Personalising

Computing Science Concepts

-Various inputs
-Sequence
-Variables
-Variable initialisation
-Algorithm
-Completion problem

National Curriculum Programs of Study

Pupils should be taught to:

design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts

use sequence, selection, and repetition in programs; **work with variables and various forms of input and output**

use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs

Research Focus

This planning uses a **completion problem** where some of the code is provided but pupils need to complete it. This one has a twist as they are provided with the idea and algorithm to help them complete the programs.

Computer scientists have **four levels** of construction. The ideas level, Planning level (which includes the algorithm), code level, and execution level (testing the code)

Personalising Monologues or Dialogues using a variable Version B Completion Problem

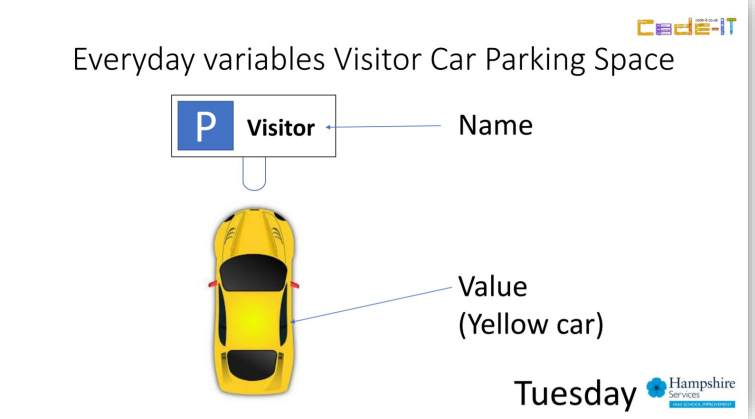
Overview

1. Introduce the concept of text variables through everyday examples
2. Extend the concept of text variables through role play
3. Introduce the challenge by sharing the idea, algorithm and part finished code of different examples [completion problems] ⟺
4. Pupils plan to personalise their own monologues or dialogues using a variable
5. Pupils turn their algorithmic planning into their personalised monologue or dialogue programs
6. Pupils fill in the summative assessment form

1. Introduce the concept of text variables through everyday examples

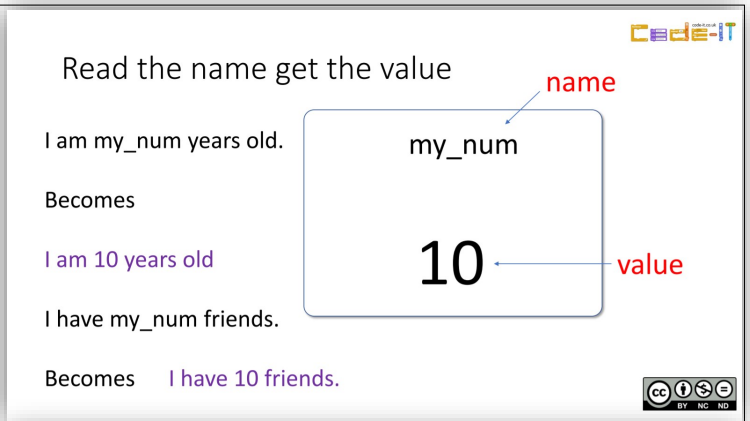
Download **Everyday computing concepts PDF** from http://code-it.co.uk/knowledge or directly from http://code-it.co.uk/wp-content/uploads/2019/04/everydaycomputingconcepts.pdf

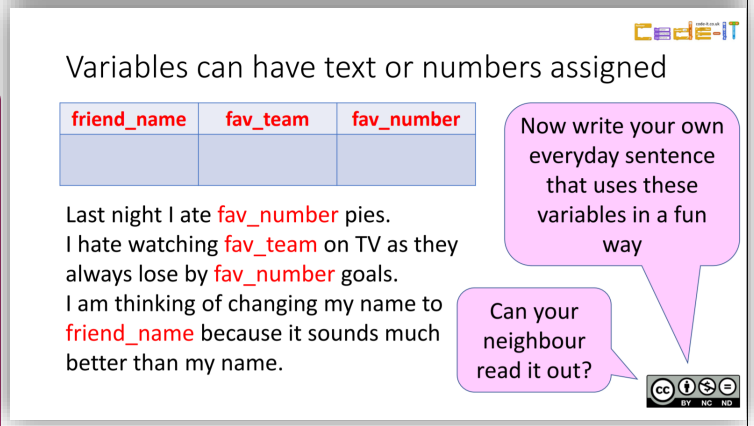
Use the visitor car park example to introduce the idea of text variables in our everyday lives. By linking the concept to its everyday use you are linking to known knowledge which means pupils are more likely to assimilate the idea.



2. Extend the concept of text variables through role play

Download Concepts before coding PDF from http://code-it.co.uk/knowledge or directly from http://code-it.co.uk/wp-content/uploads/2019/04/conceptbeforecoding.pdf
Follow the links in the menu to variables like whiteboards. Use those slides to roleplay and write simple fun text variable algorithms up until and including the second slide shown on the right.





Formative assessment opportunity

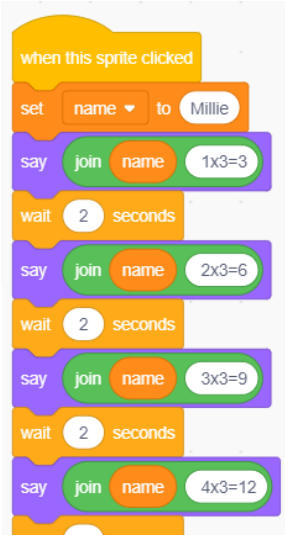
While pupils are writing their own variable algorithms go round and check them all. Is anyone struggling? Have they copied the one on the board exactly? This is often an indication that they are not sure how to create their own or that spelling is an issue. A good supportive activity is to get them to tell you about their ideas that you can use in a joint effort.

3. Introduce the challenge by sharing the idea, algorithm and part finished code of a variety of different [completion problems]

Opening template file

Give pupils either the **dialoguealgorithmvariablesplan** or **monologuealgorithmvariableplan** PDF and direct them to open the dialogue or monologue Scratch example **completion_persondialogueVBScratch** file or **completion_personmonologueVBScratch** These can be found either on the Scratch website at https://scratch.mit.edu/projects/305229088/ **Dialogue** https://scratch.mit.edu/projects/305229200/ **Monologue**

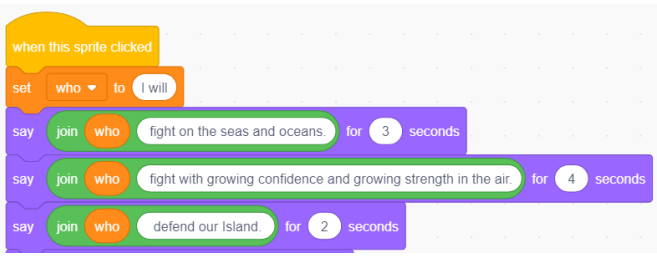
Or as a downloadable Scratch 2 or 3 file to put on your network at http://code-it.co.uk/gold/



Monologues with text variables

Introducing the challenge

Explain that the author got as far as creating their planning and they have created the blocks and put them in the right places but a few sections of code linked to personalisation using variables are not in the



Research References

[1] **Completion problems** are suggested by Sweller in his work on cognitive load theory.

You can read about it here.

[2] **Four levels of abstraction**

This article includes an example of the four levels of abstraction and sign posts the work of Waite and Armoni in using them with school level pupils.

http://code-it.co.uk/algprogdiff/

Collect variable data from user

You could also show pupils how to collect user data from the program user and assign it to a variable.

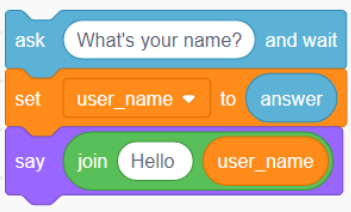
Algorithm example

Ask what is your name

Assign their answer to a variable called user_name

Say Hello user_name

Can become



right places.

Can they use the algorithm plan to help them decide where these extra blocks go?

4, Pupils plan to personalise their own monologues or dialogues using a variable

Pupils will need there original monologue or dialogue planning. Can they identify where to initialise the variable and which parts of the code could be replaced with a variable. Pupils could use a highlighter pen, circle the words to replace with variables or just cross out words to replace with variables and write in the variable (my preferred method).

On occasions some of the original monologue or dialogue choices of pupils will not lend themselves to adapting with a variable. In these cases it can be easier for pupils to start with a new planner.

Formative assessment

Check pupils ideas and algorithms as they are creating. Have they initialised the variable? Have they given it a value? Have they identified multiple places it could be used?

5, Pupils turn their algorithmic planning into their personalised monologue or dialogue programs

Give pupils time to do this and to test their creations. Does it fit in with their dialogue or monologue personalised planning?

7, Pupils fill in the summative assessment form

You can find a summative assessment quick Kahoot Quiz linked at http://code-it.co.uk/gold/

Whilst Kahoot is a limited assessment tool it is free and it is easy for teachers to pass the results back to code-it via phil.bagge@code-it.me so I can look at which method provides best short test results. Not conclusive but useful.

If you pass the results back please

1, Anonymise the results by removing the names

2, Ask the head teacher for permission

3, In the email title say which module you are doing (ie Animation D)

I recognise there is more than one way to solve/describe a problem

I don't just accept the first solution

I look for a range of solution to the same problem

I look for how a project can be extended

I can evaluate my solutions against a set criteria

Handles Ambiguity

Open Ended Problem Solver

I can break complex problems into parts

I can design criteria to evaluate my creations

Evaluates

Insert picture of your students here

I can discover / concentrate on the most important part of a problem

Copes with Complexity

I can identify patterns in problems & solutions

I can contribute useful ideas to a partner or group

I can encourage others to share their ideas

Communicates

Adapts

I can adapt existing ideas to solve new problems

I can develop, test and debug until a product is refined

I lead using all the people talent in my group

Investigates

I make predictions about what will happen

I learn from setbacks and don't let them put me off

Perseveres

I can persevere even if the solution is not obvious

I repeatedly experiment through predicting, making, testing & debugging

@baggiepr

Inspired by Behaviour Rubric created with @MarkDorling and linked at http://code-it.co.uk/attitudes/