```
define basic_procedure
```

```
run basic_procedure
```

**Computing Science Glossary**

**Condition** An action triggered by a condition that may or may not happen.

**Parsons** Code is provided but not connected for pupils to connect

**PRIMM** A strategy that promotes Predicting, Running, Investigating and Modifying code before Making something.

**Algorithm** Part of planning stage before programming written for another human to read.

**Code** Written for a digital device

**USE MODIFY MAKE** A strategy that promotes using and modifying code before creation

**Procedure** A set of instructions given a name that can be used multiple times by referring to the name. Sometimes called a sub routine.

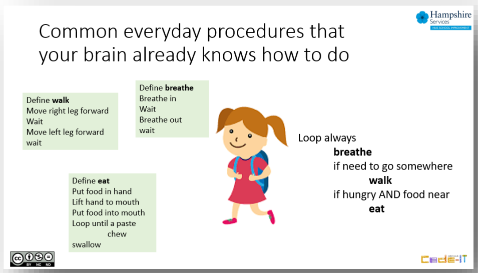
# New Crab Maze Plan

## Questions & Answers

**How does this fit in with other game programming?** See overview document **What age is this for?** KS2 (7-11 year olds) who have completed earlier gaming challenges. This could also be used by Y7 if they have never encountered procedures before). **How hard is this to teach?** Very easy as all the instructions are in the booklets. **How do we Assess learning?** Pupils use answers provided to mark their own work. **Is it in line with NC?** Yes see next page. **Why is there a choice?** All of the methods chosen have good research behind them but we don't know which are best or even if there is a best for all pupils. We do know that it helps pupils to encounter a variety of different types of method so they are continually challenged. If you work your way through all modules I recommend you switch strategies each time to keep the challenge high. **How creative is this?** This combines the best knowledge we have about how to learn something new with the opportunity at the end for pupils to create something that they want to create that uses their new knowledge.

## Unplugged Introduction to Basic Procedures

Download **GameBasicProcedures PowerPoint** Use the slides to introduce basic procedures, there are notes on the slides. The slides go through the basic premise of what procedures are. They start with a definition. Slide three lists common procedures that our brains already know how to do such as walking, breathing etc. Slide four gives an example of how three common procedure algorithms can be used by a simple script. The slide animation reveals a layer at a time. It makes it more memorable if you act it out as you reveal parts of the slide. Slide five looks at common autopilot procedures and asks pupils to order a landing procedure. Pupils could do this in pairs. Slide six has the answers to this activity.


Common everyday procedures that your brain already knows how to do

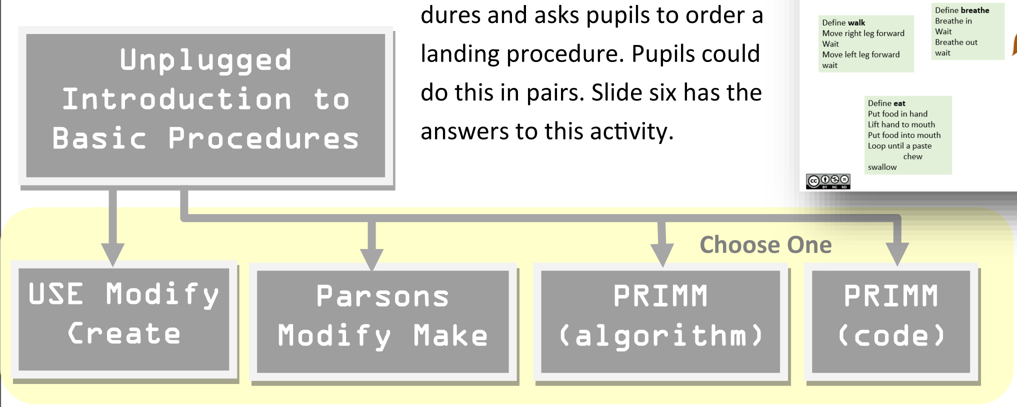
```
Define walk
Move right leg forward
Wait
Move left leg forward
wait
```

```
Define breathe
Breathe in
Wait
Breathe out
wait
```

```
Define eat
Put food in hand
Lift hand to mouth
Put food into mouth
Loop until a paste
        chew
swallow
```

```
Loop always
    breathe
    if need to go somewhere
        walk
    if hungry AND food near
        eat
```

Hampshire Services
Code-IT



**Booklet Choices** You choose from one of the options above. Each option has its own booklet which guides pupils through the stages, making them think deeply about either the code or the algorithm before modifying it and having a choice of things to make. Pupils are instructed when to work in pairs and when to work alone, but you can direct otherwise if you wish. Separate the answer sheet (last pages of the booklet) into sections as pupils will need to mark their work as part of their learning process. If you are not sure which to choose download the booklets and look at the differences. You may wish to keep the last page and give out the hints yourself.

## National Curriculum Programs of Study

(bold text is covered in this module)

**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**

## Before the module

Read the planning and download the PowerPoint. Decide which booklet variation you are going to use and download and print it out one per pupil. Remove the answer sheets for pupils and sort them for pupils to access when they need to. Download the code for your version of Scratch 2 or 3 and place it on your network where pupils can access it, or note where it is on the Scratch website if using Scratch online.

## Formative assessment support

If pupils are struggling to work together in a meaningful way then encouraging and rewarding positive attitudes to working collaboratively using the communicates stickers shown at the end helps.

Lots of misconceptions can be solved by reading the code or algorithm slowly and out loud to their partner or going back to role-play.

## Classroom Organisation

In some sections pupils are asked to work with a partner of similar programming ability. If you are not sure what programming ability they are go with Maths skills as a starting place. Move partners around between modules so that pupils benefit from different interactions. There is some evidence that pupils work better with a colleague that they like.

## Research Help

Get pupils to mark their booklets, collect in all the marks by sections. Collect the marks from the modify section and calculate a mean average for the whole class. Email or Tweet this to phil.bagge@code-it.me or @baggiepr stating clearly what age, module and version your class did. For example

Y5 9-10 Years old Diving Beetle PRIMM Algorithm Mean average 5.2/9 for 32 pupils.

## Resources

**GameBasicProcedures PowerPoint**
(20 mins)

**Parsons Modify Make**
Pupil booklets
Scratch 2 & 3 Code to download
Scratch 3 code on Scratch website

**PRIMM Algorithm & Code**
Pupil booklets
Scratch 2 & 3 Code to download
Scratch 3 code on Scratch website

**USE MODIFY CREATE**
Pupil booklets
Scratch 2 & 3 Code to download
Scratch 3 code on Scratch website

**All Resources at**
http://code-it.co.uk/goldgame/

## Further Research Reading

**Use Modify Create**

Irene Lee et al Computational thinking for Youth in practice (2011)

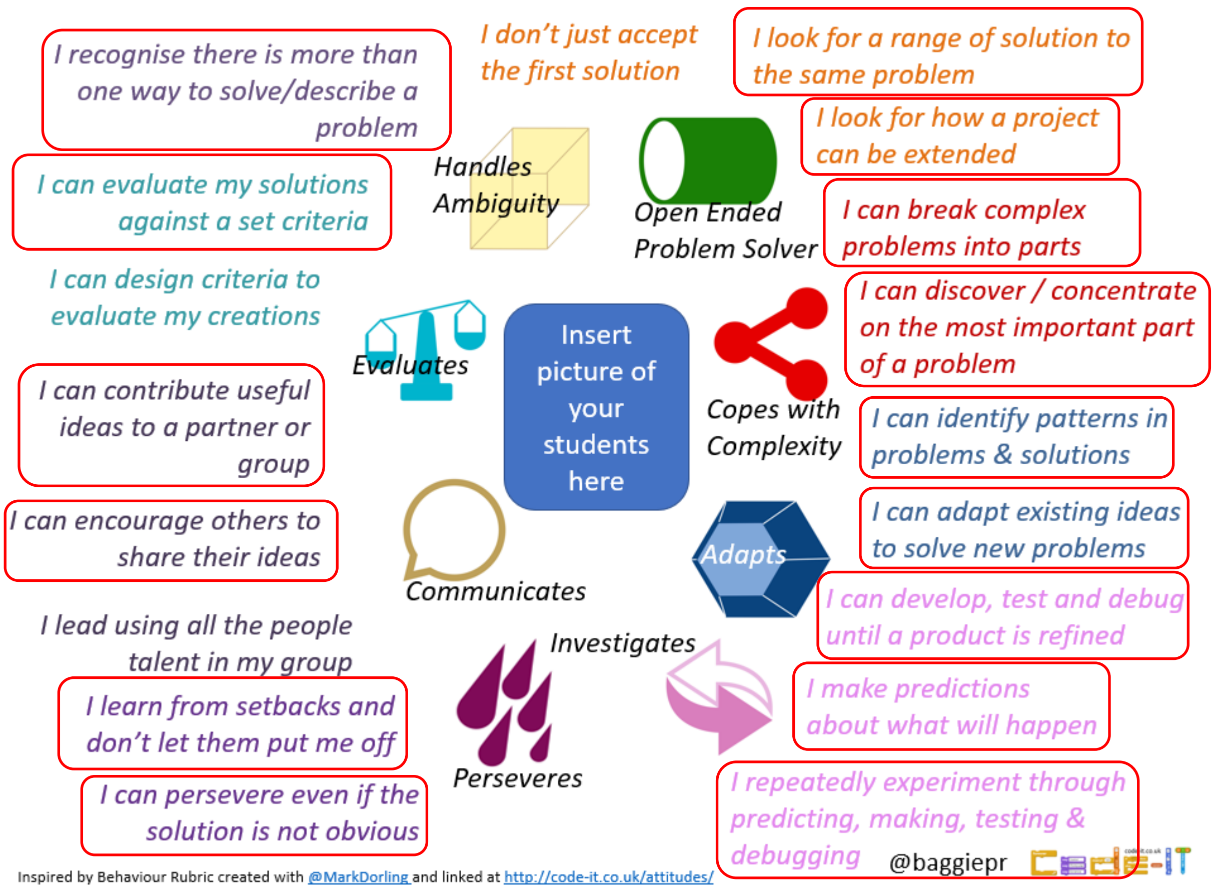
**PRIMM** Sentence

https://blogs.kcl.ac.uk/cser/2017/09/01/primm-a-structured-approach-to-teaching-programming/

**Four levels of abstraction**

This article includes an example of the four levels of abstraction and sign posts the work of Waite and Armoni

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



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

I can break complex problems into parts

I can design criteria to evaluate my creations

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

I can contribute useful ideas to a partner or group

I can identify patterns in problems & solutions

I can encourage others to share their ideas

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

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

I make predictions about what will happen

I can persevere even if the solution is not obvious

I repeatedly experiment through predicting, making, testing & debugging

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