Contents

1 Introducing artificial intelligence

Artificial intelligence offers tantalizing glimpses into a future when computers may be as smart as humans.

Artificial intelligence?

Artificial intelligence (AI) has changed our lives. It plays the stock market for us, decides if we get a bank loan, helps us to drive our cars, and even focuses our cameras. As it has moved into more and more of our devices and services, it seems increasingly mystical. Nobody ever says how it works. What is AI?

Science fiction is replete with robots that are more or less indistinguishable from humans. They think like we do or even better. The only difference is that they might not experience emotions. That's all guess-work of course; nobody has the slightest idea how to program such a robot. The robots we can build right now are about as intelligent as a snail or an ant. Soon we will be able to make them as smart as a dog. But that's not all that AI is. While some people are still investigating how to make computers and robots smart, others are using the same technology to solve problems today that even humans are incapable of handling.

When it isn't intelligence

One of the first goals was to get a computer to beat a human at chess. This, people thought, was a task that only a human could handle. In 1997, the Deep Blue computer beat the reigning chess grandmaster by winning two games and drawing three. But Deep Blue never had any pretensions of being intelligent. It was merely a fast computer and a clever program. To calculate its next move, it considered 200 million board positions every second. A human chess player evaluates only a few dozen possible moves. Deep Blue was substituting speed for skill. Even with such a restricted world, it couldn't match up to the efficiency of a human player.

Search **wikipedia.org** for more information on Deep Blue and Eugene Goostman.

Another task that people said could only be done by a human was to hold a conversation. In 2014, the Eugene Goostman program fooled a third of the judges after they held a five-minute conversation with it. Many AI researchers immediately pointed out that it was a rather simple program and only succeeded because it hid its many flaws behind pretending to be a 13 year-old Ukrainian boy for whom English was a second language. The program could be expected not to take the conversation seriously, to make jokes all the time, and to misunderstand on occasion. It didn't even try to be intelligent.

There is an old saying in the AI community: "If a computer can do it, it isn't intelligence."

But it doesn't matter

A program that can only do one thing – whether that is playing chess or deciding if it is raining – has no pretensions of being as smart as a human. That does not mean that it is useless. The same sort of technology used in Eugene Goostman is employed by the chatbots that we talk to on our bank's website, hopefully without the teenage humor. The IBM Watson program, which won the quiz show Jeopardy in 2011, is now used by medical professionals to determine the best treatment for individual patients. Watson always provides a list of possible decisions together with its level of confidence in each. That isn't true of all such programs, nor is it appropriate.

Deep learning

Some AI develops its skill without being instructed. It is trained for a single job until it can do that one task as well as possible but nobody tells it how; it has to develop the skill by itself. When it is based on neural networks it is called deep learning but there are alternatives. Depending on how it is trained, it might have flaws. For example, if the face detector in a camera is training on examples that are biased toward light skin tones then it might not detect faces with dark skin tones so well. Since nobody knows how such a program is calculating its results, it has to be taken on trust.

These algorithms are not perfect but a lot of the time they are good enough. My car knows when it is raining and it switches the wipers on. Sometimes I have to prompt it and other times, if the light is unusual, it will wipe even when it isn't raining. That isn't a serious problem and the convenience of not having to flick the lever every time is worth the occasions when it goes wrong. On the other hand, there are many times when we can't afford any mistakes. If the AI was driving the car by itself, I wouldn't want it to fail even once.

Just recently, the term "artificial intelligence" has become almost synonymous with deep learning. In this book we will build projects using both traditional AI techniques such as chatbots and expert systems, and also neural networks and deep learning. Training a deep learning program is very time consuming even on a supercomputer, so we will not be able to produce something like Chat GPT or DALL-E. However, the programs we produce will demonstrate the principles and I hope you will find them interesting.

How AI works

The history of AI dates back to the dawn of computers or even before. Perceptrons – the basic building blocks of neural networks and deep learning – were described as early as 1943. The first computers were only built in 1948. They were the size of a room, had a tiny memory of a few thousand bytes, and executed only 700 instructions per second. In comparison, a modern cell phone is small enough to hold in your hand, has a memory measured in billions of bytes, and can execute around a billion instructions per second. It stands to reason that the early computers couldn't handle complex AI tasks even if researchers knew how to program them.

The term "artificial intelligence" was invented in 1956 when a group of researchers came together for a two-month workshop at Dartmouth College, New Hampshire. Here is a very brief history of the field that shows where the projects we will be building sit in the development of AI as we know it today:

The dawn of the chatbot

In 1965, Joseph Weizenbaum wrote the ELIZA program, which pretended to be a psychiatrist talking to a patient. It was sufficiently convincing that when he tried to modify it to record conversations, the users objected because they were giving it personal information.

Expert systems

Expert systems ask the user questions and then apply rules written by subject-matter experts to provide a solution. The first of these in 1967 aided in identifying individual compounds in spectroscopy data. Another in 1969 attempted to advise doctors of the correct diagnosis for a patient, but it wasn't until the 1980s that expert systems became widespread.

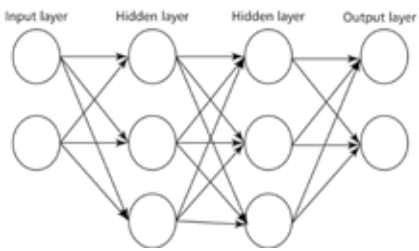
Intelligent agents

By 1987 it was becoming clear that an AI would need to handle multiple tasks at once, and trying to do that in a single program

was clumsy. Instead, a number of intelligent agents should cooperate in order to solve the problem together.

Neural networks

First described in 1943 and inspired by the neurons in human brains, perceptrons were arranged in networks in much the same way as logic gates or computer programs are. Instead, they can be laid out in layers with all of the perceptrons in one layer sending their outputs to all of those in the next layer. The whole thing can then be taught how to solve a problem by mathematically tuning the sensitivity of each input of each perceptron rather than by changing the wiring between them.



For networks consisting of more than two layers, this required an automated way to follow good and bad decisions back through the network and adjust all of the sensitivities in the correct direction. The formula for doing that wasn't found until 1980 and wasn't universal until 2010.

There have been several further advances, including changing the basic perceptron design to give it some degree of memory. This helps it handle text and music, which are streams of data rather than a single block. The arrangement of the layers can also be changed so that a network can handle two or more dimensions, such as images.

Low-code and no-code

Even with specialized libraries to help, building an AI model is complex and intricate. Many people who want to make use of AI do not have the level of skill required to program it. Applications have been created to ease this. They design a suitable form of model and then find which training method works the best for it. Some such applications still use a programming language, while others use a graphical interface.

About this book

Hot tips give some extra information on the subject being discussed.

These tips remind you about subjects earlier in the book that have a bearing on the current subject.

Notices like this warn you about difficulties you might encounter.

If you don't achieve the result illustrated in any example, simply compare your code to that in the original example files you have downloaded to discover where you went wrong.

This book is written for:

- **Enthusiasts** who want to explore AI technologies.
- **Developers** who want to use AI in their projects.
- **Programmers** who want to add AI to their skill-set.
- **Students** or those seeking a career in computing.

It does assume a familiarity with Python programming. If you are new to Python, I recommend you take the time to read **Python in easy steps** first.

When I present code for you to type in, it will be in a sans-serif font with syntax highlighting. Comments will be green; user-provided names, red; numbers or strings will be black; and everything Python provides will be cyan; like this:

```
# Find and return the first creature at a given position
def anyCollision(position):
    for creature in creatures:
        r, angle = (creature.position - position).as_polar()
        if r < 10:
            return creature
 pyreturn None
```

We will be modularizing our code into several source files. So, each time I give code, there will be an icon in the margin specifying which file it should go into, like this:

Flatland.py

The original source code from the examples in this book can be found as follows:

1. Browse to **www.ineasysteps.com** then navigate to Free Resources
2. Click on the Browse Now button under the Source code downloads and other book resources section, then find the book's title in the list and click on All Code Examples

Unzip the downloaded file into a folder of your choosing. You could create a folder called **MyProjects** in your **Documents** folder, for example.

Installing Python

If you do not already have Python installed, you can get it from **www.python.org/downloads**

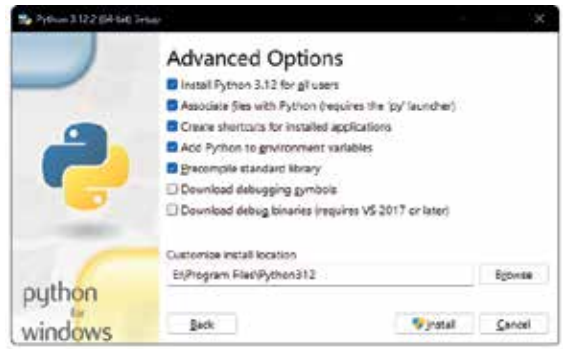
The webpage will probably offer you the appropriate installer. Choose that and when it has downloaded, execute it. You may notice that the name of the file has "amd" in it. That's okay, even if you have an Intel CPU. Python isn't that picky.

Check the **Add python.exe to the PATH** box. Then, choose **Custom Installation**.

The first page allows you to select which modules are to be installed. All of them are selected and that is fine, so just hit **Next**.

The installer asks you which packages to install. By default it installs everything and that is fine, so just hit **Next** again.

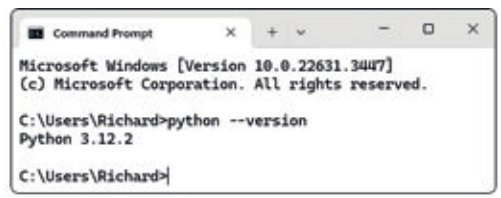
On the next page, check the top checkbox to install Python for all users and select the path where you want to install it to. I changed C: drive to E: because my C: drive is very small.



When you click on **Install**, Windows will ask you for administrator permission and then it will complete the installation.

Now you have Python installed, you can check the version number to make sure it's working. In a command prompt, type

```
python --version
```



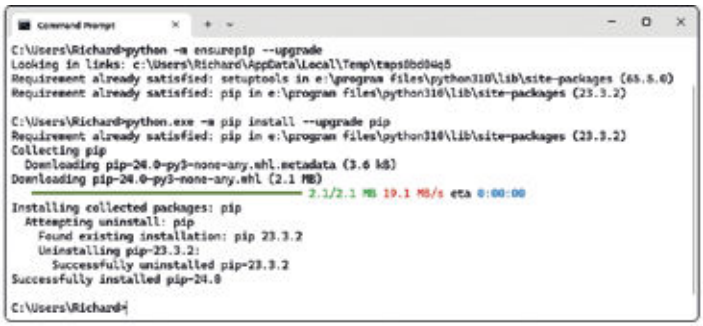
Installing PIP

PIP, the package installer for Python, will usually be installed with Python. If you already have Python installed and don't have PIP or don't know whether you do, then you can have Python install it for you. From a command prompt, type:

```
python -m ensurepip --upgrade
```

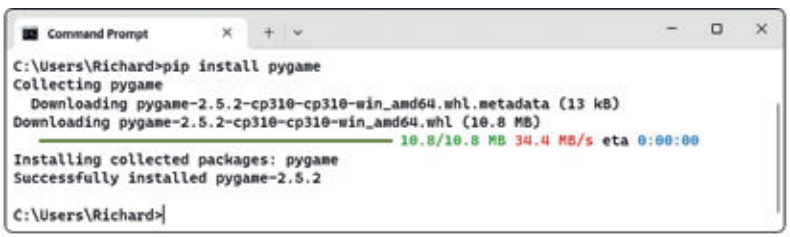
It will either install PIP or it will report that it is already installed. It might still be an old version but you can upgrade it by typing:

```
python.exe -m pip install --upgrade pip
```



We will use Pygame to produce graphics in some of our projects. Installing it is simple. Just open a command prompt and type:

```
pip install pygame
```



Optional extras

Later in the book, we will be using neural networks to build deep-learning applications. The models we are using are quite capable of running on a normal CPU. However, you might want to allow your NVIDIA GPU to be used in order to speed up the training process. This needs you to be comfortable working with Linux, and installation is rather intricate.

The installation process depends on your hardware to a large extent. I can't give exact instructions but here is a bullet list to get you started.

- Install the latest Graphics drivers.
- Install WSL2 from **learn.microsoft.com/en-us/windows/wsl/install**
- Install CuDNN and the CUDA toolkit from **docs.nvidia.com/deeplearning/cudnn/installation/linux.html**

Install the graphics drivers under Windows, as usual. Do not try to install them under Linux. However, CuDNN, the CUDA Toolkit and, on page 16, TensorFlow, are installed under Linux.

TensorFlow

For the neural network examples later in the book we will need to install a module that handles fast arithmetic. TensorFlow2 can calculate mathematical operations on matrices of hundreds of numbers very quickly and it is built specifically for implementing neural networks.

First you will need the ***Microsoft Visual C++ Redistributable*** for ***Visual Studio 2015, 2017, 2019 and 2022***. If you installed one of those releases of Visual Studio then you probably have it already. If you don't have it installed, then go to **https://learn.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170**

Click on the link for the above package, and then click on the architecture of your PC – that is probably x64. Download the package and execute it in order to install it.

To install TensorFlow2, type:

```
pip install tensorflow
```

It will take some time and install a lot of other modules but, eventually, it will tell you that the installation has been completed.

If you are interested in building more advanced projects, check out the Tensorflow2 installation page at https://www.tensorflow.org/install/pip

If you are attempting to use TensorFlow with the GPU, I suggest you also install it, and all of the following, under Windows so you can work either way:

The TensorFlow datasets module is separate so we have to install that too:

```
pip install tensorflow_datasets
```

We'll be playing with image files later, so we will need the Python Image Library, which, for historical reasons, is called Pillow:

```
pip install pillow
```

Finally, we will be using Matplotlib to easily display several images at once:

```
pip install matplotlib
```