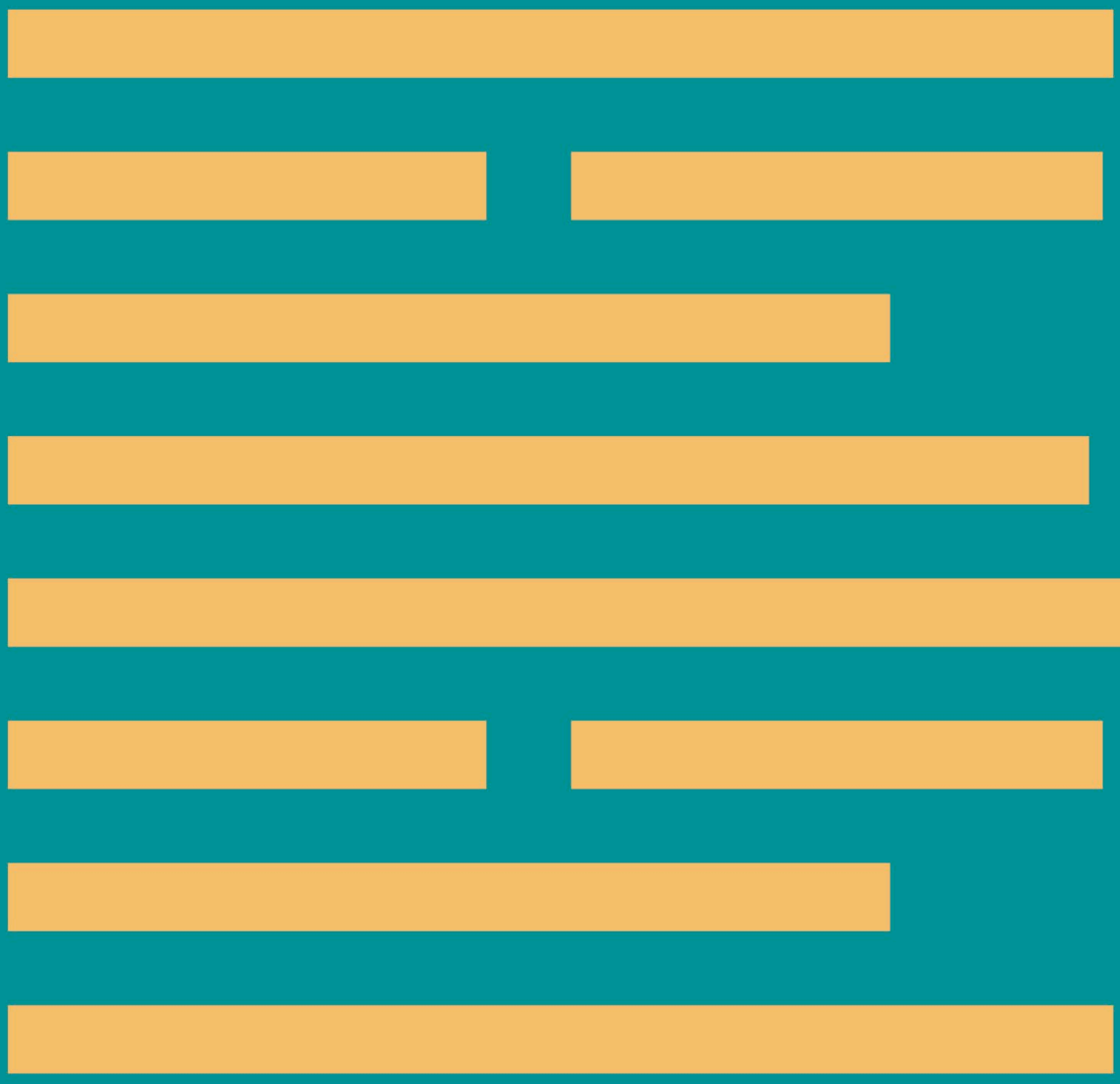


PYTHON

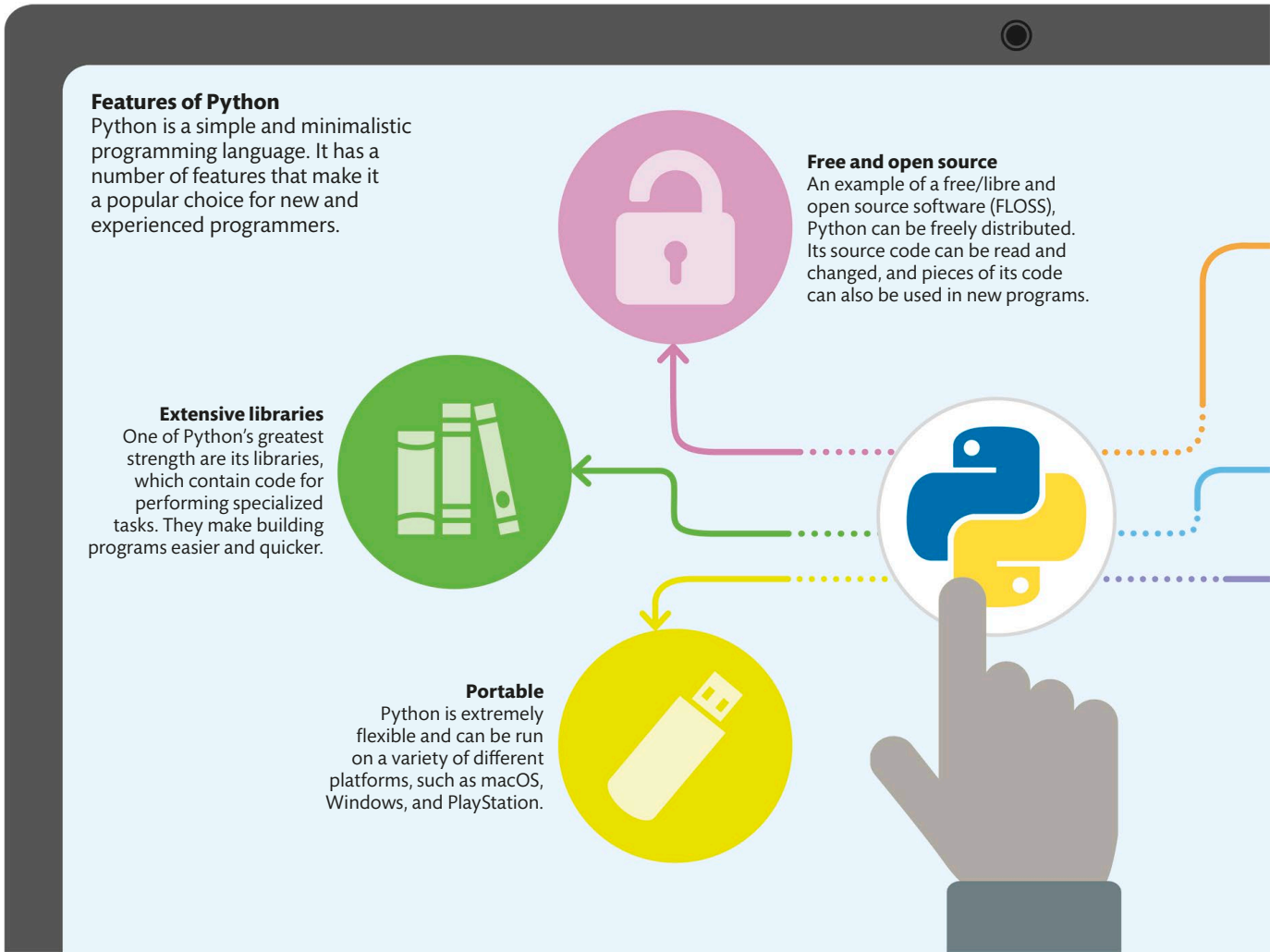
What is Python?

Python is one of the world's most popular programming languages. It is extremely versatile and can be used in many real-world situations. A text-based language, the readability and clear layout of its code makes Python less daunting for beginners.

Why use Python?

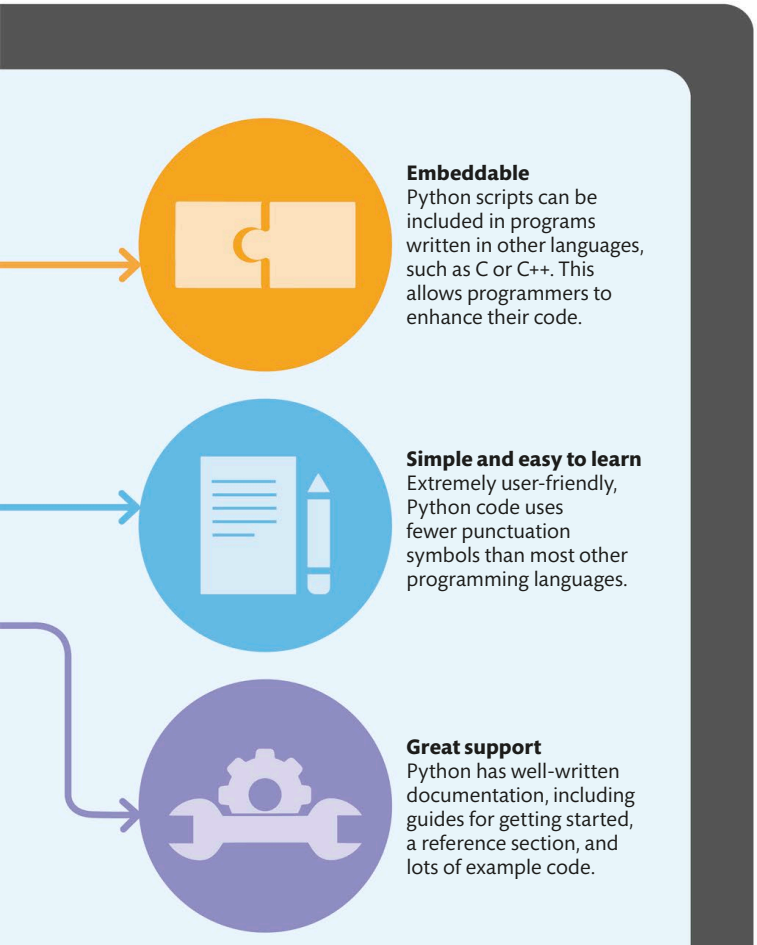
Created by Dutch programmer Guido van Rossum, Python was released in 1991. It was designed as a high-level language that would appeal to programmers familiar with the C language (see p.347) and the Unix operating system. Python lends itself to writing a wide range of programs and is used by many schools and universities to teach programming.

The syntax (arrangement of words and symbols forming the code) in Python is close to English syntax, which supports its goal of producing readable code. In addition, Python also forces programmers to lay out their code in a structured way. This is a useful skill to develop, as it makes it easier for the programmer to debug the code and also improves readability for other users.



How it works

A Python program, usually called a script, is a text file containing words, numbers, and punctuation that correspond to instructions. These instructions are formed of certain fixed patterns of words and symbols, which the programmer types in. IDLE (Integrated Development and Learning Environment) is a free app that is installed with Python. Designed for beginners, it includes a basic text editor that allows the user to write, edit, and save code before running a program.



Embeddable
Python scripts can be included in programs written in other languages, such as C or C++. This allows programmers to enhance their code.

Simple and easy to learn
Extremely user-friendly, Python code uses fewer punctuation symbols than most other programming languages.

Great support
Python has well-written documentation, including guides for getting started, a reference section, and lots of example code.

APPLICATIONS

Python is a general-purpose programming language that can be used to create systems for a variety of purposes. This, along with its many specialty libraries, makes it useful in fields as diverse as business, medicine, science, and media.

Game development
Python has various modules and libraries that support game development. These include *pygame*, for 2D games, and *PySoy*, a cloud-based 3D game engine.

Space
Software engineers have used Python to create tools for NASA's Mission Control Center. These tools help the crew prepare for and monitor the progress of each mission.

Business
Python's easy syntax makes it ideal for building large applications. It has become especially popular with the rise of Fintech (financial technology).

Scientific computing
Python has libraries that can be used in specific areas of science, such as *PyBrain* for machine learning and *pandas* for data analysis.

Web development
Python is used by software developers for automated tasks, such as build control and testing. It can also be used to create web applications.

Installing Python

It is important to download the right version of Python. This book uses the current version: Python 3. It is free and can be easily downloaded from the Python website. Follow the instructions that match your operating system.

Python on Windows

Before you install Python, you need to find out if your system has a 32-bit or 64-bit architecture. To do that, click the Start menu, right-click This PC, and choose Properties. A computer's architecture indicates how its microprocessor handles data at the lowest level. A 64-bit processor provides higher performance, as it can handle more data at once than a 32-bit processor.

FLYING CIRCUS

Python is not named after the snake, as many people think, but after the British television series *Monty Python's Flying Circus*. Guido van Rossum, who created the language, was a big fan of the program, and Python was a title that stuck. There are numerous references to Monty Python's sketches in Python's official documentation.

1 Go to the Python website

Go to *www.python.org* and click on Downloads in the menu bar on top. A list of operating systems will appear onscreen. Select Windows.

https://www.python.org

2 Download an installer

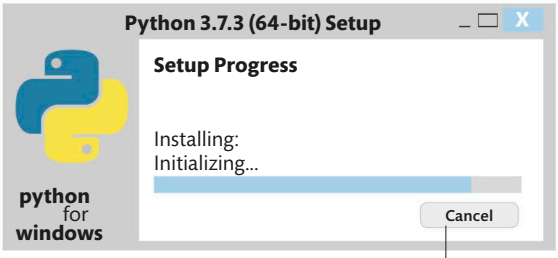
Find the most recent Python installer, which should start with 3. Be sure to select an x86 installer for 32-bit machines and an x86-64 installer for 64-bit machines. Either the web-based or executable installer will work.

The website could have a more recent version of Python

- Python 3.7.3 - 2019-03-25
  - Download Windows x86-64 web-based installer
  - Download Windows x86 web-based installer

3 Run the installer

Once downloaded, double-click the installer file and follow the instructions that appear onscreen. Remember to check the box on the initial prompt that says "Add Python to Path".



Installation can be canceled at any point

4 Open IDLE

Once the installation process is complete, go to the Applications folder and find IDLE inside the Python folder. You can also search for it in the Start menu. Double-click on IDLE to open Python's shell window. You will see IDLE's menu at the top of the window.

Python on a Mac

Before you install Python, you need to check which operating system your Mac uses. This will tell if your system has a 32-bit or 64-bit architecture. To find out, click the Apple icon in the top left of the screen and select About this Mac from the drop-down menu. If the processor is an Intel Core Solo or Intel Core Duo, it means your system has a 32-bit architecture; otherwise, it has a 64-bit architecture.

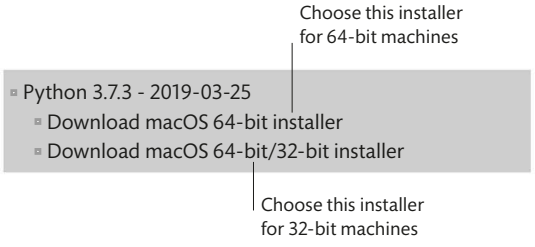
1 Go to the Python website

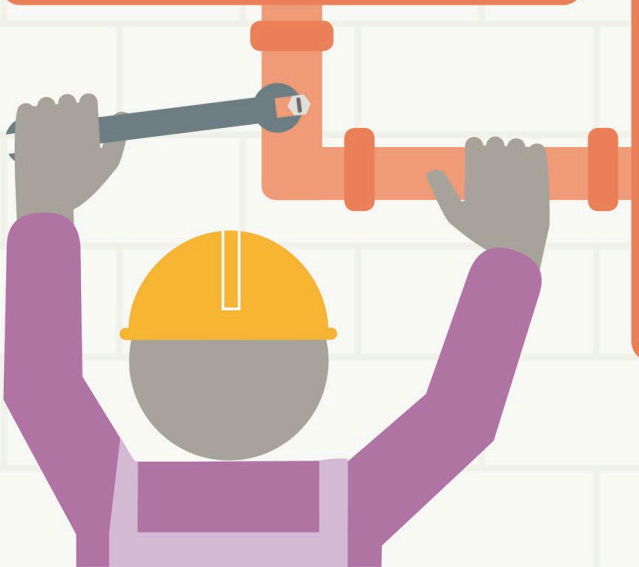
Go to *www.python.org*. Hover the cursor over the Downloads tab in the menu bar on top to generate a list of operating systems. Select the macOS option to find the installers suited to Mac computers.

https://www.python.org

2 Download an installer

Find the most recent Python 3 installer that matches your operating system and select it. The **Python.pkg** file will download to your system automatically.





3 Run the installer

Once downloaded, double-click the **.pkg** file and follow the instructions that appear. The installation process on a Mac computer is very straightforward. It will only ask you to agree to the licensing requirements and confirm the installation location (usually the Macintosh Hard Disk).

4 Open IDLE

Once the installation is complete, open the Applications folder from the Finder window's sidebar and find IDLE in the Python folder that appears. Double-click on IDLE to open Python's shell window and check that the installation has been successful.

Shell window

The shell window is opened as soon as IDLE is launched. It can be very useful to try out ideas in this window, as it gives instant feedback. However, as the shell cannot save code, it is not practical to use this window to evaluate more than a few lines of code at a time.

The shell window shows the version of Python it is running

Python 3.7.0 Shell

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jan 26 2019, 23:26:24
[Clang 6.0 (clang-600.057)] on darwin
Type "copyright", "credits" or "license()" for more information
>>>
```

This information depends on the operating system being used

Editor window

The editor window can be opened by selecting New File or Open from IDLE's File menu. This window allows programmers to type in much longer and more complex series of instructions and save them in files. Python file names are easy to identify, as they end with .py.

A Python file displayed in the editor window

helloworld.py

```
print("Hello world!")
```

COLORS IN THE CODE

To make code easier to read and errors easier to spot, IDLE displays the text in the editor and shell windows using different colors. The color used for each word depends on its role in the code.

COLORS IN THE CODE

Code	Color	Example
Built-in commands	Purple	print()
Symbols, names, and numbers	Black	25
Text within quotes	Green	"Hello world!"
Errors	Red	pront()
Keywords	Orange	if, else
Output	Blue	Hello world!

Using IDLE

Python's Integrated Development and Learning Environment (IDLE) interface has two windows for carrying out different tasks. The shell evaluates short commands immediately, while the editor window allows programmers to enter and save longer programs in files.

Running a program using IDLE

To run a program from IDLE, the file containing it must first be opened in the editor window. If it runs successfully, the shell window displays the output of the code; otherwise, the relevant error message appears.

Python 3.7.0 Shell

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jan 26 2019, 23:26:24
[Clang 6.0 (clang-600.057)] on darwin
Type "copyright", "credits" or "license()" for more information
>>>
======== RESTART: /Users/tinajind/Desktop/helloworld.py ========
Hello world!
>>>
```

Common errors

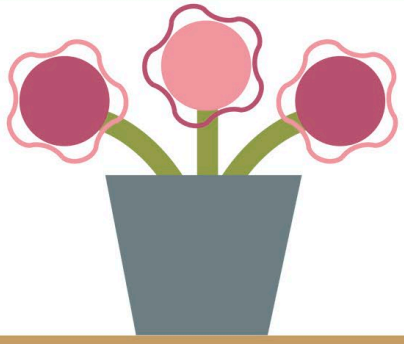
As well as being case-sensitive, Python is also very strict about the layout and spelling of code. It requires sections of code to be indented by four spaces from the line above in order to make the code more readable. These features often trip up new programmers. IDLE helps spot and fix errors with pop-up information boxes and error messages (see pp.130–133) in the shell window.

```
num = 4
if (nut == 5):
    print("Hello world!")
```

Here "num" has accidentally been typed as "nut"

The mistake in code results in this error message

```
Traceback (most recent call last):
  File "/Users/tinajind/Desktop/helloworld.py",
line 2, in <module>
    if (nut == 5):
NameError: name 'nut' is not defined
>>>
```



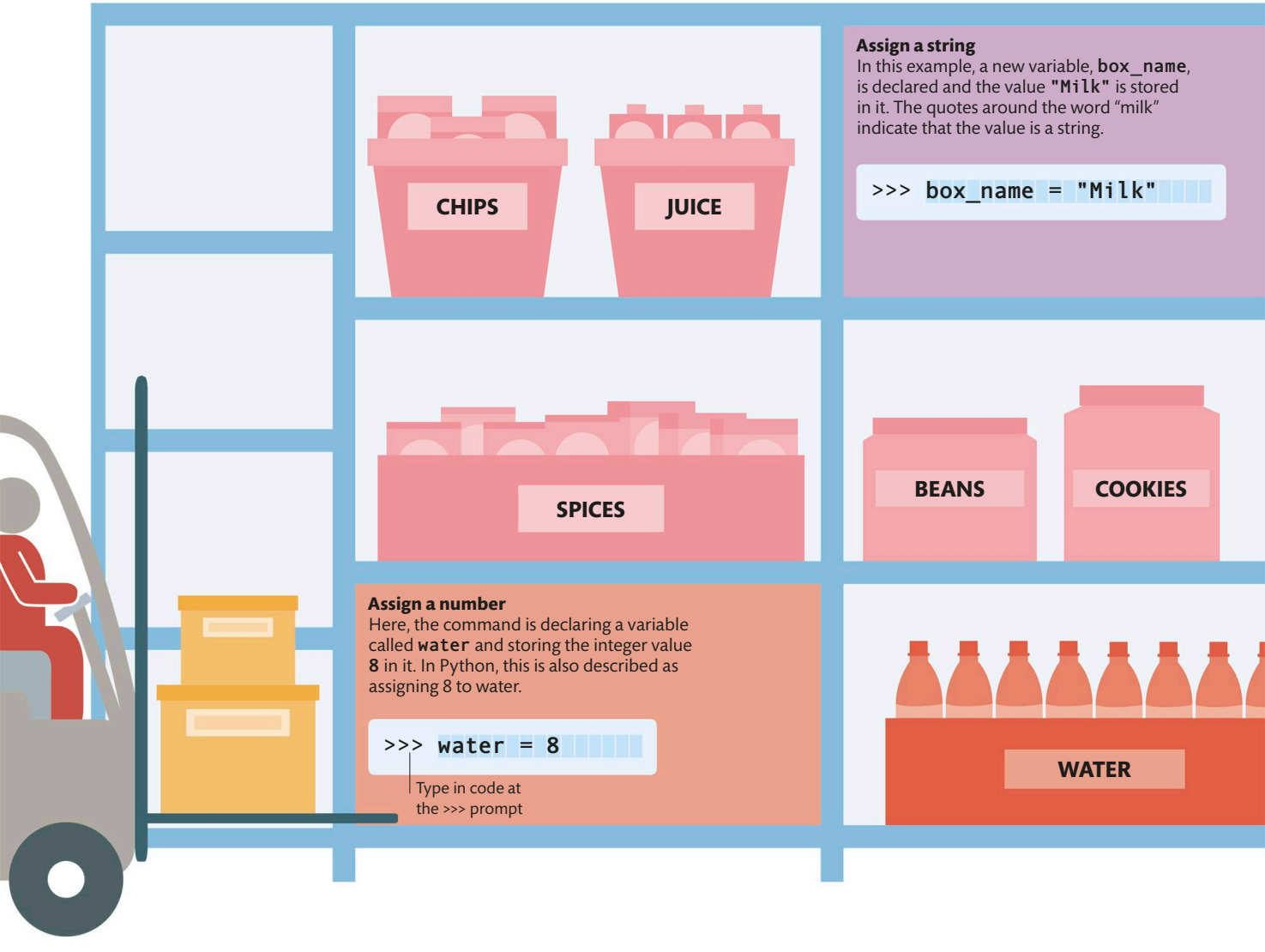
Variables in Python

Variables are one of the most basic programming tools used for storing and manipulating data. Similar to a box, they are a storage mechanism that can hold values used in a program.

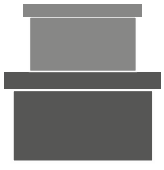
Creating variables

In order to create a variable in Python, it must be given a name and a value. The value can be one of various types, such as a number or a string (see p.103). However, as the name suggests, variables do not have a fixed value. Once data is stored in them, they can be updated to different values throughout the program.

This also allows the code to work in a variety of different situations and for a lot of different inputs. The alternative to this is known as "hard-coding," where each calculation and expression contains a specific value. This, however, would result in a situation where the programmer has to write multiple programs to cover every possible value that might be encountered.



DIFFERENT **PROGRAMMING LANGUAGES** HAVE DIFFERENT **WAYS** OF **CREATING VARIABLES** INSIDE A PROGRAM



Naming a variable
Giving a suitable name to a variable helps make a program easier to understand. Here are some rules that have to be followed when naming a variable:

- Start the variable name with a letter.
- Symbols such as -, /, #, or @ are not allowed.
- Uppercase and lowercase letters are different. Python will treat "Milk" and "milk" as two different variables.
- Avoid using Python commands, such as "print", in the name.

DECLARING VARIABLES

Creating a new variable is also called "declaring" it. In some programming languages, a special keyword is used to show that a new variable is being created. In Python, however, a variable is created as soon as a value is assigned to it. There is no need to state what sort of data will be stored in the variable. Python will work this out from the value being assigned to it. Using a variable without assigning a value to it is a common error.

Using variables

Once a variable holds a value, it can be used in various ways. The variable's current value can be used in a calculation, or the value stored can be changed to a new value.

Simple calculation
This code carries out simple multiplication. It stores the integer **2** in variable **input**, then retrieves that value and multiplies it by **3**. The result is stored in the variable **score** and then displayed onscreen.

```
>>> input = 2
>>> score = input * 3
>>> print(score)
6
```

The result of the calculation

Prints the value assigned to **score**

Changing a value
To change the value of a variable, a new value can be assigned to it. Typing this code into the shell window below the previously written code will have no effect on the value stored in **score**; it will only change the value of the variable **input**.

Changes the value of **input**

```
>>> input = 5
>>> print(score)
6
```

The result will not change

Updating a value
To get the correct result, the value of the variable **score** needs to be updated explicitly, as done in the example here.

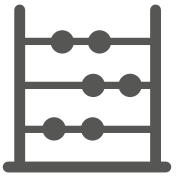
```
>>> input = 5
>>> score = input * 3
>>> print(score)
15
```

The output is updated

Adding this line assigns a new value to **score** after **input** has been changed

Data in Python

Python programs work with various data types. These types determine what can be done with different data items and how they are input, output, and stored. Errors in Python code are often the result of forgetting to consider a value's type and what it allows.



PYTHON USES THE SAME RULES OF ARITHMETIC AS HUMANS TO CARRY OUT CALCULATIONS

Integers and floats

Numbers in Python programs can be one of two data types: **integer** or **float**. An integer is a whole number with no decimal point, while a float—short for floating point—has a decimal point and numbers after it. Floats are often used for measurements or as the result of a calculation.

`pets` is an integer variable and contains the value **2**

```
>>> pets = 2
>>> print(pets)
2
```

INTEGER

```
>>> temperature = 37.5
>>> print(temperature)
37.5
```

FLOAT

The variable `temperature` contains a float

Arithmetic operators

Numbers and variables containing numbers can be combined using addition, subtraction, multiplication, and division. The symbols for these processes are called arithmetic operators. While addition and subtraction use familiar symbols, multiplication and division are slightly different and are shown as * and / respectively.

ARITHMETIC OPERATORS

Symbol	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division

Calculations
These Python commands use arithmetic operators to calculate the tax owed on an item costing $8.00.

This variable contains `price` as a float

The result will be stored in the variable `tax`

```
>>> price = 8.00
>>> tax = price * (20/100)
>>> print(tax)
1.6
```

The output is the value stored in the variable `tax`

Characters and strings

The data type Python uses for text is known as string. Made up of individual letters, numbers, or symbols called characters, strings must always have quotation marks at the beginning and the end. Python allows both single and double quotation marks in its code.

Strings
The variable `forename` contains a string made up of the characters of the word Alan.

```
>>> forename = "Alan"
>>> forename
'Alan'
```

Combining strings
Combining two or more strings to make a new one is called concatenation. Python uses the + symbol to do this. It is important to change any values with different data types into strings before concatenating them.

```
>>> happy = "happy birthday to you "
>>> name = "Emma "
>>> song = happy + happy + "happy \
birthday dear " + name + happy
>>> song
'happy birthday to you happy
birthday to you happy birthday
dear Emma happy birthday to you'
```

The variable **song** now contains a personalized version of "Happy Birthday"

Casting
It is sometimes necessary to change the data type of a value for a particular task, for example, when combining an integer and string. This is called casting, and Python provides functions, such as `str()` and `int()`, to allow it.

Change integer to string

```
>>> age = 25
>>> print ("Your age is " + str(age))
Your age is 25
```

THE LEN() FUNCTION

In a lot of programs, it can be very useful to know the length of a string or a list. Python has a built-in `len()` function that can be used for both tasks. Remember that the length of a string will also count spaces and punctuation.

```
>>> len("Hello Alan")
10
```

Lists
It is often useful to be able to group items together in a program. Python provides the list data type for this. A list can contain items that have the same data type or a variety of data types. To create a list, the values are enclosed in square brackets and are separated by commas.

The string `"two"` enclosed in double quotes

```
>>> my_list = [1, "two", \
              3, 5, 7.4]
>>> my_list
[1, 'two', 3, 5, 7.4]
```

Single quotes do not affect the value

Backslash splits code over two lines

Accessing items
To allow programmers to access items in a list, Python numbers each one. Typing the name of the list followed by the item number inside square brackets retrieves the relevant item. Python numbers the items in a list starting at 0.

```
>>> my_list[0]
1
>>> my_list[2]
3
```

First item in the list `my_list`

Logical operators and branching

Booleans, another data type in Python, have only two possible values: True or False. Booleans allow programmers to write branching statements that control which parts of a program are run.

= AND ==

It is important to distinguish between Python's two different equals signs. A single equals sign "=" means that a value is being assigned to a variable. A double equals sign "==" is a logical comparison to see whether or not the values on either side of it are equal.

Logical operators

Logical operators are symbols that allow a program to make comparisons between values. Any comparison that uses logical operators is called a Boolean expression and the result is a Boolean value. Logical operators are similar to arithmetic operators (see p.102) but produce Boolean values rather than numbers.

LOGICAL OPERATORS	
Symbol	**Meaning**
<	Less than
>	Greater than
==	Equal value
!=	Not equal value

Equality
A Boolean expression containing a double equals sign is True if the values on either side of it are equal.

Checks if the values on each side are equal

```
>>> 3 == 9
False
```

Less than
An expression containing the < symbol is True if the value on the left is less than the value on the right.

Checks if the value on the left is smaller

```
>>> 3 < 5
True
```

```
>>> oranges = 5
>>> apples = 7
>>> oranges != apples
True
```

Stores the value **5** in the variable **oranges**

Values in **oranges** and **apples** are not equal

Not equal
Logical operators also work with variables. This example stores values in two variables, then checks for the stored values being unequal.

Boolean operators

Boolean expressions can be combined using the Boolean operators "and", "or", and "not". They allow programmers to build more complex expressions that can deal with several different variables.

```
>>> (oranges < 10) and (apples > 2)
True
>>> (oranges < 10) or (apples == 3)
True
>>> not(apples == 7)
False
```

For this to be True, both expressions must be True

Only one expression has to be True for this statement to be True

Putting "not" in front of a True expression results in the value False

Branching

Computer programs often contain code that should only be run in certain situations. To allow for this, programmers create branches in the code. The decision about which branch to take depends on the result of a Boolean expression. This helps programmers tailor a program's behavior to different inputs or environments.

More than two branches

When there are more than two possible paths through the code, the **elif** command—short for **else-if**—is used. It is possible to have several **elif** branches between the **if** branch and the **else** branch.

```
quiz_score = 9
if quiz_score > 8:
    print("You're a quiz champion!")
elif quiz_score > 5:
    print("Could do better!")
else:
    print("Were you actually awake?")
```

This comparison is the first condition

If the first condition is True, this is printed

This is the second condition

If the second condition is True, this line is printed

If both conditions are False, this line is the output

One branch

The most straightforward branching command has only a single branch that the computer takes if the condition is True. This is called an **if** statement.

```
temperature = 75
if temperature > 70:
    print("Turn off heater")
```

This comparison is the condition

If the condition is True, the code runs

Two branches

A situation where a program should do one thing if a condition is True and another if it is False needs a command with two branches. This is an **if-else** statement.

```
age = 15
if age > 17:
    print("You can vote")
else:
    print("You are not old \
enough to vote")
```

The comparison is the first condition

If the condition is True, this line is printed

If the condition is False, this line is printed

A backslash is used to split a long line of code over two lines without affecting the output