extracts from

Coding for Beginners: Using Python

Written by

Louie Stowell

Published by

Usborne Publishing Ltd

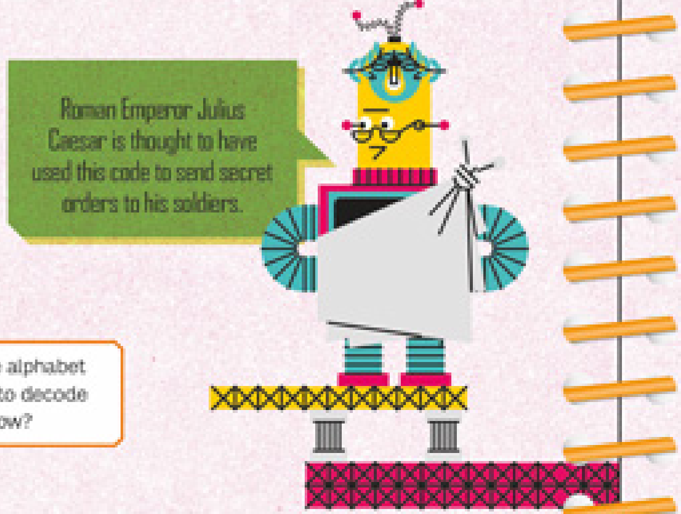
SPY MESSAGES

Spies use a technique called encryption to scramble the letters in their messages, so no one can understand them without a secret formula to unscramble them again. You can use Python to create your own encryption program.

CAESAR CIPHER

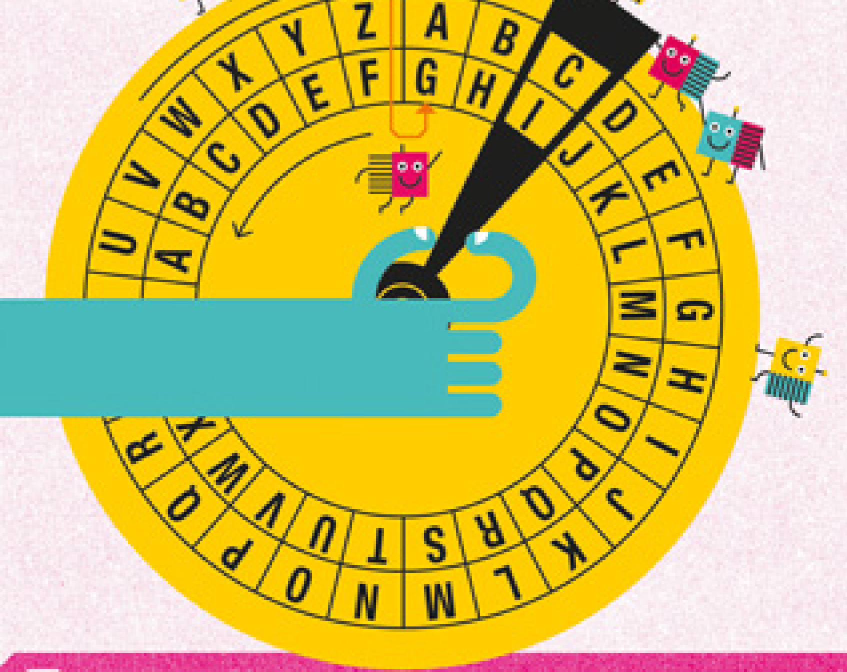
A **Caesar Cipher** is an encryption method that takes the letters of the message you want to send and 'shifts' each letter along the alphabet a certain number of places. The number of places moved along is known as the shift amount, or 'key'.

For example, if the first letter of the word you want to encrypt is C, and the key is 6, the letter will become I.

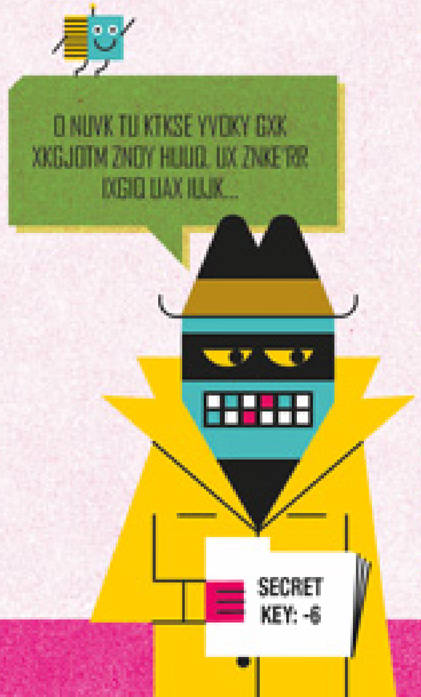


Roman Emperor Julius Caesar is thought to have used this code to send secret orders to his soldiers.

The inner ring of this wheel shows the alphabet shifted along 6 places. Can you use it to decode what the robot spy is saying below?



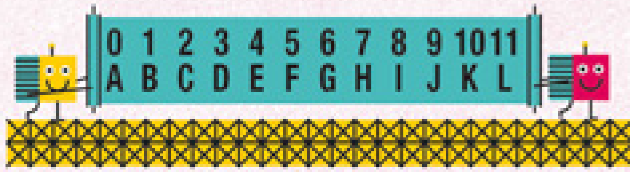
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
G H I J K L M N O P Q R S T U V W X Y Z A B C D E F



O NUVK TU KTKSE YVOKY GXK XKGJOTM ZNOY HUUQ. UX ZNKE'RR IXGIQ UAX IUJK...
SECRET KEY: -6

CREATING YOUR CIPHER

1. Open a new file and save it. On the first line, create a **variable** that contains a **string** with the whole alphabet, typed out twice.



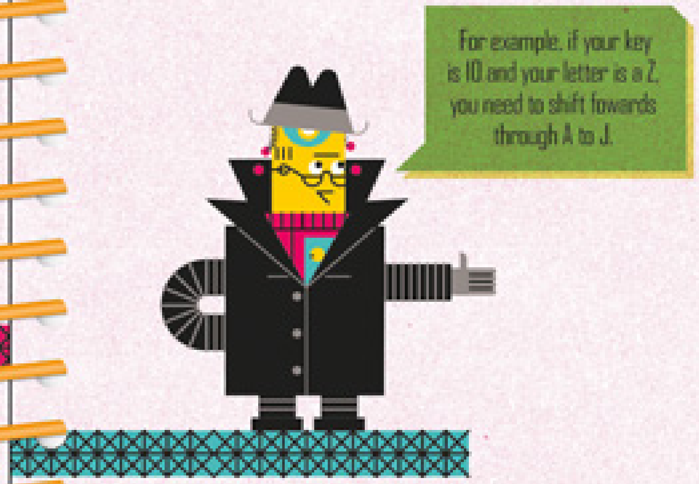
```
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

First alphabet

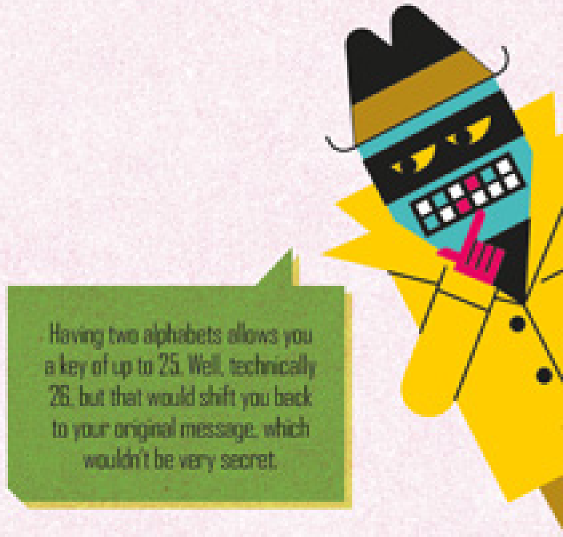
Second alphabet

Each character in a **string** has an **index**, just like each entry in a **list**.

You need the second alphabet to give you somewhere to shift to once you get to Z.



For example, if your key is 10 and your letter is a Z, you need to shift forwards through A to J.



Having two alphabets allows you a key of up to 25. Well, technically 26, but that would shift you back to your original message, which wouldn't be very secret.

2. Next, ask the user to enter a message.

```
stringToEncrypt = input("Please enter a message to encrypt: ")
```

3. In case the message includes lower case letters, make the string all upper case with the function, **upper()**.

```
stringToEncrypt = stringToEncrypt.upper()
```

This turns any lower case letters into upper case, so they match the letters in your alphabet variable.

4. Next, ask the user to type in a number. This will be used as the key to encrypt their message.

```
shiftAmount = int(input("Please enter a whole number from 1-25 to be your key."))
```

DODGE THE BOMBS

Watch out! Hidden bombs lurk beneath a green field in this puzzle game. Can you find all the safe squares without setting off an explosion?

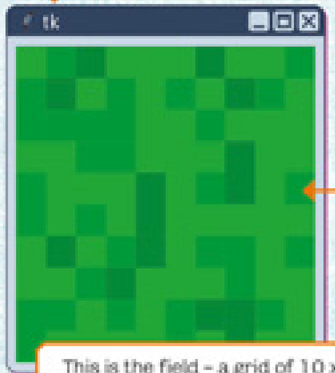
⚠ WARNING ⚠

The code for this game is quite advanced and builds on the skills you've learned earlier, so make sure you've worked through the rest of the book before you try it.

THE GAME

Here's how the game works andwill look when it's finished. The field is divided into squares.

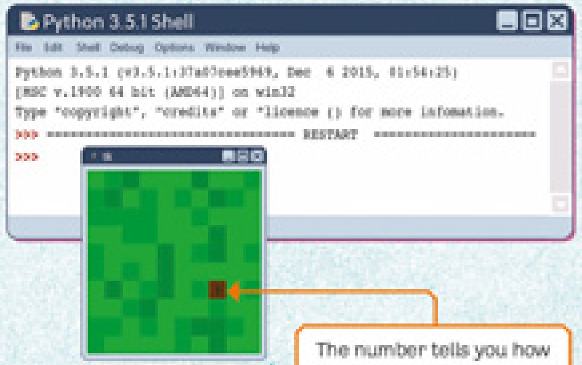
1. When you run the game, a new window pops up in front of the Shell window.



The 'tk' shows this window was created with **tkinter**.

This is the field – a grid of 10 x 10 squares, booby-trapped with hidden bombs.

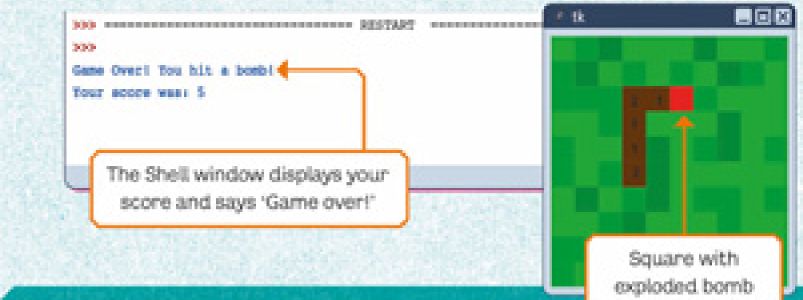
2. You click on one of the squares to test it. If it is safe, it turns brown and a number appears.



```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  6 2015, 01:54:25)
[MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "licence () for more infomation.
>>> ================================ RESTART =====================
>>>
```

The number tells you how many bombs there are in the surrounding squares.

3. If you hit a bomb, the square turns red and the game ends.



```
>>> ================================ RESTART ==============
>>>
Game Over! You hit a bomb!
Your score was: 5
```

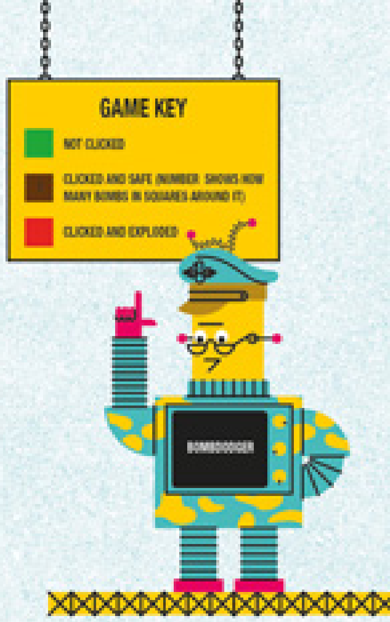
The Shell window displays your score and says 'Game over!'

Square with exploded bomb

HOW DOES IT WORK?

To make the game work, your program needs to...

- Decide where the bombs go.
- Draw the field full of bombs on the screen.
- React when the player clicks on a square, changing the colour to red for a bomb or, if the square is safe, changing it to brown and revealing the number of bombs around it.
- Keep track of the player's score (how many safe squares they find).
- End the game and show the score if a bomb is hit.
- End the game and congratulate the player if they find all the safe squares.



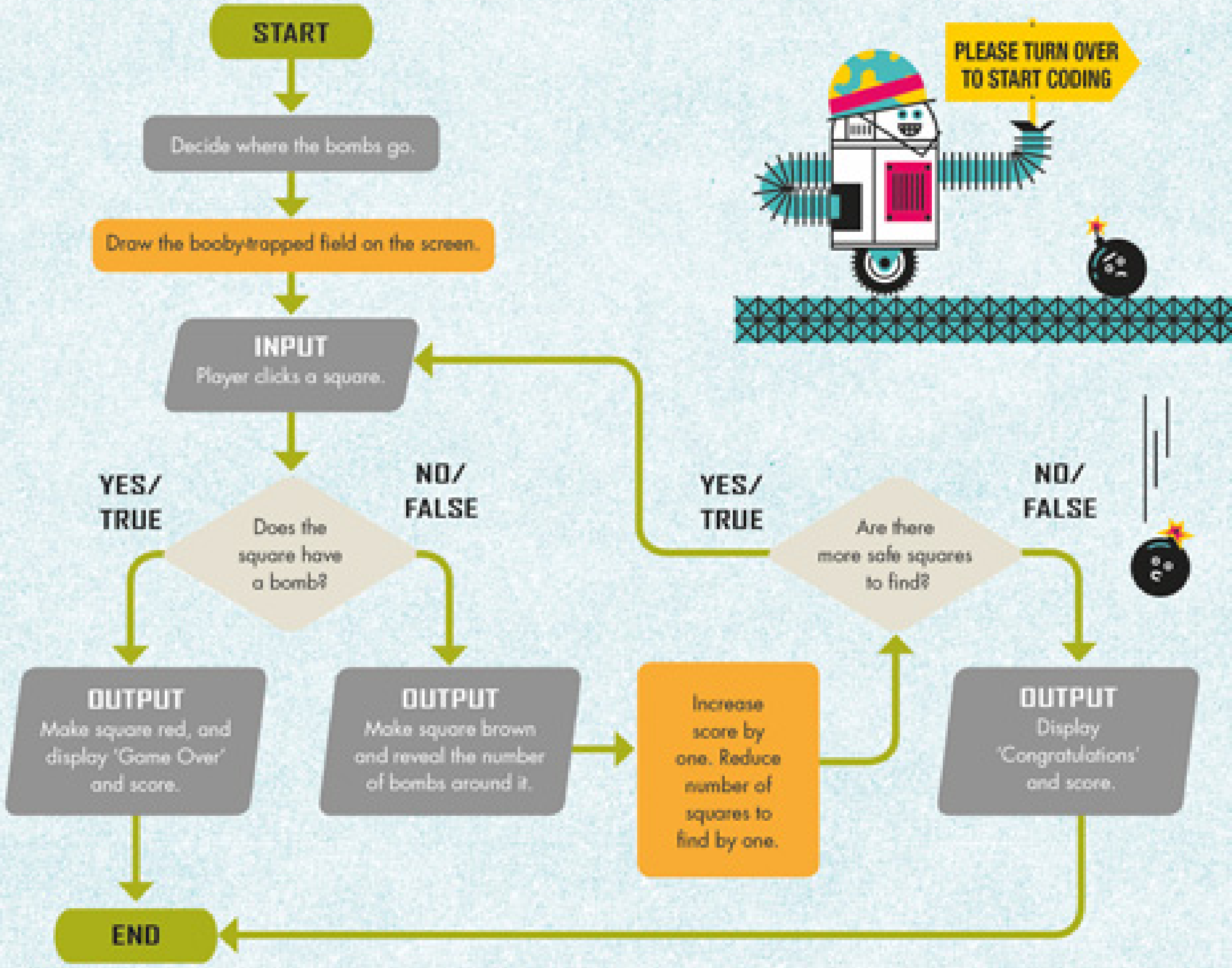
GAME KEY

🟩 NOT CLICKED

🟫 CLICKED AND SAFE (NUMBER SHOWS HOW MANY BOMBS IN SQUARES AROUND IT)

🟥 CLICKED AND EXPLODED

Here's how that looks as a flow chart.


START
Decide where the bombs go.
Draw the booby-trapped field on the screen.
INPUT
Player clicks a square.
Does the square have a bomb?
YES/TRUE
NO/FALSE
OUTPUT
Make square red, and display 'Game Over' and score.
OUTPUT
Make square brown and reveal the number of bombs around it.
Increase score by one. Reduce number of squares to find by one.
Are there more safe squares to find?
YES/TRUE
NO/FALSE
OUTPUT
Display 'Congratulations' and score.
END

PLEASE TURN OVER TO START CODING

17. To play again, you need to reset your variables.

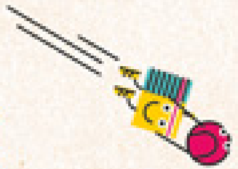
```
def reset():
    global leftPressed, rightPressed
    global ballMoveX, ballMoveY
    leftPressed = 0
    rightPressed = 0
    ballMoveX = 4
    ballMoveY = -4
    canvas.coords(bat, 10, setBatTop, 50, setBatBottom)
    canvas.coords(ball, 20, setBatTop-10, 30, setBatTop)
```

You need to update the global variables you made in step 10.

Clear any previous key presses.

Reset the ball speed for both **x** and **y** movements.

Place the bat at the bottom of the screen, with the ball just above it.



18. You also need to make the game window react to mouse clicks and key presses. This will require two different link commands: **protocol** and **bind** (see box).

window means the **tkinter** window.

When a window is closed, it automatically sends out this message.

This line binds the act of pressing the **close** button [x] to the **close()** function which stops the main loop.

```
window.protocol("WM_DELETE_WINDOW", close)
window.bind("<KeyPress>", on_key_press)
window.bind("<KeyRelease>", on_key_release)
```

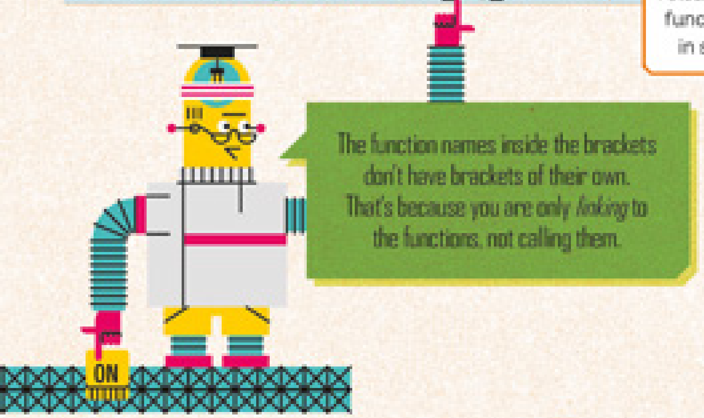
These lines bind the act of pressing and releasing a key to the functions you made in steps 5 and 6.

LINKING THINGS

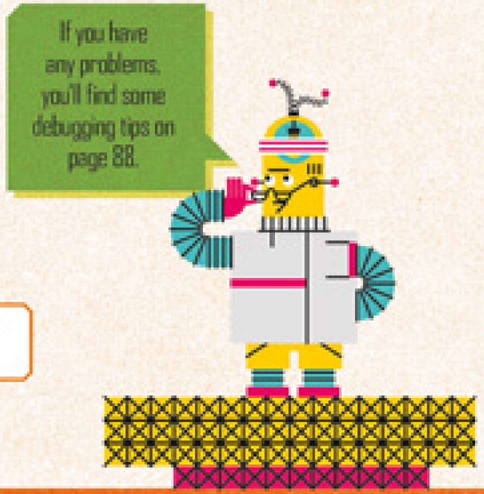
It's often useful to link a function to a specific event, so your function is called whenever that event happens.

protocol links a function to a message from a **tkinter** window.

bind links a function to other events, such as a key press or release.



The function names inside the brackets don't have brackets of their own. That's because you are only *linking* to the functions, not calling them.



If you have any problems, you'll find some debugging tips on page 88.

19. Finally, call the **reset()** and **main_loop()** functions to start the game.

```
reset()
main_loop()
```

This makes sure the game begins with all the right settings.

You've finished! Save and run the code to test it. How long can you keep the ball in the air?

KEEPING SCORE

If you want to keep track of how you're doing, you can add a score based on how many times you hit the ball.

1 To keep score, you need a new 'score' variable. Add this after the 'windowOpen' variable you made in step 4.

```
score = 0
```

Set it to zero to start.

2 The score should go up by 1 each time the ball bounces off the bat. You can do this inside the **move_ball()** function.

```
def move_ball():
    global ballMoveX, ballMoveY, score
```

Add score to the end of this line.

```
            ballMoveY = -ballMoveY
            score = score + 1
    canvas.move(ball, ballMoveX, ballMoveY)
```

Insert this line just above the final **move()** (from step 13) which makes the ball bounce off the bat.

3 Add a **print()** function to **check_game_over**.

```
    if ballTop > canvasHeight:
        print("Your score was " + str(score))
```

Add this line to display the score in the Shell window.

4 Finally, make sure the score goes back to zero when the game resets, by adding it to your **reset()** function.

```
def reset():
    global score
```

Insert this line at the start, so you can access the global 'score' variable.

```
    score = 0
```

Add this line at the end of the function.

Try the game now. You should see your score in the Shell window behind the canvas.