人类的口语。中国物联网校企联盟形象生动地把语音识别系统比作“机器的听觉系统”。

语音识别过程包括两个阶段，具体如下。

1) 特征提取阶段

这个阶段把声音转换成机器可以处理的声波信号，并从每帧声波中提取出多组声波特征向量。

2) 声波解码阶段

声波中提取出的特征向量先后经过声学模型解码、发音字典解码、语言模型解码一系列过程，根据不同语言种类分析出概率最大的词组或句子，这样就完成了从声音到文字的转换。

语音识别的应用范围非常广泛，比如语音翻译、语音开关灯、语音导航、语音播放音乐、语音对话等应用场景。在小轿车中语音识别技术得到了充分的应用，在驾驶小轿车途中，只需对着麦克风说话，小轿车驾驶系统就可以将其识别为相应的文字。

语音是人类沟通最自然、便捷的手段，语音交互比其他交互方式具备更多优势，能为人机交互带来根本性变革，是大数据和认知计算时代未来发展的关键技术，具有广阔的发展和应用前景。

1.6.3 无人驾驶

无人驾驶是人工智能的核心应用场景，它是通过机器学习方法、传感器技术、卫星定位技术等实时感知无人驾驶车周边物体(包括小车、行人、动物及交通标识等)，检测路况，规划路径等操作，从而对车辆灯光、行驶速度、转向幅度进行实时控制的一种技术。

无人驾驶采用三层架构模式，主要包括无人驾驶感知层、无人驾驶决策层及无人驾驶执行层，不同层级相互作用、相互配合，从而达到无人驾驶的目的。

1) 无人驾驶感知层

在无人驾驶系统中，感知系统就相当于人类的眼睛、耳朵等感觉器官，通过机器学习知识和传感器技术来实时感知车辆周围环境的变化。

无人驾驶感知层需要使用多种类型的传感器，主要包括计算机视觉传感器、毫米波雷达、超声波雷达、红外传感器，以及用于定位和导航的 GPS 和惯性测量单元等传感设备。

2) 无人驾驶决策层

在无人驾驶系统中，决策层是无人驾驶真正发挥优势的部分，它的作用就像无人驾驶的大脑，像人类驾驶员一样做出路线规划、灯光控制、车速控制及转向幅度控制等决策方案。

3) 无人驾驶执行层

在无人驾驶系统中，执行层是无人驾驶系统根据无人驾驶决策层做出的决策方案对车

辆的灯光、行驶速度、转向幅度等进行控制。

无人驾驶车的各个操控系统都需要通过总线与决策系统相连接，并能够按照决策系统发出的总线决策指令精确地控制加速度、制动程度、转向幅度、灯光等驾驶操作，以实现车辆的自主驾驶。

相对于传统燃油车，电动车更适合作为自动驾驶汽车，执行层就是自动驾驶的手和脚。无人驾驶汽车运动控制分为纵向控制和横向控制两大部分。

1) 纵向控制

纵向控制也叫车速控制，即油门加减速、刹车等操作，主要包括驱动与制动控制，通过对电机、发动机、传动系统和制动系统的控制实现对小轿车的纵向控制。

2) 横向控制

横向控制也叫方向控制，目标是使汽车自动按照期望或规划好的路线行驶。

1.6.4 聊天机器人—— ChatGPT

ChatGPT(Chat Generative Pre-trained Transformer)是由美国OpenAI公司研发的聊天机器人程序，于2022年11月30日发布。

ChatGPT 是一款自然语言处理工具，它能够通过理解和学习人类语言进行聊天对话，还能根据聊天的上下文进行互动，像真人一样交流、聊天，完成写文档、翻译、写代码、写文章、识别语音等多种任务。作为一种基于人工智能技术的自然语言生成模型，ChatGPT能够从大量的数据和历史对话中学习，构建一个强大的深度学习模型，然后生成与人类语言相似的优美词句。ChatGPT 的工作步骤包含数据收集、数据预处理、数据模型建立、生成文本和文本输出控制等五大步骤。

1.6.5 智能家居

智能家居(Smart Home)又称电子住宅(Electronic Home，E-Home)，是指以居民住宅为中心，利用综合布线技术、物联网技术、安防技术、自动控制技术及人工智能技术把涉及家居生活的设施进行集成的一套智能化家居系统。

智能家居系统包括家庭布线系统、家庭网络系统、中央控制系统、家居照明系统、多媒体系统、家庭安防系统、家庭环境控制系统、人工智能系统等八大系统。其中，中央控制系统、家居照明系统、家庭安防系统是智能家居的基础必备系统，家庭布线系统、家庭网络系统、多媒体系统为可选系统，人工智能系统是智能家居的关键系统。

习题

1. 什么是人工智能？人工智能有哪些特征？
2. 阐述人工智能的主要学派。
3. 符号主义学派和联结主义学派有什么区别？
4. 人工智能有哪些关键技术？
5. 人工智能有哪些应用场景？
6. 结合生活中的人工智能应用，举例说说人工智能应用的例子。

第 2 章

人工智能编程语言之 Python

2.1 搭建人工智能 Python 环境

Python 语言以灵活、容易上手、功能强大等优点受到了广大科研工作者的青睐。此外，Python 语言还具有跨平台的兼容性等特点，可以应用在 Windows、Linux、Mac OS 等操作系统上。

搭建人工智能开发环境需要安装 Python 软件，以及 tensorflow、numpy、scipy、opencv、pillow、matplotlib、h5py、keras、imageai 等扩展库。Python 与 tensorFlow、keras 版本匹配如表 2.1 所示。

表 2.1 Python 与 tensorflow、keras 版本匹配

序 号	框 架	匹配描述
1	tensorflow 2.2	tensorflow 2.2.0 + keras 2.3.1 on Python 3.7
2	tensorflow 2.1	tensorflow 2.1.0 + keras 2.3.1 on Python 3.6
3	tensorflow 2.0	tensorflow 2.0.0 + keras 2.3.1 on Python 3.6
4	tensorflow 1.15	tensorflow 1.15.0 + keras 2.3.1 on Python 3.6
5	tensorflow 1.14	tensorflow 1.14.0 + keras 2.2.5 on Python 3.6
6	tensorflow 1.13	tensorflow 1.13.0 + keras 2.2.4 on Python 3.6
7	tensorflow 1.12	tensorflow 1.12.0 + keras 2.2.4 on Python 3.6
8	tensorflow-1.12	tensorflow 1.12.0 + keras 2.2.4 on Python 2
9	tensorflow 1.11	tensorflow 1.11.0 + keras 2.2.4 on Python 3.6
10	tensorflow 1.10	tensorflow 1.10.0 + keras 2.2.0 on Python 3.6
11	tensorflow 1.9	tensorflow 1.9.0 + keras 2.2.0 on Python 3.6
12	tensorflow 1.8	tensorflow 1.8.0 + keras 2.1.6 on Python 3.6
13	tensorflow 1.7	tensorflow 1.7.0 + keras 2.1.6 on Python 3.6
14	tensorflow 1.5	tensorflow 1.5.0 + keras 2.1.6 on Python 3.6
15	tensorflow 1.4	tensorflow 1.4.0 + keras 2.0.8 on Python 3.6.6 + h5py-2.10.0
16	tensorflow 1.3	tensorflow 1.3.0 + keras 2.0.6 on Python 3.6

现以在 Windows 7 64 位操作系统上安装 Python 3.6.6、PyCharm 2021.2.1、tensorflow 1.4.0 + keras 2.0.8 on Python 3.6.6+h5py-2.10.0 为例，搭建 Python 语言开发环境。

2.1.1　搭建 Python 环境

下载安装 Python 软件的方法如下。

第 1 步　我们可以登录 Python 官网 https://www.python.org，下载 Python 3.6.6(64 位)。首先，单击 Downloads 菜单，进入下载页面，如图 2.1 所示。

图 2.1　Python 官网首页

第 2 步　进入 Downloads 页面后，单击 Windows 链接，如图 2.2 所示。

图 2.2　单击 Windows 链接

第 3 步 进入 Python(Windows 版本)下载页面后，通过上下拉动页面滚动条找到 Python 3.6.6，单击 Download Windows x86-64 embeddable zip file 选项进行下载，如图 2.3 所示。这个版本里面支持 Windows 7 64 位和 Windows 7 32 位操作系统。

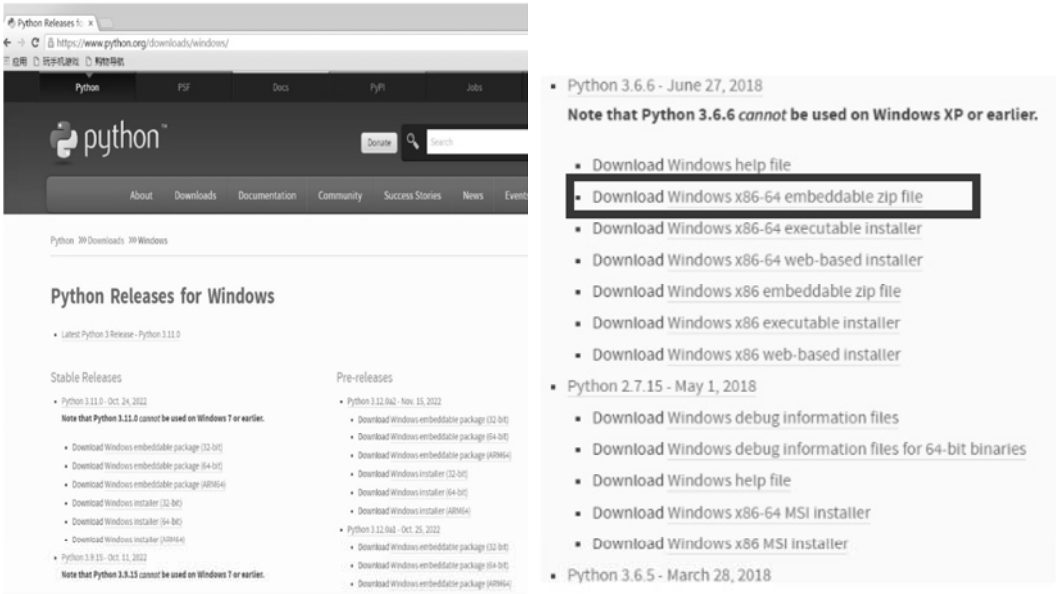

图 2.3　Python(Windows 版本)下载页面

第 4 步 安装 Python 3.6.6(64 位)软件。以“管理员”身份运行 Python 3.6.6(64 位)软件，出现如下界面，选择 Customize installation 选项，接下来出现 Documentation、pip、td/tk and IDLE 等选项，如图 2.4 所示。全部选中(默认已打 √)，直接单击 Next 按钮继续安装。

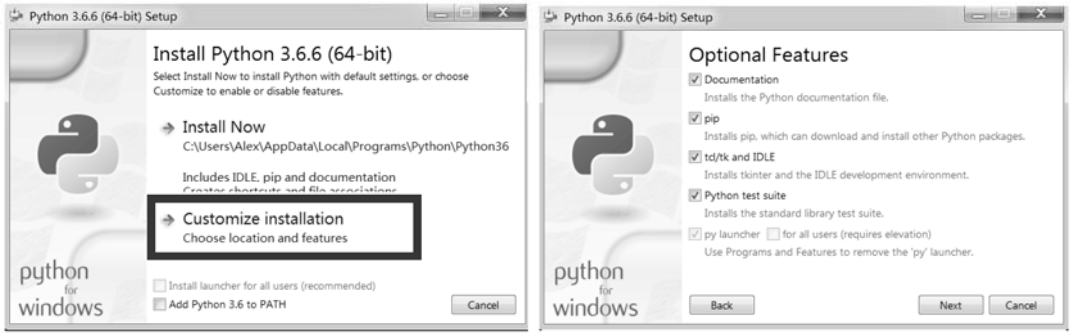

图 2.4　Python 3.6.6 安装界面

第 5 步 接下来出现安装路径选项，我们只需把安装路径设为 C:\Python36，如图 2.5 所示，然后单击 Install 按钮进行安装即可。

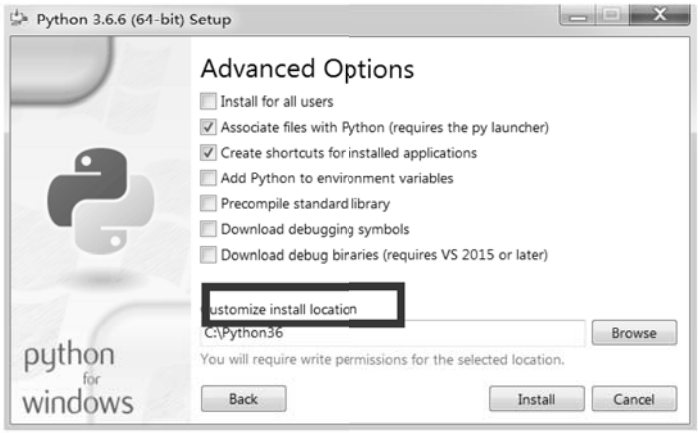

图 2.5 设置 Python 3.6.6 安装路径

第 6 步 当安装界面出现“Setup was successful”提示，表示成功安装了 Python 3.6.6(64 位)软件。

第 7 步 设置 Python 环境变量。

如果添加了 C:\python36 环境变量，就可以通过 cmd→python 操作快捷进行 Python 语言编程；如果没有对 Python 进行环境变量配置，则执行 cmd→python 操作会提示找不到 Python 命令。

如果添加了 C:\python36\Scripts 环境变量，那么可以打开“运行”对话框，输入 cmd 后按 Enter 键，进入指令输入界面，然后输入 cd C:\python36\scripts，切换到该环境变量下，调用 pip3 进行安装或卸载相关扩展模块。

打开“环境变量”对话框的方法为：右击桌面上的“计算机”图标，依次选择“属性”→“高级系统设置”→“环境变量”选项，打开的对话框如图 2.6(左)所示。

设置 Python 环境变量的方法为：在“环境变量”对话框的“系统变量”列表框中选中 Path 选项，单击“编辑”按钮，弹出“编辑系统变量”对话框，在“变量值”文本框中添加如下 Python 环境变量：“C:\python36;C:\Python36\Scripts”，如图 2.6(右)所示。

第 8 步 测试 Python 程序是否运行成功。执行 cmd→python 操作后，输入如下 Python 语言代码：print("这是我的 Python 程序")，然后按 Enter 键，如果运行结果如图 2.7 所示，则表示 Python 安装成功。

图 2.6　设置 Python 环境变量

图 2.7　测试 Python 程序是否运行成功

2.1.2　搭建 PyCharm 环境

PyCharm 是一款功能强大、方便好用的 Python 编辑软件，它提供了调试、语法高亮显示、项目管理、代码跳转、智能提示、自动完成、单元测试及版本控制等工具，可以帮助用户在进行 Python 语言开发时提高工作效率。

PyCharm 软件有 Professional(专业版)和 Community(社区版)两类，其中，专业版是收费的，而社区版是免费的。下面我们就介绍一下 PyCharm 社区版的具体安装方法，现以 PyCharm Community 2021.2.1 为例进行安装。

第 1 步 登录 PyCharm 官网(https://www.jetbrains.com/zh-cn/pycharm/)，在页面右上角的搜索框中输入 pycharm-community-2021.2.1，然后选择“其他版本-PyCharm”选项，如图 2.8 所示。

图 2.8　PyCharm 官网首页

第 2 步 进入 PyCharm 下载页面后，通过页面下拉滚动条找到 PyCharm Community Edition 的 2021.2.1 - Windows (exe)版本进行下载，如图 2.9 所示。

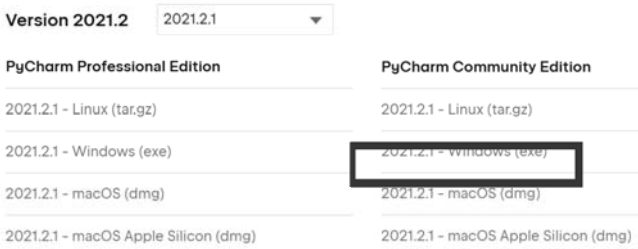

图 2.9　选择安装版本

第 3 步 安装 PyCharm 软件。

打开 PyCharm Community Edition 软件，以“管理员”身份运行，接下来都是默认安装，不需要设置，如图 2.10 所示。

第 4 步 使用 PyCharm 工具编写 Python 程序。

在 PyCharm 开发工具启动后，若要编写程序，需要先创建一个项目。选择 File→New Project 命令，命名项目名称为 Test。

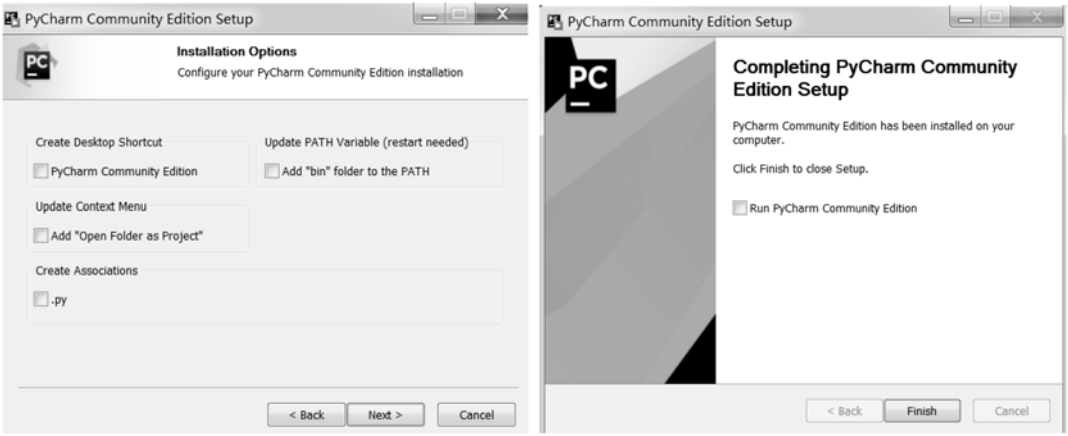

图 2.10　PyCharm 软件安装界面

选中这个 Test 项目，然后执行 File→New 命令，在打开的对话框中选择文件类型为 Python，设置文件名为 test，这样就新建好了一个 Test.py 文件。

输入简单程序：print("这是用 Pycharm 工具编写的第 1 个 Python 程序")，然后运行 Run →Test 命令，运行结果如图 2.11 所示。

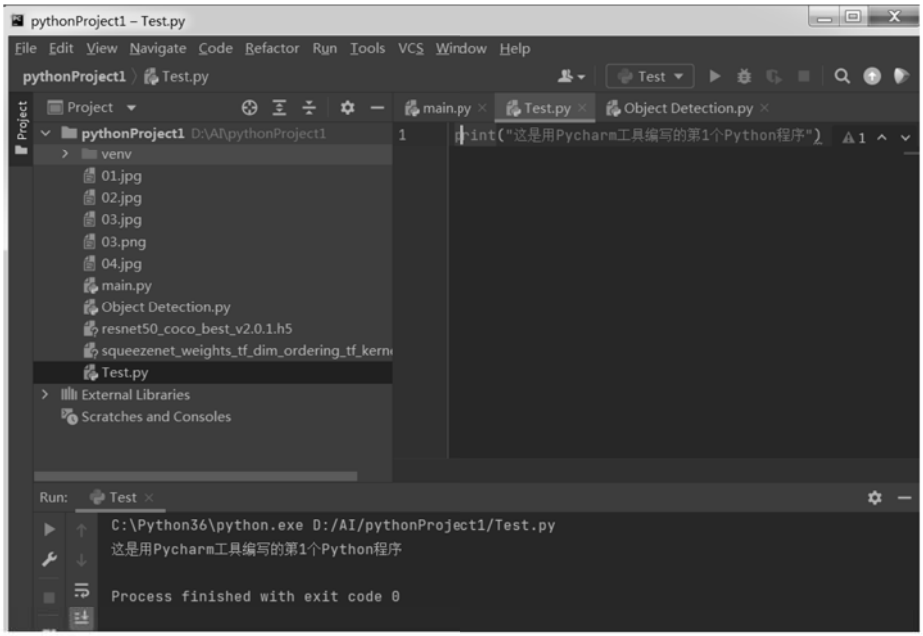

图 2.11　PyCharm 软件 Python 程序测试

2.1.3 安装扩展模块库

要安装 imageai 模块，需要安装对应的 tensorflow、numpy、scipy、opencv-python、pillow、matplotlib、h5py-2.10.0、keras2.0.8 扩展库及 imageai 插件。如果插件版本不匹配，可能引起 Python 程序编译错误，故把 imageai 和各插件名称及版本信息罗列如表 2.2 所示。

表 2.2 imageai 模块的插件列表

序 号	待安装的插件名称	备 注
1	tensorflow-1.4.0-cp36-cp36m-win_amd64.whl	人工智能工具
2	numpy	数组操作函数库
3	scipy-1.5.4	数值计算库
4	opencv-python	计算机视觉库 (比较复杂)
5	pillow	图像处理库
6	matplotlib-3.1.2-cp36-cp36m-win_amd64.whl	Python 绘图库
7	h5py-2.10.0	文件格式库
8	keras2.0.8	人工神经网络库
9	imageai-2.0.2-py3-none-any.whl	图像识别库 (比较简单)

(1) 设置插件下载或升级的服务器地址。

Python 语言中比较常用的一个命令是 pip(Python Package Index，Python 软件包索引)，它可以安装、卸载和管理 Python 软件包。pip3 专门用于 Python3，同样可以安装、卸载和管理软件包。设置插件下载或升级的服务器地址有以下两种方法。

方法 1：采用默认的国外服务器进行下载或升级插件：如果不设置服务器地址，默认采用国外服务器地址，下载速度比较慢，容易失败。

方法 2：采用国内服务器地址下载或升级插件：需要设置其服务器地址。以设置清华大学服务器为例，通过 cmd 命令进入控制台，然后采用 cd C:\python36\scripts 把当前目录切换到 c:\python36\scripts 目录下，并输入如下指令，如图 2.12 所示。

```
Python -m pip config set global.index-url
https://pypi.tuna.tsinghua.edu.cn/simple
```

下载速度比较快的国内服务器地址如表 2.3 所示。

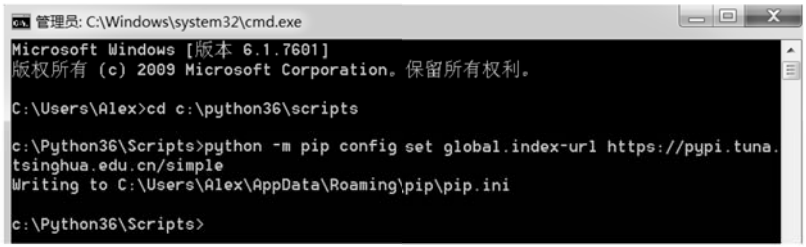

图 2.12　CMD 控制台界面

表 2.3　国内下载速度比较快的服务器地址列表

序　号	服 务 器
1	http://mirrors.aliyun.com/
2	https://pypi.tuna.tsinghua.edu.cn/simple
3	https://pypi.mirrors.ustc.edu.cn/simple/
4	http://pypi.hustunique.com/

(2) 删除插件下载或升级的服务器地址。

如果要把国内服务器地址删除，在 C:\Users\Alex\AppData\Roaming\pip 文件夹中找到 pip.ini 文件，将以下这行内容删除即可。

```
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
```

(3) 升级 pip 至最新版本。

如果不升级 pip 的话，可能会导致安装或升级某些插件速度比较慢，甚至不能顺利安装，故通过如下命令升级 pip。

```
pip  install --upgrade pip
```

(4) 安装 tensorflow 插件。

登录 tensorflow 官网，找到 tensorflow-1.4.0-cp36-cp36m-win_amd64.whl 进行下载，下载后把这个文件拷贝至 C:\python36\scripts 目录下，然后输入 pip3 命令，安装 tensorflow，具体指令为：

```
pip3  install  tensorflow-1.4.0-cp36-cp36m-win_amd64.whl
```

(5) 安装 numpy 插件。

```
pip3  install  numpy
```

(6) 安装 scipy-1.5.4 插件。

```
pip3 install scipy
```

(7) 安装 opencv-python 插件。

```
pip3 install opencv-python
```

(8) 安装 pillow 插件。

```
pip3 install pillow
```

(9) 安装 matplotlib-3.1.2 插件。

登录 matplotlib 官网，下载 matplotlib-3.1.2-cp36-cp36m-win_amd64.whl 版本，下载完后把这个插件复制到 C:\python36\scripts\目录下，然后输入如下命令：

```
pip3 install matplotlib-3.1.2-cp36-cp36m-win_amd64.whl
```

(10) 安装 h5py-2.10.0 插件。

```
pip3 install h5py-2.10.0
```

(11) 安装 keras2.0.8 插件。

```
pip3 install keras2.0.8
```

(12) 安装 imageai-2.0.2 插件。

```
pip3 install imageai-2.0.2-py3-none-any.whl
```

(13) 查看已安装的插件。

```
pip3 list
```

2.2 Python 程序简介

Python 语言是由荷兰程序员吉多·范罗苏姆(Guido van Rossum)于 1989 年设计的，是一种面向对象的程序设计语言；Python 语言具有简单易学、易读、易维护、用途广泛、运行速度较快、免费开源、可移植、可扩展等优点；Python 语言可以在不同领域中跨平台应用，受到广大科学研究人员的青睐。

2.2.1 Python 常量

在 Python 语言中，变量用来记录事物变化的状态，变量的值可以随时改变；常量用来记录事物不变的状态，常量的值不会经常改变。

实际上，在 Python 中没有真正意义上的常量，所有的名字都是变量，常量就是不变的变量，我们一般将大写的变量名看成常量，如例 2.1 所示。

常量定义的一般语法规则如下。

```
常量名 赋值符号 数据值
```

例 2.1：

```
HOST = '10.51.3.2'          # 定义一个 HOST 常量，常量值为 10.51.3.2
print(HOST)                 # 通过 Python 语言自带的 print()输出这个常量值
```

输出结果：

```
10.51.3.2
```

2.2.2 Python 变量

1) Python 变量定义的语法规则

变量和常量其实在使用的时候是一样的，无论是变量还是常量，在定义时都会给它分配一块内存空间，用于保存它的值。变量定义的一般语法规则如下。

```
变量名(常量名) 赋值符号 数据值
```

例 2.2：

```
Age = 30;
```

2) Python 变量名或常量名的命名规范

(1) 变量名字里面可以出现数字、字母、下划线。

例 2.3：

```
C2_Age = 20  #(√) 变量名定义正确
```

(2) 首字母不能用数字。

例 2.4：

```
888B = 123   #(×) 变量名定义错误
```

(3) 名字不能与 Python 中的关键字冲突。

例 2.5：

```
for = 99     # (×) 变量名定义错误
```

(4) 名字尽量做到见名知意，即按照英文单词来定义变量名，从名称就可以知道内容的意义，不能用中文。

例 2.6：

```
Age = 18  #(√) 定义一个年龄，完美
```

3) Python 语言变量命名风格

(1) 下划线(Python 推荐使用)。

```
user_name  # 每个单词之间用下划线连接
```

如果名字太长，那么每个单词可以简写。例如，user_name 可以简化成 usr_N。

(2) 驼峰体(JavaScript、Java 推荐使用，不用下划线)。

小驼峰：

```
userNameFro = '李白'   # 第一个单词小写，第二个及以后的单词首字母大写
```

大驼峰：

```
UserNameFro = '李白'   # 每个单词首字母大写
```

2.2.3 Python 注释

在编程语言中，注释就是对所写代码的解释说明。程序编译时，不对注释进行编译；程序运行时，也不运行这些注释内容。

1) 单行注释(行注释)

Python 中使用“#”表示单行注释。单行注释可以作为单独的一行放在被注释代码行之上，也可以放在语句或表达式之后。

例 2.7：

```
UniversityName = '清华大学' # 大学名字为清华大学
```

这是单行注释，“# 大学名字为清华大学”为注释部分，程序编译和运行皆不理会这些注释内容。

为了保证代码的可读性，建议在“#”后面添加一个空格，再添加注释内容。

2) 多行注释(块注释)

当注释内容有多行时，我们可以使用多行注释，Python 中使用三个单引号或三个双引号表示多行注释。

例 2.8：

```
'''a = 4
   b = 3
   c =  5
   print(a+b+c)
```

这是使用三个单引号的多行注释，表示从第三个单引号后面 a=4 开始到程序最后一行 print(a+b+c)的代码区域都是注释部分。

例 2.9：

```
"""A=12
   B=13
   C=B-A
   print(C)
```

这是使用三个双引号的多行注释，表示从第三个双引号后面 A=12 开始到程序最后一行 print(C)的代码区域都是注释部分。

2.2.4 Python 数据类型

在 Python 语言中，主要包括数字、字符串、布尔、数组、集合、字典、向量等数据类型，现对这些数据类型进行介绍。

1. 数字类型

1) 整数类型——int

与 C、C++语言不同，Python 的整数类型没有长度限制。整数类型有二进制整数、八进制整数、十进制整数、十六进制整数等类型。

(1) 二进制整数：由若干 0 和 1 数字随机组合而成。二进制数用加前缀 0b 表示，比如，0b101 表示二进制数 101。

(2) 八进制整数：由若干 0～7 的数字随机组合而成。八进制数用加前缀 0o 表示，比如，0o127 表示八进制数 127。

(3) 十进制整数：由若干 0～9 的数字随机组合而成。十进制数不用加前缀，比如，128 表示十进制数 128。

(4) 十六进制整数：由若干 0～9、A～F 的字符随机组合而成。十六进制数用加前缀 0x 表示，比如，0x100AF 表示十六进制数 100AF。

2) 浮点数类型——float

浮点数类型表示含有小数位的数，采用 8 字节空间存储，它的取值范围为[−1.7E308, 1.7E308]。

2. 字符串类型——str 类型和 unicode 类型

在 Python 中，字符串是由两个或两个以上的字符组成的，采用成对的单引号、双引号、三撇号表示。字符串有两种类型，分别是 str 类型和 unicode 类型。

在进行字符串操作时，常涉及求字符串长度、字符串连接等操作。

例 2.10：

```
Name='ChenShuMing'
Company="HuaWei"
Country='''China'''
print(Name)          # print()是用来输出变量 Name 值的函数
print(Company)       # print()是用来输出变量 Company 值的函数
print(Country)       # print()是用来输出变量 Country 值的函数
```

输出结果：

```
ChenShuMing
HuaWei
China
```

3. 布尔类型——bool

布尔类型就是我们常说的逻辑类型，常常理解为 true 或 false。布尔类型常用于两个数的逻辑判定，如两数是否相等(==)、大于(>)、小于(<)的判断。

例 2.11：

```
print(88==77); print(88==66)
```

输出结果：

```
False False
```

4. 数组类型

1) 数组概述

在 Python 中，常用列表类型 list、元组类型 tuple 表示数组，数组中每个元素类型相同。常用的数组有一维数组和二维数组：一维数组由一行若干列相同类型的元素构成，元素下标从 0 开始标号；二维数组由若干行若干列相同类型的元素构成。

(1) 创建数组。

例 2.12：

```
A = [1,2,3,4]                    # 定义一个一维数组 A，里面有 1、2、3、4 共 4 个元素
B = [[1,2,3],[4,5,6],[7,8,9]]    # 定义一个三行三列的二维数组 B
```

(2) 引用数组元素。

例 2.13：

```
A =[1,2,3,4]; B=[[1,2,3],[4,5,6],[7,8,9]]
print(A[0])          # A[0]表示数组 A 的第 1 个元素
print(A[3])          # A[3]表示数组 A 的第 4 个元素
print(B[1][2])       # B[1][2]表示数组 B 的第 2 行第 3 列元素
```

输出结果：

```
1
4
6
```

(3) 给数组元素赋值。

例 2.14：

```
A =[1,2,3,4];
A[0] =88    # 给 A[0]赋值 88
print(A)
```

输出结果：

```
88  2  3  4
```

2) 列表类型

列表是 Python 特有的数据类型，在 Python 中，列表数据类型可以存储由多个值构成的序列，可以同时存储多种数据类型，也可以嵌套在其他列表中。

(1) 创建列表。

不同数据项之间由逗号分隔，整体放在一个方括号里，这就是列表。

例 2.15：

```
list = [1, 2, 3, 4, 'Alex', 'blank', [8, 5, 7]]
```

创建一个列表，名为 list，里面有整数类型、字符串类型、列表类型(如[8,5,7])。

(2) 列表元素引用操作。

列表元素引用方法：列表名 [索引] (索引从 0 开始计数)。

例 2.16：

```
list  = [1, 2, 3, 4, 'a', 'b', [8, 5, 7]]
print(list[6],list[6][0])
```

输出结果：

```
[8, 5, 7] 8
```

(3) 列表连接操作。

列表连接操作直接采用“+”进行列表连接。

当同一行中有多个 Python 语言程序时，需要用“;”隔开。

例 2.17：

```
ls1 = [1, 2, 3] ls2 = ['a','b','c']; print(ls1 + ls2)
```

输出结果：

```
[1,2,3,'a','b','c']
```

(4) 列表元素复制。

在 Python 编程中，可以通过列表乘法实现列表的快速复制。具体来说，将列表与一个整数相乘，可以生成一个新的列表，其中包含原列表元素按照指定次数的复制。

例 2.18：

```
ls = [1, 2, 3]; print(ls*3)
```

输出结果：

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

(5) 求列表长度。

通过函数 len()求得列表长度。

例 2.19：

```
ls  = [1, 2, 3, [8, 5, 7]] ;  print(len(ls))
```

输出结果：

```
4
```

3) 元组类型

元组是 Python 中另一种特有的数据类型，和列表相似，同样可以存放不同类型的数据，但它是不可变对象，即创建后就不可以进行任何元素的修改操作。

(1) 创建元组。

元组用逗号对元素进行分隔，但是为了美观及代码的可读性，一般还会加上小括号。

例 2.20：

```
a  = 1, 2, 3; b = (1, 2, 3)
print(type(a)); print(type(b)); print(a == b)
```

输出结果：

```
<class 'tuple'> <class 'tuple'> True
```

让我们看看修改不可变对象元素会报什么错。

例 2.21：

```
Tp = (1,2,3,4,'a','b',[8, 5, 7]) ; Tp[0] = 100 ; print(Tp)
```

结果报错：

```
" TypeError 'tuple' object does not support item assignment "
```

即因违反了“元组的元素值一旦被设置就不能改变”原则而报错。

(2) 索引和切片操作。

对于元组来说，只能通过索引和切片操作获取元组元素的数值，但不能修改元组里的数值。索引操作只能获取序列中一个下标对应的元素，切片操作则可以获取序列中一个范围对应的元素。

例 2.22：

```
Stu = (11,22,"alex",[(33,44)])
# 元组 Stu 第 4 个元素[(33,44)]是列表类型，这个列表类型元素(33,44)又是元组类型
print(Stu[2])
```

```
# 正数索引，取出从左边开始编号的第 3 个元素 (下标从 0 开始)
print(Stu[-1])
# 负数索引，取出从最右边开始编号的第 1 个元素 (从-1 往左编号)
print(Stu[0:2])   # 切片，取出元组下标从 0 开始的两个元素
```

输出结果：

```
alex
[(33, 44)]
(11, 22)
```

(3) 元组的方法。

元组方法包括 count()方法和 index()方法：count()方法用于计算元组的指定元素出现的次数，index()方法用于获取指定元素第一次出现的索引位置。

例 2.23：

```
Stu = (11,22,"alex",[(33,44)],22,)
print(Stu.count(22))  # 输出数值 22 在元组中出现的次数
```

输出结果：

```
2
```

例 2.24：

```
Stu = (11,22,"alex",[(33,44)],22,)
print(Stu.index(22))   # 输出数值 22 在元组中出现的下标号
```

输出结果：

```
1
```

5. 集合数据类型

集合是一种不重复的无序集，用花括号“{}”来定义。集合的元素是无序的、不可重复的，必须是不可变类型。

1) 创建集合

集合中的元素可以是数字、字符串、元组等。创建集合的方法有三种。

(1) 直接使用大括号创建。

例 2.25：

```
st = {1, 2, 3, 'python', ('a', 1)}
print(st); print(type(st))
```

输出结果：

```
{1, 2, 3, 'python', ('a', 1)}
<class 'set'>
```

(2) 使用列表或元组创建。

使用 set()函数，将不包含可变对象元素的列表或元组转换成集合数据类型。

例 2.26：

```
ls = [1, 2, 3, 4]
st = set(ls)   # 使用 set()函数将列表 ls 转换成集合数据类型
print(st); print(type(st))
```

输出结果：

```
{1, 2, 3, 4}
<class 'set'>
```

(3) 使用字符串创建。

例 2.27：

```
st = set('Python')
print(st); print(type(st))
```

输出结果：

```
{ 'y', 'P', 'o', 'n', 'h', 't'}
< class 'set'>
```

由字符串创建的集合，集合元素是所有不重复的字符，可以很直观地看出，集合的元素是无序的。

2) 集合的常用方法

(1) add(x)向集合中添加元素 x，并将其放在集合最前面。

例 2.28：

```
st = {1, 2, 3, 4}
print(st)
st.add(0)
print(st)
```

输出结果：

```
{1, 2, 3, 4}
{0, 1, 2, 3, 4}
```

(2) update()将一个可迭代对象的元素添加到集合中。

例 2.29：

```
st = {1, 2, 3, 4}
iterable = ['a', 'b', 'c']
st.update(iterable)
print(st)
```

输出结果：

```
{'b', 1, 2, 3, 4, 'c', 'a'}
```

(3) pop()将集合的第一个元素删除，并返回被删除元素的值。

例 2.30：

```
st = {1, 2, 3, 4}
iterable = ['a', 'b', 'c']
st.update(iterable)
print(st)
p = st.pop()
print(st, p)
```

输出结果：

```
{ 'b', 1, 2, 3, 4, 'c', 'a' }
{ 1, 2, 3, 4, 'c', 'a' } b
```

6. 字典数据类型

字典是一个无序、可变和有索引的集合。在 Python 中，字典用花括号“{}”编写。字典定义了键和值之间一对一的关系，但它们是以无序的方式存储的，其中，键和值之间用“:”隔开。

(1) 创建字典。

例 2.31：

```
Info = { "Name": "ChenShuMing","Sex": "men"," Country ": "ChangTing" }
print(Info);
```

输出结果：

```
{ 'Name': 'ChenShuMing', 'Sex': 'men', 'Country': 'ChangTing' }
```

(2) 访问项目。

通过在方括号内引用其键名来访问字典的项目。

例 2.32：

```
name= Info["Name"]  #获取 "Name" 键的值
```

输出结果：

```
'ChenShuMing'
```

(3) 修改键值。

通过引用其键名来修改特定项的值。

例 2.33：把"Country"键值改为"LongYan"。

```
Info = { "Name": "ChenShuMing","Sex": "men"," Country ": "ChangTing" }
Info["Country"] = "LongYan"
```

(4) 遍历字典。

使用 for 循环遍历字典。循环遍历字典时，返回值是字典的键，但也有返回键值的方法。

例 2.34：

```
Info = { "Name": "ChenShuMing","Sex": "men"," Country ": "ChangTing" }
for x in Info:
  print(x)
```

输出结果：

```
Name
Sex
Country
```

例 2.35：

```
Info =  { "Name": "ChenShuMing","Sex": "men"," Country ": "ChangTing" }
for x in Info:
  print(Info[x])
```

输出结果：

```
ChenShuMing
Men
ChangTing
```

(5) 获取字典的值。

使用 values()函数返回字典的值。

例 2.36：

```
Info = { "Name": "ChenShuMing","Sex": "men"," Country ": "ChangTing" }
for x in Info.values():
  print(x)
```

输出结果：

```
ChenShuMing
Men
ChangTing
```

(6) 遍历键和值。

使用 items()函数遍历键和值。

例 2.37：

```
for x, y in Info.items():
  print(x, y)
```

输出结果：

```
Name ChenShuMing
Sex men
Country ChangTing
```

(7) 计算字典长度。

要确定字典有多少项目(键值对)，可使用 len()方法。

例 2.38： 打印字典中的项目数。

```
Info = { "Name": "ChenShuMing","Sex": "men"," Country ": "ChangTing" }
print(len(Info))
```

输出结果：

```
3
```

(8) 添加项目。

使用新的索引键并为其赋值，可以将项目添加到字典中。

例 2.39：

```
Info =  {"Name": "ChenShuMing","Sex": "men"," Country ": "ChangTing"}
Info["ID"] = "350801"
print(Info)
```

输出结果：

```
{'Name':'ChenShuMing','Sex':'men',' Country ':'ChangTing','ID': '350801'}
```

(9) 删除项目。

pop()方法可以删除具有指定键名的项目。

例 2.40：

```
Info =  {"Name": "ChenShuMing","Sex": "men"," Country ": "ChangTing"}
Info.pop("Name")
print(Info)
```

输出结果：

```
{'Sex': 'men', ' Country ': 'ChangTing'}
```

7. 向量数据类型

向量(Vector)是存储分组数据的简单工具。向量的值是由数字、字符或逻辑数据构成的字符串，向量中的每一项就是一个元素，每个元素的数据类型相同。

1) 创建向量

可以通过中括号“[]”来创建向量，向量元素用逗号隔开。

例 2.41：创建一个出生年份向量。

```
Birthday = [1984,1986,2016,2020]
```

例 2.42：创建一个时间向量，向量里面的值为空。

```
Time = []
```

2) 向量的运算

通过如下函数可以对向量进行运算。

(1) append()函数。

append()函数可以将元素添加到向量中。

例 2.43：

```
Birthday = [1984,1986,2016,2020]
```

```
Birthday.append(2024) # 将 2024 插入到向量的末尾
print(Birthday)
```

输出结果：

```
[1984,1986,2016,2020,2024]
```

(2) len()函数。

len()函数可以查看向量中有多少个元素。

例 2.44：

```
Months = [1,2,3,4,5,6]
print(len(Months)) # 输出向量 Months 的元素个数
```

输出结果：

```
6
```

(3) sort()函数。

sort()函数可以对向量进行排序。

例 2.45：

```
Data = [30,20,50,10,90]
Data.sort()
print(Data)
```

输出结果：

```
10  20  30  50  90
```

(4) max()函数。

max()函数可以获得向量的最大值。

(5) min()函数。

min()函数可以获得向量的最小值。

例 2.46：

```
Months =[2,3,5,7,10,12]
print(max(Months))  # 输出向量 Months 中元素的最大值
```

输出结果：

```
12
```

```
Months =[2,3,5,7,10,12]
print(min(Months))  # 输出向量 Months 中元素的最小值
```

输出结果：

```
2
```

2.2.5 Python 程序结构

在 Python 语言中，一般包括顺序结构、分支结构及循环结构三种程序设计结构。

1. 顺序结构

顺序结构就是程序自上而下执行，一条语句执行完成之后再执行下一条语句，一直执行到程序的末尾。Python 顺序结构流程图如图 2.13 所示。

例 2.47：计算半径为 30 的圆面积。

```
R=30; Area=3.14*R*R;
print(" 面积为: ");
print(Area);
```

本例的顺序结构流程图如图 2.14 所示。

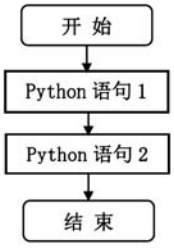

图 2.13 Python 顺序结构流程图

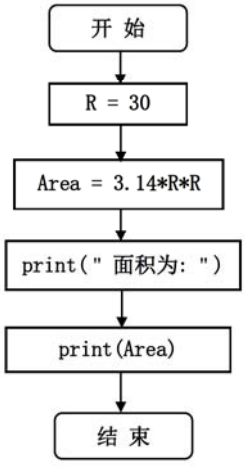

图 2.14 例 2.47Python 程序顺序结构流程图

2. 分支结构

Python 分支结构包括单分支结构、双分支结构和多分支结构。如果满足条件就执行某

条语句，否则就执行其他语句。

1) 单分支结构

单分支结构可用 if 单分支语句来实现，其一般格式为：

```
if 表达式：语句块
```

Python 单分支结构流程图如图 2.15 所示。

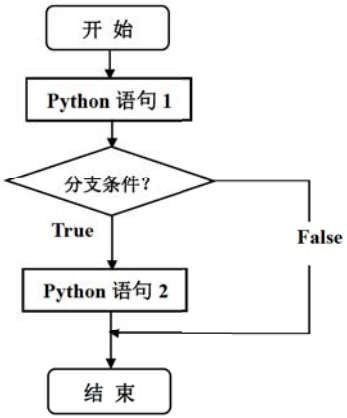

图 2.15 Python 单分支结构流程图

例 2.48：

```
age = int( input(' 请输入年龄 '))
if age >= 18:
print(" 恭喜你成年了 ")
```

2) 双分支结构

双分支结构包括两个条件，其基本格式为：

```
if 判断条件：
        执行语句 1
else:
        执行语句 2
```

其中，判断条件成立(非 0)时，就执行语句 1，否则执行语句 2。

例 2.49：从键盘上输入一个年龄数字，如果年龄大于或等于 18 岁，那么输出信息"恭喜您，您成年了!"，否则，输出信息"祝贺您，小朋友!"。

```
Grade = int(input(" 请输入年龄: "))
if  Grade >= 18:
```

```
            print(" 恭喜您，您成年了！ ")
else:
            print(" 祝贺您，小朋友！ ")
```

3) 多分支结构

多分支结构的一般格式如下：

```
if 判断条件 1:
      执行语句 1
elif 判断条件 2:
      执行语句 2
      ……
elif 判断条件 N:
      执行语句 N
```

Python 多分支结构流程图如图 2.16 所示。

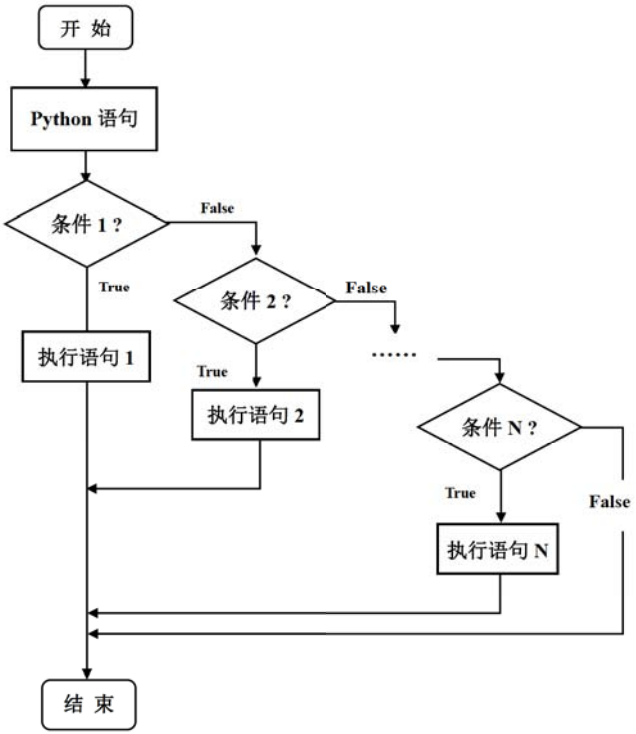

图 2.16　Python 多分支结构流程图

3. 循环结构

在 Python 编程中，循环语句用于循环执行程序，即在某种条件下，循环执行某段程序，从而处理需要重复处理的相同任务，直到不满足给定条件时，才会结束循环。Python 循环结构流程图如图 2.17 所示。

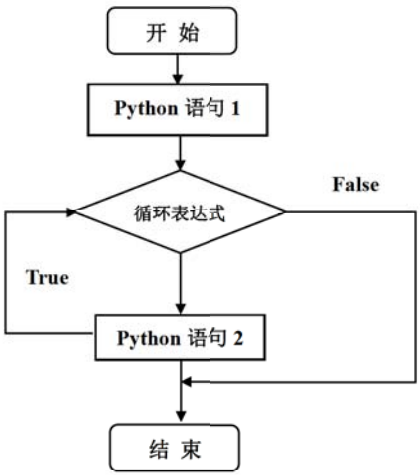

图 2.17 Python 循环结构流程图

Python 中的循环语句有 while 循环结构和 for 循环结构两种。

1) while 循环结构

在 while 循环结构中，当 while 条件为 true 时，就执行语句 1；当 while 条件为 false 时，则执行语句 2。其语法格式如下：

```
while condition:
执行语句 1
else:
执行语句 2
```

例 2.50：采用 while 循环输出小于或等于 5 的值。

```
i = 1              # 指定从 1 开始数
while i <= 5:      # 满足 i≤5 时，就接着运行这个循环
    print(i)       # 输出 i 值
    i += 1         # 给 i 值 + 1
                   # 以此类推，一旦 i 大于 5，循环就停止，整个程序也将到此结束
```

输出结果：

```
1
2
3
4
5
```

2) for 循环结构

for 循环用于顺序遍历，主要用在遍历列表、字符串或数组等操作中。在 Python 中，使用的是 for…in…循环，其语法格式如下：

```
for iterator_var in sequence:
statements(s)
```

例 2.51：采用 for 循环输出数组 A 的所有元素。

```
A=[1,2,3,4,5,6,7,8]
for i in range(0,8):
print(A[i])
```

输出结果：

```
1
2
3
4
5
6
7
8
```

2.2.6 Python 函数

在 Python 中，函数的应用非常广泛，前面调用的函数有 input()、print()、len()等，这些都是 Python 自带的函数，可以直接使用。除了 Python 自带的函数外，用户还可以自定义函数并调用自己已定义的函数。

1. Python 函数的定义

定义 Python 函数可以理解成创建一个满足用户需要的工具。定义函数一般用关键字

def 来实现，其一般格式如下：

```
def 函数名(参数列表):
    语句 1
    语句 2
        ……
return [返回值]
```

其中，返回值是可选择部分，可以是实际值，也可以省略。

例 2.52：定义一个比较数值大小的函数。

```
def  Max(a,b):
   if a> b:
      max = a
   else:
      max = b
   return max
```

2. Python 函数的调用

调用函数也就是使用函数、执行函数。如果函数已经定义，那么就可以直接调用已定义好的函数。调用函数的一般格式如下：

```
[返回值]  =  函数名 ([形参值])
```

其中，函数名是指调用函数的名称，形参值就是已定义好的函数需要传入的形参的值。

例 2.53：调用例 2.52 中的函数 Max()输出对应的值。

```
max = Max(12,99);
print(max)
```

输出结果：

```
99
```

2.3 NumPy 基础应用

人工智能的应用离不开数组计算和矩阵运算，NumPy 是 Python 的一个扩展库，支持大量的维度数组和矩阵运算，还针对方程求解、矩阵运算、求解特征值、数组计算、向量运算、直方图绘制等方面提供大量的数学函数，并与 Pandas、Matplotlib 合称为数据分析的“三

剑客”，不断推动着人工智能的发展。

2.3.1 方程求解

NumPy 不是 Python 内置的函数库，要通过如下方式安装后才可以使用。首先进入 cmd 命令窗口，然后使用 Python 的 pip3 工具执行以下命令：

```
pip3 install numpy
```

几分钟后出现如下界面就说明安装成功了，如图 2.18 所示。

```
C:\Users\Alex>pip3 install numpy
Collecting numpy
  Using cached numpy-1.19.5-cp36-cp36m-win_amd64.whl (13.2 MB)
Installing collected packages: numpy
Successfully installed numpy-1.19.5

C:\Users\Alex>
```

图 2.18　NumPy 扩展库安装成功界面

例 2.54：利用 NumPy 库求解下列二元一次方程组。

$$\begin{cases} x+2y=5 \\ 2x+y=4 \end{cases}$$

其中，x, y 为方程的位置变量，也就是即将求解的变量；$\begin{bmatrix} [1 & 2] \\ [2 & 1] \end{bmatrix}$ 为变量 x, y 的系数矩阵，$[5 \quad 4]$ 为常数矩阵，即方程等式右边的数值。

mat()函数可以将多种数据类型(包括列表、元组等)转换成矩阵形式。

解题步骤：

第 1 步　打开 PyCharm 软件，创建项目后新建一个 py 文件。

第 2 步　引入已安装的 NumPy 扩展库到本项目中。

第 3 步　利用 Python 语言求解二元一次方程组，代码如下。

```
01.  # 导入 NumPy 库，并区别名 Np
02.  import  numpy  as  Np
03.  # 利用 Np 库中的 mat()函数生成一个二维矩阵 A
04.  A=Np.mat([[1,2],[2,1]]);
05.  # 利用 Np 库中的 array()函数生成一个一维数组 B
06.  B=Np.array([5,4]);
```

```
# 利用 Np 库 linalg 模块中的 solve()函数求解二元一次方程组
X=Np.linalg.solve(A,B);
# 输出方程的解
print(X)
```

运行结果如图 2.19 所示。

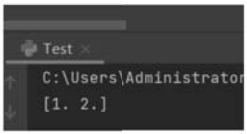

图 2.19　例 2.54 二元一次方程组的解

2.3.2　数组运算

例 2.55：某班学生成绩如表 2.4 所示，求该班学生大学语文成绩的最低分、最高分、平均值、标准值和方差。

表 2.4　某班学生成绩信息表

学　号	大学语文	大学英语	人工智能	数据库原理
100	80	90	78	80
101	81	91	79	81
102	86	92	80	82
103	83	90	80	80
104	84	100	82	84
105	90	95	90	85
106	91	96	84	70
107	87	97	80	87
108	88	98	86	88
109	89	99	87	89

本例参考代码如下。

```
# 导入 NumPy 库，并区别名 Np
import  numpy  as  Np
# 利用 Np 库中的 loadtxt()函数导入 csv 格式的文件(其中，数据与数据间用“,”隔开)
grade_data=Np.loadtxt("./data/grade.csv",delimiter=",",skiprows=1)
```

```
# 利用 Np 库中的 min()函数求得矩阵 grade_data 第 2 列数据的最小数
print('大学语文成绩最低分',Np.min(grade_data[:,1]))
# 利用 Np 库中的 max()函数求得矩阵 grade_data 第 2 列数据的最大数
print('大学语文成绩最高分',Np.max(grade_data[:,1]))
# 利用 Np 库中的 mean()函数求得矩阵 grade_data 第 2 列数据的平均数
print('大学语文成绩平均值',Np.mean(grade_data[:,1]))
# 利用 Np 库中的 std()函数求得矩阵 grade_data 第 2 列数据的标准值
print('大学语文成绩标准值',Np.std(grade_data[:,1]))
# 利用 Np 库中的 var()函数求得矩阵 grade_data 第 2 列数据的方差
print('大学语文成绩方差',Np.var(grade_data[:,1]))
```

运行结果如图 2.20 所示。

```
Test
C:\Users\Administrator\PycharmProjects\pythonProject\venv\Scripts\python.exe
大学语文成绩最低分 80.0
大学语文成绩最高分 91.0
大学语文成绩平均值 85.9
大学语文成绩标准值 3.5902646142032486
大学语文成绩方差 12.89
```

图 2.20　例 2.55 输出结果

习题

1. 采用 PyCharm 创建一个 Test.py 文件，运行输出如下结果：I Love Python。

2. 利用 Python 语言编写程序：创建一个元组，元组数据为：0 1 2 3 4 5 6 7 8 9，并逐个反序号输出元组数据：9 8 7 6 5 4 3 2 1 0。

3. 利用 Python 语言编写程序：输入一个年龄数字，如果年龄大于 18 岁，就输出：恭喜您，您已经成年了；否则输出：祝贺您，您还是少年。

4. 编写 Python 语言求解如下二元一次方程组：

$$\begin{cases} 3x+y=12 \\ 3y-x=16 \end{cases}$$

5. 编写 Python 语言求下列数据的平均值、标准值、方差。

80　90　70　60　50　70　80　85　89　90

80　90　70　60　50　70　80　85　89　90