

Tshwane University of Technology

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Principles of Programming A (Extended)

PPAF05D

Unit 1

Introduction to Scratch

DEVELOPED: 2019-2020

© COPYRIGHT: TSHWANE UNIVERSITY OF TECHNOLOGY (2019)






All rights reserved. Apart from any reasonable quotations for the purposes of research criticism or review as permitted under the Copyright Act, no part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy and recording, without permission in writing from the publisher.








| | |
|---|-----------|
| Legend for Icons | 2 |
| Unit 1: Introduction to Scratch..... | 3 |
| Purpose of this unit | 3 |
| What is programming..... | 4 |
| First acquaintance with Scratch | 4 |
| Planning a program | 17 |
| Representing algorithms in pseudocode..... | 18 |
| Reflection..... | 23 |
| Sources consulted..... | 25 |

You OWN your learning – therefore you need to reflect on your learning process. At the end of each lesson, you need to have a discussion with yourself and/or fellow students and/or lecturer to reflect on the lesson of the day. See the paragraph called ‘Reflection’ at the end of the chapter for examples of questions you can include in your discussions.



Legend for Icons

| | |
|-----------------|--|
| Activity |  |
| Checklist |  |
| Discussion |  |
| Example |  |
| Explore |  |
| Fact/ Note/ Tip |  |
| Knowledge |  |
| Video |  |

| | |
|------------|---|
| Link |  |
| Quiz |  |
| Reflect |  |
| Calculate |  |
| Reminder |  |
| Vocabulary |  |
| Skills |  |

Unit 1: Introduction to Scratch

Purpose of this unit

The skills and knowledge you will obtain in engaging with this unit, will prepare you to achieve the learning outcomes stated in the Study Guide. In this unit, the following skills and knowledge will be addressed:

You will be able to:

- Use programming-related terminology in correct manner.
- Explain the characteristics of an algorithm.
- Write a basic program to implement animation.
- Rename a sprite.
- Add a new background.
- Add new sprites.
- Save a file in the correct folder or subfolder.
- Find relevant code blocks based on the colour of a block.
- Make changes to existing Scratch programs.
- Write scripts for more than one sprite.
- Create a Scratch program when an algorithm in pseudocode is provided.
- Copy (duplicate) a stack of blocks to another sprite.
- Write comments inside a script.
- Add a code block into an existing stack of blocks.
- Change the name of a sprite.
- Determine the coordinates of the position of a sprite.
- Break a stack of blocks apart.
- Remove an unwanted block from a stack.

What is programming

Computers cannot perform any tasks on their own. They need instructions to 'tell' them what to do. These instructions are called programs. These programs are written by people (programmers) using different programming languages. The programming language we are going to use first is called Scratch 3.0¹. Once you can write basic programs, you will learn how to write programs using a programming language called Java.

Definitions

Programming is the process of writing computer programs.

A *program* is a series of instructions, written in a language understood by the computer that tells it exactly what to do.

A *programming language* is a set of commands, instructions following certain syntax rules that we use to create a program (application).

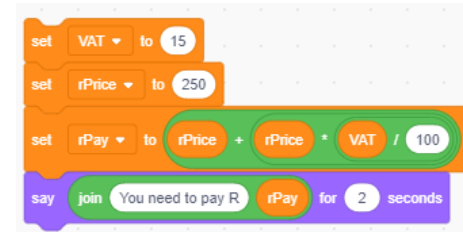
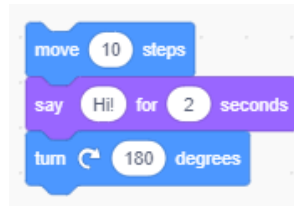


- Series
- Syntax rules
- Software application
- Commands



First acquaintance with Scratch

Scratch is a programming language that provides colourful blocks that can be used to create programs. Programs can be created that can perform calculations, and it can easily include animation and sound. To learn how to use Scratch to write, save and execute programs, let's first watch a video demonstrating the basic skills. Thereafter you can work through the activities we have provided so that you can become familiar with the Scratch environment.



1_GettingStartedWithScratch

Know this before you continue

- The Scratch programming language works like a play. There are actors called sprites, which perform actions on a Stage.



- Playwright
- Costume

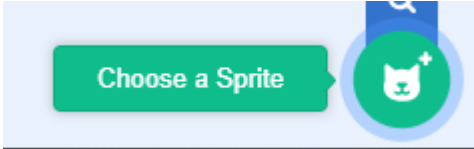
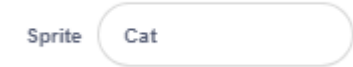

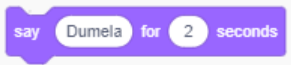
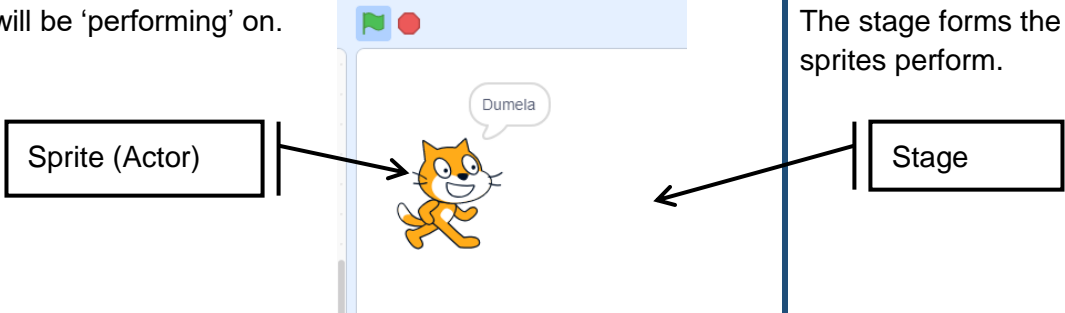


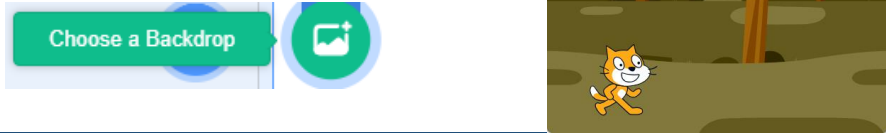
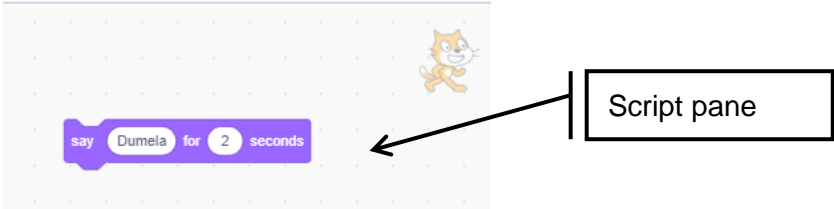
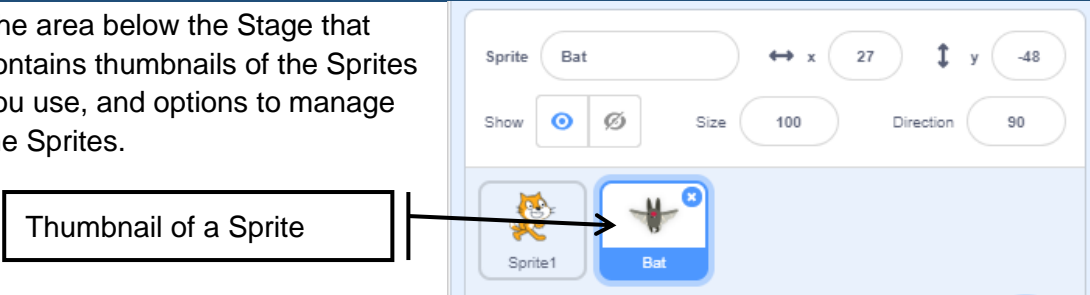
¹ Scratch is a project of the Lifelong Kindergarten Group at the MIT Media Lab. See <http://scratch.mit.edu>

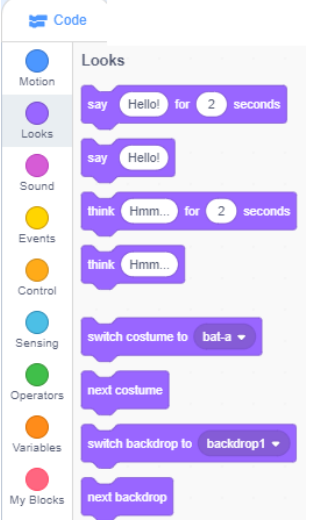
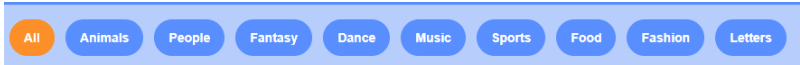
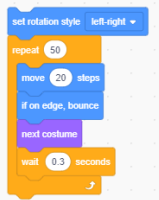
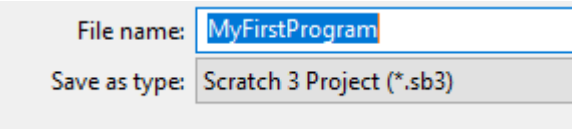
- You, the programmer, are the playwright - you tell the story.
- The playwright decides what the actors will do and say and how they will look and move.
- As the programmer you decide how many sprites you want to use, what they should do and say, what they will look like (costumes they will wear), and how they will move.
- To manipulate the actors on the stage you use code blocks.

Important terminologies/concepts



| Name | What it is | What it does |
|-------------|--|--|
| Sprite | <ul style="list-style-type: none"> • The 'actor' in our story. You can choose a new Sprite using the icon. • You can remove a sprite by clicking on the white cross on the thumbnail of the sprite. • You can rename a sprite by entering a name in the Sprite pane.    | The sprite performs the actions the programmer decides on through the code blocks - it does what the code blocks 'tells' it to do. |
| Code blocks | Puzzle shaped pieces that slot together. These pieces build up the instructions for the computer (code).  | A code block contains an instruction or a command to make a sprite 'do' something. |
| Stage | The area that the sprites will be 'performing' on.  | The stage forms the space in which the sprites perform. |

| Name | What it is | What it does |
|-------------|--|---|
| Backdrop | <p>The background picture against which the story is performed. Backdrops can be changed by using code blocks or clicking on an icon.</p>  | It makes a story more colourful and interesting, for instance if the story takes place in a forest, you can use a background with trees. |
| Script pane | <p>The area where you add the code to control any Sprite.</p>  | Displays the code for the Sprite selected. |
| Sprite pane | <p>The area below the Stage that contains thumbnails of the Sprites you use, and options to manage the Sprites.</p>  | <p>Provides options to change, for example, the size, position and direction of a sprite.</p> <p>If you click on the thumbnail of a Sprite, the code for that Sprite will appear in the Scripts pane.</p> |

| Name | What it is | What it does |
|-------------------|---|--|
| Blocks Palette | <p>All the available code blocks appear here, arranged as part of different classes.</p> <p>Examples of classes are Looks, Motion, Sound.</p>  | Choose your code blocks from the blocks palette. |
| Sprites Library | <p>The list of all the Sprites that are available for you to choose from.</p>  | It allows the programmer to review available Sprites from various categories and select one. |
| Script | <p>One set of code blocks linked together.</p>  | It allows the Sprite to execute a set of instructions. |
| Program / project | <p>All the scripts of all the Sprites stored in one file.</p>  | Contains the code blocks for one or more Sprites to 'tell a story' and/or do calculations. |



Use this image of the Scratch Screen to help you find your way.



The Block Palette: contains code blocks in different classes.

Script pane: drag your code blocks here.

Stage: see what your program does here.

Menu bar

A Sprite: A character or an object that will perform the actions indicated in the code blocks.

Sprites list: Sprites used in your project will appear here.

Sprite Library: choose new Sprites (characters) here.

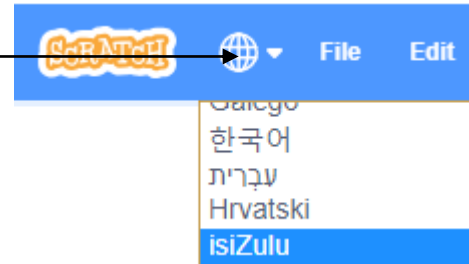
Choose new backdrops here.

- Puzzle shaped.
- Icon
- Slot together
- Thumbnail
- Set of
- Pane
- File
- Category (categories)

Tip

You can change the language to code in from English to isiZulu.

- Click on the globe icon.
- Choose isiZulu.


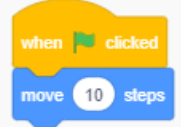

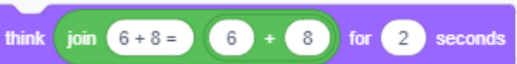


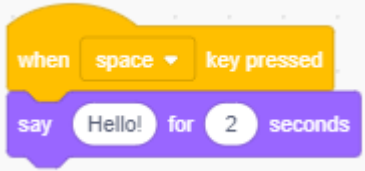
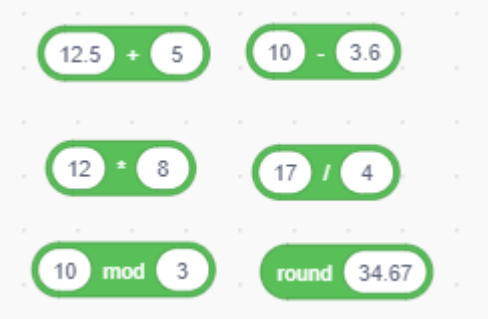
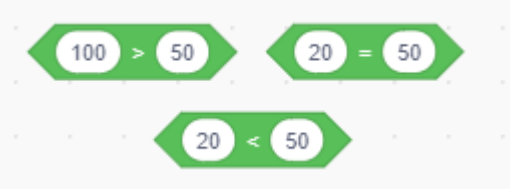
- Globe



Activity 1: Play with code blocks

Open Scratch. Add the following code block(s) to the scripts area. Follow the instructions provided, note the results, and determine what you should have learnt. Look at the colour of each block to determine in which class you will find the block.

| Code block(s) | Instructions | What should you learn |
|---|--|---|
|  | Click on the code block repeatedly, | The sprite executes the instruction in the code block in the script area. |
|  | Click on the green flag above the stage. | Clicking on the green flag is an <i>event</i> (something that happened). When you connect a code block to an event, the code is executed when the event takes place. We say the event is <i>triggered</i> . |
|  | Click on the code block. | Different code blocks can be combined. |
|  | Click on the code block. | Code blocks can be combined. |

| | | |
|---|---|--|
|  | <p>Press the space bar. Change the words 'Hello! ' to a greeting in your language. Press the space bar.</p> | <p>A sprite can react to different events. You can determine the words that must be displayed in the speech bubble of a sprite.</p> |
|  | <p>Click on each block.</p> | <p>Operators are available in Scratch to do mathematical calculations. Number values need to be entered in the white spaces of the blocks.</p> |
|  | <p>Click on each block. Change the values in the white spaces. Notice how the results change.</p> | <p>The code blocks can be used to do logical operations.</p> |

Tip

You can always move individual blocks to the script area to test what they will do. Do not be afraid to ask yourself 'What will happen if '.

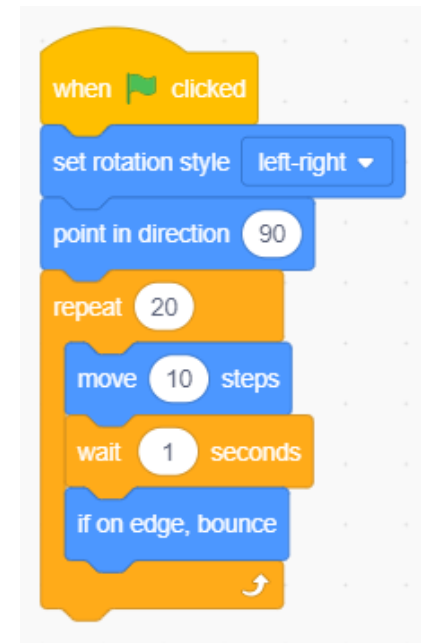


Activity 2: Explore Scratch

- The script for the program demonstrated in the video 1_GettingStartedWithScratch is shown below:
- Working in groups of two, write this program yourselves and save it in a folder with an appropriate name. Find ways to change the instructions to do the following (let each person have a chance to make changes to the program – it should not be the same person who works on the keyboard).

- a) Make the sprite walk faster.
- b) Make the sprite give bigger steps.
- c) Make the sprite walk for a longer time.
- d) Answer the following questions (written exercise):
 - i. What is the function of the block [when green flag clicked]?
 - ii. Why do the colours of the blocks differ?
 - iii. Which decisions were made by program instructions?
 - iv. Which instructions had to be repeated?
 - v. Why does the repeat block enclose the other blocks?
 - vi. How many steps does the sprite take in total?

Save the program



Tips for saving your work

Use the *File, Save to your computer* option to save each program in a file with a unique name before you start working on it. Save the program again once you made a few changes. Scratch gives you the opportunity to change the name of the file each time you save the program. Just click on the file name you used the first time you saved the program, then click on *Yes* if Scratch asks if the file should be replaced.

Decide on your own naming and storing convention and follow it consistently. For example, you can create a single folder for each unit's activities, and then name each program according to the activity number. What is important is that you set up a logical storage system and use names that describe what the program is about, so that it will be easy to find a specific program through its folder and file name.



- Create folders and subfolders in *My Documents*.
- Store a program in a specific folder.
- Move a file to a different folder

Make sure you know the following before you continue!

1. In Scratch program blocks are connected to form a *stack* (one block is tacked on top of another).
2. Each block in the stack is executed in order, starting from the top to the bottom.
3. The program blocks can also be called program statements.
4. The stack can be called an *event handler* because it indicates what should happen when a certain event takes place.
5. The program blocks are grouped together in classes depending on the type of instruction the block contains.
6. Each class has a different colour to help beginner programmers to find the class a certain block belongs to.

Activity 3: Modify an existing program

- a) Working in groups of two (and taking turns at the keyboard), find ways to change the instructions of the program in Activity 2 to do the following:
- i. Change the name of the sprite to *Cat*.
 - ii. Add a [next costume] block from the Looks class in an appropriate position in the script, so that the Cat sprite will move in a way which resembles a person walking.
 - iii. Make the Cat sprite greet you before it starts walking.
 - iv. Make the Cat sprite change colour while it is walking.
 - v. Make the Cat sprite become bigger while it is walking.

Hint: Place the following blocks from the Looks class anywhere in the script area. You can click on any of them to return the sprite to its default state (the way it used to be):

clear graphic effects

This clears all colour changes.

set size to 100 %

Returns the sprite to its normal size.

Save the program.

- b) Add the event handler (stack of program blocks) displayed alongside to the script area.
- i. Which key do you have to press to let the sprite say I am running!!?
 - ii. How can you make the sprite stop saying I am running!!?

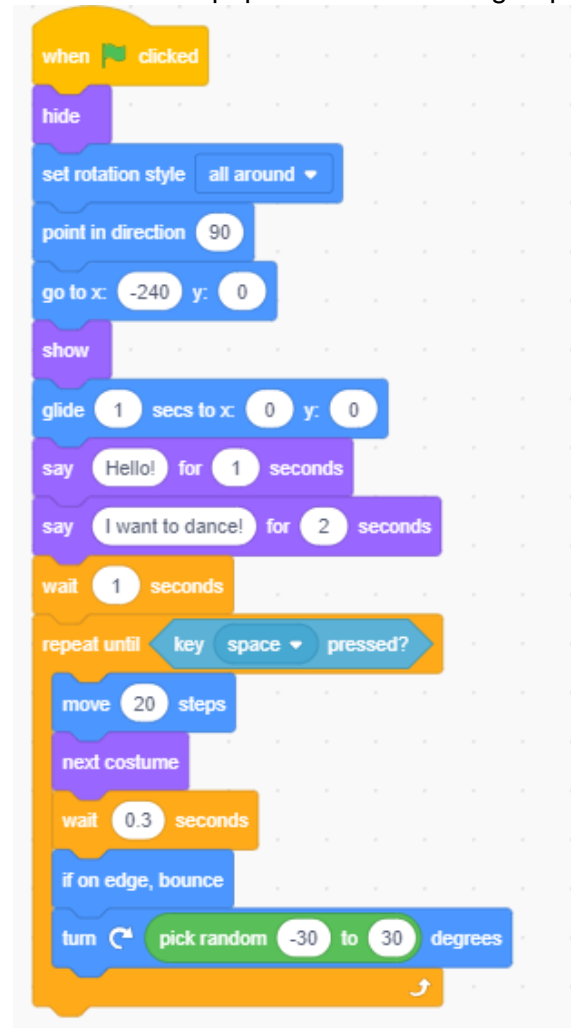


What did you learn?

1. More than one event handler can be written for one sprite.
2. One sprite can act on different events.

Activity 4: Analyse a worked example (Additional activity)

Review the script provided below. In groups of two, discuss what the given program does. Once you agree, write the program to see if you were correct. (Store the program in a folder with an appropriate name. Choose *Champ99* as Sprite, and *Concert* as Backdrop.)

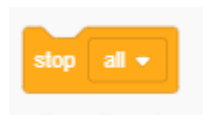


[hide] is in the Looks class

90 degrees will let a sprite look to the right

[pick random] is in the Operators class

- Discuss how the following changes to the script will affect the outcome of the program. Once you agree on the expected results, make the changes and see if you were correct.
 - a) Set the rotation style to [don't rotate].
 - b) Use [go to x:240 y:-180] in step 4.
 - c) Use [pick random -60 to 60] in the last step.
- Add comments to blocks to provide more information. For example, where did you find the <key space pressed> condition? (Right click on a block, left click on Add Comment. Type a comment.)
- Make the following changes:
 - a) Make the sprite dance faster.
 - b) Make the sprite think 'I am dizzy!' when it stops dancing.
 - c) Make the sprite stop moving when you click on it with the mouse pointer. (Tip: Use the [stop all] code block in the Control class.)

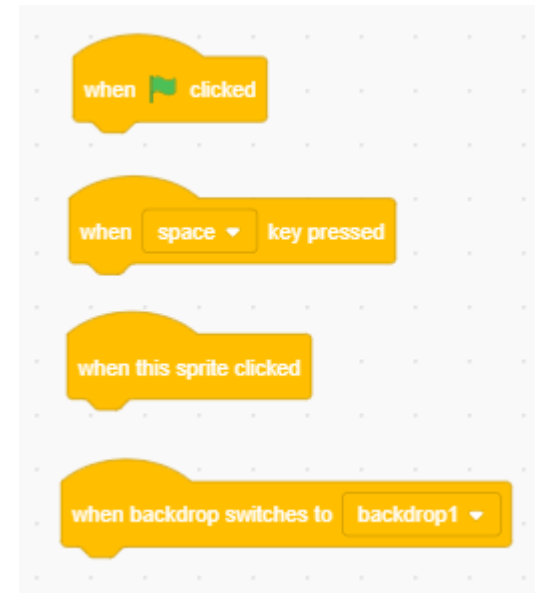


Important terminologies

- *Program blocks* (instructions / code) are used to write a script for a sprite.
- The unit of blocks creating the script is called a *stack*.
- Blocks rounded at the top are called *hats*.
- Blocks which can click into one another are called *stack blocks*.
- A hat is always placed at the top of a stack.
- A hat always waits for an *event* to take place.
- An event can be one of many different actions, for example:
 - Click on the green flag.
 - Press the space bar.
 - Click on a sprite.
 - Press a key on the keyboard.
 - Change the backdrop.
- The same sprite can react to more than one event.
- You can also refer to a stack as an *event handler*.
- The sprites as well as the stage are referred to as *objects*.



- Unit
- Event



Activity 5: Apply terminology

Answer the following questions about the program you wrote in Activity 4.

- Which hat blocks did you use?
- Make a list of all the events that occurred.
- Which decisions were made by the program code?
- Which actions were repeated?

Tip

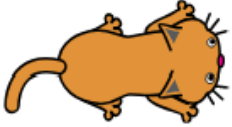

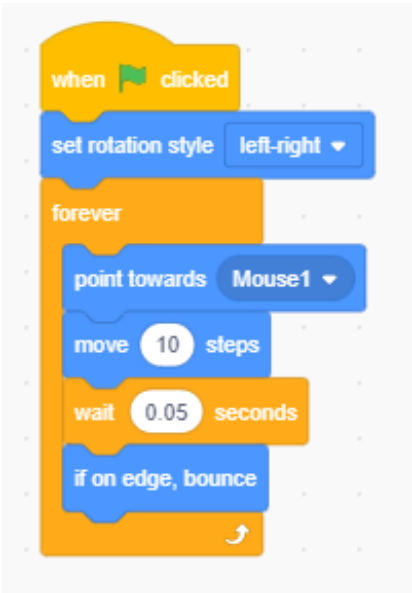

Scratch 3.0 has Tutorials that provide useful tips. Click on the **Tutorials** option in the menu bar. This will provide access to short tutorials on writing



2_WorkingWithMoreSprites.

Activity 6: Analyse a worked example

- This program uses two different sprites. Each sprite has its own script.
- Review the scripts provided below. In groups of two, discuss what the given program does. Once you agree, write the program to see if you were correct. (Store the program with an appropriate name.)

| Cat 2  | Mouse 1  |
|---|---|
|  |  |

- Random



Add comments to the relevant program blocks to indicate:

- In which class did you find the [pick random] code block.
- In which class did you find the [say] code block?

- What does x and y refer to in the script for the Mouse1 sprite?

Activity 7: Re-arrange code blocks

Create a Scratch program with one sprite. Choose any sprite in the *Dance* class. Write three different scripts using only the blocks that are shown alongside. The sprite should dance while a drum is playing, and it should stop when the space bar is pressed.



Use your earphones for this activity!



Activity 8: About me (additional activity)

Combine sprites, costumes, backdrops, looks and sounds to create a Scratch program that can be used by other people to learn more about you.





Planning a program

The programs you have written so far have been quite simple and contained only a few code blocks (instructions). Therefore, you could plan these programs in your head and choose the code blocks you thought were necessary without much effort. However, as your programs become more complex, you will find that you first must plan the solution by writing it down in a step-by-step way. This step-by-step solution is referred to as an *algorithm*. An algorithm is a step-by-step instruction set (a sort of manual or 'recipe' explaining how to perform a task).

We are already familiar with the planning and executing of algorithms. Almost all the tasks we perform every minute of the day are 'algorithms' of one kind or another – when we answer the phone, eat, make coffee or tea, tie shoelaces, etc. Most of these everyday activities follow a sequence of steps that you learned and can now apply without even thinking about it. Some tasks, such as how to bake bread, how to install a specific software program or how to fix a printer, can be a little more complex. However, a well-defined set of instructions can make these complex tasks just as easy to do!

Activity 9: Exploring characteristics of algorithms

In this activity you are going to write down the step-by-step instructions (an algorithm) to perform a simple day-to-day task.

Group activity: Compile an algorithm that can be used to explain to someone how to do a small familiar task in class. For example, how to take the Scratch textbook from a suitcase. Write down the steps in the order they should be executed.

Each group must nominate someone from another group to execute their algorithm. The group will read their solution one step at a time, while the nominated person must do exactly as instructed. The class must be extremely critical and identify the flaws in each algorithm. Write on the board the pros and cons of each algorithm and use these to compile a list of characteristics of a good algorithm.

Some characteristics of a good algorithm are:

- It contains one activity per step.
- The instructions are clear (no ambiguity).
- It must have a clear beginning and end.
- It must be complete – there must be no missing steps.
- The steps must be in the correct order.
- All the steps must be related to the task to be completed – no unnecessary steps or information should be included.

The algorithm must be efficient. Make use of decision and repetition structures to make the algorithm as short as possible.

Representing algorithms in pseudocode

Algorithms can be represented in different ways, for example flowcharts, block diagrams and pseudocode. We will first use pseudocode. Pseudocode is a step-by-step written solution to a program, using a combination of human language and programming language, which can then easily be transcribed into a programming language.

- Transcribed
- Corresponding



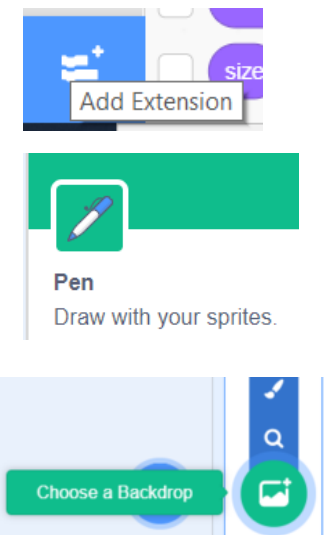
Activity 10: Execute an algorithm

The following table contains an example of pseudocode, with the corresponding Scratch code on the right. Let's try again to execute this algorithm, in much the same way as we did in the group exercise above. This time, work in groups of two. One person reads the algorithm and the other executes it physically by using a pen and paper. What image is drawn?

Once you are done working through the algorithm, create the Scratch program. Note how each step in the algorithm is translated to a code block.

Do the following before you write the program:

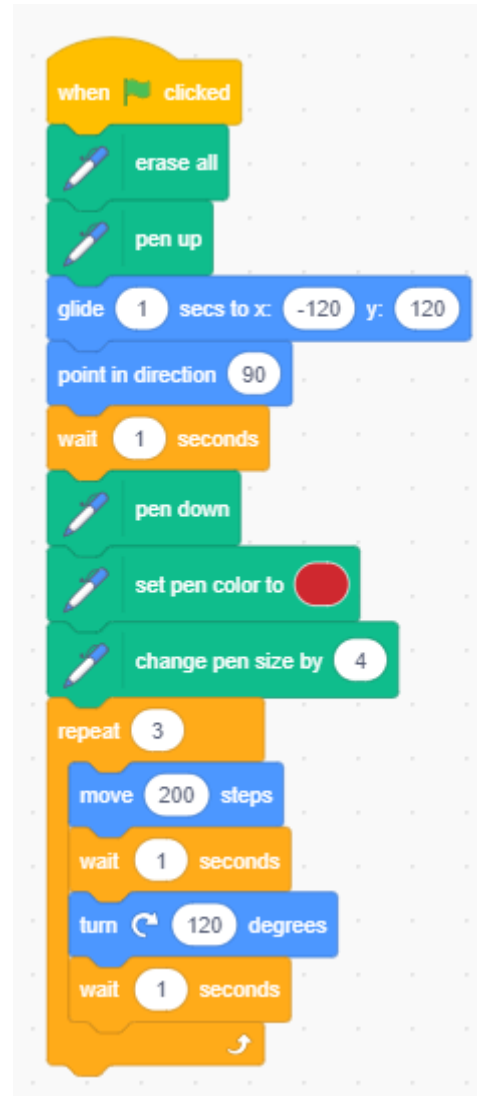
- Delete the cat sprite.
- Add an Arrow1 sprite.
- Make the Pen blocks visible. (Click on the Add Extension icon in the bottom left corner of the screen. Click on Pen.)
- Change the backdrop to Xy-grid. (Click on the Choose a Backdrop icon in the bottom right corner of the screen.)



```

when the user clicks the green flag
    clear the screen
    lift up the pen
    move to position (-120, 120)
    look to the right of the screen
    wait 1 second
    put down the pen
    choose the pen colour
    choose the pen thickness
    repeat 3 times
        move 200 steps
        wait 1 second
        turn clockwise 120 degrees
        wait 1 second
    endRepeat
end

```



We use an *end* statement to indicate the end of the algorithm. In Scratch we do not need to use a specific block to end the script. However, in other programming languages you may also have to use a specific instruction to end a script (part of a program).

You can use pseudocode

- to plan a program;
- as a tool to think through a program solution and test it;
- to communicate your ideas to other people.

Unit 1: Introduction to Scratch



You will learn more about decision-making and

Here are some basic guidelines for compiling pseudocode:

- Use short, clear instructions.
- Clearly indicate the start and the end of the task to be executed.
- Use keywords to indicate decision-making and repetition, for example *If* and *Repeat*.
- Indent instructions that need to be repeated or that indicate decisions made within an *If* statement. (Study the pseudocode above, noting the indentation of instructions within the *Repeat* loop).
- Usually, one pseudocode instruction translates into one programming statement (block), but it can happen that several program blocks are needed to perform a task indicated in one pseudocode instruction.

There is no standard set of rules applicable to pseudocode. However, within a company or a course, there may be a prescribed or preferred way of writing pseudocode. Remember, the main purpose of using pseudocode is that:

- you and others working with you should easily be able to understand the intent of the program,
- that the logic of the code may be tested beforehand and
- that the pseudocode may easily be translated into the programming language used.

Programmers tend to make their pseudocode instructions approximate the programming language they use. Since we program in Scratch in this course, we refer to functions and other statements as they are presented in Scratch. Therefore, the way pseudocode instructions are formulated in this book may differ slightly from those in other sources. You will use pseudocode and flowdiagrams later in this module to write programs in the programming language called Java.

Activity 11: Create a program following an algorithm

Create a new Scratch program. Use the *Ball2* sprite. Create 2 separate scripts using the pseudocode below:

when the user clicks the green flag

```
let the sprite be able to turn in any direction
repeat until the user presses the space bar
    move forward 10 steps
    display the next costume
```

Unit 1: Introduction to Scratch

repetition when you learn
how to code in Java.

- Indent



- Comment:

Programmers write comments in programs to explain what a statement is doing, or to make different parts of a program clearer. In pseudocode and in many programming languages, two slashes (//) are used to



```
        wait 0.09 seconds
        if the sprite touches the edge, bounce away
    end repeat
end
```

when the user clicks on the sprite

```
    go to a random position
    play the sound 'boing' //obtain the block from the Sound class
end
```

Store the program – we will use it again in Unit 2.



indicate a comment. In a comment you can use any language – e.g. English, isiZulu, Northern Sotho – comments do not follow any syntax rules.

Activity 12: Knowledge questions

- What do Scratch programs consist of?
- How do you tell a sprite what to do?
- What is the stage?
- Where is the (0,0) coordinate of the stage? (*Tip: Change the Backdrop to Xy-grid in a scratch program.*)
- In which area do you choose different code blocks?
- What type of files can be used for a sprite, a costume or a backdrop?
- What is the difference between a sprite and a costume?
- What is the difference between a hat block and a stack block?
- Name a few program blocks you can use to communicate with the user of the program.
- Which program block can change the size of a sprite?
- Which program blocks can cause a script to pause or be delayed?
- Which program blocks can cause a sprite to move to another position on the stage?
- Which program blocks can cause a sprite to change direction on the stage?

- n) Name two objects.
- o) Name a few events a sprite can react to.
- p) Who or what causes an event to occur?
- q) Name one decision structure.
- r) Name two repetition structures.
- s) Which repetition structure will cause the program blocks inside it to be executed until the program is stopped by clicking on the red dot?
- t) What is an algorithm?
- u) List 3 ways of presenting an algorithm.
- v) List 3 guidelines for compiling a solution in pseudocode.

Activity 13: Checklist

Ensure you can do the following:



| | I can do it ✓ |
|---|---------------|
| • Use programming-related terminologies in correct manner. | |
| • Explain the characteristics of an algorithm. | |
| Use the programming language Scratch 3.0 to do the following: | |
| • Write a basic program to implement animation. | |
| • Rename a sprite. | |
| • Add a new background. | |
| • Add new sprites. | |
| • Save a file in the correct folder or subfolder. | |
| • Find relevant code blocks based on the colour of a block. | |
| • Make changes to existing scratch programs. | |
| • Write scripts for more than one sprite. | |

| | |
|---|--|
| • Create a scratch program when an algorithm in pseudocode is provided. | |
| • Copy (duplicate) a stack of blocks to another sprite. | |
| • Write comments inside a script. | |
| • Add a code block into an existing stack of blocks. | |
| • Change the name of a Sprite. | |
| • Determine the coordinates of the position of a Sprite. | |
| • Break a stack of blocks apart. | |
| • Remove an unwanted block from a stack. | |

Programming language terminologies referring to Scratch

- Scratch is an *event-driven* programming language.
- In event-driven programming the flow of program execution is determined by *events* – for example a user action such as a mouse click, a key press, or a message from an object. An event-driven application is designed to detect events as they occur, and then deal with them using an appropriate *event-handling procedure* (a script in Scratch).
- Scratch has a visual **integrated development environment** (IDE).
- Scratch produces programs with a **graphical user interface** (GUI).
- Scratch has characteristics of an object-oriented programming language (OOP), because a Scratch program consists of objects (Sprites) that can react to events.



3_AdditionalScratchTips

Reflection



1. What was the purpose of the lesson content you re discussing?
2. What did you learn that will advance your future learning?

3. What new knowledge and skills did you learn today? (Tip: To help you to answer this question – if someone asks you tonight – “So, what did you learn in class today?” – what will you answer them? Your answers can, for example, start with “I am now able to....I managed to ... I learned how to)
4. What did you figure out on your own?
5. What is still challenging or confusing for you to get your mind around?
6. What did you find most challenging today?
7. Are you now confident that you have mastered that difficult concept / skill?
8. If not – what are you going to do to make it work?
9. How was your thinking extended or pushed today?
10. How are the knowledge and skills you learned today connected to what you already knew?
11. Did you discover connections between concepts today that you were not aware of previously?
12. What did you enjoy doing and why?
13. What do you think could be done differently in today’s lesson?
14. What did you discover that may make someone else’s life easier in future classes?

(Sackson, 2011)

(Harvard Graduate School of Education, n.d.)

Sources consulted

- 2018 Scratch Foundation. (2015, June 28). Retrieved from Code-to-Learn Foundation: <http://scratchfoundation.org/about-us>
- Coetzee, C., & Wassermann, U. (2016). Reflections on the application of scaffolding principles to support transfer of mathematical skills to entry-level algorithm design in a novice programming course. *EdMedia*. Vancouver.
- edSurge. (2015, June 28). Retrieved from Learn to Code, Code to Learn: <https://www.edsurge.com/n/2013-05-08-learn-to-code-code-to-learn>
- Erasmus, H. G., & Pretorius, C. M. (2012). *Basic Programming Principles* (2nd ed.). Cape Town, South Africa: Pearson Education South Africa (Pty) Ltd.
- Harvard Graduate School of Education. (n.d.). *Connect Extend Challenge*. Retrieved December 2, 2019, from Visible Thinking: http://www.visiblethinkingpz.org/VisibleThinking_html_files/03_ThinkingRoutines/03d_UnderstandingRoutines/ConnectExtendChallenge/ConnectExtend_Routine.html
- Liang, D. Y. (2013). *Introduction to Java Programming: Comprehensive Version* (9th ed.). Harlow, England: Pearson Education Limited.
- Murach, J. (2017). *Murach's Java programming* (5th ed.). United States of America: Mike Murach & Associates, Inc.
- Sackson, E. (2011, June 11). *10 ways to encourage student reflection*. Retrieved December 2, 2019, from What Ed said: <https://whatedsaid.wordpress.com/2011/06/11/10-ways-to-encourage-student-reflection-2/>
- ScratchEd. (2015, June 28). *What is computational thinking?* Retrieved from ScratchEd: <http://scratched.gse.harvard.edu/ct/defining.html>
- Wassermann, U., Gentle, E., Gibson, K., Noomé, C., Van Zyl, S., & Zeeman, M. (2014). *IT is gr8! @ Grade 11: Delphi*. Pretoria, South Africa: Study Opportunities.
- Wassermann, U., Noomé, C., Gentle, E., Gibson, K., Macmillan, P., & Zeeman, M. (2011). *IT is gr8! @ Grade 10*. Pretoria, South Africa: Study Opportunities.