2024

Algorithms & Computational Thinking

Orb education

Dan Collingbourne

Do This	Section
☐	1. Wordlist
☐	2. Computational Thinking
☐	3. Making Hamburgers
☐	4. Problem Solving
☐	5. Meal Planning
☐	6. Flowcharts
☐	7. Branching (or Selection)
☐	8. Iteration
☐	9. Scratch Programming
☐	10. Pseudocode
☐	11. Calculating Change
☐	12. Robot Mazes
☐	13. Introducing Python
☐	14. Errors and Tracing
☐	15. Evaluating Solutions
☐	16. Your Age and other Algorithms
☐	17. Creating a Quiz
☐	18. Calculating Change in Python
☐	19. Searching
☐	20. Sorting

In the tasks below, we will try and solve a variety of problems. Whilst doing this, we will consider how we might write down a set of instructions for the solutions. Writing instructions is the first step to creating algorithms.

Aim: *To practise problem decomposition using approaches such as divide and conquer.*

Task 1 – Discussing a Problem

You're on a holiday in Thailand and end up with a pocket full of change. You want to know how much money you have in total.

The Thai currency is made of Satangs and Baht, where 100 Satangs equals 1 Baht. There are 1, 5, 10, 25 and 50 Satang coins. There are also 1, 2, 5 and 10 Baht coins.

Discuss with your partner or group how you would add up all your change and find the total. Once you have a plan, start to think through instructions that could be written down for someone else to follow.

Task 2 – Logic Problems

The logic problems below will help you think about the steps required to find a solution. There are answers to these problems online, of course, but you gain nothing by looking them up before you have really tried to think about the issues yourself.

a. The Wolf, Goat and Cabbage Problem

A farmer with a wolf, a goat and a cabbage must cross a river by boat. The boat can carry only the farmer and a single item. If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage. How can they cross the river without anything being eaten?

b. The Failing Torch

Four people want to cross a bridge at night but it can only hold two people at a time. They also need a light for the crossing but only have one torch. Alan can cross the bridge in 1 minute, Belle in 2 minutes, Chloe in 5 minutes and Drew in 8 minutes. When two people cross the bridge together, they must move at the slower person's pace. Can they all get across the bridge if the torch only has 15 minutes of power remaining?

Extension: The Fastest 3 Horses

You need to find the 3 fastest horses from a group of 25. There is no stopwatch and the racetrack has only 5 lanes. No more than 5 horses can be raced at a time. How many races are necessary to find the 3 fastest horses? ***Hint:*** *Set up some races and then work out which horses you can eliminate.*

Branching occurs when there are options. For example, if you were making a cup of tea for a friend, you might ask the question "Do you take sugar". The action taken would depend on the answer you get.

Aim: *To learn about branching (also known as selection) in algorithms.*

Ask the question "Do you take sugar".
IF *your friend says "Yes"* ***THEN*** *add sugar.*

When programming, branching often involves the words **IF** and **THEN**.

Task 1 – Hotdogs ('IF…THEN' in an Algorithm)

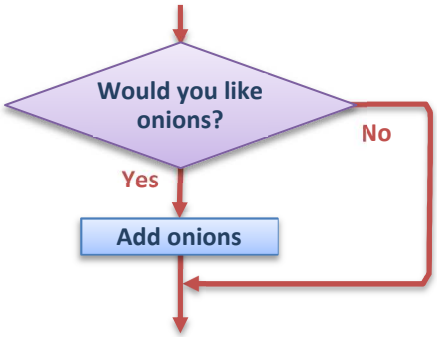
Write some simple instructions for making a hotdog for your friend starting with a cooked sausage, cooked onions, a bread roll and some sauce. Include a part where you find out if your friend would like onions and add these if they do. Use the words **IF** and **THEN**. Do the same for the sauce.

Task 2 – The Hotdogs Flowchart

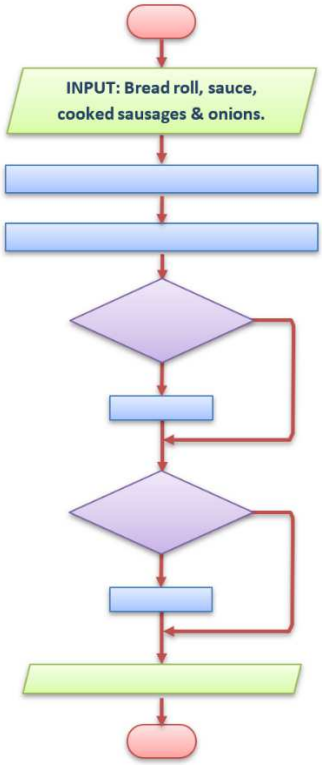
We will create a flowchart for our hotdog instructions. A branch is created in the flow each time a selection is made. This selection is shown by the diamond shape, as shown below. The diamond can be of any colour.

A decision, usually a 'Yes' or 'No'

The section of flowchart below shows where the decision about onions is made. There will be a similar section like this for the sauce.



Create a complete flowchart for your hotdog instructions from Task 1. It should look something like the one on the right but include instructions, questions and 'Yes / No' answers etc.



Note: *If you have the skills, you may design the flowchart using the shapes in Microsoft Word.*

Scratch is a visual programming language used to create animations and games. It provides a great stepping stone to the more advanced world of computer programming. It can also be used for maths and science projects, animated presentations and interactive art and music.

Aim: *To learn some visual programming using Scratch.*

Task 1 – Having a Look

a. Go to https://scratch.mit.edu/ (or type 'Scratch' into your search engine).

b. Watch the introductory video.

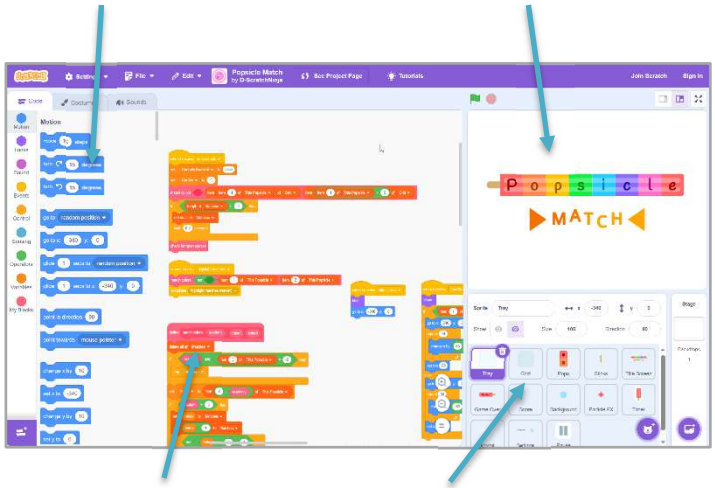
c. Have a look at a few of the *Featured Projects*. Click on the green flags to run the programs.

d. Choose a simple looking project where not too much is going on and click the *See Inside* button. The screen should look a bit like the one on the right. Click on the different *sprites* in the bottom right section. You may notice that the blocks in the script area change. Each sprite can have different scripts (and actions) associated with it.

e. Have a look at the scripts and see if anything makes sense. You might recognise actions from the project such as *move* or *change color*.

Script Blocks
Drag these blocks into the script area.

Stage
See your project in action.



Script Area
Build your scripts here.

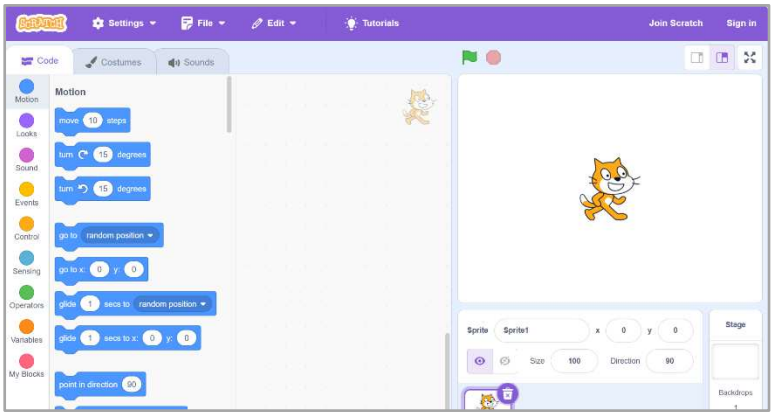
Sprites
All the things that will be visible in your project.

Task 2 – Having a Go

It's never too soon to jump in and have a go. You can't break anything.

a. From the homepage, click on the *Start Creating* button.

b. You should see a cat on the stage which doesn't seem to do anything. You may close the tutorial for now.



The activities below will help you create a computer program that asks questions for a quiz or test. The program checks the answer entered by the user and displays a message if it is correct. If the answer is incorrect then it will let the user try again.

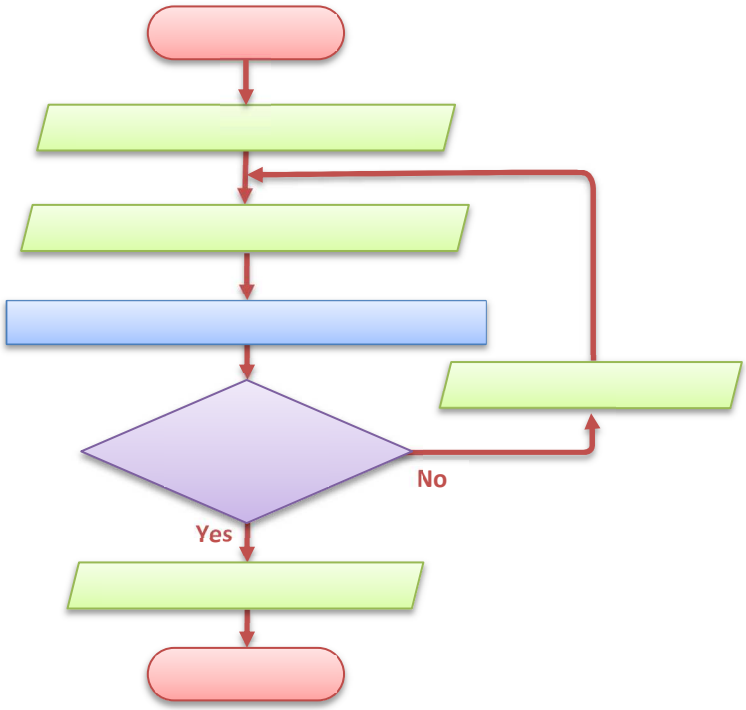
Aim: *To practice flowcharts, pseudocode, Scratch and Python.*

Task 1 – Single Question Flowchart

The flowchart on the right should describe the algorithm for a single question and answer.

Use the terms below to complete the flowchart.

- **STORE answer in *answer* variable**
- **End**
- **OUTPUT 'Try again'**
- **INPUT user enters an answer**
- **OUTPUT 'Well done'**
- **Start**
- **OUTPUT 'What is 22 x 3?'**
- **Is *answer* equal to 66?**



Task 2 – The Pseudocode

Complete the pseudocode for the flowchart in the previous task.

OUTPUT 'What is 22 x 3?'

REPEAT

INPUT user inputs their answer

STORE the input in the *answer* variable

Task 3 – Tracing the Algorithm

Work through the flowchart, imagining exactly what will happen if the answer given is correct or incorrect. Are you happy with the way it works or do you think it could be improved?

Hint: *Think through exactly what the user will see if their answer is incorrect? How will they reread the question and recalculate their answer?*

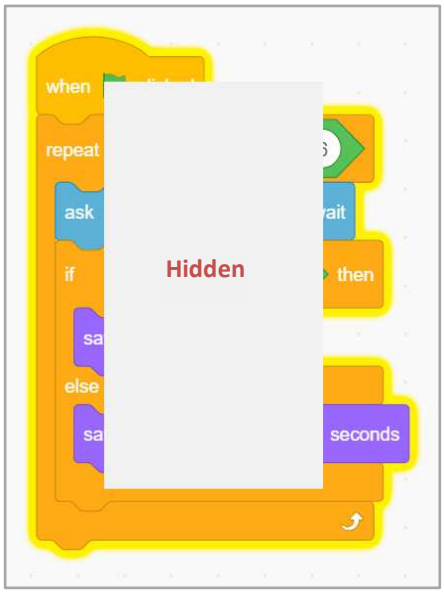
If you can improve the algorithm, sketch or explain the changes you would make to the flowchart and pseudocode.

Task 4 – Programming in Scratch

The Scratch script on the right will ask the question and display the different messages mentioned previously. The script has been partly hidden.

We have improved the algorithm so that if the answer given is incorrect, then the message 'Try again' is displayed for 2 seconds and then the question displayed again.

Go to https://scratch.mit.edu/ and create a script for the algorithm. Either log in to your account or download the script file for later use.



Task 5 – Programming in Python

Use the website www.online-python.com (or another coding website of your choice) to create the program shown below right. Answer the questions.

```
1  import time
2  answer = ""
3
4  while (answer != 66):
5
6      answer = input("What is 22 x 3?")
7
8      if (answer == 66):
9          print ("Well done")
10     else:
11         print ("Try again")
12         time.sleep(3)
13
```

1. There is a semantic error in the program which is preventing it from working. Find and fix the error.

   Copy the working script to a document on your computer and give it the title "**17.5 Single Question**".

2. We have used the *comparison operator* 'not equal to' on line 4. What is the symbol for this operator?

3. We use another comparison operator with a symbol ==. What do you think this operator means?

4. Line 2 defines a variable called *answer*. What happens if the script is run without this line?

5. Line 12 uses the *sleep* function to delay the program for 3 seconds. Before using this function, we must import the *time* module. On which line does this import occur?

Extension Project

Your challenge is to build on these ideas and develop a quiz with 10 questions. You should create the program using the Python programming language.

Although it would be easy to simply copy the previous code 10 times and edit the questions, we will use more efficient programming. Our solution will involve a list of 10 questions and another list containing the answers. The program will run through the questions in the list.

Guide notes:

1. A list is an ordered set of data. The code on the right creates a list called *numbers_list*. It has 4 elements.

```
numbers_list = ["One", "Two", "Three", "Four"]
```

2. You may refer to the list elements using indexes as shown. Indexes start at 0, so in our list of 4 elements the indexes are 0, 1, 2 and 3.

```
print(numbers_list[0])
print(numbers_list[2])
```

3. Run through the whole list using the range function as shown. Note that the second number is where the loop stops; it is not included.

```
for i in range(0, 4):
    print(numbers_list[i])
```

4. For your question bank, it may be clearer to create the list as on the right. It starts by defining an empty list then adds (appends) questions one at a time.

```
question_list = []
question_list.append("This is question 1")
question_list.append("This is question 2")
```

5. Use either method shown above to set up your list of answers.

6.

```
11 question_list.a
12
13 answer_list = [
14
15 question = 0
16 answer = ""
17 correct = 0
18
19 while (question
20
21     answer = in
22
23     if (answer
24         correct
25
26     question =
27
28 print("You scor
```

…Continued