Computational Thinking

Design a spam filter

Overview
Computers cannot think for themselves. In order to use computers to solve problems we have to think through the problem first. Making sure you understand a problem and thinking up possible solutions is called computational thinking.

In this chapter you will learn some basic principles of computational thinking and use them to design a simple spam filter.

Learning outcomes
By the end of this chapter you will know how to:

- describe computational thinking
- use decomposition to break a problem down into smaller parts
- describe an algorithm
- use pattern recognition
- use a flow chart to describe your problem-solving
- describe selection
- use if... then... else to navigate through a problem
- describe iteration
- use loops to navigate through a problem
- evaluate other people’s work
- give helpful feedback.

Talk about...
Computational thinking helps us understand and solve problems. Some of these problems are in the real world. You already use computational thinking in your everyday life! Some examples are:

- planning a story before you write it
- arranging a party with your friends.

What other activities do you do that might use computational thinking?

FACT
Seymour Papert (1928–2016)
The term ‘computational thinking’ was first used by Seymour Papert in 1980. In 1968 Papert and his colleagues created Logo—the first programming language designed for children. The LEGO Group named their robotic products Mindstorms in honour of Papert’s book Mindstorms: Children, Computers and Powerful Ideas.
What is computational thinking?

Learning outcomes
When you have completed this lesson you will be able to:
- describe computational thinking
- use decomposition to break a problem down into smaller parts.

Learn about...
Computational thinking is a process for solving problems. Using computational thinking, you can understand a problem and break it down into smaller parts. You can work out what is important and find a solution.

Your computational thinking toolbox has a set of useful tools for you to use.

Decomposition and algorithms are two tools in your computational thinking toolbox.

Decomposition
Just as you would climb a mountain by taking one careful step at a time, you can decompose a problem by breaking it down into smaller parts. You can understand each part in more detail and solve it, one step at a time. If you do not decompose a problem, it can be harder to solve because you are trying to deal with too many parts at once.

Algorithm
An algorithm is a set of instructions or rules we can follow to carry out a task. We build algorithms by decomposing a problem so that we can carry out one task at a time. We carry out each task in a logical order.

How to...
You can use computational thinking to help break down a problem into smaller parts. This is called decomposition.

Imagine you are the manager of a chocolate factory and you want to make a new type of chocolate.

You would not just tell the workers to do anything they wanted! You would first decompose the problem into smaller parts.

- What type of chocolate would sell well?
- How would you make it?
- How would you get it to customers?

You can decompose the parts into even smaller parts until you have a list of simple tasks. You can then make a set of instructions to carry out each task.

What type of chocolate would sell well?
- Find out what type of chocolate customers would like to buy.
- How much would they be willing to pay for it?

How would we make it?
- Work out what ingredients and machines you would need to make the chocolate.
- Work with designers to create a wrapper.
- Test the chocolate.
- Set up a process in the factory for making the chocolate.
- Test the process.

How would we get it to customers?
- Get some shops to agree to stock the chocolate.
- Work out whether the chocolate should get to the shops by rail, air or road.
- Make the chocolate.
- Transport the chocolate to the shops.

The sets of instructions here are algorithms. This is an algorithm for the task: Find out what type of chocolate customers would like to buy.

1. Invent some chocolate ideas.
2. Make a poster of each chocolate.
3. Find a group of people to look at the ideas.
4. Show them the chocolate ideas.
5. Give them a survey to say which idea they like best.

You can use decomposition in computer programming in exactly the same way. You can break your problem down into small chunks and carry out one task at a time. This can save time. If there is a mistake in a program with many lines of computer code, you can check each section of code separately.
Computational Thinking

1. What is computational thinking?

1.1 Now you do it...

In this chapter you are going to design a simple spam filter. Email spam involves sending messages to many people who have not asked for the emails to be sent. We call this unsolicited email. Spam emails often contain links that look as if they might be from well-known and trusted websites, such as a bank’s website. A spam filter is a computer program that spots unsolicited email. A spam filter can stop unsolicited email from getting into your inbox. Your spam filter will detect spam relating to online banking, and will also spot emails from a bank that ask for personal details. Any email you receive that asks for personal banking details is likely to be untrustworthy.

What are the differences between an honest email and a fake one?

Work with a partner to answer these questions. Use the email examples from Honest Bank and Mrs Smith to help you.

- Who is the email from?
- Does the name on the email match the email address?
- Is the content of the email addressed to the email owner?
- Does the email ask for personal details?
- Does the email ask for money?
- What are the key words that your spam filter will detect?

When you answer these questions, you decompose the problem of understanding the differences between an honest email and a fake one. You can use the information to help you in future work.

If you have time...

Work with a partner. Each of you should make a list of four items. Three items on your list should have things in common and one item should be the odd one out. For example, you could choose a baseball, a tennis ball, a balloon and a basketball.

Which is the odd one out? The balloon is the odd one out because you cannot play a sport with a balloon.

The balloon does have something in common with the other items. All of the items can be spherical.

Can your partner identify the odd one out on your list? Why is it the odd one out? Can you think of something all four things have in common?

Test yourself...

1. What is an algorithm?
2. Decompose the problem of getting dressed in the morning.
3. What should you do if you receive a spam email?
4. Give an example of how computational thinking is relevant to everyday life.

From: jstyles@cooltunes.com
Heading: Stranded in Florida

How are you doing? We have not been introduced, but I need your help and have no one else to turn to. I have had to send this in a rush because I am in desperate trouble.

My family and I visited Florida for a short vacation recently and we were mugged outside our hotel. The thieves stole all our cash, cell phones and credit cards. Now we can’t go home because we can’t pay our hotel bill. We are stranded!

Would you please urgently send a cheque for any amount you can spare (even $100 would help) to a temporary post office box that we have set up? The address is below:

PO Box 2014
Maple Bay
Florida, USA 02956

I really need your help. Thank you.

Your friend,
Mrs Smith

HONEST BANK

HONEST BANK™ jstyles@honestbank.cooltunes.com

6:30 AM (10 hours ago)

Dear Valued Customer

We understand that you have withdrawn $5,000 from your checking account while in another country. If this information is incorrect, someone you do not know may be using your account.

Please visit our website and give us your account details so that we can be sure all the information we have for you is correct.

Click here to update your details. www.cqhy457.com

Thank you for banking with us.

Yours sincerely

Honest Bank

Key words

Algorithm: An algorithm is a set of instructions or rules that we can follow to carry out a task.

Computational thinking: Computational thinking is a way of thinking through problems and finding good solutions for them. We can apply the principles of computational thinking to computer programs and to solving problems in our everyday lives.

Decomposition: Decomposition is breaking a problem down into smaller parts.

Process: A process is a set of steps we can take to solve a problem or reach a goal. A process can include a set of algorithms to solve a problem.

Spam: Email spam involves sending messages to many people who have not asked for the emails to be sent.

Spam filter: A spam filter is a computer program that spots unsolicited email. A spam filter can stop unsolicited email from getting into your inbox.
Why is it important to identify patterns?

He was a Hungarian doctor who worked in a hospital where babies were being born. At that time, many women having babies in hospitals died of an illness called puerperal fever. Semmelweiss worked in the First Clinic at the hospital. This is where medical students were trained to deliver babies. In another clinic at the hospital, the Second Clinic, midwives were trained to deliver babies. Semmelweiss saw that more women died in the First Clinic than in the Second Clinic.

Semmelweiss decided to try to find out why so many women were dying in the First Clinic. He wrote down all the similarities and differences between the two clinics. For example, he studied the number of patients and the climate in both clinics. He could not find any patterns that would explain the high death rate in the First Clinic.

Then one of his doctor friends died after cutting his finger during an autopsy, which is an examination of a dead body. Semmelweiss knew that almost all of the medical students did autopsies. Student midwives did not. In those days doctors did not wear gloves or wash their hands. Semmelweiss realised that the medical students were taking something dangerous from the dead bodies to the women patients in the First Clinic. He told all of his colleagues to start washing their hands after autopsies. The death rate in the First Clinic dropped by 90 per cent. The fall in the death rate happened because Semmelweiss had found the pattern and then the right solution.

Patterns and computational thinking

Pattern recognition is finding the similarities and differences between things. You can use pattern recognition to help write algorithms. If you do not look for patterns when you decompose your problems, you might take longer to find the right solution or you might not find the right solution at all.

We often ask computers to recognise patterns for us.
Computational Thinking

1.2 Pattern recognition

How to...

You are designing a spam filter in this chapter. You can do this by finding a set of features that helps you tell the difference between spam and non-spam email. You can write an algorithm that tells the computer to look for these differences in your emails.

What kinds of words or phrases do you think you would find in spam emails? Here are some examples.

- bargain
- million
- dear friend
- opportunity
- money
- billion
- fees
- cost
- fast
- debt
- credit card
- urgent
- please read
- please help
- $ or other currency symbol
- funds
- bank
- guarantee
- rates
- income

Many spam emails use these words and phrases. Computer algorithms can spot these words across many possible spam emails. The algorithm is finding a word pattern. If an email has many of these types of words, it is possible that it is a spam email.

Now you do it...

Look at your decomposed spam filter problem.

- Can you identify any patterns that might be useful?
- What parts of the email could you think of as patterns?

Remember that there may be patterns in an individual spam email. For example, you might notice the repeated use of a fake business name or email account. There may also be patterns between several spam emails. For example, spam emails may ask for personal banking details.

If you have time...

Work with a partner. Compare making a cup of tea with making a cup of instant coffee. Which tasks are similar? Which tasks are different? Where could you use the same instructions in both processes?

Test yourself...

1. What is pattern recognition?
2. Look at the image of some robots on the next page. Write any patterns you see.

Key words

Pattern recognition: Pattern recognition is finding things that decomposed problems have in common.
Learning outcomes
When you have completed this lesson you will be able to:
- describe an algorithm
- use a flow chart to describe your problem-solving.

Learn about...
You know that an algorithm is a set of instructions or rules you can follow to carry out a task. You can use algorithmic thinking to work through and solve problems.

A computer cannot think for itself. We must do the thinking to design algorithms so that the computer has instructions to follow. When we give the computer a correct algorithm, it will work on the problems we want it to solve.

Computer programs that are given incorrect or inefficient algorithms will not work well. An inefficient algorithm means the computer has to work harder to solve the problem. Computer programs work well when they are given carefully planned algorithms. In a well-designed algorithm, each instruction is clear. The order for carrying out instructions is also clear.

What is a flow chart?
Writing algorithms as long lists is not a good way of communicating a set of instructions. Why do you think that is?

A list is not a good way of giving a set of instructions because it does not show the relationships between the steps needed to carry out a task. You can use a flow chart instead. A flow chart is a diagram that explains a process. Each step in the process is shown in a box. Different types of box mean different actions.

How to...
We use symbols in flow charts to show what action to take.

- **Start/Stop**
  This shape shows the beginning and end of a sequence. A sequence is the order in which one task follows another.

- **Process**
  This shape shows a command. A command is a specific instruction.

- **Decision**
  This shape shows a ‘yes’ or ‘no’ decision.

- **Input/Output**
  This shape shows where data are being received or sent from a computer program. Data are facts and figures.

These arrows show the way we move through the sequence from symbol to symbol. The arrows show the direction of flow.

A variable is anything that can be stored by the computer, and changed, controlled or measured. We use variables in flow charts to show what is being changed, controlled or measured in our algorithm.
Making a flow chart

This is a flow chart for a simple program that shows whether a named user likes reading. The user is the person who is using the program.

Start

Output

“What is your name?”

Input

User types in name.

Store input in variable NAME

Output

“Do you love reading?”

Input

User types in ‘yes’ or ‘no’.

Does the user love reading?

Yes

Output

NAME loves reading!

No

Output

NAME could try a different author

Stop

Why do you think NAME is in uppe-case letters and bold font? It is to show that NAME is a variable, so that the program can store the user’s name and output it at the end of the process.

For example, when your program scans an incoming email to see whether it contains the word ‘MONEY’, would that be a process or a decision, or both? First the program would need to carry out a process for scanning the incoming email. You could show the process as:

Scan email for word MONEY

Then the program would need to make a decision about whether or not the email contained the word ‘MONEY’. You could show the decision as:

Does scan show MONEY?

If you have time...

Can you think of any disadvantages of using flow charts to show your computational thinking?

Test yourself...

1. Why are flow charts useful in computational thinking?
2. Draw and label the four main shapes we use to make flow charts.
3. Decompose the task of brushing your teeth.
4. Make a flow chart to describe the process of brushing your teeth.

FACT

Flow charts are not just used in computing. Project managers, engineers, designers and surgeons all use flow charts in their work.

Key words

Command: A command is an instruction that tells the computer to do something.
Data: Data are facts and figures.
Flow chart: A flow chart is a diagram that explains a process. Each step in the process is shown in a box. Different types of box mean different actions.
Sequence: A sequence is the order in which one task follows another.
Variable: A variable is anything that can be changed, controlled or measured.

Now you do it...

Make a flow chart to describe the program for your spam filter. Remember that your spam filter is going to try to detect spam relating to online banking.

- Look at your decomposition of the spam filter problem
- Work with a partner to decide on symbols to represent each step in your flow chart.
Learning outcomes

When you have completed this lesson you will be able to:

★ describe selection
★ use if... then... else to navigate through a problem.

Learn about...

You already know that an algorithm is a set of instructions or rules that you can use to solve a problem or carry out a task. We write computer programs to implement algorithms. To implement an algorithm means to put it into action.

You also know that we can use flow charts to represent the sequence of instructions in an algorithm. Sequencing is one tool that you can use to design algorithms. Another tool is selection.

★ Selection can be a decision.
★ Selection can be a question.

Selection happens when you reach a step in the algorithm that has two or more possibilities. The program then needs to ask a question. Depending on the answer to the question, the program chooses a particular direction of flow. The program will ignore any other possibility. This selects the path through the flow chart.

Selection

You can use your flow charts to show choices and decisions as questions. The flow chart can show what would happen if the answer to the question is ‘yes’ or ‘no’.

How to...

Using the example from Lesson 1.3, the algorithm’s message depends on whether you enjoy reading. This is how we might show selection in plain English.

1 Ask whether the user loves reading.
2 if the user loves reading, then say, ‘NAME loves reading!’
3 Or else say, ‘NAME could try a different author.’

We can show these statements in a flow chart using selection.

Selection and if... then... else statements

The green shape is the question. if the answer is ‘yes’, then the output is ‘NAME loves reading!’
if the answer is ‘no’, then the output is ‘NAME could try a different author.’

Many computer programs use the words if, then and else to make a selection.

Sometimes you will want to show more than two paths in answer to a question.

In our example, we ask whether the user loves reading. Some users will love reading. Some users will only like reading. Some users will not love or like reading at all.
**Now you do it...**

Work with your partner. Apply what you know about selection to the flow chart showing your design of a spam filter.

When you reach your decision points, think of the possible answers to the question. Show the different paths your design could take.

**If you have time...**

Can you think of any other examples of computer programs where selection would be useful?

**Test yourself...**

1. Finish this sentence: Selection happens when...
2. What does implement mean?
Learning outcomes

When you have completed this lesson you will be able to:

- describe iteration
- use loops in a program.

Learn about...

You already know about two tools that we can use to design algorithms. These are sequencing and selection. Computer scientists call these tools programming constructs. Constructs are the tools that you can use to build algorithms.

In this lesson you will learn about another important tool, called iteration.

Sometimes an algorithm has to repeat a step. For example, imagine you are programming a robot to work in a factory. The robot picks up an item and puts it down somewhere else. The robot then moves back to the first position and picks up the next item, and so on. The robot needs to repeat the motion steps many times.

We call this iteration.

Loops

Using iteration, we do not need to write out the same instruction many times. We can simply say that we will repeat (or iterate) particular steps until we tell the algorithm to stop.

Most programmers use the word loop when they talk about iteration. When a program iterates, it loops back to a previous step.

Iteration is an important element of computational thinking. You can apply it to almost any activity in your everyday life. For example, if you want to become good at playing a musical instrument you need to practise. For each piece of music you learn to play, you need to repeat sections. You improve each time you repeat correctly.

How to...

Imagine your avatar from Lesson 1.4 sees treasure down each woodland path and wants to collect it. How might you represent this in a flow chart?

The flow chart starts by stating that there are three paths. The flow chart defines the current path as Path 1. The avatar is instructed to walk down the current path. The avatar collects the treasure. The avatar walks back to the starting point. The loop means that the next path is now the current path. After Path 1 comes Path 2, then Path 3.
Now you do it...

Work with your partner. Apply what you know about loops to the flow chart that describes your spam filter program. You may want to change your flow chart in some way. Ask yourself these questions:

- Where are there repeated actions in your flow chart? For example, do you want your spam filter to repeat a sequence of steps for a second email, or more?
- Can you show these in a more efficient and elegant way by using loops?
- Do you need to say how many loops there will be?

If you have time...

How would you change the flow chart if you had ten paths? How would you change it if you had one million paths?

Test yourself...

1. Complete the words naming the three algorithm constructs we use in computational thinking:
   S __________  S __________  I __________

2. Why is iteration important in programming?

FACT

Alan Turing (1912–1954)

Alan Turing was a famous British mathematician and early computer scientist. In 1936 he invented the Turing machine, which is an idea-based model of how a computer could work. The Turing machine would not work without the idea of iteration.

Key words

Iteration: Iteration means repeating a step or task in an algorithm or computer program.

Loop: To loop means to go back to a previous step. Looping happens in iteration.
Learning outcomes
When you have completed this lesson you will be able to:
* evaluate other people’s work
* give helpful feedback.

Learn about...
Once you have decomposed a problem and developed a possible solution, it is time to evaluate what you have done. You need to be sure that the solution is fit for purpose. This means that the solution is able to completely solve the problem.

Sometimes, when you have been working on an idea, it is difficult to look at your work critically. You have worked so hard that it is not easy to see where the faults are. There are two simple ways to deal with this.

1. You can evaluate your design against set criteria. The criteria are the things to judge the program against. You can use criteria to tell you whether a program works well or not.
2. You can pass your design to someone else so that person can test it and give you helpful feedback.

How to...
A good algorithm should have characteristics that fit together like a jigsaw puzzle.

As you write an algorithm, you need to ask whether the problem has been properly decomposed. Decomposed means being broken down into parts. Efficient means doing the best work for the least effort. Elegant means that the algorithm is very clear for someone else to understand. Correct means that there are no mistakes in the algorithm.

If an algorithm has all of these pieces it can solve the whole problem quickly, using as little work as possible. If you have agreed design criteria at the start of a project, it is easier to evaluate an algorithm. If you do not have agreed design criteria, these puzzle pieces are a good place to start.

Here is an example of someone starting to make a cup of tea. Can you see where the process might be improved?

Using this feedback, the algorithm could look like this:

The algorithm is not correct. There are some steps missing. The process could be improved by thinking through the decomposed problem more carefully.

The algorithm is not properly decomposed, so it is not efficient. It could be improved by breaking the steps down further. For example, break the algorithm down further with these steps: getting a cup, getting water, choosing what kind of tea. Each of these steps would also have steps within it.

Getting a cup
- Take a cup out of the cupboard in the kitchen.
- Put the cup on the table.

Getting water
- Pick up the kettle.
- Take the kettle to the sink.
- Turn on the tap.
- Pour water into the kettle.
- Turn off the tap.
- Return the kettle to the plug.
- Heat the water in the kettle.

Choosing tea
- Walk to the cupboard that has tea in it.
- Open the cupboard.
- Look at the kinds of tea in the cupboard.
- Pick up the mint herbal teabags.
- Close the cupboard.
- Walk to the cup
- Put a mint herbal teabag in the cup.

Decomposed  Efficient  Correct  Elegant
Giving others feedback

There are three important things to remember when you are evaluating other people’s work and giving them feedback.

1. **Pick up the kettle**
2. **Take the kettle to the sink**
3. **Turn on the tap**

1. **Be specific**
   - Don’t say: You’ve missed some steps.
   - Do say: You need to add steps here and here.

2. **Keep it about the work**
   - Don’t say: Your work is dull.
   - Do say: You could make your algorithm shorter by adding a loop here.

3. **Show where things are working well**
   - Point out things that have been done well.

**Giving helpful feedback**

**If you have time...**

Improve your algorithm based on the feedback you have been given.

**Test yourself...**

1. What criteria can we use to judge whether an algorithm is fit for purpose?
2. What are the characteristics of good feedback?
3. Summarise the main strengths and weaknesses of the spam filter you have designed.
4. How could you address the weaknesses?

**Key words**

- **Criteria**: The criteria are the things to judge a program against.
- **Efficient**: Efficient means doing the best work for the least effort.
- **Elegant**: Elegant means to be very clear for someone else to understand.
- **Fit for purpose**: Fit for purpose means that a program does the job it is supposed to do.

**Now you do it...**

By now, you should have a spam filter design that is ready to share.

1. Work in a group. Develop evaluation criteria for your spam filters. Is your design efficient, correct, decomposed and elegant?
2. Now use your criteria to evaluate the design of another group. Show where the design is strong. Show the areas that need improvement.
3. Give your feedback to the other group.
Overview

In this chapter you learned some basic principles of computational thinking. You used computational thinking principles to design a simple spam filter. You have learned how to:

- describe computational thinking
- use decomposition to break a problem down into smaller parts
- describe an algorithm
- use pattern recognition
- use a flow chart to describe your problem-solving

Test questions

Answer these questions to check how well you have learned this topic.

1. What is computational thinking?
2. Why is computational thinking important to programming?
3. Give an example of how you can use computational thinking in everyday life.
4. What are the three constructs you can use to design algorithms?
5. Explain what these words and phrases mean using your own words.
   - a. pattern recognition
   - b. sequencing
   - c. selection
   - d. iteration
6. What is a loop?
7. Give an example of an algorithm where a loop would be useful.
8. Think back to the algorithm evaluation jigsaw. Which four words can you use to help decide whether an algorithm is fit for purpose?

Assessment activities

Imagine you want to make a simple meal.

Starter activity

Choose one of these meals and decompose the task of making the meal.

Intermediate activity

Draw a flow chart to show how you would make the meal.

Extension activity

Look at the flow chart made by a partner. Evaluate your partner’s flow chart. Give your partner helpful feedback on how to improve the flow chart.