











**100 Ideas for  
Secondary Teachers:  
Outstanding Computing  
(Bonus) Lessons**

**Simon Johnson**



# Contents

Introduction	i
Testimonials	ii
How to use this book	iii
<b>Part 11: Bonus Ideas</b>	<b>1</b>
 What's your elf name?	2
 Guess / _ _ e / word	3
 Cards against humanities	5
 Shakespearean complement generator	7
 Hacking the news	8
 Data science detectives	9
 Code tracing	10
 Make me happy	11
 Worked examples	12
 Team teaching	14

# Introduction

The hardest part of writing 100 ideas for *Secondary Teachers: Outstanding Computing Lessons* was not coming up with 100 ideas. Instead, the biggest challenge was trying to find the right balance of pedagogy and practical activities. Because of this, some ideas did not make it into the final book.

So, as a thank you to all those who have already purchased a copy of my book and as a little teaser for those who are still undecided, here are 10 hand-picked ideas that did not make the final cut.

## **What is 100 ideas for Secondary Teachers: Outstanding Computing?**

Part of the 100 ideas series, published by Bloomsbury Education, *100 ideas for Secondary Teachers: Outstanding Computing* is a collection of 100 practical, tried-and-tested ideas for teaching computing. It is aimed at computing teachers of all levels, whether specialist or non-specialist, newly qualified or experienced. To find out more about 100 ideas for *Secondary Teachers: Outstanding Computing Lessons* or to read testimonials from people who have purchased the book, please visit: [teachwithict.com/100ideas](http://teachwithict.com/100ideas)

To find more about the 100 ideas series, or to purchase a copy of the original book, visit: [bit.ly/100IdeasSecCS](http://bit.ly/100IdeasSecCS)

## **Acknowledgements**

IDEA 107. Code tracing – NCCE Pedagogy Quick reads, <https://blog.teachcomputing.org/code-tracing/>

IDEA 109. Worked examples – NCCE Pedagogy Quick reads, <https://blog.teachcomputing.org/using-worked-examples-to-support-novice-learners/>

# Testimonials

*9.1/10 – "Another great book within this fantastic series of practical teaching books, offering great support and resource ideas to support computing at secondary level." – UKEdChat*

*"A pocketful of inspiration for your next computing lesson." – Hello World Magazine*

*"An outstanding book that can simply be picked up off the shelf as a means to inspire any teacher when delivering computing. Simon has shared these super quick ideas that will have impact in every classroom, from rubber duck debugging to pedagogical approaches in the subject. This is an excellent read!" – Matt Warne, Head of Computing and Digital Learning at RGS The Grange, CAS Master Teacher and Computing Champion, @MattWarne*

*"Read this if you teach Computing! Packed with lesson ideas but more importantly loads of nuggets of wisdom about pedagogy: research-informed techniques that work, all packaged into bitesize chunks you can read in your break or put this on your summer reading pile to come back refreshed and raring to go in September. I will definitely be doing "code golf" (write a program to solve a problem in as few lines as possible), "crazy characters" (back-to-back drawing to teach the importance of clear instructions) and "intelligent pieces of paper" (introducing AI by playing noughts and crosses against a written algorithm). Simon has pulled together a goldmine of inspirational and powerful ideas which is essential reading for the Computing teacher." –*

*Alan Harrison, Head of Computing at William Hulme's Grammar School, CAS Master Teacher, @MrAHarrisonCS*

## How to use this book

This book includes quick, easy and practical ideas for you to dip in and out of to help you deliver effective and engaging computing lessons. Each idea includes:

- a catchy title, easy to refer to and share with your colleagues
- an interesting quote linked to the idea
- a summary of the idea in bold, making it easy to flick through the book and identify an idea you want to use at a glance
- a step-by-step guide to implementing the idea.

Each idea also includes one or more of the following:

### Teaching tip

Practical tips and advice for how and how not to run the activity or put the idea into practice.

### Taking it further

Ideas and advice for how to extend the idea or develop it further.

### Bonus idea



**Bonus ideas in this book that are extra-exciting, extra-original and extra-interesting.**

Share how you use these ideas and find out what other practitioners have done using #100ideas.

Online resources to accompany this book can be found at [www.teachwithict.com/100ideas](http://www.teachwithict.com/100ideas). Here you will find editable versions of all the code needed to run the ideas. You can copy and paste the code or tweak it to suit your requirements. There are also a variety of downloadable resources such as pre-prepared cards and worksheets to help you put the ideas into practice.



**Bonus  
ideas**



**Part 11**



# What's your elf name?

'An elf is for life - not just for Christmas!'

To find your elf's name, take the first letters of your first and last name and match them to the list of elf names. For example, if your first name is "Ben", then your elf's first name is "Blustery". If your last name is "Barnes", your elf's last name is "Bells".

First initial	Elf first name	Last initial	Elf last name
A	Angelic	A	Angel
B	Blustery	B	Bells

### Taking it further

Have the program ask the user to enter their full name and automatically extract the first letter from the first name and last name. For example:

```

firstName =
input("What is your
first name? ")

firstInitial =
firstName[0].upper()
    
```

### Bonus idea

Challenge students to create their own name generator. For example, superhero or game name generator etc.

Start by displaying the following code:

```

SantasList = {"ANGELA": "Nice", "BILLY":
"Naughty", "DUNCAN": "Nice"}

childsName = input("Enter child's name: ")

result = SantasList[childsName]

print("Naughty or nice?: " + result)
    
```

Challenge the students to predict what will happen before revealing the answer. Explain that the code uses a feature in Python called a dictionary which is a collection of items containing a "key" and a "value".

Have the students copy the code then modify it by changing some of the names. Ask the students to test what happens if they don't enter a name all in upper case. Explain that we can convert a user's input to uppercase using `.upper()`. Have students add the following code after line 2:

```

childsName = childsName.upper()
    
```

Explain how the elf name generator works (see description above) and share the worked example (see idea 109). Challenge the students to create their own elf name generator by completing the worked example. The full list of elf names, including worked example, can be found at:

[www.teachwithict.com/elf.html](http://www.teachwithict.com/elf.html)

# Guess / \_ \_ e / word

'Sir, is the answer Donald Trump – again?'

**The game will start by picking a random word or phrase from a list and then replace each letter with a dash '-' or 'underscore '\_'.** e.g., CAT becomes \_ \_ \_. **The player must then guess the word by suggesting different letters. If the player guesses correctly, the letter is revealed in the phrase. However, if the player makes an incorrect guess, the player loses a life!**

## Part 1 – String manipulation

Start by demonstrating how to replace a character at specific index in a string using the string slicing method (see example below). Have the students predict what will happen before copying and running the code for themselves:

```
string = "TRUMP"
position = 3
string = string[:position] + "A" +
string[position+1:]
print(string)
```

Inform the students that their game needs to replace each letter in the chosen word or phrase with an 'underscore '\_' . Challenge the students to modify their code so that it changes the 3rd letter to an underscore '\_' .

Explain to the students that, in order to change all the letters in their word or phrase to a dash, they are going to need to use a loop. Share the following example code:

```
string = "TRUMP"
character = "-"
for i in range (len(string)):
    string = string[:i] + "-" + string[i+1:]
print(string)
```

## Teaching tip

Differentiate the activity by providing different (or faded) worked examples (See idea 109).

Ask the students to run the sample code and identify the problem. Draw out answers such as 'the program just prints out one long line' or 'there are no spaces in between each of the hidden letters'. Have the students replace the last line in their code with the following and run their program again:

```
print(" ".join(string))
```

## Part 2 – Dealing with lists.

Explain that, to make the game more challenging, each time the player plays the game it must pick a new word at random from a list. Share the following code with the students. Explain that this will form the basis for their 'guess the word' game.

```
import random  
names = ["Bob", "Dave", "Stuart"]  
print(random.choice(names))
```

Challenge the students to modify the code so that it adds the word 'Minion' before the randomly selected name. For example:

```
print("Minion" + " " + random.choice(names))
```

## Making the game

Share the worked example (See idea 109) with the students and have them fill in the gaps in the code using what they have learned. The worked example, including a sample solution to this problem, can be found here:

[www.teachwithict.com/guessing.html](http://www.teachwithict.com/guessing.html)

### Taking it further

Challenge the students to add a 'life counter' which removes a life every time a player guesses incorrectly.

The game ends if the player guesses correctly or they run out of lives (whichever comes first).

# Cards against humanities

'A phrasal template word game for students who don't like History, Geography, Modern Foreign Languages, or Religious Studies etc...'

**Cards Against Humanities is a phrasal template word game, where one player prompts other players for a list of words to substitute blanks in a quote from History, phrase in French or Spanish, or question about Geography etc., before reading out the completed phrase or question aloud. For example:**

**"Je voudrais un ananas is French for I would like a blank".**

### Taking it further

Challenge the students to modify their code so that it gives the player a choice of words, chosen at random, from a list or file.

There are several ways to create a Cards Against Humanities game with code. In this example, using Python, the program will pick a phrase or question at random from a list of humanities subjects (History, Geography, Modern Foreign Languages or Religious Studies etc.) before asking the player to substitute the missing word (noun, verb etc.).

Start by having the students create a simplified versions of the game using the code sample below. Have the students tinker with the code, for example change the phrase or add more than one input etc.

```
noun = input("Enter a noun: ")
print("Rumble in the " + noun)
```

Next, introduce the students to the string replace method. Explain that the students will need to use the **.replace()** method to replace the blank(s) in their Cards Against Humanities game with the player's suggestions. Example:

```
string = "Rumble in the blank"
string = string.replace("blank", "toilet")
print(string)
```

Explain that, in the previous example, the line `string = string.replace("blank", "toilet")` looks for the word "blank" in the sentence and replaces it with the word "toilet". Challenge the students to modify their code so that it replaces the word "blank" with a word input by the user.

Next, inform the students that the game needs to pick a phrase or question at random from a text file. Share the following code:

```
import random
# Open the Cards Against text file
f = open('cards.txt','r')
# Read the whole file and store each line
in a list
cardList = f.readlines()
# Choose a random line from the list
card = random.choice(cardList)
# Print out the question/phrase card
print(card)
```

**Note:** In order for the game to work, the students will need to create a text file, called 'cards.txt', containing at least 3 phrases or questions, and save this in the same folder as their python script.

Example:

Je voudrais un ananas is French for I would like a **blank**  
The 45th president of the United States is **blank**  
Plucking is a process of erosion that occurs during **blank**

Finally, have the students use everything they have learned to create their own 'Cards Against' game which picks a question / phrase at random and replaces the blank or blanks with suggestions from the user.

Example solutions to this problem can be found here:  
[www.teachwithict.com/pythoncards.html](http://www.teachwithict.com/pythoncards.html)

#### Bonus idea

Have the students modify their code to create a Mad Libs® game (See idea 85).

# Shakespearean compliment generator

'Shall I compare thee to a ~~summer's day~~?  
beautiful sunset?'

If I had to pick just one, my favourite python activity would have to be the 'Shakespearean insult generator' (See idea 81). However, whilst also being one of my more popular activities, it's not without its critics! So, by popular demand, let me introduce the 'Shakespearean compliment generator'!

## Taking it further

Explain to the students that having very long lists is not very efficient, plus it makes the code more difficult to debug.

Challenge students to improve their solution by having a file for each list of nouns / adjectives.

The program works by taking the familiar Shakespearean sonnet 'Shall I compare thee to a summer's day?' but replaces the last two words with a random *adjective* followed by a random *noun*.

Start by displaying the following code on the board / screen and have the students predict what will happen before revealing the answer:

```
import random
coinFlip = ["Heads", "Tails"]
print(random.choice(coinFlip))
```

Have the students copy the code then modify it to simulate the roll of a dice (or pick a name at random). See idea 86: Sorting hat.

Next, share the following worked example (see idea 109). Have the students predict what will happen before running the code and then challenge them to create a 'Shakespearean compliment generator' using what they have learnt.

```
#Superhero name generator
import random

firstname = ["Red", "Green", "Yellow"]
surname = ["Claw", "Spider", "Dynamo"]

print(random.choice(firstname) + " " +
random.choice(surname))
```

# Hacking the news

'Miss, is the world really going to end tomorrow?'

## A fun way to teach HTML!

Ever wanted to change the main headline on the BBC News page or change the picture on the school's homepage to Hogwarts? Well, now you can thanks to a free plugin from Mouse.org! X-Ray Goggles, originally developed by Mozilla, is a free browser extension which lets you see the code behind any webpage then lets you change it!

### How it works

Start by having students visit <https://x-ray-goggles.mouse.org/> and follow the instructions for their browser.

Once the students have installed the plugin, have them load a suitable online news page. For example, [bbc.co.uk/news](http://bbc.co.uk/news) (or, if using with younger students, [bbc.co.uk/newsround](http://bbc.co.uk/newsround))

Next, instruct the students to click on the x-ray goggles icon in their bookmarks toolbar and hover their mouse over the page elements to see the HTML/CSS code. Then, by clicking on and editing the HTML code, have them change some of the content. For example, students could change the main headline or replace the featured image. Once complete, instruct the students to take a screenshot of the finished news article and share it with their friends.

### Teaching tip

Have the students take a screenshot of their chosen webpage before and after remixing it.

### Bonus idea



Use X-ray Goggles to create a starter activity to introduce a lesson on 'Fake News' (See idea 24).

# Data science detectives

'The ability to take data - to be able to understand it, to process it, to extract value from it, to visualise it, to communicate it - that's going to be a hugely important skill in decades to come.' - Hal Varian (Google)

**At its heart, data science is all about solving problems. The ability to think analytically and to recognise patterns are not just key computational thinking skills, they are key employability skills too!**

## Teaching tip

Gamify the activity by placing the students into teams and making it a competition to see which team can come up with the most innovative solution.

A famous example of pattern recognition, an essential component of data analytics, is the case of John Snow vs. cholera. John Snow, not to be confused with the character from Game of Thrones, was an English physician most famous for his work in tracing the source of a cholera outbreak in Soho, London, in 1854. At that time, cholera was believed to be spread through the air but, with the help of data visualisation, Snow was able to substantiate his theory that all the deaths could be attributed to a particular water pump used for drinking water. This information helped convince the local council to immediately remove the pump handle resulting in countless lives being saved!

Start by giving students a copy of the very same map used by John Snow along with testimonies of people living in the area at the time ([teachwithict.com/data.html](http://teachwithict.com/data.html)). Next, have the students 'decompose' the problem by dividing the city into zones and interrogating the data for each zone. Using pattern recognition, have the students narrow down the origin of the infections by identifying the zones with the earliest reported cases and highest density of infection rates. Then, using abstraction, have the students eliminate the unnecessary details from the interviews taken at the height of the epidemic. Using all of the available data, the students should reach the same conclusion as John Snow but should also be encouraged to come up with their own innovative solutions to the problem.



# Code tracing

‘A simple strategy for reducing cognitive load.’

**It is common practice to teach children to read before they learn to write. So why is it that, when teaching them to code, we often get children to write code before they can read it?**

Code tracing is a well-established approach designed to help children with code comprehension. Similar to strategies often used for teaching reading and writing, code tracing requires learners to read, understand and analyse code before learning to write code for themselves.

### Taking it further

Once the students have successfully traced the given problem / solution, challenge them to create their own program based on the code they have traced.

### How it works

Start by giving students some sample code to read before asking them to predict the outcome; this should preferably be done away from the computer so that the students focus on reading the code rather than executing it. Next, instruct the students to step through the code, line-by-line, and record the expected behaviour (or outcomes) and execution flow (see example below).

<pre> countdown = 5 while countdown &gt; 0:     print(countdown)     countdown = countdown - 1 print("Lift off!")         </pre>	<ol style="list-style-type: none"> <li>1. Highlight all the expressions.</li> <li>2. Use arrows to show the order of execution.</li> <li>3. Step through the program and fill in the variables and output tables.</li> </ol>
<p><b>Variables</b></p> <p>countdown</p> <p>5</p> <p>4</p> <p>3</p> <p>2</p> <p>1</p>	<p><b>Output</b></p> <p>5</p> <p>4</p> <p>3</p> <p>2</p> <p>1</p> <p>Lift off!</p>

# Make me happy

'Be kind whenever possible. It is always possible.' – Dalai Lama

**A fun way to teach kindness online is to have students create their own sentiment analysis bot which reacts to what they say!**

## Taking it further

Have students 'Think, Pair, Share' why is it important to be kind online?

Machine Learning for Kids is a free online tool for introducing children to how Machine Learning systems are trained, how they are used, and some of the real-world applications of Artificial Intelligence (AI).

Explain to the students that they will be making a chatbot that reacts to what they say. If they compliment it, it will look happy. If they insult it, it will look sad.

Direct students to [machinelearningforkids.co.uk](http://machinelearningforkids.co.uk) and have them click on 'try it now'. Ask the students to follow the on-screen instructions to create a machine learning project which recognises 'text'.

Next, have the students 'train' their bot – the students will need to create two labels (one called 'kind things' and another called 'mean things') and populate each label with examples. The students will need to provide at least 6 examples of each.

Once the training is complete, have the students test their model to see what the computer has learned. It is recommended that the students start by testing the model using the examples they supplied before trying examples that the computer hasn't seen. Once the students are happy with their trained model, they can move on to coding their sentiment analysis bot in Scratch.

Step-by-step instructions for this activity, including code, can be downloaded at: [teachwithict.com/happy.html](http://teachwithict.com/happy.html)

# Worked examples

‘An effective strategy for reducing cognitive load for novice learners.’

**Commonly used in a range of subjects, including Maths and Science, worked examples are a form of ‘scaffolding’ designed to help reduce cognitive load placed on novice learners.**

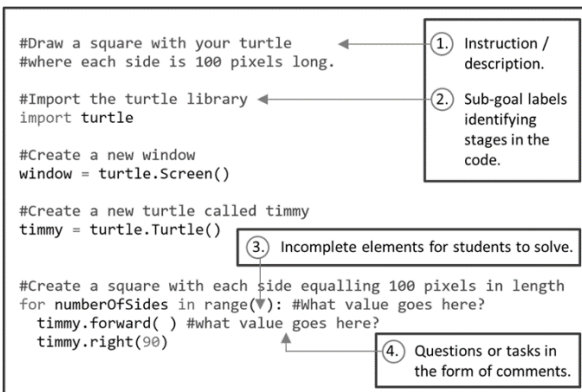
When a learner is given a partial (or near complete) solution to a problem, they do not have to rely so much on their long-term memory. Worked examples aim to reduce the superfluous cognitive load, often placed on a learner’s working memory when faced with new challenges, by providing a model solution to a problem.

### Teaching tip

Use worked examples while a concept is new, then fade the support over time.

### How it works:

- Prepare a partial solution to a problem for the students to complete.
- Provide a brief description of what the code is meant to do, this can be in the form of comments or verbal explanation.
- Use sub-goal labelling to identify each of the important steps in the code.
- Add questions / prompts with the aid of comments or annotations.



As learners become more proficient, the benefits of using worked examples are reduced to the point at which they may start to become a hindrance rather than a help! As with any types of scaffolding and support, worked examples should be faded to provide stretch and challenge.

#### Teaching tip

Worked examples, when used in conjunction with traditional practice problems, can also help learners develop strategies for solving similar problems.

### Faded examples

Imagine a faded example as a sort of fill-in-the-blank challenge, but with code. Start by giving students a near complete solution to a problem and challenge them to fill in the blanks. Over time, add more and more blanks until the student is essentially writing the solution for themselves.

The following demonstrates a greatly simplified example of a fading using the Turtle library in Python.

#### *Example 1 (First phase of fading)*

```
#Import the turtle library
import turtle
#Create a new window
window = turtle.Screen()
#Create a new turtle called timmy
timmy = turtle.Turtle()

#Create a square with each side
#equalling 100 pixels in length
for loopCounter in range( ): #What value goes here?
    timmy.forward( ) #what value goes here?
    timmy.right(90)
```

#### *Example 2 (Final phase of fading)*

```
#Import the turtle library
#Place missing code here
#Create a new window
#Place missing code here
#Create a new turtle called timmy
#Place missing code here

#Create an equilateral triangle with each side
#equalling 100 pixels in length
for numberOfSides in range( ): #What value goes here?
    #Place missing code here
```

# Team teaching

'The most valuable resource that all teachers have is each other. Without collaboration our growth is limited to our own perspectives' – Robert John Meehan

---

**Sometimes we scan the galaxy for good practice only to find it was right on our doorstep all along!**

---

## **Taking it further**

Plan an extended project with a colleague. Strategies such as Project Based Learning (PBL) and Genius Hour are great for this as they allow you to combine multiple subjects around a common goal.

Two heads are better than one, and that goes for teaching too! Planning and teaching a lesson with a colleague is not only a great way to build good working relationships but it's a great way to share best practice too! Here are some things to consider:

### **Plan a lesson with a colleague**

Planning a lesson with a colleague in your department is a great way to share best practice but also a great way to play on each other's strengths. Agree on an upcoming topic and plan a lesson together.

### **Leave your comfort zone**

Team teaching with a colleague can be extremely rewarding, even more so if it's with someone from another department. You may have an area / topic on your curriculum that you're unfamiliar with or not sure how to teach but a colleague from another subject area may be able to help. In return, they may learn something from you.

# **100 Ideas for Secondary Teachers: Outstanding Computing Lessons**

**Simon Johnson**

**Bloomsbury**  
[bit.ly/100IdeasSecCS](http://bit.ly/100IdeasSecCS)

**Amazon**  
[bit.ly/100ideasCS](http://bit.ly/100ideasCS)