Coding Robotics

K-12 Resources

binarylogic

Computing and ICT
are the new literacy

Information and Communications Technologies (ICT) are now part of the educational experience of children and teenagers in most parts of the world. Taught as a separate subject, as well as being embedded within the curriculum, Computing and ICT is increasingly regarded as a new literacy, alongside reading, writing and numeracy.

Digital Kids and Digital Teens are designed to introduce students to the key Computing concepts and ICT applications they need to use in order to acquire that literacy and to help them understand the impact of technology on our daily lives. The curriculum provides a framework in which Computing and ICT competences and practical skills can be developed within an environment that is appropriate for the age of the students.

40 years
working with technology in schools

> Serving the learning community

Binary Logic has been working actively with schools, universities and Ministries of Education around the world since 1982 and is well known for the quality of its educational resources and services. The company belongs to the MM Educational Group which was founded in 1974 and since then it has been dedicated to excellence in education. The founders of Binary Logic are educators who decided to incorporate technology early on as they saw the need for innovative ways and methods to enrich students' learning experience. With Belt Study System and ELT SKILLS, we've made English language learning practical, flexible and fun through learning experiences that are interactive and tailored to students' specific needs. In today's everchanging society, we are focusing on the subject of Computing and ICT in schools. Through our innovative curriculum and academic support we have become a worldwide leader.

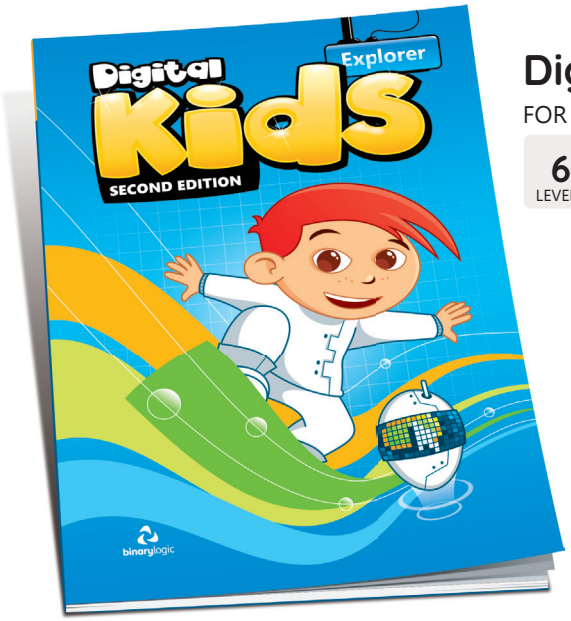
> Our experience in school environments

We design complete solutions for real classroom conditions. The students' needs determine the form of our educational material and with our extensive experience in educational technology we are well positioned to meet the challenges in a wide variety of school environments. There are thousands of schools and universities and millions of students in Europe, the Middle East, Asia and Latin America using educational solutions created by Binary Logic.

mm
educational group
mmedugroup.com

- mm publications
- binary logic
- vector math & science
- mm schools
- combo books
- argus logistics
- prime edu software
- focus on digital services
- abacus fcs

binarylogic
binarylogic.net



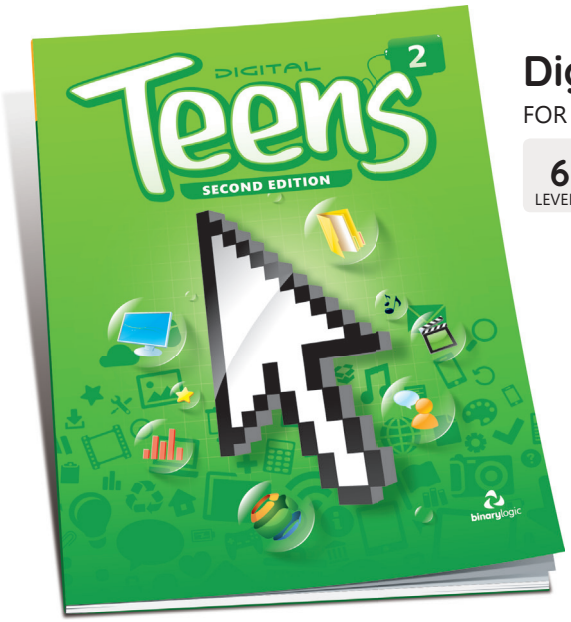
Digital Kids

FOR PRIMARY SCHOOLS

6 LEVELS

ISTE Standards Seal of Alignment Student Standards Jan 2021-23

DQ ADOPTER



Digital Teens

FOR SECONDARY SCHOOLS

6 LEVELS

ISTE Standards Seal of Alignment Student Standards Jan 2021-23

DQ ADOPTER

Digital World

FOR KINDERGARTEN

coming soon

ICT SKILLS

SECOND EDITION

coming soon

eSkills

FOR SCHOOLS

12 LEVELS

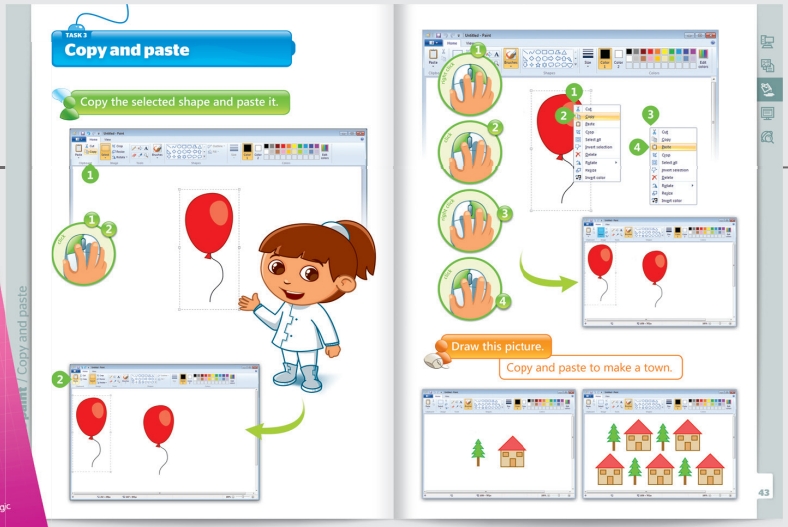
- *Student-centered learning through a fun, hands-on approach*
- *Written and designed by educators*
- *Modern educational material that meets various learning styles*
- *Fully graded and designed for schools*
- *Content aligned to student needs in each age group*
- *Activities based on school subjects in each grade*
- *Language in English edition is graded to facilitate non-native speakers*
- *Available in several languages*
- *Coding and robotics available in different grades*

Local education with global standards

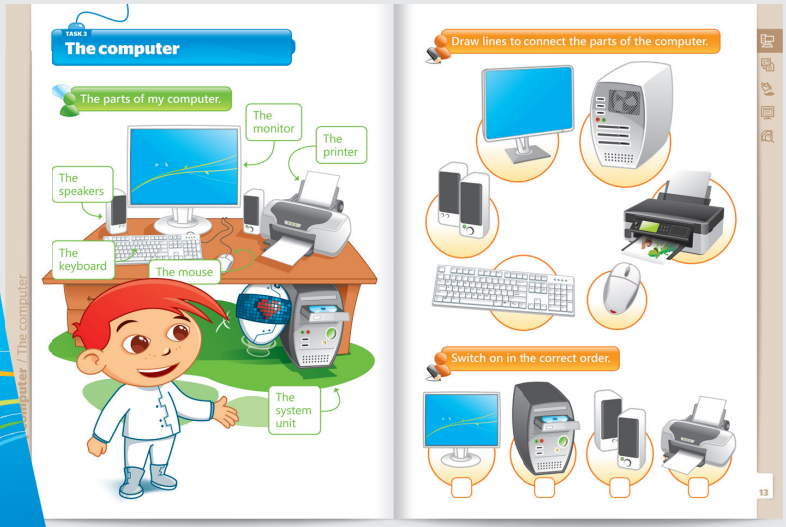
Contact us for custom localized editions

Digital Kids Grades 1-6

for Primary schools



Grade 1



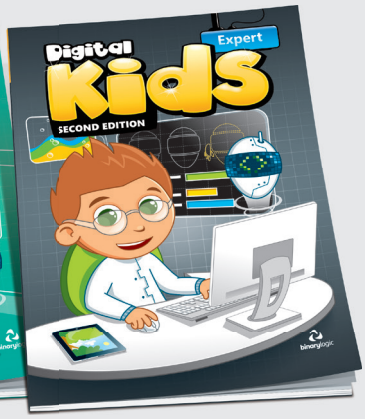
Grade 2

Digital Kids Starter and Explorer are specifically created for very young learners!

Grade 3

Grade 4

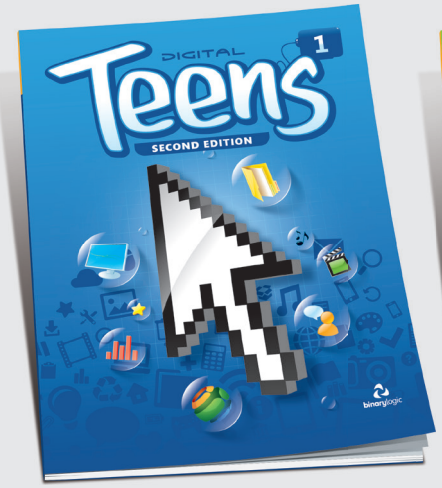
Grade 5



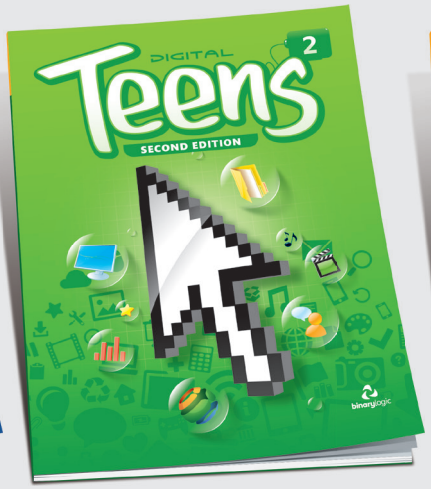
Grade 6

Digital Teens Grades 7-12

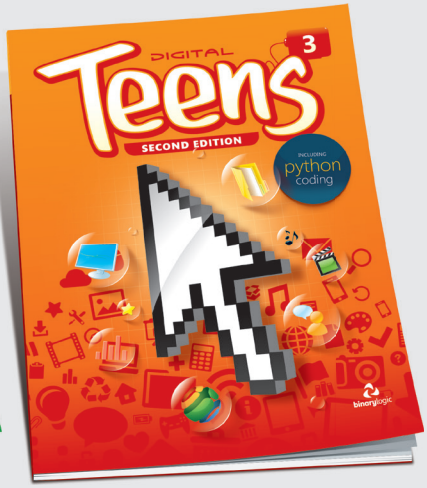
for Secondary schools



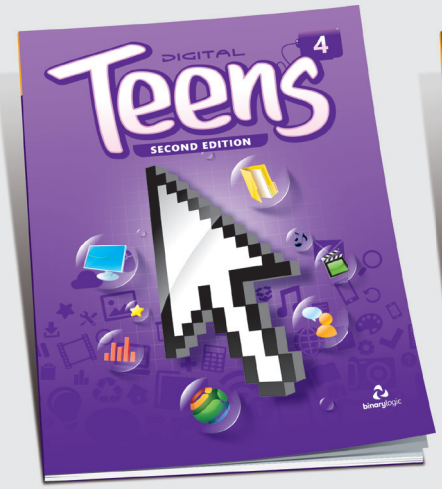
Grade 7



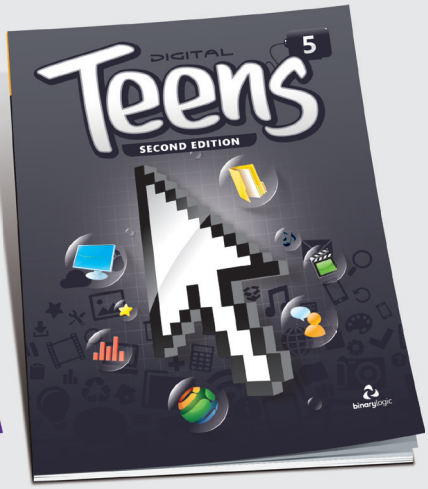
Grade 8



Grade 9



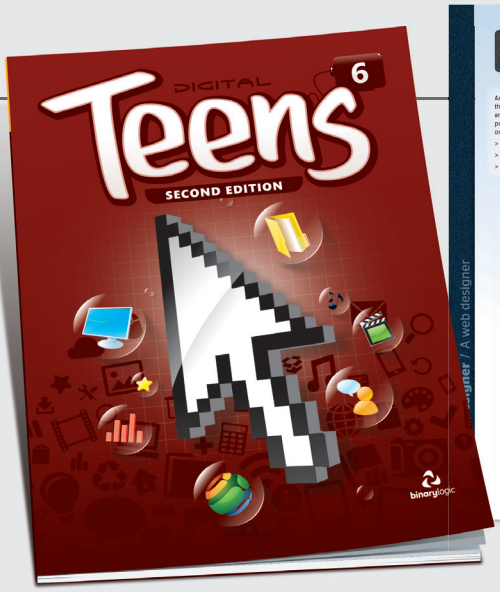
Grade 10



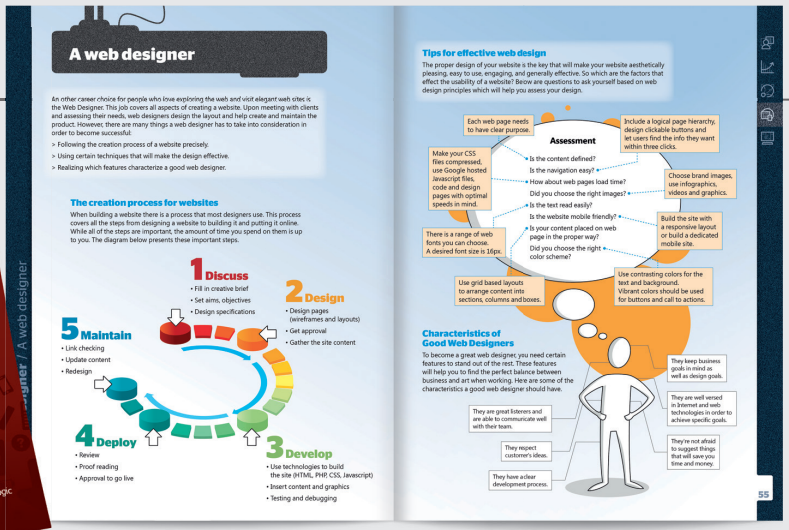
Grade 11

ISTE Standards
Seal of Alignment
Student Standards
Jan 2021-23

Global Standards for Digital Literacy, Skills, and Readiness
DQ ADOPTER
2019



Grade 12



A web designer

An other career choice for people who love exploring the web and visit elegant web sites is the Web Designer. This job covers all aspects of creating a website. Upon meeting with clients and assessing their needs, web designers design the layout and help create and maintain the product. However, there are many things a web designer has to take into consideration in order to become successful:

> Following the creation process of a website precisely.
> Using certain techniques that will make the design effective.
> Realizing which features characterize a good web designer.

The creation process for websites

When building a website there is a process that most designers use. This process covers all the steps from designing a website to building it and putting it online. While all of the steps are important, the amount of time you spend on them is up to you. The diagram below presents these important steps.

1 Discuss
• Fill in creative brief
• Set aims, objectives
• Design specifications

2 Design
• Design pages (wireframes and layouts)
• Get approval
• Gather the site content

3 Develop
• Use technologies to build the site (HTML, PHP, CSS, Javascript)
• Insert content and graphics
• Testing and debugging

4 Deploy
• Review
• Proof reading
• Approval to go live

5 Maintain
• Link checking
• Update content
• Redesign

Tips for effective web design

The proper design of your website is the key that will make your website aesthetically pleasing, easy to use, engaging, and generally effective. So which are the factors that effect the usability of a website? Below are questions to ask yourself based on web design principles which will help you assess your design.

Assessment
Is the content defined?
Is the navigation easy?
How about web pages load time?
Did you choose the right images?
Is the text read easily?
Is the website mobile friendly?
Is your content placed on web page in the proper way?
Did you choose the right color scheme?

Each web page needs to have clear purpose.
Include a logical page hierarchy, design clickable buttons and let users find the info they want within three clicks.
Make your CSS files compressed, use Google hosted Javascript files, code and design pages with optimal speeds in mind.
Choose brand images, use infographics, videos and graphics.
There is a range of web fonts you can choose. A desired font size is 16px.
Build the site with a responsive layout or build a dedicated mobile site.
Use grid based layouts to arrange content into sections, columns and boxes.
Use contrasting colors for the text and background. Vibrant colors should be used for buttons and call to actions.

Characteristics of Good Web Designers

To become a great web designer, you need certain features to stand out of the rest. These features will help you to find the perfect balance between business and art when working. Here are some of the characteristics a good web designer should have.

They keep business goals in mind as well as design goals.
They are well versed in Internet and web technologies in order to achieve specific goals.
They're not afraid to suggest things that will save you time and money.
They have a clear development process.
They respect customer's ideas.
They are great listeners and are able to communicate well with their team.

Digital Teens 6 is entirely project-based and helps students practice the Computing and ICT skills they acquired in previous years.

Computational Thinking

Programming helps students understand and apply the fundamental principles and concepts of computing and computer science, including logic, algorithms and data representation.

Our educational material follows a spiral, project-based approach based on the age and school grade of the students.

Programming is introduced at various stages and in various complexity both in primary and secondary grades with different programming tools and languages. Robotics labs are supported with resources for different educational robot kits and virtual platforms.

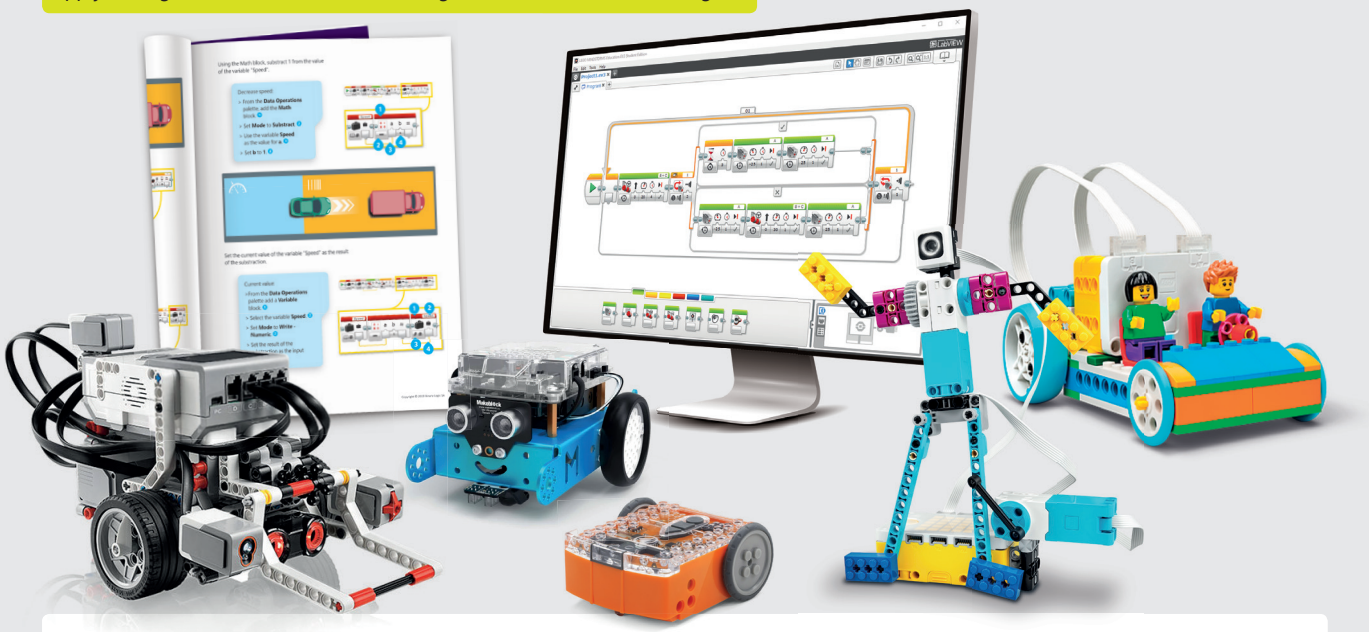
Short lessons that can match the time that is available in the school curriculum.

Extra coding and robotics material for all grades.

Learn how to code in:

Logo, Small Basic, Scratch Jr, Scratch, Microsoft MakeCode, Microbit, Python, Visual Basic, HTML, MIT App Inventor.

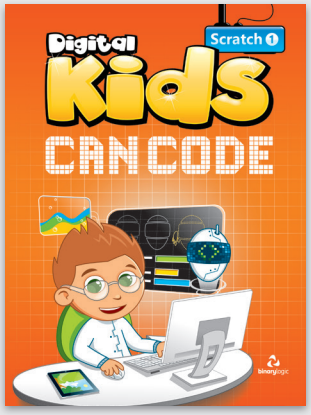
Apply coding skills to robotics for the new generation of kids and teenagers.

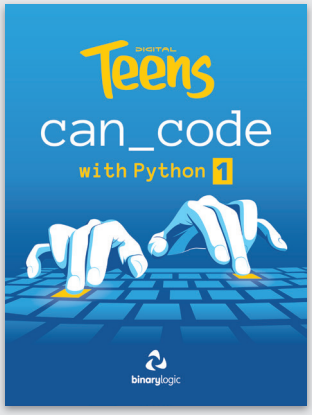


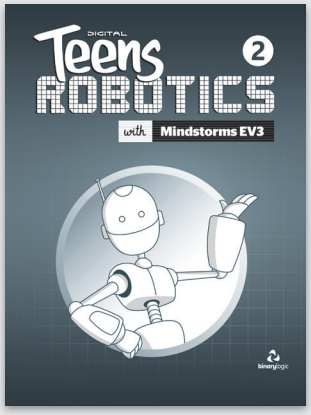
Free virtual robotics platforms support remote and hybrid teaching of robotics.

Programming - Coding - Robotics

Starting in Grade 1 for both topics, very young students are gradually introduced to the concepts of computational thinking with “unplugged” and technology-based activities. The curriculum continues in all grades up to 12 with advanced Computer Science concepts preparing the students for their college or university studies.





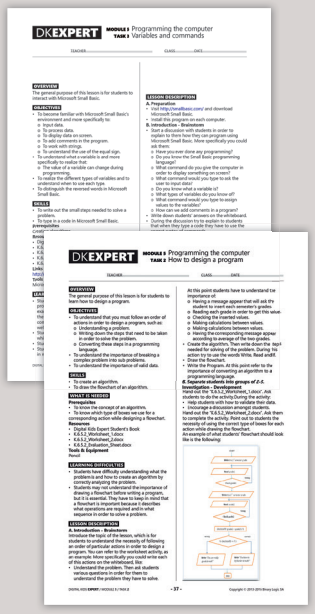


	Grade	1	2	3	4	5	6	7	8	9	10	11	12
Coding / Programming	**Unplugged**												
	Bumblebee Alda												
	Digital Kids Go!												
	LOGO												
	ScratchJr												
	MIT Scratch												
	Microsoft Small BASIC												
	Microsoft Kodu												
	Python 3												
	IoT: MakeCode & Micro:bit												
	IoT: Python & Raspberry Pi												
	MIT App Inventor												
	HTML5 - CSS3 - PHP - JavaScript												
	Visual Basic												
Robotics	**Unplugged**												
	BeeBot												
	LEGO® WeDo 2.0 (WeDo Blocks)												
	LEGO® WeDo 2.0 (Scratch)												
	LEGO® Spike Essential (Icon Blocks)												
	LEGO® Spike Essential (Scratch)												
	LEGO® Spike Prime (Scratch)												
	LEGO® Spike Prime (Python)												
	LEGO® EV3 (Mindstorms Blocks)												
	LEGO® EV3 (Scratch/Makecode)												
	LEGO® EV3 (Python)												
	Edison Robot (EdBlocks)												
	Edison Robot (EdScratch)												
	Edison Robot (EdPython)												
	Makeblock mBot (mBlock Scratch)												
	Makeblock mBot (mBlock Python)												
	Open Roberta Lab (Virtual/Blocks)												
	VEXcode VR (Virtual/Blocks)												
	VEXcode VR (Virtual/Python)												

Printed books · Custom editions · Online eBooks · Coming soon

Teacher support

Teachers get full support to be effective in the computer lab, easily, even if they do not have experience in teaching programming.



Coding | Syllabus G1-6

Grade	Syllabus	Tools
1	> Solve a problem > Follow instructions > Sequence > Find the error > Storytelling	> Digital Kids Go!
2	> ScratchJr programming environment > Drawing > Display a message > Control Blocks	> MIT ScratchJr > LOGO
3	> Flow control > Loop (Repeat) > Simple events (Key Press) > Input/Output	> MIT ScratchJr

Coding | Syllabus G1-6

Grade	Syllabus	Tools
4	> Scratch programming environment > Display information > Sound effects > Use Pen to draw shapes	> MIT Scratch 3
5	> Design a program > Flowchart > Sensing Blocks > Flow control > Conditional operators > Selections/Decisions (IF) > Events (Key Press) > Movement Blocks	> MIT Scratch 3
6	> Sensing Block > Loop (Repeat Until) > Variables > Calculations > Complex decisions (If else) > Conditional operators	> MIT Scratch 3 > Microsoft Small BASIC

Coding | Syllabus G7-12

Grade	Syllabus		Tools
7	> Solve a problem > Flowchart > Sequence (commands) > Coordinates > Display information (Print) > Get information (Input) > Events (Wait Until) > Complex decisions (If...else)	> Operators > Logical operators (AND, OR, NOT) > Variables (naming) > Numbers/Strings > Constants > Calculations > Comments > Use code to control an IoT device	> MIT Scratch 3 > Python 3 (IDLE) > MakeCode & Micro:bit

Grade	Syllabus		Tools
8	> Visual Studio Code programming environment > Conditional operators > Simple decisions (If) > Complex decisions (If...elif, If...elif ...else, nested if)	> Loop (For, range(), while, infinite loop) > Exit loop (Break) > Use code to control an IoT device	> Python 3 (Visual Studio Code) > MakeCode & Micro:bit

Grade	Syllabus		Tools
9	> Modular programming > Functions (parameters, arguments, Return) > Local and global variables > Data structures	(Lists, tuples) > Draw shapes with code (tkinter library) > Events (Key press, mouse click)	> Python 3 (Visual Studio Code) > MIT App Inventor

Coding | Syllabus G7-12

Grade	Syllabus		Tools
10	> Loop (Nested loops) > Drawing/graphing > Data structures (Nested lists) > Functions (len, sum, max, min) > Mobile app development > Mobile user interface design > Create a website with Visual Studio Code > HTML grammar and syntax	> HTML elements > Database management > Classes, objects & inheritance > User interface and testing Python 3 (Visual Studio Code) > Python 3 with Raspberry Pi > MIT App Inventor > HTML5 (Visual Studio Code) > VisualBasic.NET	> MIT Scratch 3 > Python 3 (IDLE) > MakeCode & Micro:bit > App Inv > HTML 5 > visual basic > rasberry

Grade	Syllabus		Tools
11	> Dictionaries > Files (read/write sequential) > Recursion > Global variables > IoT – GPIO programming > Mobile application development with accessibility standards > Design a user interface for people with special needs	> Create a mobile app prototype > Test mobile app for accessibility > Use HTML and CSS tags to format a web page > Responsive web pages with CSS Python 3 (Visual Studio Code) > Python 3 on Raspberry Pi > MIT App Inventor > HTML5 - CSS3 (Visual Studio Code)	> Python 3 (Visual Studio Code) > MIT App Inventor > HTML5, CSS3 (Visual Studio Code)

Grade	Syllabus		Tools
12	> Algorithms > Bubble, Selection & Insertion Sort > Linear & Binary Search > Interactive elements with JavaScript > Server-side Form processing	> Web server & RDBMS Python 3 (Visual Studio Code) > HTML5 - CSS3 - JavaScript (Visual Studio Code) > Node.js	> Python 3 (Visual Studio Code) > HTML5, CSS3, Javascript (Visual Studio Code) > note js

Robotics | Syllabus G1-6

Grade	Syllabus	Tools
1	> Solve a problem > Follow instructions > Sequence > Find the error > Storytelling  	> Bee-Bot Robot
2	> Draw shapes > Assembling simple robot model > Motors/Power > Simple calculations    	> LEGO® WeDo 2.0 > Edison Robot (EdScratch) > Makeblock mBot (with remote control)
3	> Flow control > Loop (Repeat) > Simple events (Key Press) > Input/Output > Sensors/Information Processing (Motion Sensor, Tilt Sensor) > Gears and other mechanical systems    	> LEGO® WeDo 2.0 > Edison Robot (EdScratch) > Makeblock mBot (Scratch)

Robotics | Syllabus G1-6

Grade	Syllabus		Tools
4	> Types of robots > Positive and negative impacts of robotics > Autonomous driving > Scratch programming environment > Loop (Repeat Until, Forever) > EV3 Brick	> Movement Blocks > Selections/Decisions (IF) > Conditional operators (Comparisons) > Test & debug > Open Roberta Lab environment > Gears and other mechanical systems	> LEGO® WeDo 2.0 (Scratch/ Makecode) > LEGO® EV3 (Mindstorms programming environment) > LEGO® EV3 (Scratch/Makecode) > Edison Robot (EdScratch) > Makeblock mBot (Scratch) > Open Roberta Lab (Virtual platform)

Grade	Syllabus	Tools
5	> EV3 Mindstorms programming environment > EV3 Brick settings > EV3 connections > Creating shapes with precision movement	> LEGO® WeDo 2.0 (Scratch/ Makecode) > LEGO® EV3 (Mindstorms programming environment) > LEGO® EV3 (Scratch/Makecode) > Edison Robot (EdScratch) > Makeblock mBot (Scratch) > Open Roberta Lab (Virtual platform)

Grade	Syllabus		Tools
6	> Complex decisions (IF Else) > Sensing Blocks > Control movements of two robots	> Variables > More calculations > Coordinates > Moving autonomously	> LEGO® WeDo 2.0 (Scratch/ Makecode) > LEGO® EV3 (Mindstorms programming environment) > Edison Robot (EdScratch) > Makeblock mBot (Scratch) > Open Roberta Lab (Virtual platform)

Robotics | Syllabus G7-12

Grade	Syllabus		Tools
7	> Follow instructions > Assembling a robot model (Driving Base, Loader) > Data Wires > Variables > Calculations (Comparison symbols, arithmetic operations)	> Strings > Display messages > Complex decisions (IF Else) > Loop (Forever) > Sensors/Information Processing (Ultrasonic Sensor) > Motors (Large/Medium)	> LEGO® EV3 (Mindstorms programming environment) > LEGO® EV3 (Scratch/Makecode) > Edison Robot (EdPython) > Makeblock mBot (Python) > VEXcode VR (Scratch) > Open Roberta Lab (Virtual platform)

Grade	Syllabus		Tools
8	> Conditional operators (Logic operators) > Loop (Until) > Sensors (Touch /Color) > Flow Control	> Acceleration > Deceleration > Cruise control	> LEGO® EV3 (Mindstorms programming environment) > LEGO® EV3 (Scratch/Makecode) > Edison Robot (EdPython) > Makeblock mBot (Python) > VEXcode VR (Scratch)

Grade	Syllabus		Tools
9	> Modular programming > Code reuse > Code organisation > Modules (My Block)		> LEGO® EV3 (Mindstorms programming environment) > LEGO® EV3 (Scratch/Makecode) > Edison Robot (EdPython) > Makeblock mBot (Python)

Robotics | Syllabus G7-12

Grade	Syllabus	Tools
10	> Data Logging > EV3 sensors for data collection > Export EV3 data file to Excel > Import EV3 data file from Excel > Display data diagrams	> LEGO® EV3 (Mindstorms programming environment) > LEGO® EV3 (Scratch/Makecode)

Grade	Syllabus	Tools
11	> Use Python to control a robot > Use Python to draw shapes > Use Python to detects obstacles > Design a robot model with a mechanical arm and lifting system (Prototype) > Visual Studio Code programming environment for LEGO® EV3 > ev3dev Visual Studio Code extension	> LEGO® EV3 (MicroPython)

Grade	Syllabus	Tools
12	> Assembling a robotic arm > Gears and other mechanical systems > 3D Coordinates > Robotic Arm Calibration > Fundamentals of Kinematics > Operating a robotic arm with Python	> LEGO® EV3 (MicroPython)

Coding G1

LESSON 2

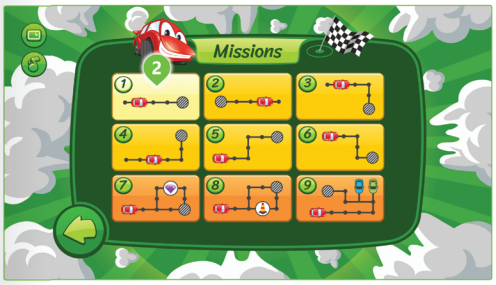
Give commands

This is my computer.

I always start my work here.

Let's meet GO!

Start a mission!



You can find **Digital Kids GO!** on Student's Digital Resources (http://www.binary-academy.com), or you can start the Digital Kids GO! application. To complete a mission, you give instructions to the car to move on a specific path.

Match.

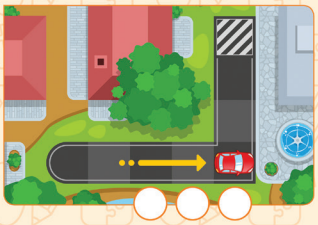
Delete a command.

Go to starting position.

Draw the path.

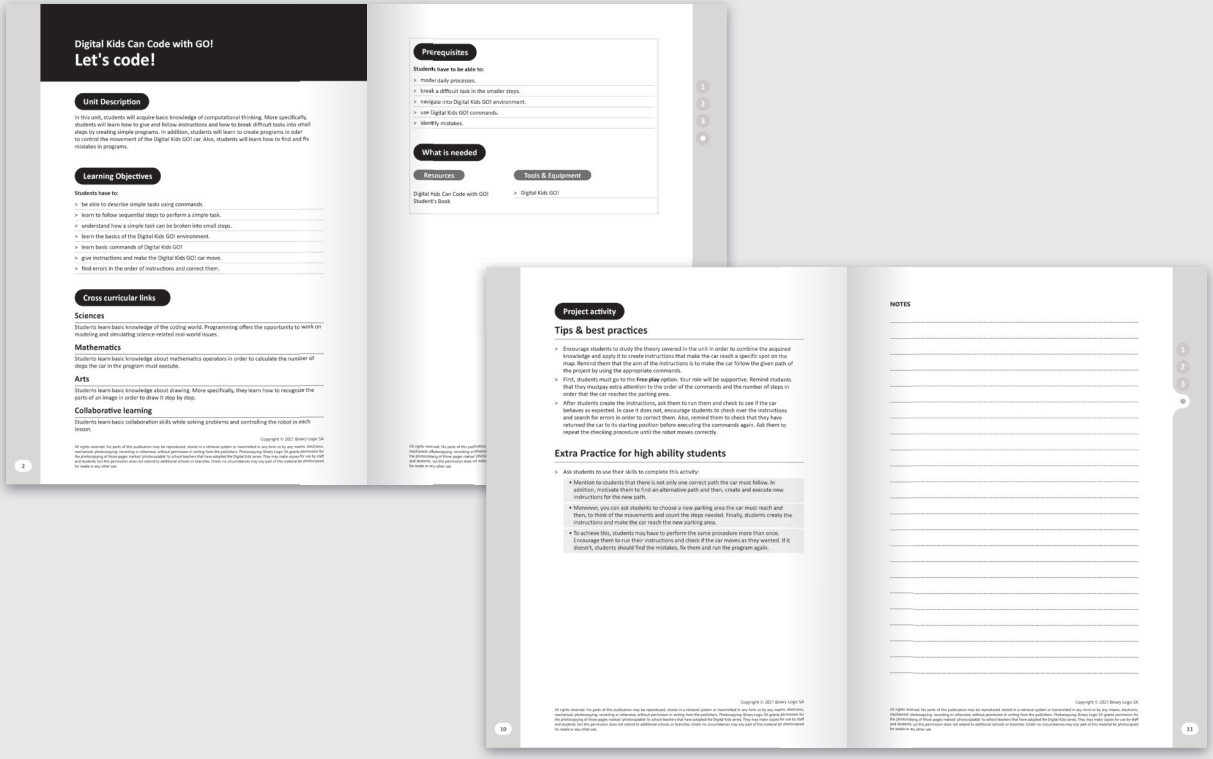
Fiil in the gaps.

Step 1



Step 2

Step 3



Digital Kids Can Code with GO!

Let's code!

Unit Description

In this unit, students will acquire basic knowledge of computational thinking. More specifically, students will learn how to give and follow instructions and how to break difficult tasks into small steps by creating simple programs. In addition, students will learn to create programs in order to control the movement of the Digital Kids GO! car. Also, students will learn how to find and fix mistakes in programs.

Learning Objectives

Students have to:

- be able to describe simple tasks using commands.
- learn to follow sequential steps to perform a simple task.
- understand how a simple task can be broken into small steps.
- learn the basics of the Digital Kids GO! environment.
- learn basic commands of Digital Kids GO!
- give instructions and make the Digital Kids GO! car move.
- find errors in the order of instructions and correct them.

Cross curricular links

Sciences

Students learn basic knowledge of the coding world. Programming offers the opportunity to work on modeling and simulating science-related real-world issues.

Mathematics

Students learn basic knowledge about mathematics operators in order to calculate the number of steps the car in the program must execute.

Arts

Students learn basic knowledge about drawing. More specifically, they learn how to recognize the parts of an image in order to draw it step by step.

Collaborative learning

Students learn basic collaboration skills while solving problems and controlling the robot in each lesson.

2

Prerequisites

Students have to be able to:

- model daily processes.
- break a difficult task in the smaller steps.
- navigate into Digital Kids GO! environment.
- use Digital Kids GO! commands.
- identify mistakes.

What is needed

Resources

Digital Kids Can Code with GO! Student's Book

Tools & Equipment

- Digital Kids GO!

Project activity

Tips & best practices

- Encourage students to study the theory covered in the unit in order to combine the acquired knowledge and apply it to create instructions that make the car reach a specific spot on the map. Remind them that the aim of the instructions is to make the car follow the given path of the project by using the appropriate commands.
- First, students must go to the **Free play** option. Your role will be supportive. Remind students that they must pay extra attention to the order of the commands and the number of steps in order that the car reaches the parking area.
- After students create the instructions, ask them to run them and check to see if the car behaves as expected. In case it does not, encourage students to check over the instructions and search for errors in order to correct them. Also, remind them to check that they have returned the car to its starting position before executing the commands again. Ask them to repeat the checking procedure until the robot moves correctly.

Extra Practice for high ability students

- Ask students to use their skills to complete this activity:
  - Mention to students that there is not only one correct path the car must follow. In addition, motivate them to find an alternative path and then, create and execute new instructions for the new path.
  - Moreover, you can ask students to choose a new parking area the car must reach and then, to think of the movements and count the steps needed. Finally, students create the instructions and make the car reach the new parking area.
  - To achieve this, students may have to perform the same procedure more than once. Encourage them to run their instructions and check if the car moves as they wanted. If it doesn't, students should find the mistakes, fix them and run the program again.

10

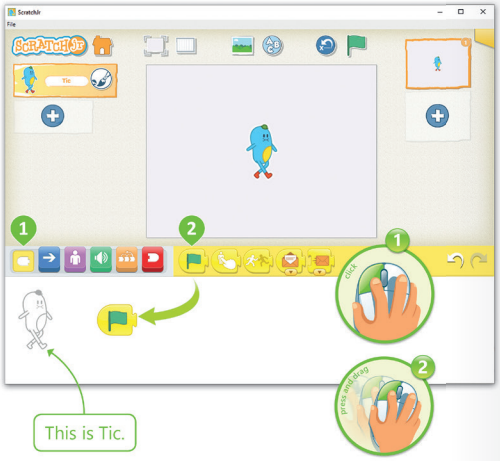
NOTES

11

Coding G2

Create a program.

By clicking the Green Flag our program will start.

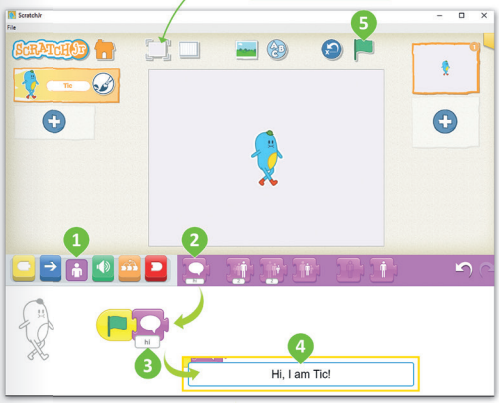


This is Tic.

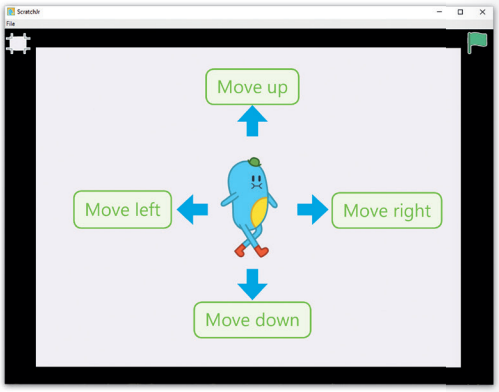
Drag and join your blocks together in the programming area to code Tic. The program always starts with an Event block. When this event happens the program is activated.

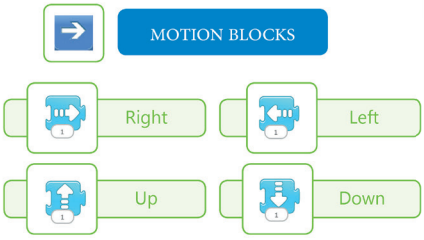
Tic says Hi!

Click here to see Tic in full screen.



Use blocks to move around.





Make the first move.

To delete a block or a group of blocks, drag and drop them out of the programming area.

TASK 3

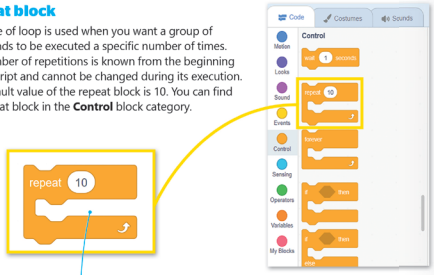
Loops in Scratch

The scripts you have created so far, execute the commands sequentially, one after another. Sometimes you want the computer to execute the same lines of code several times. Loops allow us to re-execute the same commands over and over again. Scratch supports three types of loops: Repeat, Forever and Repeat Until. In this lesson you will use the **repeat** block.

You can find the three loop blocks in the **Control** block category.

Repeat block

This type of loop is used when you want a group of commands to be executed a specific number of times. The number of repetitions is known from the beginning of the script and cannot be changed during its execution. The default value of the repeat block is 10. You can find the repeat block in the **Control** block category.



The blocks you want to be repeated have to be placed inside the repeat block.

To create a script using the repeat block:

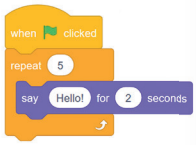
- Add the **when flag clicked** block from the **Events** block category. 1
- Click the **Control** block category. 2
- Drag and drop the **repeat** block into the script area. 3
- Put the **move 10 steps** block from the **Motion** block category inside **repeat** block. 4
- Set **steps** to 5. 5

The following script makes the cat to move 5 steps 10 times.

SMART TIP

The type of loop which is used for a specific number of repetitions, is called a fixed loop.

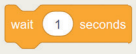
Try the following code. Can you see the cat saying "Hello" 5 times?

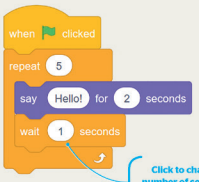


*This happens because there is no pause between the execution of the loops. The block that will help us solve this problem is the **wait** block.*

Wait ( ) seconds block

This block stops the code from running for a specified number of seconds. You can find the wait ( ) secs block in the **Control** block category.





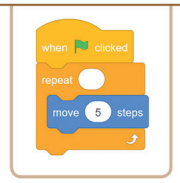
Click to change the number of seconds you want the script to wait.

Place the wait block and try again. Do you see the difference?

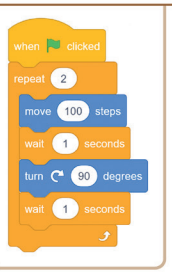
hands on!

Fill in the number on the repeat block to make the sprite reach the edge of the stage.

At the end, add a Say block that will say "I reached the edge!"

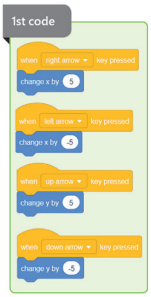


In the next script, we wanted the sprite to form a square as it moved, but something went wrong. Can you find and fix the mistake?

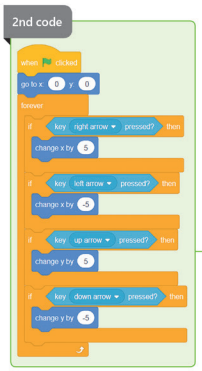


Let's see the difference between these blocks:

1st code



2nd code



The second code is used more frequently for movement, because it moves the sprite faster and gives the illusion of movement. The **key ( ) pressed?** block, executes its script faster and makes the movement smoother.

hands on!

Choose whether the sentence is true or false:

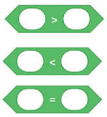
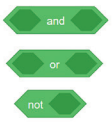
1. *The y value specifies the location of the sprite on the horizontal axis.* **True** ☐ **False** ☐
2. *If the coordinates of the location of the sprite x and y are zero, this sprite is located in the center of the stage.* **True** ☐ **False** ☐
3. *The x value specifies the location of the sprite on the horizontal axis.* **True** ☐ **False** ☐
4. *You can move the sprite to a random location on the stage.* **True** ☐ **False** ☐
5. *You can not visualize data using Scratch.* **True** ☐ **False** ☐

TASK 2

Make complex decisions

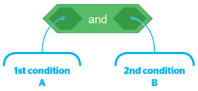
Scratch operators

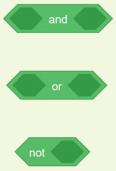
In Scratch there are three categories of **Operators** blocks. The **Conditional Operators**, the **Calculation Operators** and the **Logical Operators**. Let's remember what we have learned about Conditional Operators and Calculation Operators and identify the Logical Operators.

Conditional Operators	You can use Conditional Operators to compare values and act depending on the result. The result of a conditional check can be either "true" or "false".	> 
Calculation Operators	Calculation Operators blocks are used to perform arithmetical calculations such as addition, subtraction, multiplication, division.	+ 
Logical Operators	The Logical Operators blocks allow different actions by changing control flow depending on whether a condition is "true" or "false".	

Logical operators

In this lesson, you will learn how to use the three types of Logical Operators: **( ) and ( ), ( ) or ( ), not ( )**. These operators are used to create complex decisions, by checking conditions.





The **( ) and ( ) block** joins two logical blocks. If even a single condition is false, the block returns false.

The **( ) or ( )** block joins two logical blocks, if even a single condition is true this block returns true.

The **not ( )** block checks the condition inside it. If it is false it returns true. If it is true it returns false.

The following table shows the results of applying the logical operators to a series of logical true-false scalar pairs. This table is called a "**Truth table**" and displays the output of a **Logical Operator** for various inputs.

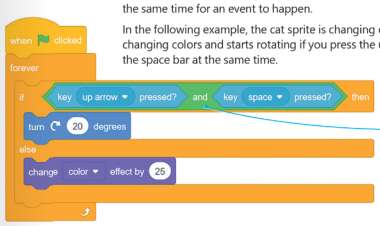
Truth table

A	B	A and B	A or B	not A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Logical operator: ( ) and ( )

There are some cases that you need two conditions to be true at the same time for an event to happen.

In the following example, the cat sprite is changing colors. It stops changing colors and starts rotating if you press the up arrow and the space bar at the same time.



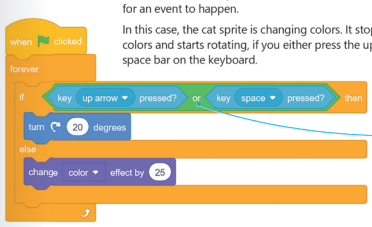
The cat turns only if both keys are pressed.

Remember the table:
Both conditions (A & B) must be true to run the code inside the first space.
Else, if one is false the code in the second space code will run.

Logical operator: ( ) or ( )

In some other cases, you need one or more conditions to be true for an event to happen.

In this case, the cat sprite is changing colors. It stops changing colors and starts rotating, if you either press the up arrow or the space bar on the keyboard.



The cat turns if either one of the buttons is pressed.

Remember the table:
One condition (A/B) must be true to run the code inside the first space.
Else, if both are false the code in the second space will run.

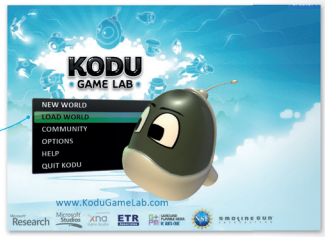
TASK 4

Create your computer game

Now you have an idea about how a program works. Are you ready to go further? Do you want to create your own computer game? There are a lot of game development applications. Some are easy to use and some are very complex and need experienced programmers and game designers. Here you will get an idea of what you can do with **Kodu Game Lab**.

Kodu is a visual programming language made specifically for creating games. It is designed to express advanced game design concepts in a simple, direct, and intuitive manner. With **Kodu**, you have to analyze what you want to do and create a solution. It is free and you can download it from **fuse.microsoft.com/kodu**. There you can also find the **Classroom Kit** with a lot of information and examples.

Start Kodu and choose Load World to view existing games or start a new game.

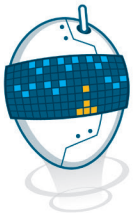


First explore the existing games. Some are full games and others are just worlds without a specific game play. Some are tutorials to help you learn how to create games and some are titled "Technique" to show you specific procedures. Play with some games to get the feeling and then use the tutorials to start learning.

You have terrain tools to create your game's world.

The core of Kodu is the programming user interface. The language is simple and entirely icon-based. Programs are composed of pages, which are broken down into rules, which are further divided into conditions and actions. The Kodu language is designed specifically for game development. Programs are expressed in physical terms, using concepts like vision, hearing and time to control your character's behavior.

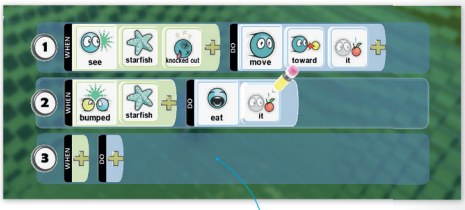
To view the code, press Esc on the keyboard to enter edit mode. Then choose the kodu tool from the toolbar at the bottom, move to an object in the world, and right-click it. To practice coding, start with Tutorial 01 and select the Kodu character that wants to get to the castle.



The games that you create with Kodu can be played with either your mouse and keyboard or an Xbox controller connected to your PC.

SMART TIP

From the Load World menu, select your game. A menu will pop out to the right with a choice of Play, Export or Delete. Choose Export. The game will be saved in the folder My Documents/SavedGames/Boku/Player1/Exports. It's a small file and can easily be emailed. You can also share it with others at www.planetkodu.com



In Kodu, you can program by selecting visual tiles for a condition (WHEN) and an action (DO).

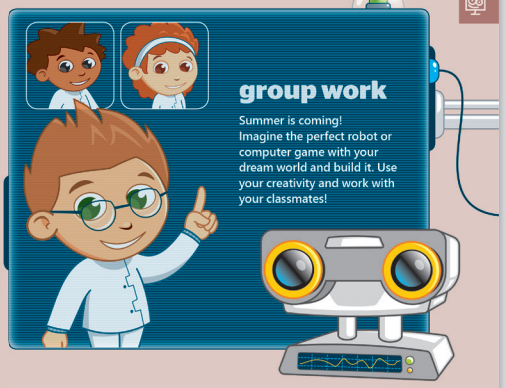
hands on!

Work with the first 5 tutorials (First Tutorial, Programming Kodu to Find Apples, Add/Paint Terrain, Score Tutorial, Glass Walls Tutorial) and get ready for the final group work assignment!

wrap up

Now you have learned how to:

> create shapes and graphics with code.
> use a datalogger for science experiments.
> start a robotics project.
> create your own computer game.



group work

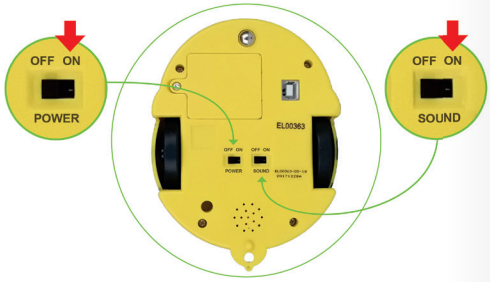
Summer is coming!
Imagine the perfect robot or computer game with your dream world and build it. Use your creativity and work with your classmates!

GLOSSARY

action	datalogger	icon-based language	touch sensor
background color	datalogging	motor	turtle graphics
computer game	drawing canvas	pen color	ultrasonic sensor
condition	game development	robotics	x, y coordinates
		sensor	

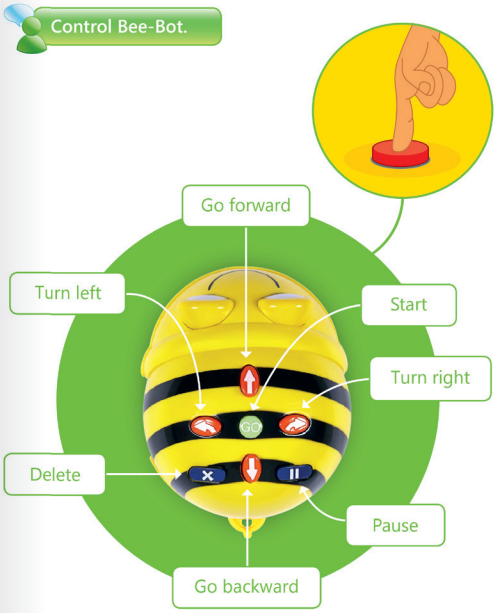
Robotics G1

Let's meet Bee-Bot.



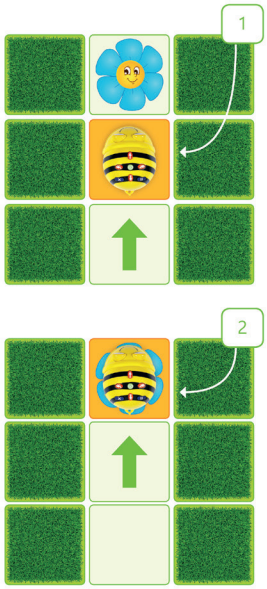
The Bee-Bot is a bee robot that moves! To use the Bee-Bot, first turn the POWER ON and then turn the SOUND ON.

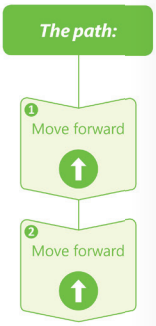
Control Bee-Bot.



To make the Bee-Bot move, you must press the buttons. The "Pause" button will stop the Bee-Bot for 1 second and the "Delete" button will delete all the previous instructions form the memory of the robot.

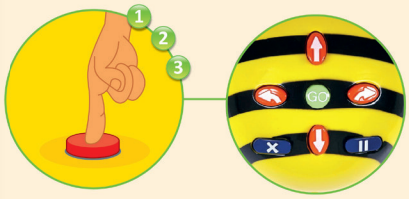
Follow the steps.





One step for the Bee-Bot, is one square on the map. Find out how many steps the Bee-Bot must make to reach the flower.

Move Bee-Bot.

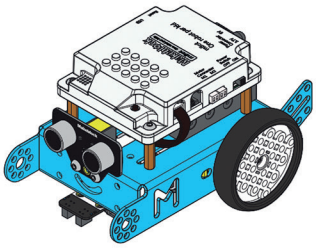


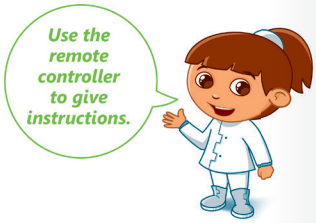
Put the Bee-Bot on the map and use the buttons to make the Bee-Bot reach the flower!

LESSON 1

Movement

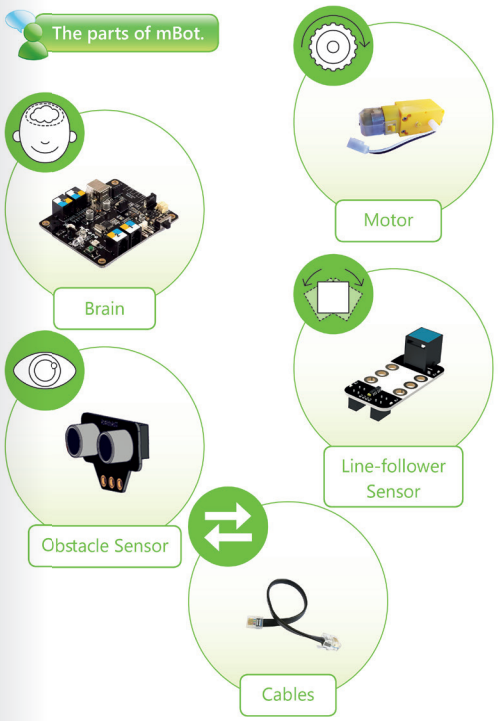
Let's meet mBot.



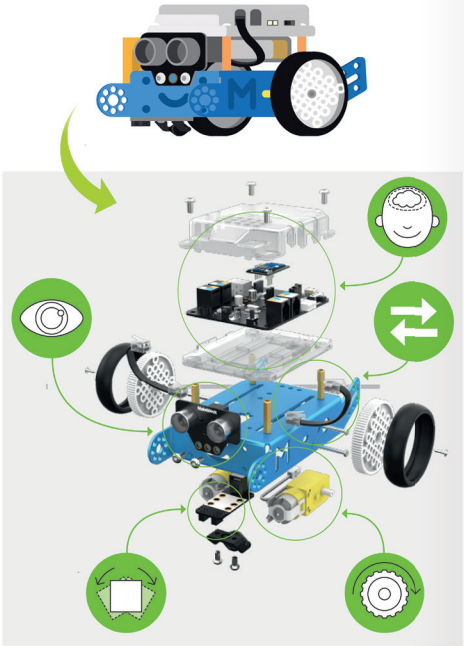


MBot is a kit that can teach students about a variety of robotic machinery and electronic parts. MBot will aslo help students to develop their logical thinking and design skills.

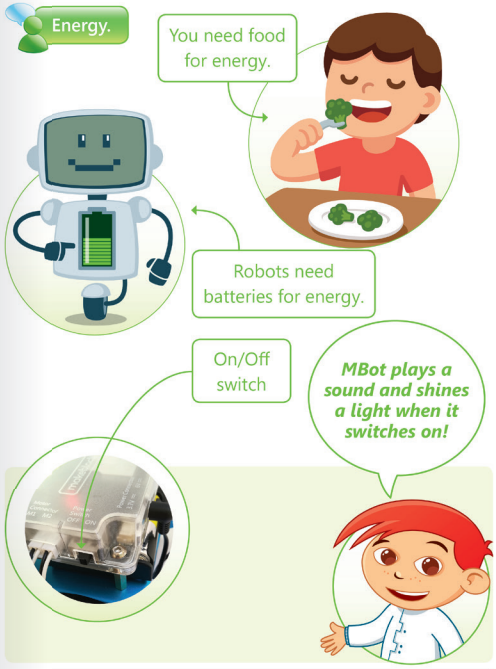
The parts of mBot.



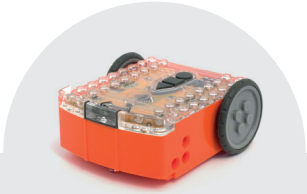
What are the parts...



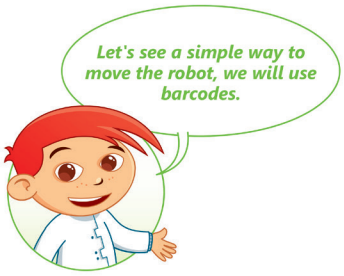
Energy.



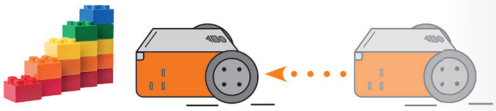
In order to have energy, you need to eat your daily meals. Robots also need energy and they can draw it from batteries.



Avoid obstacles.



An obstacle

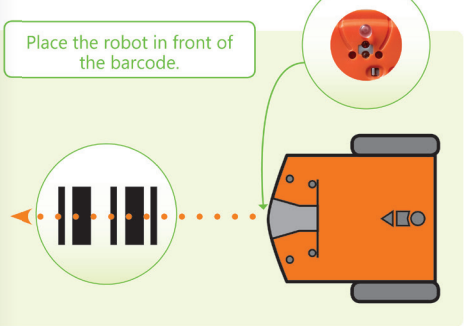


In this activity we want to make the robot move toward an obstacle and when it gets close to it, stop moving. When you do this activity, you have to be careful because Edison can only see obstacles that are the same height or taller than it.

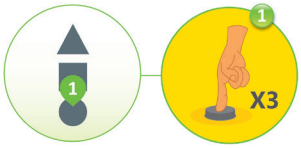
Program Edison.

The sensor for reading.

Place the robot in front of the barcode.



Press the record button 3 times.



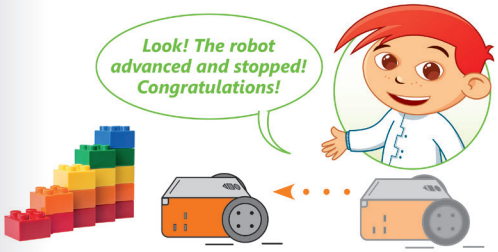
The robot advances and reads the code using the sensor at the bottom, which it uses to see the difference between light and dark surfaces.

Try the activity.

Read the barcode.

Put an obstacle in front of him.

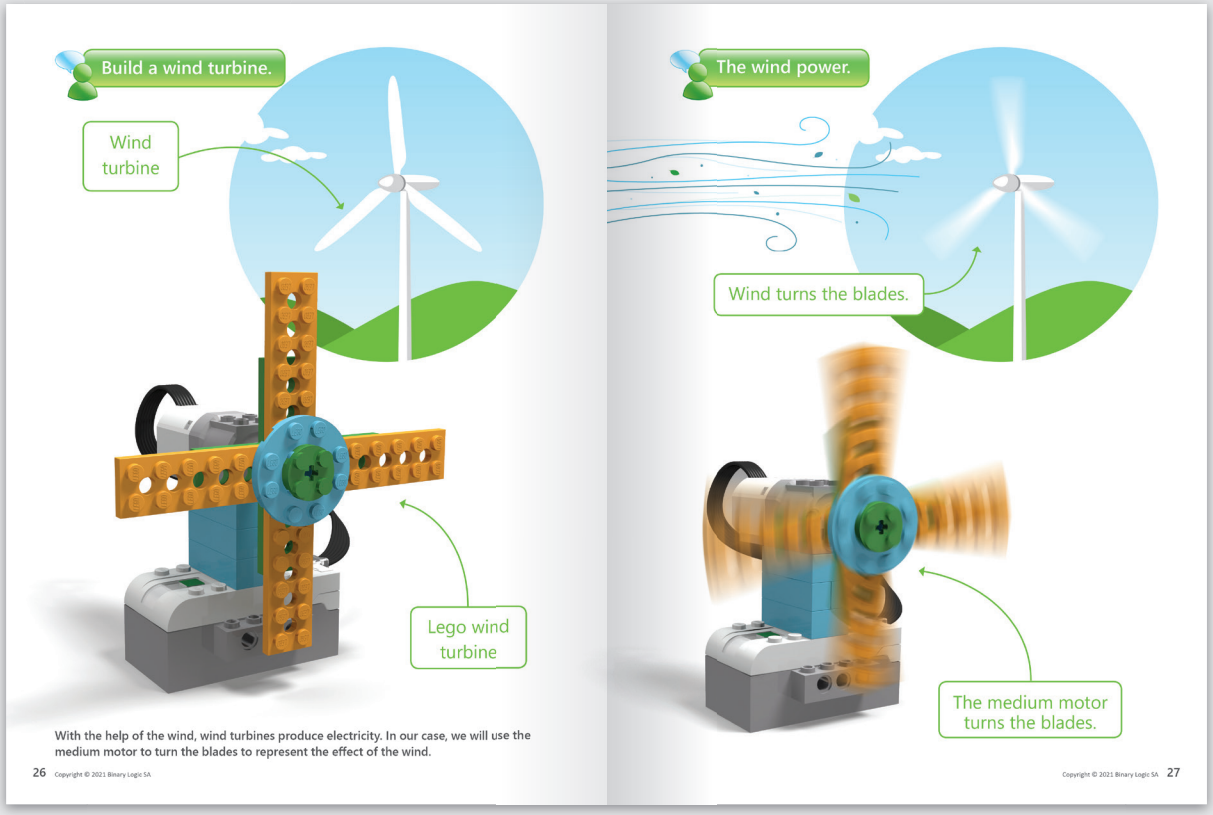
There are several barcodes and each one is for one activity. In this activity, we use the code that makes the robot stop when it sees an obstacle.



Use different obstacles.

An interesting activity would be to form a circle with your classmates using your hands to trap Edison. Try it and see how the robot moves.

G2 Robotics



Build a wind turbine.

Wind turbine

Lego wind turbine

With the help of the wind, wind turbines produce electricity. In our case, we will use the medium motor to turn the blades to represent the effect of the wind.

26 Copyright © 2021 Binary Logic SA

The wind power.

Wind turns the blades.

The medium motor turns the blades.

Copyright © 2021 Binary Logic SA 27

Motor That Way Block.

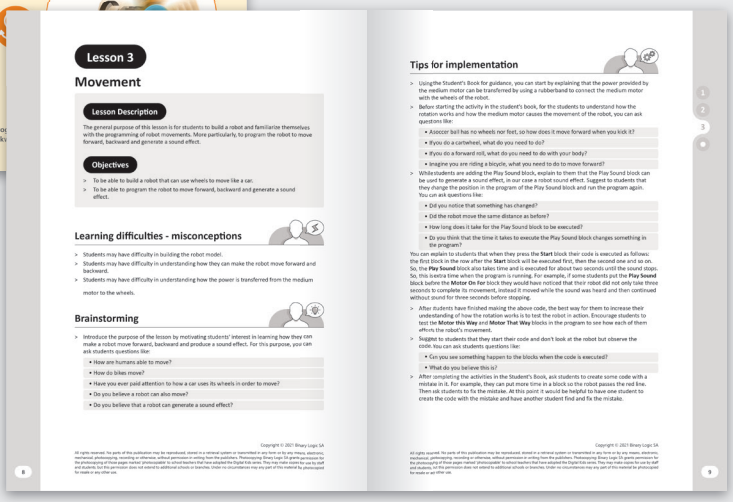
Drag the Motor That Way Block and drop it after the Motor Block. We use the Motor That Way Block to turn the wind turbine clockwise. Click the Start block to start the program and when you want to stop it, click the Stop program button.

34 Copyright © 2021 Binary Logic SA

Change your program.

The wind changed!

Change the previous pro
and second, turn it clock



Lesson 3

Movement

Lesson Description

The general purpose of this lesson is for students to build a robot and familiarize themselves with the programming of robot movements. More particularly, to program the robot to move forward, backward and generate a sound effect.

Objectives

- To be able to build a robot that can use wheels to move like a car.
- To be able to program the robot to move forward, backward and generate a sound effect.

Learning difficulties - misconceptions

- Students may have difficulty in building the robot model.
- Students may have difficulty in understanding how they can make the robot move forward and backward.
- Students may have difficulty in understanding how the power is transferred from the medium motor to the wheels.

Brainstorming

- Introduce the purpose of the lesson by motivating students' interest in learning how they can make a robot move forward, backward and produce a sound effect. For this purpose, you can ask students questions like:
  - How are humans able to move?
  - How do bikes move?
  - Have you ever paid attention to how a car uses its wheels in order to move?
  - Do you believe a robot can also move?
  - Do you believe that a robot can generate a sound effect?

Copyright © 2021 Binary Logic SA

All rights reserved. No parts of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without permission in writing from the publishers. Photocopying Binary Logic SA grants permission for the purchasing institution, to make photocopies for use within that institution. Photocopying Binary Logic SA grants permission for the purchasing teachers and students, but the permission does not extend to additional schools or branches. Under no circumstances may any part of this material be photocopied for resale or any other use.

8

Tips for implementation

- Using the Student's Book for guidance, you can start by explaining that the power provided by the medium motor can be transferred by using a rubberband to connect the medium motor with the wheels of the robot.
- Before starting the activity in the student's book, for the students to understand how the rotation works and how the medium motor causes the movement of the robot, you can ask questions like:
  - A soccer ball has no wheels nor feet, so how does it move forward when you kick it?
  - If you do a cartwheel, what do you need to do?
  - If you do a forward roll, what do you need to do with your body?
  - Imagine you are riding a bicycle, what you need to do to move forward?
- While students are adding the Play Sound block, explain to them that the Play Sound block can be used to generate a sound effect, in our case a robot sound effect. Suggest to students that they change the position in the program of the Play Sound block and run the program again. You can ask questions like:
  - Did you notice that something has changed?
  - Did the robot move the same distance as before?
  - How long does it take for the Play Sound block to be executed?
  - Do you think that the time it takes to execute the Play Sound block changes something in the program?

You can explain to students that when they press the **Start** block their code is executed as follows: the first block in the row after the **Start** block will be executed first, then the second one and so on. So, the **Play Sound** block also takes time and is executed for about two seconds until the sound stops. So, this leads to time when the program is running. For example, if some students put the **Play Sound** block before the **Motor On For** block they would have noticed that their robot did not only take three seconds to complete its movement, instead it moved while the sound was heard and then continued without sound for three seconds before stopping.

- After students have finished making the above code, the best way for them to increase their understanding of how the rotation works is to test the robot in action. Encourage students to test the **Motor this Way** and **Motor That Way** blocks in the program to see how each of them affects the robot's movement.
- Suggest to students that they start their code and don't look at the robot but observe the code. You can ask students questions like:
  - Can you see something happen to the blocks when the code is executed?
  - What do you believe this is?
- After completing the activities in the Student's Book, ask students to create some code with a mistake in it. For example, they can put more time in a block so the robot passes the red line. Then ask students to fix the mistake. At this point it would be helpful to have one student to create the code with the mistake and have another student find and fix the mistake.

Copyright © 2021 Binary Logic SA

All rights reserved. No parts of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without permission in writing from the publishers. Photocopying Binary Logic SA grants permission for the purchasing institution, to make photocopies for use within that institution. Photocopying Binary Logic SA grants permission for the purchasing teachers and students, but the permission does not extend to additional schools or branches. Under no circumstances may any part of this material be photocopied for resale or any other use.

9

Robotics G4

LESSON 2

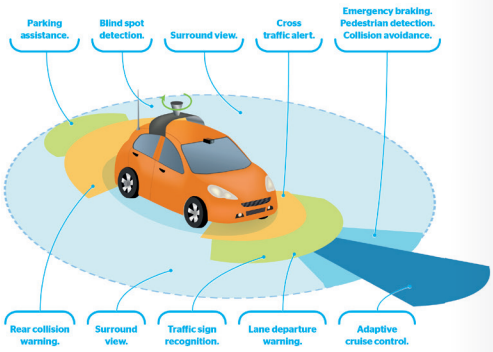
Conditions

Can you imagine a car driving by itself instead of being driven by a human? Wouldn't it be great if it could interact with the environment and navigate without human intervention? If it could avoid collisions, park, change direction based on road surface markings and stop or start based on traffic lights all by itself, wouldn't that be fantastic?

Autonomous driving

Nowadays, the car industry is moving toward this goal, working on creating fully self-driving cars. Adaptive cruise control, lane departure, and parking assist are a few of the increasing number of driver assist features that are becoming standard on new cars.

Autonomous car



Autonomous mBot

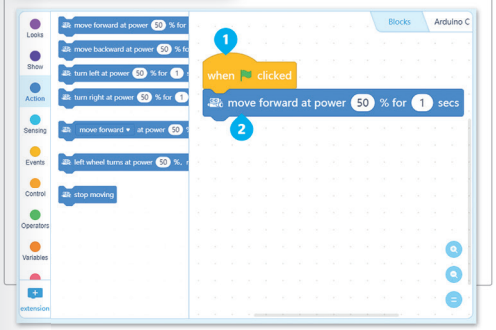
To create a fully-autonomous car requires a lot of sensors and complex programming scripts. We are going to use our mBot and its Ultrasonic sensor. The robot will move forward and when it detects an obstacle it will turn right and continue its movement. In this way, it will avoid any obstacles it finds in its way. Add the mBot to the Device tab and connect the mBot with mBlock so the robot can follow the instructions of the scripts.

But, how exactly can a car drive by itself? The correct answer is "with the use of sensors". Sensors can give cars some "human senses", a fact that makes them autonomous. Let's create our Autonomous robot!

To start the script:

> Drag and drop the **when flag clicked** block into the script area. 1

> Add the **move forward at power ( ) % for ( ) secs** block from the **Action** block category. 2



HISTORY

Ultrasound is used in many different fields. Apart from detecting objects and measuring distances, ultrasound imaging or sonography is often used in medicine or for accelerating chemical processes.

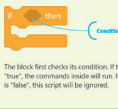
Decisions and conditions

Making decisions is an important part of everyday life. As humans, we make decisions based on what we observe or know to be "true". For example, if it is raining outside, we will use an umbrella. Conditions deal with cause and effect. A computer cannot decide by itself how to react, and that's the reason we use conditional statements. In this way, we tell the computer what to do and when to do it.

How the if () then block works

Conditional statements allow us to control what a computer program does and to make a computer perform different actions based on logical statements. The program executes a particular section of code based on whether a condition is "true" or "false". The instruction most often used to make a decision in a program is the **if () then** statement. If statements control the flow of a program.

In mBlock, the **if () then** block belongs to the orange **Control** blocks category and it controls the flow of the script.



The block first checks its condition. If the condition is "true", the commands inside will run. If the condition is "false", this script will be ignored.

*One of the most important parts of programming, is to check conditions. Using the **if () then** block is the simplest way to do that. When you need to check more than one condition, you can use more **if () then** blocks. And so, this block is used in many cases. You can use it to compare values, to check the given input or to control objects!*

Conditional operators in mBlock

When coding conditions, you can use conditional operators to compare values and act depending on the result. The result of a conditional check can be either "true" or "false".

Three of the **Conditional Operator** blocks are the **more than** block **( ) > ( )**, the **less than** block **( ) < ( )** and the **equal to** block **( ) = ( )**. Each one has two white boxes in which you type text or put a value.

The **( ) more than ( )** block checks if the first value is greater than the second value. If the first is more, the block returns "true", if not, it returns "false".

The **( ) less than ( )** block checks if the first value is less than the second value. If the first is less, the block returns "true", if not, it returns "false".

The **( ) equal to ( )** block checks if the first value is equal to the second value. If the values are equal, the block returns "true", if not, it returns "false".

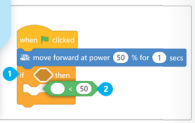
Is there any obstacle?

In the following example, the robot moves forward and checks if there is any obstacle in its way. When the mBot detects any obstacle with its Ultrasonic sensor it turns left and continues moving forward. Use the **if () then** block to check if the object is close enough.

To create a script that checks for obstacles:

> From the **Control** block category, add the **if () then** block. 1

> Click the **Operators** block category. Add the less than **( ) < ( )** block into the **if () then** block. 2



Continue creating the script as shown below:

This block has two new words **if** and **then**. The word **if** is followed by a condition. In this case, **distance(cm) < 20**. The computer compares the data in **distance(cm)** with the condition, and if the number is smaller than 20, it executes the commands in the **if** branch. This is called conditional execution.

This command is executed if the condition is "true".

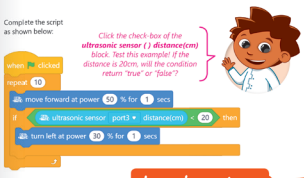
Try running this script! The mBot will move forward only once and then it will stop. Do you remember which block runs a group of blocks over and over again?

BE SAFE

Make sure that the robot is in a safe place before testing your programs.

Complete the script as shown below:

*Click the check-box of the **ultrasonic sensor ( ) distance(cm)** block. Test this example! If the distance is 20cm, will the condition return "true" or "false"?*



hands on!

Create a program so that the robot shows different colors according to the distance of an object from its Ultrasonic sensor.

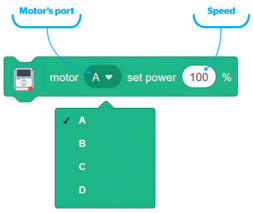
*Answer the following **true-false** questions. You can use your computer if needed.*

1. The **ultrasonic sensor ( ) distance(cm)** block reports the distance of an obstacle sensed by the specified Ultrasonic sensor. *True* ☐ *False* ☐
2. The commands inside the **if () then** block are executed if the condition is "**false**". *True* ☐ *False* ☐
3. The **( ) equal to ( )** block is located in the **Control** block category. *True* ☐ *False* ☐
4. The **if () then** block belongs to the **Events** block category. *True* ☐ *False* ☐
5. The result of a conditional check can be either "**true**" or "**false**". *True* ☐ *False* ☐

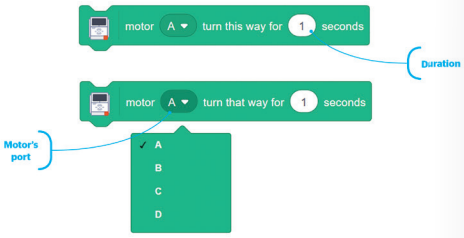
Move your robot

In order to make the robot drive forward or backward, you use the **motor () set power ()%**, **motor () turn this way for () seconds** and **motor () turn that way for () seconds** blocks. These blocks control the movement of the motors of the robot.
More specifically, **motor () set power ()%** block changes how fast the motor operates, **motor () turn this way for () seconds** makes the motor turn clockwise for the specified number of seconds and **motor () turn that way for () seconds** makes the motor turn counter-clockwise. You can find them on the LEGO EV3 palette.

> **Motor () set power ()% block properties**

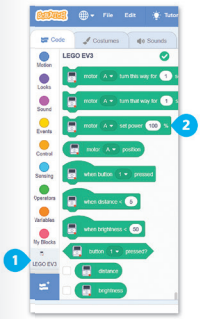


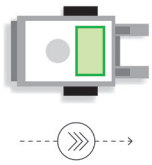
> **Motor () turn this way for () seconds** and **motor () turn that way for () seconds block properties**

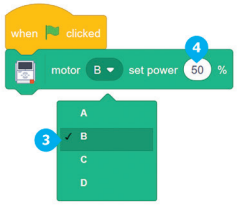


Let's make the robot move forward for 5 seconds with a speed of 50.
Make a script to control motor B and then duplicate it, to control motor C.

To set motor B's power:
> From the **LEGO EV3** palette (1), add the **motor () set power ()%** block. (2)
> Set **motor** to **B**. (3)
> Set **power** to **50**. (4)

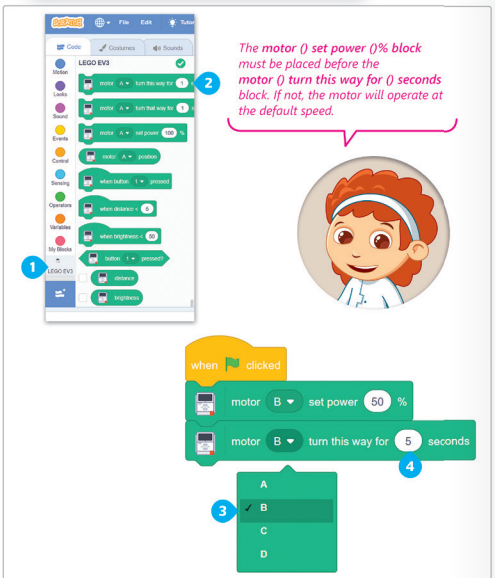






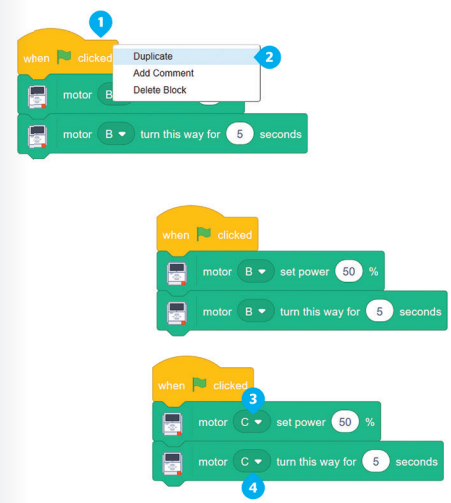
BE SAFE
Make sure that the robot is in a safe place before testing your programs.

To set motor B's direction:
> From the **LEGO EV3** palette (1), add the **motor () turn this way for () seconds** block. (2)
> Set **motor** to **B**. (3)
> Set **seconds** to **5**. (4)



In order to make your robot move forward, motor C must move exactly like motor B.

To control motor C:
> Right click on the **when flag clicked** block (1) and choose the **Duplicate** option. (2)
> Set **motor** to **C**. (3) (4)



Robotics G5

LESSON 1

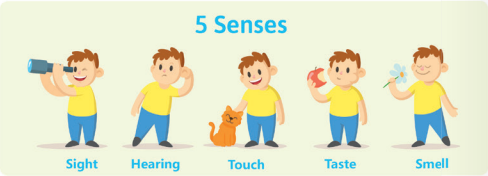
What are sensors?

Human senses

Humans have sensors too. These are the five basic senses: **sight**, **hearing**, **touch**, **taste** and **smell**.

The sensing organs associated with each sense send information to the brain to help you understand and recognize the world around you.

For example, sight is the sense that stops you from crashing into objects when you walk around.



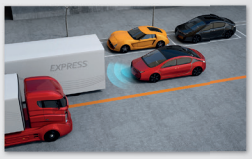
Sensors

Sensors are devices that are really important in the modern world. They can detect movement, weight, frequency and temperature.

Examples of sensors

A thermometer is a sensor, it can measure temperature.

Some cars have sensors that help the driver park or drive by detecting obstacles around them.



What is a Motion Sensor?

The WeDo 2.0 robotics kit contains a Motion Sensor. This specific sensor can detect objects within a range of 10 cm when programmed using the WeDo 2.0 software.

Modes of the Motion Sensor

The WeDo 2.0 Motion Sensor detects changes in distance of an object within its range in three different ways and also has an extra mode to input a value detected from the sensor to a block.

Any Distance Change
Set the Motion Sensor to detect any change in distance to an object.

Distance Change Closer
Set the Motion Sensor to detect if an object is moving closer.

Distance Change Further
Set the Motion Sensor to detect if an object is moving further away.

Distance Sensor Input
Set the Motion Sensor to input the value detected (from 0 to 10) to a block.

SMART TIP

Make sure you have the correct icon in your program that corresponds to the mode you want to use to detect an object.

What is an input?

You want to make a sandwich. You can put in a lot of ingredients to make it very tasty.

In programming, all the ingredients you put between the slices of bread in the sandwich you would call inputs.

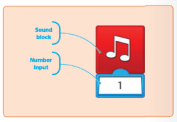
You use inputs when you program in the WeDo 2.0 environment.

WeDo 2.0 Inputs

The WeDo 2.0 environment has programming blocks and inputs.

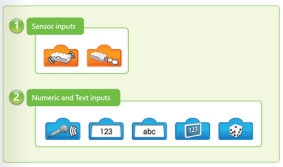
Remember you have to use both!

You have changed the number of a **Number Input** to choose a sound in the **Sound** block in a previous grade.



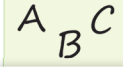
Input categories

The WeDo 2.0 software has two input categories.



Numeric Inputs can be numbers.

Text Inputs can be letters.

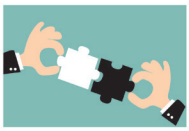


Motion Sensor

The Motion Sensor Inputs to Numeric Inputs. That is the Motion Sensor car det distance as a number of c For example, an obstacle i away from the Motion Sen

Puzzles

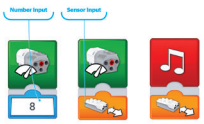
Puzzles are great for helping your brain develop and grow. Each piece of a puzzle is unique, and if you connect all the pieces together, you can complete the puzzle.



Look! I put some blocks and drag and drop inputs to each one.

WeDo 2.0 puzzles

When you program in the WeDo 2.0 environment, it is kind of similar to building a puzzle, but you have only two pieces. The blocks and the inputs. You can match any kind of block with any kind of input.

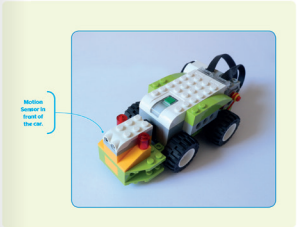


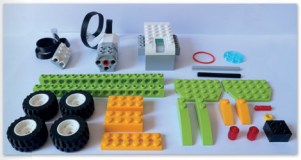
SMART TIP

Before you match a block with an input, you can see a gray color appear between them to help you combine them.

Build the car

In this lesson, you're going to build a car with the Motion Sensor.



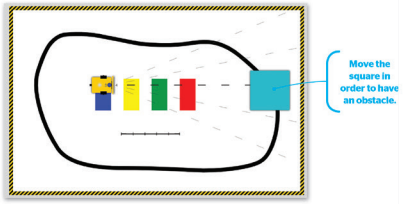


Testing the EV3 sensors

Before you start programming with the EV3, it is recommended that you check the sensors you are about to use, in order to see how they work. Now that you know what testing and debugging are, you can use them in order to test the Ultrasonic sensor and the Color sensor of the EV3 robot.

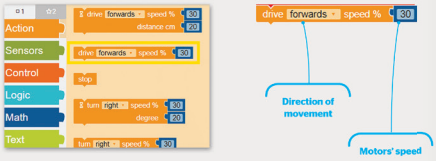
Ultrasonic sensor

Creating a simple program like the one below, gives you the opportunity to test whether the Ultrasonic sensor can detect an object at a distance of less than 15 centimeters. More specifically, the robot will move forward till it detects an object at a distance of less than 15 cm. In such a case, it will stop.



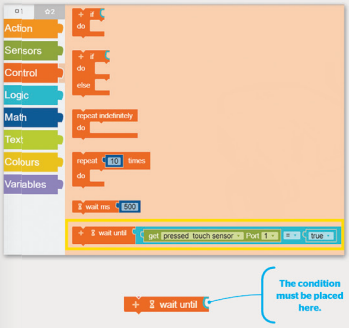
drive block

You can program the robot's direction (fowards and backwards) and speed with the **drive** block. The speed of your robot is set in the **speed %** parameter. The motors move until they are stopped by detecting a block.
You can find the **drive** block in the **Action** category.



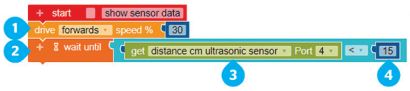
wait until block

This block stops the code from running until the condition is true.
You can find the **wait until** block in the **Control** category.



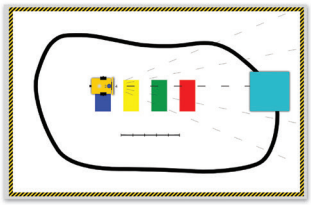
Test the Ultrasonic sensor:

- > From the **Action** category, add the **drive** block. 1
- > From the **Control** category, add the **wait until** block. 2
- > Set the first value to **distance cm ultrasonic sensor**. 3
- > Set the last value to **15**. 4



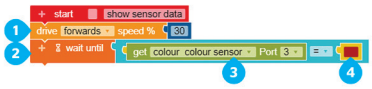
Color sensor

After checking the Ultrasonic sensor, you can create a simple program like the following one in order to test how the Color sensor detects a specific color. More specifically, the robot will move forward till it detects a red line. When it detects the red line, it will stop.



Test the Color sensor:

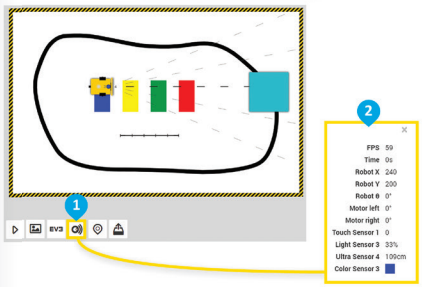
- > From the **Action** category, add the **drive** block. 1
- > From the **Control** category, add the **wait until** block. 2
- > Set the first value to **colour colour sensor**. 3
- > Make sure the color is set to **red**. 4



The sensors of your EV3 robot can observe its environment. You can be informed of the position of your robot, what it "sees" and "feels" and the ports of the sensors in the sensors' data view.

Open the sensors' data view:

- > Click the **Sensors** button. 1
- > The sensors' view window appears. 2



hands on!

Select the sensor you will use to do the following:

- > *Measure the temperature of the room.* _____
- > *Create a burglar alarm.* _____
- > *Control the distance of the vehicle from other cars.* _____
- > *Stop at a red traffic light.* _____

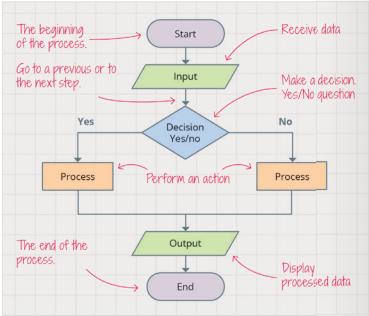
python™

Flowchart

A flowchart is a type of diagram that represents an algorithm and shows the steps you need to follow and their correct order. This diagram gives a clear step-by-step solution to a problem broken into smaller tasks or specific instructions. You can make flowcharts to describe your thoughts about how to solve a problem using a computer before you actually start writing the program.

You can visualize the steps of an algorithm by drawing 4 different types of boxes to reflect different actions and connect the boxes with arrows to show their order.

Type of box	Description
Start/End	To mark the beginning and end of the process.
Input/Output	To receive data and display processed data (input and output).
Process	To perform an action (do calculations or give commands).
Decision	To make a decision, usually a Yes/No question or a True/False test.
↓	Use arrows to show the order in which the steps flow.



Flowchart Best Practices

> A flowchart must contain a starting point and an end point.

> The arrow lines that connect the actions should not touch each other.

> There should not be any actions that are not included in the flow.

Stages of program creation

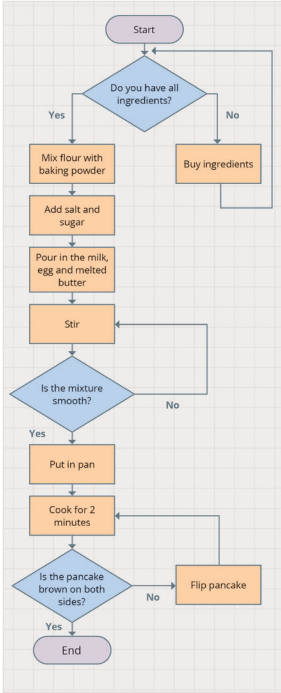
The first thing you have to do is to identify the problem and write the steps needed to solve it.

You put the steps in a logical and sequential order to form the algorithm.

The next step is to draw the flowchart, which shows the logical sequence of the algorithm.

The last step is to write the program in Python.

Here is the flowchart of creating pancakes



Define the problem

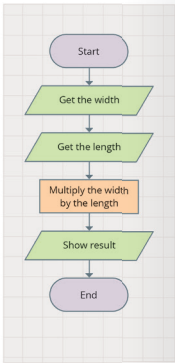
Before you start designing a program, you have to define and understand the problem you have to solve and and what has to be done to accomplish your aim. For example, let's say that you want to calculate the area of a rectangle.

First, you have to think about the steps that are needed to get your answer. In this example, you need to know the length of the two sides of the shape (width and length). Area = Width x Length.

1. Get the width.
2. Get the length.
3. Multiply the width by the length.
4. Show result.

Steps of the algorithm

The flowchart



Geek talk

The tool that is used to convert the programmatic section from the high-level programming language into the machine language for the computer to understand is called a compiler.

Let's code

To write your first program you must convert the flowchart to a programming language. The following program section will calculate the area of a rectangle in Python. You will soon learn how you can write each instruction by yourself.

The code

```
print("Let's calculate the area of a rectangle")
print("Type the length of the rectangle: ")
length=int(input())
print("Type the width of the rectangle: ")
width=int(input())
area=length * width
print("The area of the rectangle is: ",area)
```


Practice

1 Write an algorithm to calculate a student's final grade. The final grade is calculated as the average of three grades.

Below are the steps to create the program algorithm in random order. Arrange the steps correctly, and then create the flowchart of the algorithm.

- Calculate student's final grade by summing and dividing by 3.
- Ask the user to enter the three grades.
- Display the result on the screen.
- Read the three grades.

Steps of the algorithm

1. _____
2. _____
3. _____
4. _____

The flowchart

Lesson 3

Input data

Lesson Description

The general purpose of this lesson is for students to learn how to interact with users, to get data or to provide a result. They will also learn how to use Python in order to make calculations.

Objectives

Students have to:

> learn how to ask a user to enter a value to a variable.
> understand how to use arithmetic operators to perform calculations with numbers and variables.

Learning difficulties - misconceptions

> Students have difficulty in understanding that while they are programming, they have to view the program from the users perspective. For example, if the programmer wants the user to iput two numbers, the programmerhas to use the proper function in order to ask the user clearly to input the correct type of data the program needs.
> Students may have difficulty in understanding that when they use input commands, the user has to type a value for the variable. Give students time to practice on some examples.
> Also, students sometimes don't have the basic mathematics knowledge which is required in order to make calculations. For example, students might not remember the method we use in order to calculate the average of a group of numbers.

Brainstorming

> Introduce the purpose of the lesson by motivating students' interest in programming using Python.
> Start by asking students questions such as:
- What command would you use to ask the user to input data?
- Do you know the mathematical operators and when each one is used?

Tips for implementation

> Using the examples in the Student's Book, mention that in programming we are always working with different types of data. Continue explaining the five types of data we use in Python and in programming in general. Try to focus on their similarities and differences. At this point, it is important for students to recognize the different uses of each category.
> When learning how to make calculations using Python, inform students that it is time to use their knowledge of mathematics. At this point, it is important to take some time and remind students the basic mathematical rules that are required for this lesson, such as the priority of calculations.

Project activity

Tips & best practices

> Encourage students to first read the project carefully in order to break down the problem into small steps. After completing the process, ask them to write down the steps and create the algorithm of the project.
> Then, ask students to start creating the flowchart. Let them experiment and help them if it is necessary. Remind them to use the correct shape that is required for each step.
> After completing the algorithm and the flowchart, students can test if the process is the same in both methods. Make sure that the program follows the steps according to the flowchart.
> While typing the code students must take care when using parenthesis. Remind them that when they open a parenthesis, they always have to close it.
> After completing the code, encourage students to compare it with the algorithm and the flowchart of the project.
> Finally, they have to run and test the program. Students must enter different numbers each time they run the program to be sure that it works correctly in different cases. Encourage them to use integers and float numbers.
>

Extra Practice for high ability students

> Ask students to use their skills to complete this activity:
- Using the project of the unit, ask them to go on to a more advanced level.
- Ask students to type comments into the code so another programmer that didn't write the program can understand its function.
- Ask students to change the program so it can work with any payment terms. The percent value paid in advance and the amount of equal installments must both be given by the user.
- To achieve this, encourage students to run their code and check if the results are the same as their calculations on paper. Ifthey aren't, ask students to find the mistakes, fix them and run the program again.

1.2

Lesson 2
Variables and loops

So far, you have learned how to work with fixed numbers and text. In this lesson, you are going to learn how to assign values to the variables of a program. Variables are associated with data storage locations. In addition, we sometimes need a part of the code to be repeated several times in programming. Almost all the programming languages provide a function called a loop.

Variables

Variables are associated with data storage locations. A symbolic name is given to a variable that permits it to be used independent of the information it represents. The value of a variable can change during the execution of the program. Variables can represent different types of data. The two main categories of variables are numbers and text. Python supports two types of numbers - integers and floating point (decimal) numbers. As we mentioned in Scratch, text variables are also called strings.

A variable can have a short name (like x or y) or a more descriptive name (like age, carname, total_volume etc.).

Syntax inspector
Some names cannot be used, because they are special words already used by the programming language. These are called **reserved words:**

```
def     and
return  not
while   True
else    False
global  None
if      import
```

Numbers (numeric variables)

```
MyAge=12
level=3
score=1200
```

Text (string variables)

```
MyName="Nicky"
EmailAddress="nicky@binary-academy.com"
color="Green"
```


1.2

Declare a variable

Declaring a variable is simply a matter of assigning a value and an identifier (a unique name) to a variable. To declare a variable, you use the equal sign. In coding, the equal sign (=) is not used like it is in mathematics. For example, **MyAge=12** means that you take the value **12** as a number and assign it to the variable named **MyAge**.

You can also calculate anything on the right side of the equal sign and then assign the result to the variable on the left side. Let's see an example!

Syntax inspector
When using text variables, you should always type the text between quotes " ".

To set a value to a number variable:

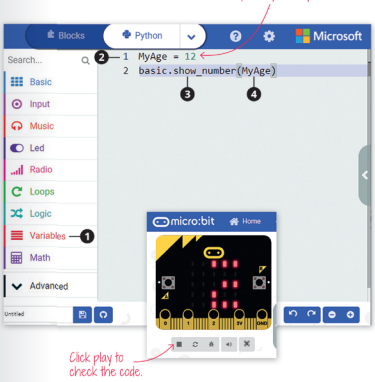
> Click the **Variables** command category. ❶

> Drag and drop the **item=0** command, and set the variable name to **MyAge** and its value to **12**. ❷

> From the **Basic** command category, drag and drop the **show** number command. ❸

> Type the variable name inside the parentheses. ❹

You can input any value you want.



Click play to check the code.

1.2

String variables

Variables cannot only store numbers. You can use them to store text too! Variables which store text are called string variables. To assign text to a variable you just put the text inside quotes.

Syntax inspector
In a program, every variable has a unique name and a value.

To set a value to a string variable:

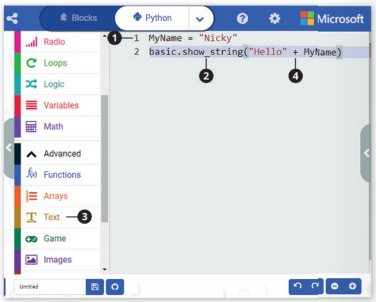
> Type the variable name and its value. ❶

> From the **Basic** command category, drag and drop the **show string** command. ❷

> Click the **Text** command category from the **Advanced** commands. ❸

> Drag and drop the **concat** command inside the **show string** command and type the string **Hello** and the name of the variable. ❹

Go further!
While programming in Python, as it is a text based programming language, you can type the commands you remember. It is not always necessary to choose them from the commands categories.



Change command

Variables can be used for a variety of tasks. For example, you might want to change the value of a variable in order to use it as a counter. This **Variable** command raises the value of a specified variable by a defined amount.

```
item += 1
```

You can input any value you want.

You can use it only with numeric variables.

1.2

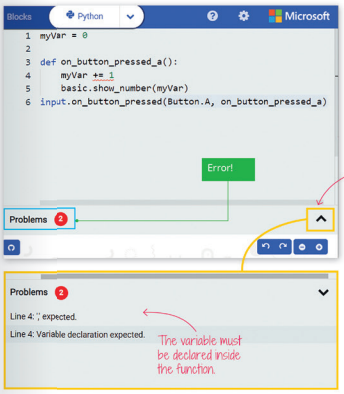
Local and global variables

When you declare variables within a function definition, they do not affect and are not affected by other variables of the same name that are used outside of that function. The part of a program where a variable can be accessed and used is called the scope of the variable. Local variables have the scope of the part where they are declared, starting from the point in a function where the name is defined and ending when the function stops executing.

Let's see an example in a function where the first time we use a value named **myVar**, Python uses the value of the parameter declared inside that function.

Let's create a program in which every time you press the button **A** of the micro:bit, the value of the variable myVar changes by 1.

Create the following code:



Geek talk
Local variables can be accessed only inside the function in which they are declared.

Click to see the problems.

The variable must be declared inside the function.

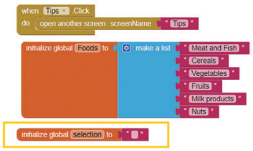
Geek talk
Debugging is the process of checking, detecting and correcting the errors in a program.

If you want to assign a value to a name defined at the top level of the program (i.e. not within a particular scope such as in functions or classes), then you need to tell Python that the name is not local but global. This is done using the **global** command.

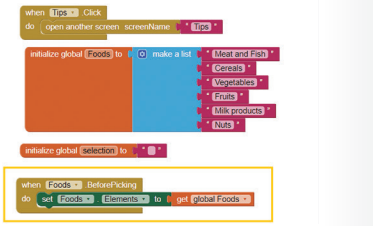
Programming the menu

We must create a new variable and assign a specific value to it from the list in order for the item list selection process to take place.

Create a new variable called "selection" and connect it to the Text String block..



To display the list of the items, we have to add the following code.



To display the list:

> From the Blocks panel, click the "Foods" menu button.❶

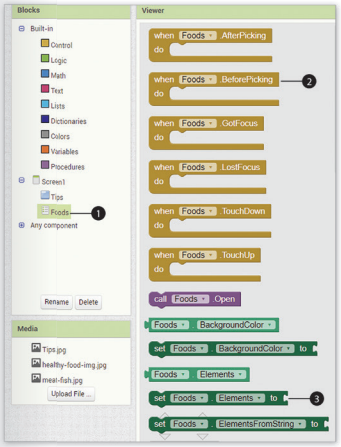
> Click the when Foods.BeforePicking do block, then drag and drop it into the programming area.❷

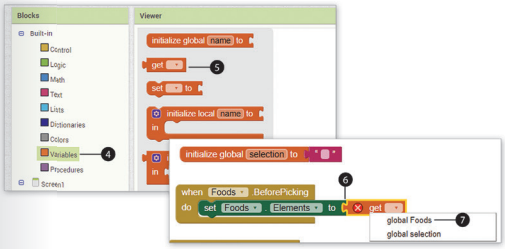
> Click the set Foods.Elements to block, then drag and drop it into the when Foods.Before.Picking do block.❸

> In the Blocks panel, click the Variables section.❹

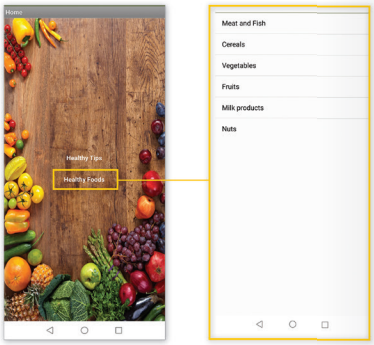
> Click the get block,❺ then drag and drop it into when Foods. BeforePicking do block.❻

> Click the arrow in the get block and select global Foods.❼





At this point, when we open the application on your phone you will see the following:



Try it out!

Change the appearance of the list using the options in Properties for the ListPicker button.

ItemBackgroundColor
Default

ItemTextColor
Default

Selection

So, when we click the Healthy Foods button, the list appears. You will create a new screen so when you choose an item, for example "Meat and Fish", a new screen for that item will open.

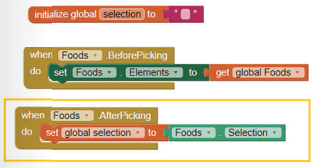
We will create a new screen and add the following tools to it just as we already have learned to do in the previous lesson:

> Labels

> Image

In this way, we will program the list of foods that we have created, using the following blocks, where the menu will be activated and ready to use.

This event will be activated after an item is selected from the list. ListPicker returns its results and the properties that are added to it.



3.2

Lesson 2
Play some music!

The micro:bit is a simple miniature computer in the form of a microchip. As do all computers it contains one or more processors along with memory and supports different programmable input/output peripherals. You can use your micro:bit in a wide array of projects. For example, even if the micro:bit does not have a speaker, you can produce sound using an output device!

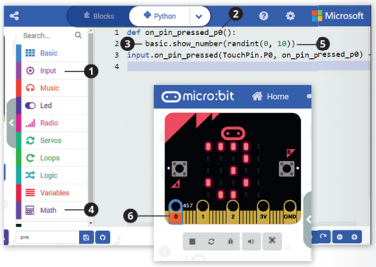
In this module, you will meet the Piezo buzzer.

It's time to see how you can use the micro:bit crocodile clips. You are going to create a new project in which every time you press P0, the screen displays a random number from 0 to 10.

Start by creating a new project.

To create a program for P0:

> Click the **Input** command category. ❶
> Drag and drop the **run code on pin ( ) pressed** function. ❷
> From the **Basic** category, drag and drop the **show number ( )** command inside the function. ❸
> Click the **Math** command category. ❹
> Drag and drop the **randint** command. ❺
> Click the **P0** pin to complete the circuit. ❻



3.2

Output pins

Pins connect the of micro:bit with the outside world. Through pins, the micro:bit can send output signals. Output signal can be digital or analog. A digital signal is either 1 or 0, an analog signal can be any number between 0 to 9. At this point, you will learn about digital output signals.

Digital signal

A digital signal uses discrete 0s and 1s to represent information. In electronic circuits like micro:bit, high voltages represent a 1 and low voltages a 0. So, for example, you send a 1 to the sound producer when you want to hear a sound and a 0 when you want to stop the sound.

Bits 'n' tips

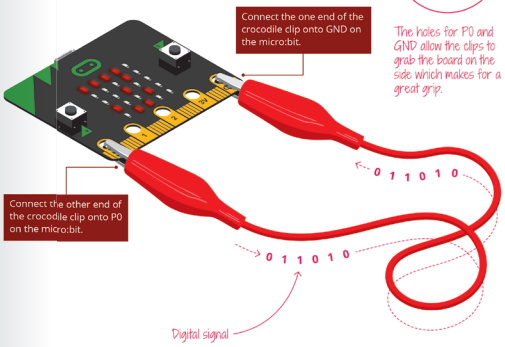
For the center holes, P1 and P2, the clips can also grab the bottom of the board, but they are a bit harder to grip.

Connecting crocodile clips

While working with hardware, you can use crocodile clips as wires of a circuit. The large holes at the bottom of the board are designed to attach crocodile clips to create electrical circuits with other components. Each time the crocodile clip is connected and then disconnected from the P0 pin, the micro:bit will return a random number between 0 and 10.

Complete the circuit as shown below:

The holes for P0 and GND allow the clips to grab the board on the side which makes for a great grip.



After completing the connection, download the program to the micro:bit.
Test your program and check that works as expected.

3.2

Play a melody

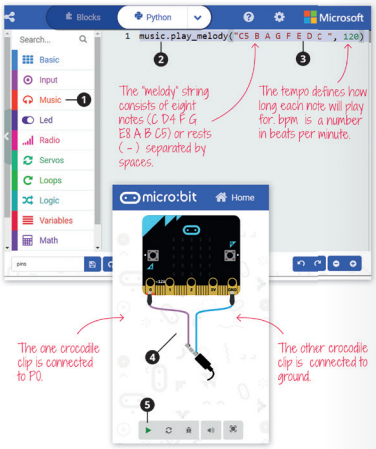
Another thing you can do with the micro:bit is to produce sound. To play music, you have to use a **Music** command. In this project, you will use the **play melody ( ) at tempo ( ) bpm** command.

To play a melody:

> Click the **Music** command category. ❶
> Drag and drop the **play melody ( ) at tempo ( ) bpm** command. ❷
> Type the melody string inside the parentheses. ❸
> MakeCode creates the connection in the simulation automatically. ❹
> Click the **play** button to hear the music. ❺

Bits 'n' tips

To find ready melodies, go to the Blocks editor and select one from the drop down list of the play melody ( ) at tempo ( ) bpm block.



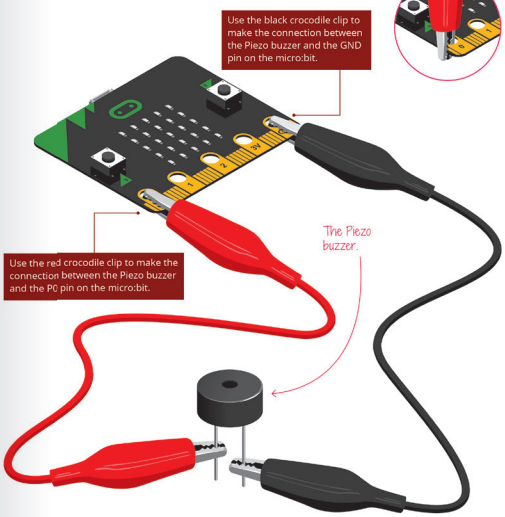
3.2

The Piezo buzzer

To produce sound in for hardware micro:bit, you will need a buzzer. The Piezo buzzer is an electronic device which produces sound based on the reverse of the piezoelectric effect. The generation of pressure variation or strain by the application of an electric potential across a piezoelectric material is the underlying principle. The pressure variations are what create the sounds.

Let's complete the connection of the micro:bit and buzzer.

You can also, pass one jaw through the hole and grab the side of the board with the other jaw.



Try it out!

Create the following code in MakeCode.

```
music.play_melody("G B A G C5 B A B ", 40)
music.play_melody("G B A G C5 B A B ", 120)
music.play_melody("G B A G C5 B A B ", 470)
```

Describe the function of the code.

Controlling the volume

MakeCode allows you to increase or decrease the volume of the sound played. The volume can be set between 0 and 255. The 0 value silence your sound.

In the following example, you are going to use the A button to control the sound volume. Each time you press the A button, the volume will decreases by 10.

To decrease the volume:

> From the **Music** category, drag and drop the **set volume ( )** command and set the volume to **255**. ❶

> From the **Basic** category, drag and drop the **run code forever** function. ❷

> From the **Music** category, drag and drop the **play melody ( ) at tempo ( ) bpm** command. ❸

> Type the melody string inside the parentheses. ❹

> From the **Input** category, drag and drop the **run code on button ( ) pressed** function. ❺

> Drag and drop the **set volume ( )** command. ❻

> From the **Music** category, drag and drop the volume command and type **- 10**. ❼


```
1  #sets the volume to maximum
2  music.set_volume(255)
3
4  #plays the melody
5  def on_forever():
6      music.play_melody("C5 B A G F E D C ", 120)
7  basic.forever(on_forever)
8
9  #decreases the volume
10 def on_button_pressed_a():
11     music.set_volume(music.volume() - 10)
12 input.on_button_pressed(Button.A, on_button_pressed_a)
13
```

Add comments in your code.

Continue the above example. Add some new code so that every time the user presses the B button the volume of the sound increases.

Write your code:

Try it out!

Do you remember?
You can find the "if else" command in the "Logic" command category.

Is the pin pressed?

Apart from the **run code on pin (TouchPin.P0) pressed** function, the micro:bit provides alternative ways to check whether a pin is pressed. The boolean **pin (TouchPin.P0) is pressed** command allows you to use it inside the same set of code as a condition. This is something that a function like a header block cannot do. This command gets the pin's state (pressed or not) and returns **true** if the pin is pressed or **false** if the pin is not pressed.

Let's try some code to see how it works!

You will create a program which will produce different melodies on each pin pressed. The program will be activated when the button A is pressed.

```
def on_button_pressed_a():
    def on_forever():
        if input.pin_is_pressed(TouchPin.P0):
            music.play_melody("A F E F D G E F ", 120)
        elif input.pin_is_pressed(TouchPin.P1):
            music.play_melody("B A G A G F A C5 ", 120)
        elif input.pin_is_pressed(TouchPin.P2):
            music.play_melody("G F G A - F E D ", 120)
    basic.forever(on_forever)
input.on_button_pressed(Button.A, on_button_pressed_a)
```

First press button A and then click the pins to listen to the different melodies.

Try it out!

Let's use our micro:bit device!

> First, create the circuit using the micro:bit, the Piezo buzzer and two crocodile clips.

> Download the program to the micro:bit.

> Test your program, by connecting the one clicodile clip to the GND pin and the second one to pin 0, then pin 1 and then pin 2, in order to listen to all the melodies.

Practice

1 Create a program which on start increases or decreases the sound produced by 20 using the A and B buttons of the micro:bit. Include the following commands in your code:

- if else
- button (Button.A) is pressed
- forever

2 Create your own jukebox program in which each time you press the button B, a random song plays. Add comments to your program. Below are some helpful tips for your program.

- On button B pressed the song changes.
- The program will pick a random number from 0 to 2.
- Each number corresponds to a pin of the micro:bit.
- According to the picked number, a different song will be heard.

Draw the circuit:

Hyperlinks

It is very useful to use links on your website as they allow you to move from one webpage to another.

Examples of links:

- Links from one page to another page on the same site.
- Links from one part of a web page to another part of the same page.
- Links from one website to another.
- Links open in a new browser window.
- Links start your email application, to compose a new email message.

Syntax inspector
If the href attribute is not present, the <a> tag will not be a hyperlink.

Links are created using the <a> tag. Everything that is between the opening tag <a> and the closing tag becomes clickable, using the href property we can specify the target title (the page that will open when the link is clicked).

The page to open

The text the user will click on

```
<a href="https://www.nasa.gov"> Click here</a>
```

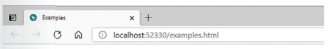
specifies the URL

End tag

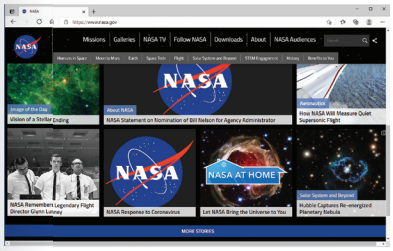
This is a link to a website (URL). The href attribute value here is the name of an entire website, just like the address you would type into your browser to visit that page.

Let's see an example of a hyperlink that links to other websites.

```
<!DOCTYPE html>
<html>
<head>
  <title>Examples</title>
</head>
<body>
  <a href="https://www.nasa.gov">Click here</a>
</body>
</html>
```



Clicking on this text will direct you to the specified location.

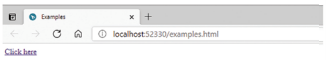


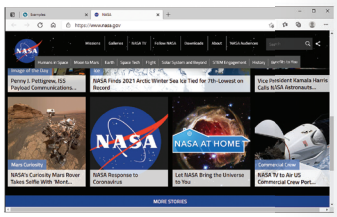
Target attribute

When using the target attribute in hyperllink information, we specify where the page linked to that URL will open. This property can take the following values:

Target	
Value	**Description**
_blank	The page will open in a new tab.
_self	The page will open in the same tab.
_parent	The page will open in the parent window.
_top	The page will open in the body of the window.

```
<!DOCTYPE html>
<html>
<head>
  <title>Examples</title>
</head>
<body>
  <a href="https://www.nasa.gov" target="_blank">Click here</a>
</body>
</html>
```





Football fan page

Create a navigation bar

In our project we have included a list arranged as a navigation bar. This list consists of a group of links. Generally, some elements of this list must be linked to a specific part of the page, while the item "Contact Us" is linked to another page on the same site.

Link to a specific part of the same page

Before we start linking to the specific part of the page, we need to highlight the part of the page that will be referred to via the link. For this purpose we will use the "id" attribute.

The "id" attribute is used with every HTML element to distinguish the element from the rest of the web page.

```
<h2 id="history">History</h2>
<p>Football, also called soccer has a long history.Football
in its current form arose in England in the middle of the 19th
century.</br>Football clubs have existed since the 15th century,
but were unorganized and without official status.</br> Lots of
football clubs were formed in the late-nineteenth century, but
only a few survived. Most historians believe that the clubs
</br>which lasted tended to be situated in slightly more
affluent areas, where skilled and semi-skilled workers
would have Saturday afternoon off work and would be able to
afford to spend money on attending football. </p>
<h2 id="gallery">Gallery</h2>
<h2 id="about"> About</h2>
<p>This is a page where we can exchange ideas and views about
the football team we support or football in general nowadays.</
br>
We can also communicate through the contact form to add more
photos to the gallery or articles.</p>
</body>
</html>
```

The id can be assigned with a word beginning with a letter or an underscore (_), and the same name cannot be assigned to two different items on the same page.

1.1

Lesson 1
Nested loops

Until now, you have used different types of loops one at a time. Now, you have to see what happens when you combine looping procedures. The placing of one loop inside the body of another loop is called nesting.

Any loop type can be nested within another type. The most commonly nested loops are the for loops.

Outer loop

Inner loop

```
for i in range(3):
    for j in range(2):
        print("i= ",i,"j= ",j)
```

```
i= 0 j= 0
i= 0 j= 1
i= 1 j= 0
i= 1 j= 1
i= 2 j= 0
i= 2 j= 1
```

The steps of execution:

1. i gets the value 0, the inner loop will be executed 2 times: for j=0, j=1.
2. now we increase the value of i, and for i=1, the inner loop will be executed again 2 times: for j=0, j=1.
3. now we increase the value of i, and for i=2, the inner loop will be executed again 2 times: for j=0, j=1.

At the end the outer loop has made 3 repetitions and the inner loop 6.

values		
i	j	
0	0	1
	1	
1	0	2
	1	
2	0	3
	1	

Don't forget!
The main advantage of functions is that we can call them repeatedly from the main program.

1.1

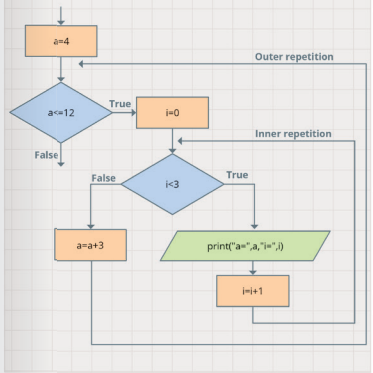
Let's see another example of nested loops.

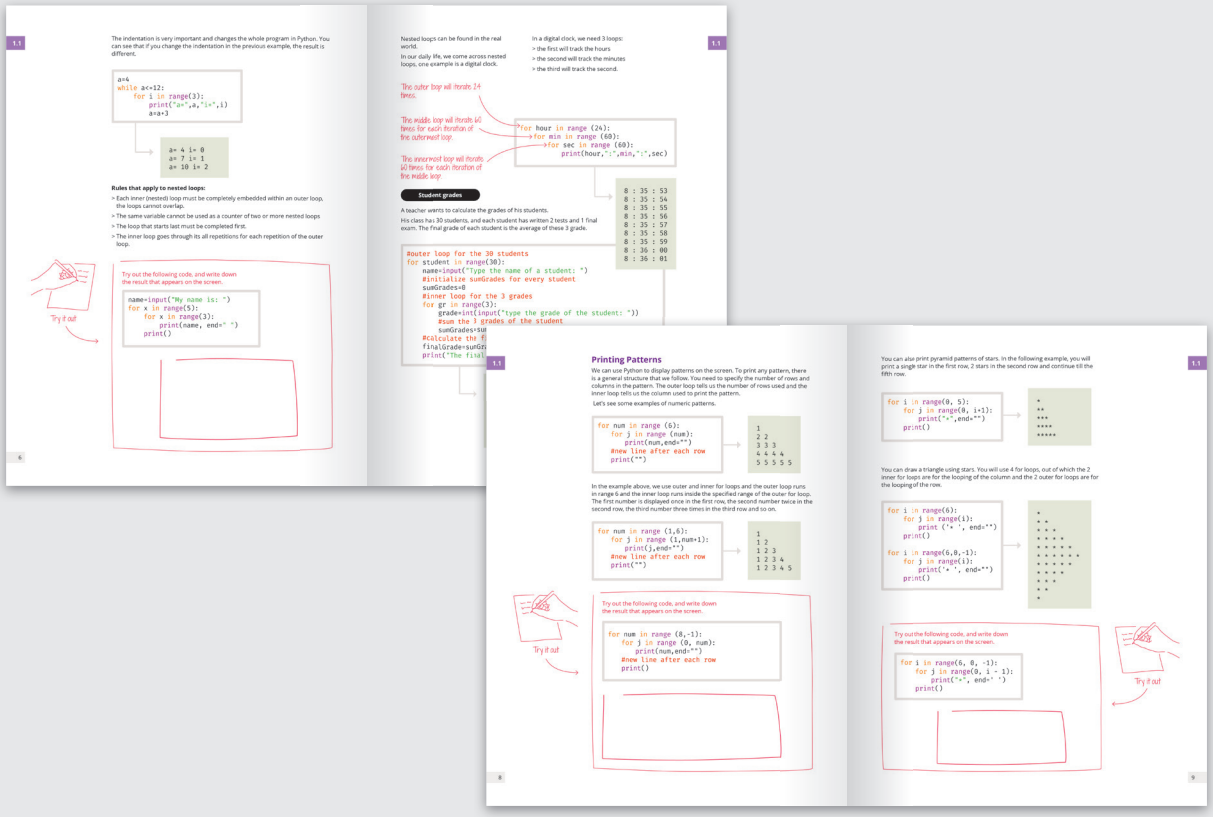
```
a=4
while a<=12:
    for i in range(3):

print("a=",a,"i=",i)
    a=a+3
```

```
a= 4 i= 0
a= 4 i= 1
a= 4 i= 2
a= 7 i= 0
a= 7 i= 1
a= 7 i= 2
a= 10 i= 0
a= 10 i= 1
a= 10 i= 2
```

Syntax inspector
You have to be very careful with the indentation in nested loops. Indentation determines the commands that are included in each loop.

The flowchart of the code





python™

1.1

Dictionary

Now that you are familiar with Python, it is time to familiarize yourself with the data structure of the Dictionary and some of the ready-made functions for dictionaries in Python.

The dictionary is a changeable data structure that contains a number of elements. Each element in the dictionary is composed of two values, the first representing the key and the second representing the same item. The dictionary differs from the previous structures in that the element is accessed via the key and not via the site number, as is the case in lists, rows and arrays. Key values can be of any data type.

A dictionary is a data structure that stores data in pairs, each pair consisting of two parts, a key and a value.

The general form of defining a dictionary

```
dictionary_name={key1:item1,key2:item2,...,key n:itemn}
```

A variable representing thenameofthedictionary.

Dictionary items.

> The curly braces {} are used when defining the dictionary, and a colon : is used to separate the element and key.

> There cannot be two items in the dictionary that have the same key, each key allows you to access one of the values in the dictionary.


```
Europe={"France":"Paris","Italy":"Rome","Spain":"Madrid"}
print(Europe)
```

```
{'France': 'Paris', 'Italy': 'Rome', 'Spain': 'Madrid'}
```

The difference between List and Dictionary

> A list is a series of consecutive items, while a dictionary includes unordered pairs of items.

> The main difference is the way you access the items. List items are placed in a list that is accessed by the site number, while the dictionary items are accessed through the keys.

Create the dictionary

We can create the dictionary by using the create command dict().

```
Europe=dict(France="Paris", Italy="Rome", Spain="Madrid")
print(Europe)
```

```
{'France': 'Paris', 'Italy': 'Rome', 'Spain': 'Madrid'}
```

You can also create a dictionary whose entries are filled in by the user.

```
myDict = dict()
key = input("Enter the key: ")
value = input("Enter the value: ")
myDict[key] = value
print(myDict)
```

Createanemptydictionary.

```
Enter the key: United Kingdom
Enter the value: London
{'United Kingdom': 'London'}
```


Tryitout

What would you add to the code to create a triple dictionary?

Functions used with the dictionary

Python offers us a set of built-in functions that can be used with dictionaries.

Function	Description
dictName.get(x)	Returns the value associated with key x, and if the key is not found in the dictionary, it returns None.
dictName.update(x)	Adds new item pair(s) to the dictionary if the keys are not already present in it. Or, it updates the value content associated with existing keys.
dictName.values()	Returns all values in the dictionary.
dictName.keys()	Returns all keys in the dictionary.
dictName.clear()	Deletes all items in the dictionary.

Access to dictionary items

The dictionary item does not have an index number, but there are two ways to access the items:

> Using the key of the element written inside the square brackets [ ].

> Using the get () function.

Let's look at the following example to understand it:

```
Europe= {
    "France" : "Paris",
    "Italy" : "Rome",
    "Spain" : "Madrid",
    }

capital1=Europe["Spain"]
print(capital1)

#use the get operation
capital2=Europe.get("France")
print(capital2)
```

```
Madrid
Paris
```

To change the value of an item within the dictionary, you can use the following commands:

```
Europe= {
    "France" : "Paris",
    "Italy" : "Rome",
    "Spain" : "Madrid",
    }

Europe["Italy"]= "Venice"
print(Europe)
```

```
{'France': 'Paris', 'Italy':
'Venice', 'Spain': 'Madrid'}
```

Lesson 3
Linear Search and Binary Search

Searching and sorting are two problems of particular interest in the field of Computer Science, due to their usefulness in a variety of applications. In this lesson, you will learn about searching algorithms.

Searching

Oftentimes we need to look for something specific in the dataset. Operating systems, software applications, and websites contain various search features to find specific data, regardless of the nature of that data, be it numeric or alphanumeric. Usually, we search for specific data such as a last name (surname), identity number, etc.
There are several search algorithms that are used depending on whether the data is arranged or not.

The two basic searching algorithms are:

> Linear Search

> Binary Search

A search is the process of finding a value in a collection of data.

Searching in everyday life

> Search for a book in the library using the title of the book or name of the author.

> Search for a specific telephone number in the contact list of your smartphone.

> Search engines like Google are the most popular example of a search program. You type a keyword and you get web pages that contain that keyword.

Linear Searching

The linear or serial search algorithm is the simplest search algorithm, and it relies on performing the search comprehensively on all the elements in the data collection.

This algorithm checks all menu items one by one, starting with the first one, and if the item we are looking for is found it returns True, otherwise it returns False.

Let's see the algorithm steps of the linear search algorithm.

1. The beginning of the algorithm.
2. Read the search element from the user.
3. Compare the search element with the first element in the list.
4. If they match, return the value True.
5. If they do not match, then compare the search element with the next element in the list.
6. Repeat steps 3 and 4 until you reach the end of the list.
7. If the last element in the list doesn't match, return the value False.
8. The end of the algorithm.

Python implementation of linear search.

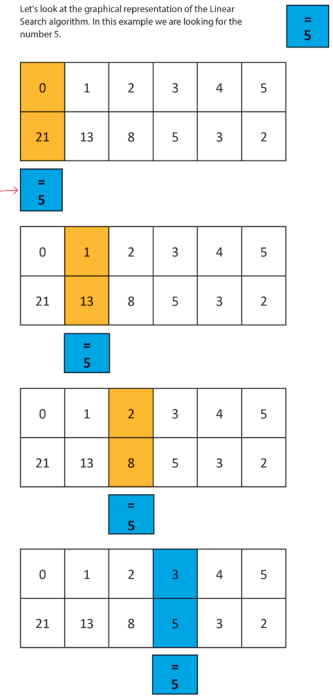
```
myList=[21,13,8,5,3,2]

key=int(input("Enter a number to search for:"))
N=len(myList)
found = False
for i in range(0,N) :
    if myList[i] == key :
        found = True

print("Does the number exist?",found)
```

```
Enter a number to search for:5
Does the number exist? True
```

Bits 'n' tips

You can use the Ctrl+F keyboard shortcut to open the search box in the web page or document you are viewing.

Let's look at the graphical representation of the Linear Search algorithm. In this example we are looking for the number 5.



The value 5 will be compared with the value of the first item, then the value of the second item ... and so on until we reach an item whose value matches 5, and then the value will be returned True.

Let's now see what happens if the element doesn't exist in the list.

```
myList=[21,13,8,5,3,2]

key=int(input("Enter a number to search for:"))
N=len(myList)
found = False
for i in range(0,N) :
    if myList[i] == key :
        found = True

print("Does the number exist?",found)
```

```
Enter a number to search for:32
Does the number exist? False
```

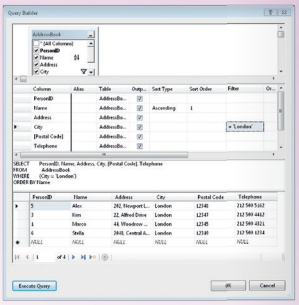
Sometimes you need to know the item's index (its location in the list) and not the item itself, so you'll add a new variable to keep the position or index where the item is found.

```
myList=[21,13,8,5,3,2]

key=int(input("Enter a number to search for:"))
N=len(myList)
found = False
for i in range(0,N) :
    if myList[i] == key :
        found = True
        pos=i

if found==True:
    print("We found the number in index:",pos)
else:
    print("The number doesn't exist in this list")
```

```
Enter a number to search for:5
We found the number in index:3
```

Let's add another query to our example program to demonstrate the use of the **WHERE** condition.

Follow the instructions in our previous example to create a query named *FromLondon*. We want our query to display only our contacts that live in *London*, sorted by their name.

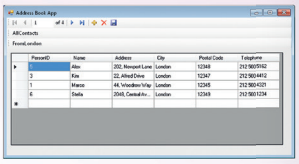


The condition here is `(City = 'London')`. Conditions in an SQL statement are inside parentheses to distinguish one condition from another when we have more than one. For example, if we wanted to display all our contacts that live in *London* and their name is not *Kim*, the condition would be

```
WHERE (City = 'London')
AND (Name <> 'Kim')
```

Notice that all text strings in our conditions must be inside single quotes `''`.

You can use any of the usual comparison operators in a condition and you can connect multiple conditions using the logical operators **AND**, **OR** and **NOT**.



hands on!

Create a quiz game program that reads questions and answers from a text file, stored in the form "Question, Answer", and asks the user to answer those questions. When checking the answers, the program should keep a score, how many correct answers out of the total number of questions the user achieved. The score is saved to a scores text file at the end.

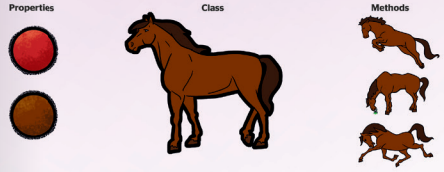
TASK 4

Classes, objects and inheritance

In programming, it's always a good idea to break your code down to small pieces that are easy to manage and understand. We have already seen a way to do that with functions and subs. Another popular approach in that direction is known as **Object Oriented Programming** (**OOP**). In OOP, we group together variables and functions or subs, to create classes and objects. A **class** is a piece of code that represents something that has certain characteristics and behavior and can be seen as an individual entity in your program. You have already seen a class in all our previous examples. Remember the first line in the code window:

```
Public Class Form1
```

and also `End Class` at the end of your code, indicating the end of the class named `Form1`. That means that all the things we write between those two lines, belong to `Class Form1`. To understand classes better, let's take an example from real life concepts. Let's say you want to represent the concept of a horse in your program. This can be represented by the class `Horse` for example. A horse has certain defining characteristics like its color or its name if it has one. These are the class' **Properties**. Properties are just variables that belong to a class and describe its characteristics. A horse also exhibits certain behavior, or can perform certain actions, like run, jump etc. These are the class' **Methods**. The methods of a class are functions or subs that implement the functionality of your class.



You have already seen many classes in your previous programs, even if you didn't know they were classes. Remember when we were reading text files in our programs, we used something called `StreamReader`. This is a class that describes something that can read streams of text and also has useful methods like `ReadLine` or `ReadToEnd`.

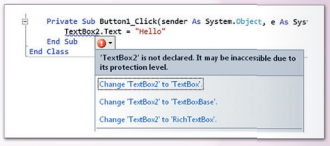
All the controls that we use in our forms are also classes! For example, a **Button** control is a class that describes buttons and has properties like *Name* and *Text*.

Classes are used to create **objects**. The difference between a class and an object is that a class is the code that we write to describe a concept, while an object is the instance of a class while it is being used in our program. For example, the class `Horse` describes what horses are and what they can do with its properties and methods, but an object of the class `Horse` is a specific horse that we have created, with a specific color and name. So, an object is an instance of a class that resides in memory and can execute various methods that are described in its class description.

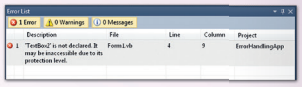
Error handling

It's very difficult to write perfect code every time. The usual program development process is writing the program, testing it out, discovering any errors and correcting them. Errors are bound to appear when programming. So you should be able to recognize them in order to be able to correct them. Programming errors are broken down into three types: Design-time, Runtime and Logic errors.

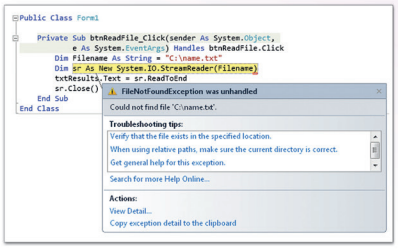
Design time errors, also known as syntax errors, are the most easy to find and correct. These errors occur when you mistype an instruction. The programming environment does not recognize the instruction and informs you with a blue wiggly line. In the following example, we are trying to set the *Text* property of a **TextBox** named *'TextBox2'*, but there is no control with such name in our form. If we click the red exclamation mark that appears next to the error, the environment informs us about the problem.



At the bottom of our code window, there is the **Error List** panel, which lists all errors found in the current code tab.



Runtime errors are harder to find because, as the name suggests, they occur when the program is running. These errors usually result in your program crashing. Runtime errors are the errors that the programmer should have predicted but didn't. For example, your program is trying to open a file that doesn't exist, or is trying to divide by zero and fails. In the following example you can see what happens when you are trying to open a file that doesn't exist.



Logic errors are the most difficult to find. They also occur while your program is running, but unlike runtime errors they usually don't cause your program to crash. A typical example of a logic error is infinite loops. For example, you have the loop

```
Do Until i > 5
```

but `i` never becomes greater than **5**, so the loop never ends and runs indefinitely. These errors don't crash your program but they prevent it from running as it was supposed to.

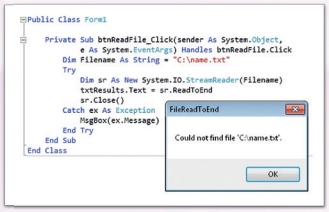
Try...Catch Statement

In order to prevent your code from crashing when an error occurs, you can use the **Try...Catch** statement. The general structure of the statement is as follows.

```
Try
    [code in which an error might occur]
Catch ex As Exception
    [code to run if an error occurs]
End Try
```

This essentially means "Try to execute this code and catch any errors that might occur". The **ex** is an object variable of the class `Exception`. This object of this class is created automatically when an error occurs.

Let's try and catch the "File not found" error from the previous example.

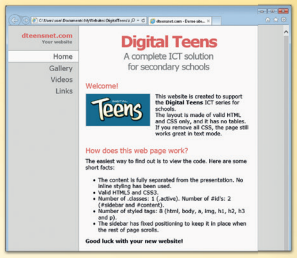


The **ex** variable is an object of class **Exception** and has its own properties, one of which is the **Message** property which contains a description of the error that occurred. You can see that this description matches the description that appears on the error window that appears on our code when we run the program without a **Try...Catch** statement.

You should make it a habit to surround any error prone parts of your programs in **Try...Catch** statements and deal with any problems accordingly. This way, your programs won't crash but simply inform the user that something went wrong.

Now, let's add the appropriate styling with some CSS rules.

```
a {
    color: #d85d5d;
    font-weight: bold;
    text-decoration: none;
}

a:hover {
    color: #505050;
    text-decoration: underline;
}

img {
    float: left;
    margin: 0 15px 15px 0;
    padding: 1px;
    background: #ffffff;
    border: 1px solid #d0d0d0;
}

#content h1 {
    margin: 0;
    font-size: 300%;
    color: #D85D5D;
    text-align: center;
    letter-spacing: -4px;
}

#content h2 {
    margin: 0;
    font-size: 170%;
    font-weight: normal;
    color: #808080;
    text-align: center;
    letter-spacing: -2px;
}

#content h3 {
    margin: 30px 0 10px 0;
    font-size: 140%;
    font-weight: normal;
    color: #D85D5D;
    letter-spacing: -1px;
}

#content p {
    margin: 0 0 20px 0;
}
```



The rules for the **<a>** elements and the **a:hover** special case define the behavior of all links everywhere on our page. Note however that because in CSS the rules cascade, all properties that are defined here are inherited by other CSS rules that define a more special case of the same selector, for example remember the **#sidebar a** selector. If, however, the more specialized selector redefines an inherited property, the inherited value is overridden.

All the other rules are just basic styling. Note however the **float: left;** property in the CSS rule of the **<img>** element. This allows all images to float to the left and allows other elements, like the text of the paragraph element in this case, to take up the space remaining to the right of the image.

This concludes our two-column web page layout. Now, you can use the same CSS file for the other pages and fill each one with the appropriate content. Remember, when designing a new page for the website, make sure to apply the active class to the link of the current page in the navigation panel. This gives the impression that the current page is "selected".

Make sure your code works in various browsers. Check it against the latest versions of the top five: Internet Explorer, Firefox, Chrome, Safari and Opera.

SMART TIP

Naming conventions for CSS selectors are important. Never use names that describe physical attributes (such as redLink), because changing the value later could render the name inappropriate.

CSS media types

With media types a page can have one layout for screen, one for print, one for handheld devices, etc. You can create different css files for each device and link them all to your page. In the following example we define two different style sheets for two different media types (screen and print).

```
<head>
<link rel="stylesheet" type="text/css" href="style.css">
<link rel="stylesheet" type="text/css" href="print.css" media="print">
</head>
```

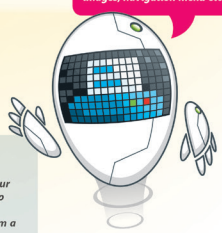
If we ommit the media property from the link tag, then the default value is screen, intended for computer monitors. Generally, you will want to use different values for the same properties between the different devices. For example, a document usually needs a larger font-size on a screen than on paper, and sans-serif fonts are easier to read on the screen, while serif fonts are easier to read on paper.

The @media rule

The **@media** rule allows different style rules for different media in the same style sheet. The style in the example below tells the browser to display a 14 pixels Verdana font on the screen. But if the page is printed, it will be in a 10 pixels Times font. Notice that the font-weight is set to bold, both on screen and on paper:

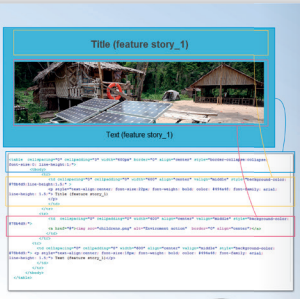
```
@media screen
  {
  p {font-family:verdana,sans-serif;font-size:14px;}
  }
@media print
  {
  p {font-family:times,serif;font-size:10px;}
  }
@media screen,print
  {
  p {font-weight:bold;}
  }
```

Printer-friendly stylesheets are very important if you want to save your visitors' ink and paper. Use the @ media print rule and remove anything that isn't necessary on the printed page, like colored backgrounds, stylistic images, navigation menu etc.



hands on!

Create a contact page with an HTML form for your own website and set the email address in the php script so that you receive the messages in your email. Visit your friends' websites and leave them a message using their contact form.



Work with external tables | Work with multiple tables | Test email clients

3. Test in all major clients

Finally, you have to check your newsletters on all major email clients for desktop, web and smartphone, at the very least on some of the more common clients such as Gmail, Outlook or Yahoo! Mail. If you know any friends or colleagues who use alternative software, you may wish to ask for their help on testing the newsletter.

Project 3

Design a one column website

The scenario

You are working in a website design company and your boss has assigned you to design the layout of a website for a restaurant. The website will have a one column layout which will include a header area, a horizontal navigation menu, an area for the contents of the page, and a footer.

The suggested way

First, plan the layout of the website. Next, using an image editing program design the look and layout of the site. Then slice your layout into pieces and export the pieces so that they can be later used and modified in a HTML editor.

1. Plan the website layout | Design the website layout | Export the images

1a. Fill in a creative brief

First, you have to identify your target audience and design around what they are comfortable with. Discuss with your customer how many pages your site is going to have and what the content of each page is going to be.

- *Start your project filling a creative brief with all this information.*
- *This will help you steer your project in the right direction by providing information about the restaurant, what you are trying to communicate and who is going to visit the website.*
- *Set aims, objectives and design specifications.*

1b. Plan the website layout

Before designing the website layout you should have at least three distinct elements:

- *The header. It will include the banner. Also links and buttons page etc.*
- *The body. It will include the text and pictures or graphics.*
- *The footer. It will include copyright and the 'back to top' link.*

Add controls and labels | Add code to save images | Add code to save new records

3. Add functionality to the "Save the movie" button

To insert new records directly into a database you can use the TableAdapter.Insert method. Therefore you should:

- *Call the TableAdapter's Insert method, using the values for each column as parameters. For example, the following instruction passes the fields "Genre", "Title" and "Poster" as parameters :*

```
All_movies.MoviesTableAdapter.Insert(Me.ComboBox1.SelectedValue, Me.TextBox2.Text, Img)
```

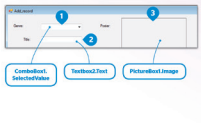
- *For the "Poster" field, you have to convert the image to a byte array in order to yield it into the*

```
Dim converter As New ImageConverter
Dim img() As Byte = converter.ConvertTo(PictureBox1.Image, GetType(Byte()))
```

- *Fill your Dataset with data: calling the data adapter's Fill method. Add the appropriate code where you need to load a data table with data. For example, below the TableAdapter.Insert method you can add the following:*

```
All_movies.MoviesTableAdapter.Fill(List.Movies_applicationDataSet.Movies)
```

- *Now clear the input fields. We want the user to start with clean fields before typing the new record. For example, you can clear the field "Title" giving the instruction* `TextBox2.Text = ""` *You can also set the minimum allowed value for the spin box.*
- *Finally, add a message box in your form like "Movie added succesfully".*
- *And now it's time for some more testing. Try to use the application as a normal user but also as someone that doesn't follow the rules or wants to create a problem to the system.*



SMART TIP

Use the TableAdapter.Insert method when your application uses objects to store data, or when you want fine control over creating new records in the database.

Project 6

Search with a filter

The scenario

The user needs to search for a particular movie or navigate between the best movies based on their IMDb rating.

The suggested way

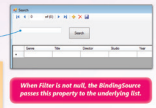
First, you have to create a search form. You should add the proper controls and labels into this form and also the required functionality. Then, create another form where the user will be able to navigate between a set of records based on specific filter criteria.

1. Display a particular record | Create a query | Display a set of records

1a. Create a "Search" form

Create a new form where the user will be able to search for a particular movie giving the movie's title.

- *Add a new Windows form. Give a relevant name to your form.*
- *Add a "Search" Button control and a Textbox control where the user will type the title of a movie in order to display the relevant records.*
- *Drag and drop the table "Movies" from the Data Sources Window onto the form to create a DataGridView control.*
- *Select which fields (columns) you want to be displayed in your DataGridView control.*



1b. Add the functionality

- *What happens when you click the "Search" button? Use the Filter property that allows you to view a subset of the DataSource. For example try the following code:*

```
Me.MoviesBindingSource.Filter = "[Title]='" & Me.searchTextBox.Text & "'"
```

- *Add the form to your menu with the correct code in the proper control.*

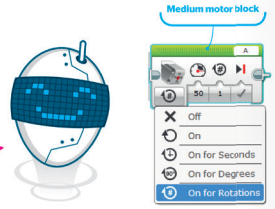
When Filter is not null, the BindingSource passes this property to the underlying list.

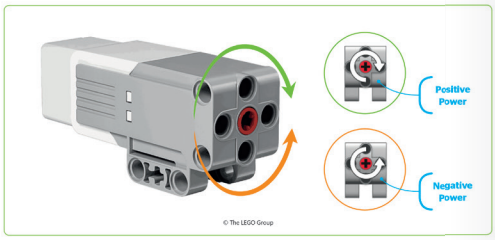
TASK 2

Robotic arm and calculations

In addition to moving or detecting obstacles and colors, the Mindstorms EV3 robot can also lift or carry obstacles. For this purpose, it has a loader that is controlled by its Medium motor. We can program the Medium motor using the Medium motor block.

When you use the Medium Motor block, the power value can accept numbers between -100 and 100. A positive number rotates the Medium Motor clockwise, while a negative number rotates it in a counterclockwise direction.





BE SAFE

The speed of rotation of the medium motor is approximately proportional to the power level, although the speed of rotation can also be affected by the amount of load placed on the engine.

Before starting to program, make sure that the medium motor and the loader are connected to the front of the driving base and that its wire is connected to port A.

Front view

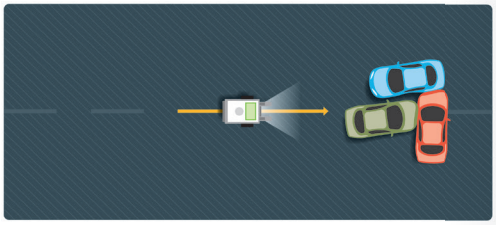
When you use the Medium Motor block to control the loader of the robot, a positive power value means that the loader will move up. On the other hand, a negative power value means that the loader will move downward.

Detect and clear the road of obstacles

Let's suppose that your robot wants to take a trip, driving on a highway for hours. So, you can program it to move forward at a steady speed, but what happens if there is a car accident blocking on the road and the robot can not cross?

Knowing that the robot has an ultrasonic sensor and a loader, we can program it to react if it detects obstacles that are in front of it. More specifically, we can make the robot check for obstacles closer than 15 cm and, if it detects any vehicle crashed in front of it, it picks it up with its loader to clean the road, so it can continue its way forward.

To make the robot move forward along the road at a constant speed, you are going to use the Move Steering block, with power equal to 30.



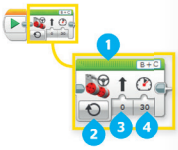
Make the robot move forward:

> From the Action palette add the **Move Steering** block. 1

> Set **Mode** to **ON**. 2

> Set **Steering** to **0**. 3

> Set **Power** to **30**. 4



SMART TIP

In the constructions of EV3, the medium motor usually serves as a robotic arm. In our construction, the loader can be considered a robotic arm.

While the robot is moving forward, it has to keep looking for obstacles. For that purpose, you will use the Wait block and adjust the Mode of the Ultrasonic Sensor. The robot will continue moving forward until it detects an obstacle closer than 15 centimeters.

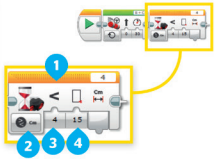
Make the robot detect an obstacle:

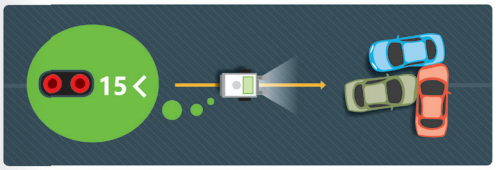
> From the Flow Control palette add the **Wait** block. 1

> Set **Mode** to **Ultrasonic Sensor-Compare - Distances in Centimetres**. 2

> Set **Compare Type** to **4**. 3

> Set **Threshold Value** to **15**. 4





If you want the robot to make some decisions, you can use the Wait block or the Switch block. With the Wait block we make the program wait for something to happen before continuing with the next block of the sequence, while with the Switch block we can add two or more sequences of programming blocks.

SMART TIP

The ultrasonic sensor works best when it has to detect sound waves that are reflected off objects with hard surfaces. Soft objects such as clothes can absorb these waves and that means they will not be detected. It is also more difficult to detect objects with rounded or angled surfaces.

Import the data to the computer:

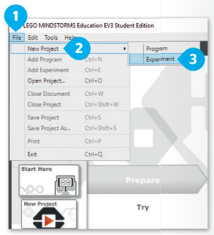
> In the tab **File** 1, choose **New Project** 2 and then click **Experiment**. 3

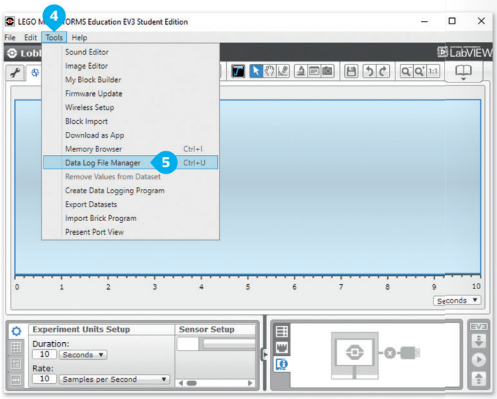
> In the **Tools** tab 4, choose **Data Log File Manager**. 5

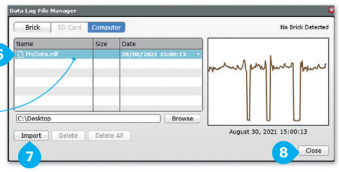
> Choose the file **MyData** 6 and click **Import**. 7

> Click **Close**. 8

> The data diagram will appear in the experiment window. 9







You can import this file into a spreadsheet program in order to analyze data in a more advanced way.

Here are three indications of pollution hotspots.

As we mentioned at the beginning of the lesson, when the robot scans a homogeneous area we expect to see homogeneous data values of the reflected light intensity, however a value that is extremely low might indicate a possible pollution hotspot. In the scanned area, the reflected light intensity values are for the most past stable between 60% and 70%. However, there are three places where the sensor registered values between 0% and 10% and this means that these places might be pollution hotspots.

TASK 2

Pollution-detecting robot

When we talk about soil pollution we refer to the land degradation or other alterations in the natural soil environment. It is typically caused by industrial activity or improper disposal of waste.

Currently, the detection of soil pollution is done by manual surveys which are inefficient, unreliable, time consuming and very expensive. The need for a better and low-cost technical approach, that would be able to assess contaminated areas before programing the required remediation actions, has become more intense due to public awareness of environmental issues. Robots can provide an adequate sollution to this problem, scanning large areas with their sensors to show us if there are hotspots of pollution or not.

Soil pollution hotspots

A soil pollution hotspot is a location with a high level of pollution. When a sensor scans a homogeneous area, the data of the reflected light intensity is expected to be homogeneous as well. If the sensor registers a value that is extremely low or extremely high in comparison with all the other observed values, this means that the specific part of the scanned area where the different values have been registered, might indicate a possible pollution hotspot.

HISTORY

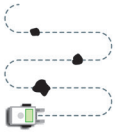
Pollution has accompanied humankind ever since groups of people first congregated and remained for a long time in any one place. Indeed, their waste is often recognizable in ancient human settlements – shell mounds and rubble heaps, for example.

Program a robot to detect pollution hotspots

Open Mindstorms EV3 and start a new project. The robot will be programmed to scan a specific surface and detect potential soil pollution hotspots. More specifically, create a program in order to make the robot:

> Scan the area with the Color Sensor, following a specific route.

> Collect Reflected light intensity data while scanning the area.



For our experiment we will need:

> the Color sensor, which will collect Reflected light intensity data.

> the Datalog Experiment window in order to import and analyze the selected data.

Before you start programming make sure that the color sensor is connected to the front right part of the driving base facing downward and that its wire is connected to port 3.

Attach the co the front part base, facing

Create the robot's route

Let's start creating our program. First of all we have to create the route that the robot will follow, scanning the study area. The robot will move forward at 40% power for 2 seconds.

Move forward:

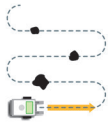
> Add the **Move Steering** block 1

> Set **Mode** to **ON for seconds** 2

> Set **Steering** to **0**. 3

> Set **Power** to **40**. 4

> Set **Seconds** to **1**. 5



Then, the robot will turn left (-30) at 50% power for 3 seconds

Move left:

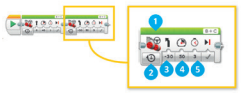
> Add the **Move Steering** block 1

> Set **Mode** to **ON for seconds** 2

> Set **Steering** to **-30**. 3

> Set **Power** to **50**. 4

> Set **Seconds** to **3**. 5



The robot will move forward at 40% power for 2 seconds.

Move forward:

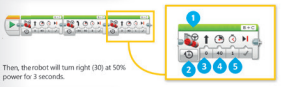
> Add the **Move Steering** block 1

> Set **Mode** to **ON for seconds** 2

> Set **Steering** to **0**. 3

> Set **Power** to **40**. 4

> Set **Seconds** to **1**. 5



Then, the robot will turn right (30) at 50% power for 3 seconds.

Turn right:

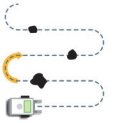
> Add the **Move Steering** block 1

> Set **Mode** to **ON for seconds** 2

> Set **Steering** to **30**. 3

> Set **Power** to **50**. 4

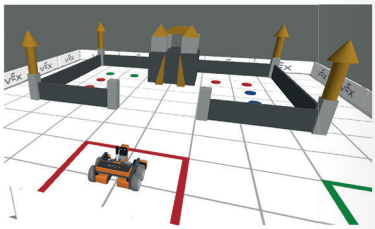
> Set **Seconds** to **3**. 5



1.1

Lesson 1
Virtual robots

It is good to have a robotics kit to build and program new robots, but if you don't, you can always use a virtual robotics toolkit to build, program and simulate your robot. Virtual robotics involves simulated robots used to generate programs for robots. Simulation is an important way of learning how physical concepts like force and motion work in real life.



Advantages of using Virtual Robotics

- Little/no risk of damaging the equipment.
- Faster trial and error method.
- Use components that you don't have to create more advanced robots.
- Lower/no cost because most of the virtual robotics tools are free to use.
- Sometimes more enjoyable because of the terrains that you can use.
- Sometimes more enjoyable because you can use different robots.
- Suitable for different learning styles. Some students can gain a better understanding.

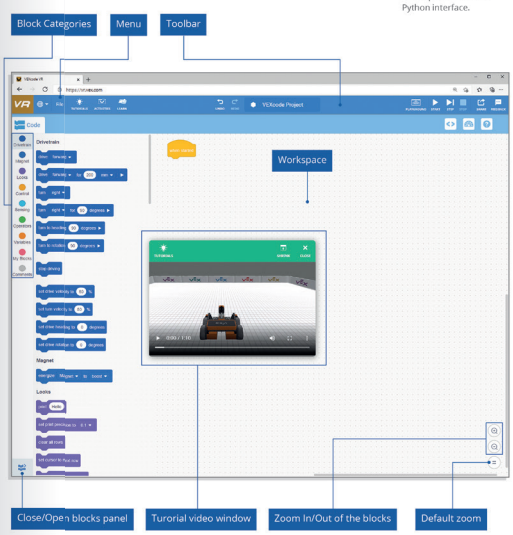
1.1

VEXcode VR

VEXcode VR is a block-based coding platform powered by Scratch Blocks that allows you to code a virtual robot. Due to VEXcode VR's simple interface, you can create your own program without writing complex code. The only thing you have do is to drag blocks into the workspace and link them together, just like Scratch blocks.

Go to **https://vr.vex.com/** and explore VEXcode VR.

Go further!
In VEXcode VR, you also have the opportunity to code by using a custom-developed Text-based Python interface.

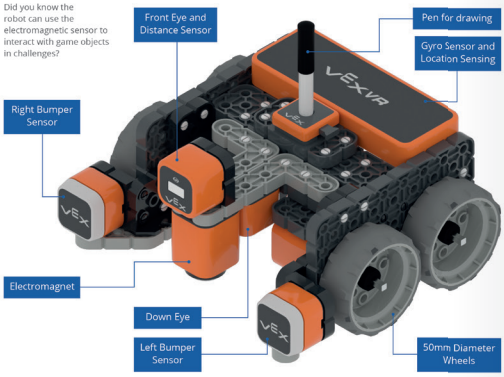


1.1

Geek talk
Did you know the robot can use the electromagnetic sensor to interact with game objects in challenges?

VEXcode Virtual Robot

In your projects, you will use a virtual robot that is pre-built. It has wheels so you can move it around, it has a lot of different sensors mounted on it to interact with the environment and a pen so you can draw lines or shapes on the playgrounds.



How to create a program

In VEXcode VR, you can create programs by using blocks or Python code. In this unit, you will only create programs using blocks.

Let's take a look to see the coding options when you use VEXcode VR.

Coding in VEXcode VR

You can code in three different ways in VEXcode VR.

1. **Blocks**: Block-based coding powered by Scratch Blocks.
2. **Blocks + Text**: Creating code with Blocks while you can see the corresponding Python code generated in real time with the Code Viewer.
3. **Text**: Text-based coding with Python and even dragging and dropping predefined code lines.

1.1

Block categories

There are a variety of programming blocks that you can use to create a program.

Each of them is color-coded, and all of them are grouped into block categories according to their type and use. Let's have a look!

Operation	Block category
Controls the movement of the robot on the playground.	Drivetrain
Used to capture disks on specific playgrounds.	Magnet
Used to control the Print Console and the pen of the robot.	Looks
Controls the flow of the program.	Control
Used for reading the sensor values of the robot	Sensing
Contains various math and logic operators.	Operators
Used to create new variables.	Variables
Used to create your own unique blocks.	My Blocks
Used to add comments to code.	Comments

Bits 'n' tips
A program can be executed by pressing the start button on the toolbar or the start button in the playground.

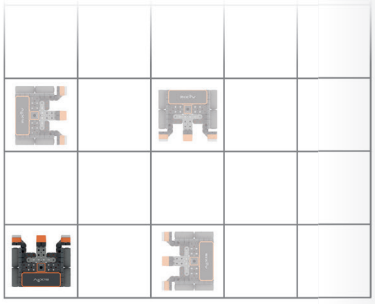
Syntax inspector
The blocks are interconnected and executed by the robot according to their order. This concept is known as "sequential operation." When running the program, only blocks that are connected to each other are executed.

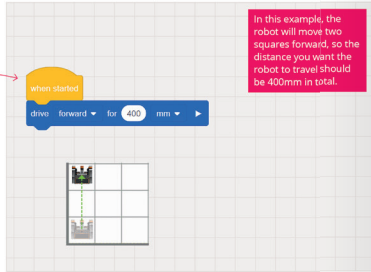
1.2

Create a program

You will use the **Grid Map** in the **Playground** which is a good playground to understand how to program the robot to move. You want your robot to move from point A and form a 3x3 square like the one in the picture. In order to do that, you will use blocks from the **Drivetrain** category.

Do you remember?
Remember, each square of the Grid Map has 200mm sides.





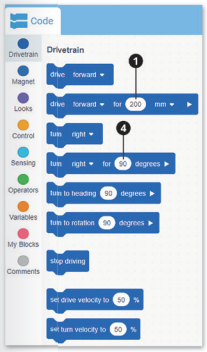
1.2

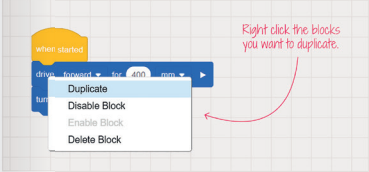
First you will create one side and corner of the square.

Side and corner:

> From the **Drivetrain** category, click the **drive for** block, ❶ drag and drop it after the **when started** block ❷ and set the distance to 400. ❸

> From the **Drivetrain** category, click the **turn for** block, ❹ and drag and drop it after the when **drive for** block. ❺





Don't forget!
To save time, you can duplicate the blocks. Remember, a square has 4 equal sides and corners.

1.3

Loop commands

Sometimes we want a program to execute the same lines of code several times. When we need to execute the same commands more than once, we use the loop commands.

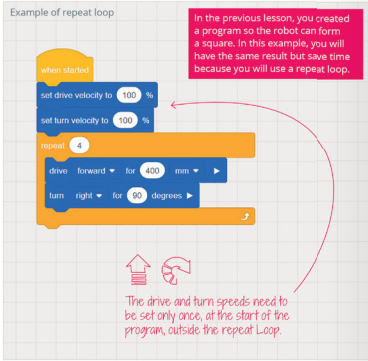
The most frequently used loop commands in VEXcode VR are the **repeat ( )**, **forever**, **repeat until ( )** and **while ()** blocks. They belong to the orange Control blocks category, and they control the flow of the program.

Loop blocks in VEXcode VR

The repeat () block

In this lesson, you will see how the repeat () block works. The **repeat** () block will execute the blocks inside it a specific number of times.

Example of repeat loop



Don't forget!
A square has four equal sides and four 90° angles.

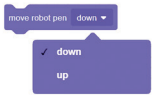
1.3

Draw shapes

To have a better view of what shape the robot draws, you can use the pen. The pen goes through the center of the robot of the robot, and you can use it to draw while the robot moves. The blocks you can use to draw are the **move robot pen ()** and **set robot pen to color ()**. Both of these blocks belong to the purple **Looks** blocks category.

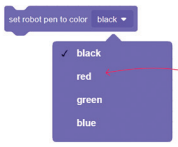
Move robot pen ()

can be used to move the pen tool down so the robot can draw on the playground and up to stop drawing. Imagine you have a real pencil, to write something, you need to move it down and start moving your hand and when you move it up, it stops writing.



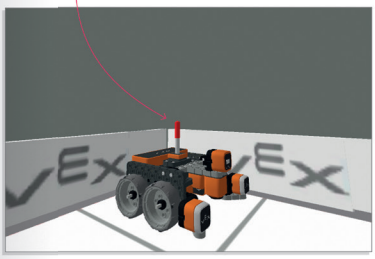
Set robot pen to color ()

can be used to change the color of the pen.



For example, if you use the chase camera and set your robot to write in red, you will see that the pen through the center of the robot has changed color.

Choose between four different colors



CODING AND ROBOTICS | K12 RECOURSES

binarylogic

CYPRUS FRANCE GREECE POLAND UK USA

e-mail: info@binarylogic.net | Internet: www.binarylogic.net

binarylogic

Coding Robotics

K-12 Resources

binarylogic.net