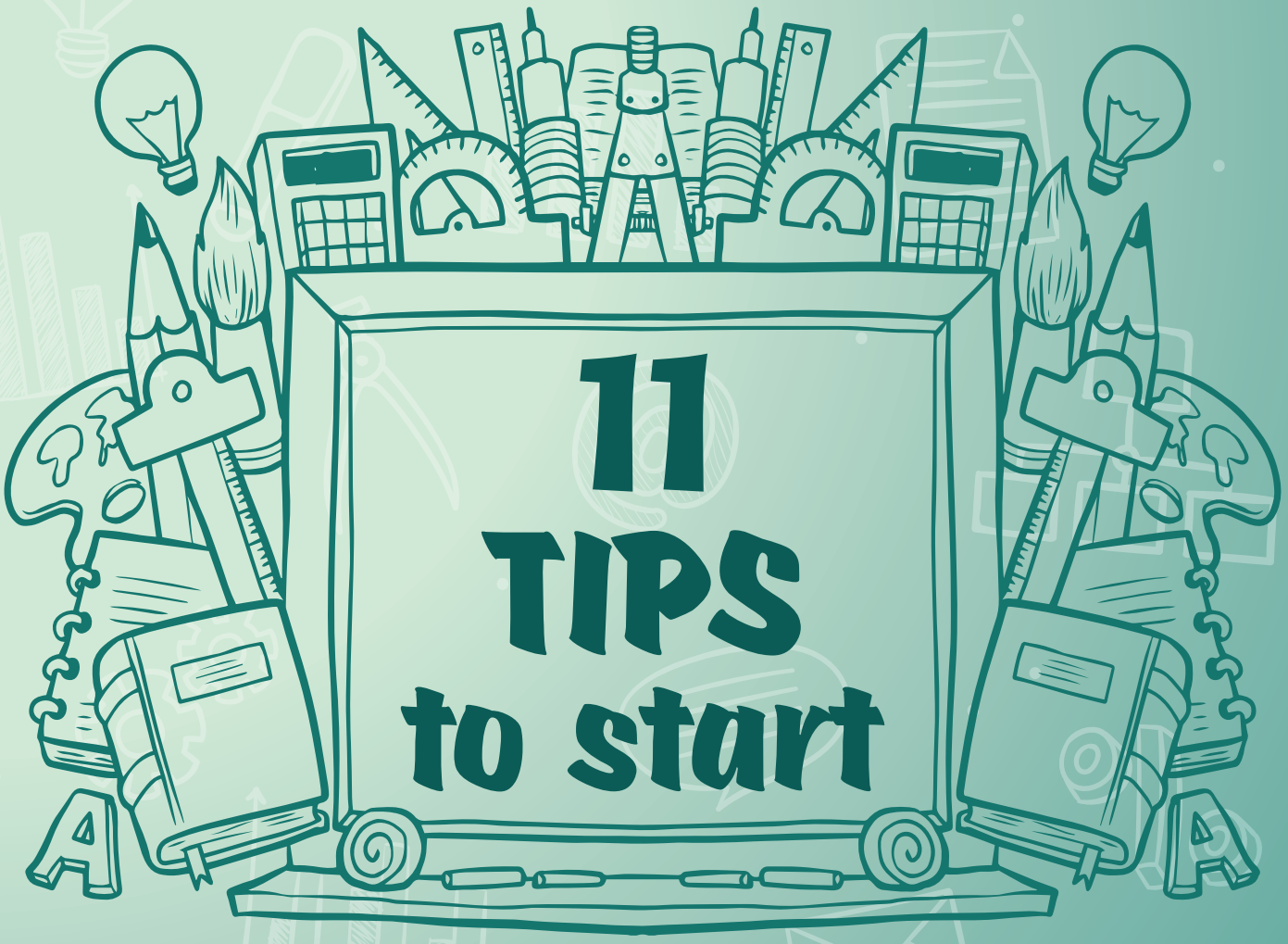


# 11 TIPS to start CODING & ROBOTICS at your school

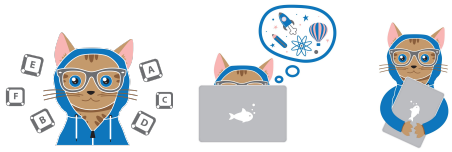
*for teachers*

Emily de la Peña

# 11 TIPS to start CODING & ROBOTICS at your school

*for teachers*

Emily de la Peña



# About the Author

Emily de la Pena, Founder of Coding Kids (www.codingkids.com.au) and Advance Queensland's Community Digital Champion, is developing the next generation of coders, inventors and change makers.

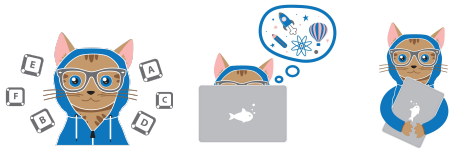
After spending over a decade building a successful career in civil engineering, Emily now enjoys sharing her passion for science, technology, engineering, and mathematics (STEM) with the next generation. She balances her time between empowering children to become technology creators and engaging with teachers and librarians to integrate the new Digital Technologies subject (i.e. computational thinking, design thinking, coding, and robotics) into their classrooms and schools. Emily is also a mentor with CoderDojo, where she teaches children to code.

Emily's organisation, Coding Kids, runs in-school coding classes, after-school coding clubs, school holiday code camps, and professional development workshops for educators. At Coding Kids, children build their own computer games, animation movies, digital solutions, and interactive digital art. Children discover computer programming concepts, algorithmic thinking, debugging, logic, problem solving and design through fun and play.

Emily also judges several robotics and coding competitions in Queensland: Robocup Junior, Young ICT Explorers, and First Lego League.

Email hello@codingkids.com.au to bring Coding Kids to your school, and explore the resources below to learn more about Emily de la Pena and how Coding Kids can help introduce coding and robotics to your school community.

1. Emily's interview on 612 ABC Radio Brisbane: https://soundcloud.com/612abcbrisbane/emily-de-la-pena-coding.
2. Emily on Twitter: https://twitter.com/EmilyFdelaPena.
3. Coding Kids on Twitter: https://twitter.com/coding_kids.
4. Emily's LinkedIn profile: https://www.linkedin.com/in/emilydelapena.
5. Advance Queensland's Community Digital Champion profile: http://godigitalqld.dsiti.qld.gov.au/EmilyDeLaPena.
6. Leanne Enoch MP's Vine video of Emily receiving her award as Advance Queensland's Community Digital Champion: https://vine.co/v/iY9137lOz1e.
7. CoderDojo Mentor profile: https://coderdojobrisbane.com.au/about/our-mentors/.



# Table of Contents


**Benefits of Learning to Code** **4**

**1. Learning Progression** **7**

**2. Unplugged Activities** **11**

Place the Flag on the Target 11

Sorting Data 12

**3. Coding and Robotics** **17**

**4. Scratch** **19**

Leap over the Frog Game 20

Forest of Danger Game 22

Dance Party Animation 24

Tips on Delivering these Classes to Students 25

**5. Python** **26**

**6. Lego Mindstorms** **28**

Easy Challenges 29

Intermediate Challenges 29

Advanced Challenges 29

**7. Timetables and Group Sizes** **30**

Fitting coding and robotics into a school timetable 30

**8. Other Gadgets to Use** **31**

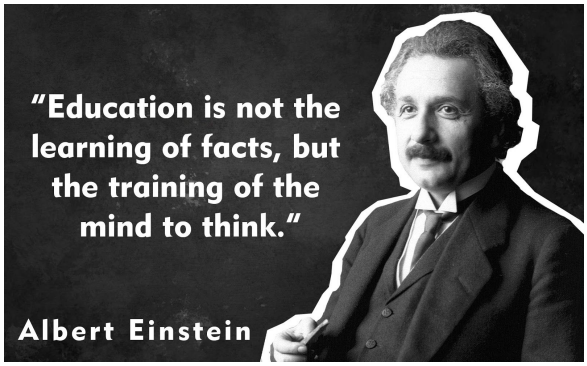
**9. Coding, Robotics, and Invention Competitions** **38**

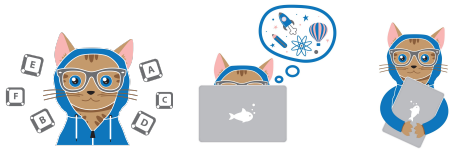
**10. School Tech Excursions in South East Queensland** **39**

**11. Where to Find Help** **40**

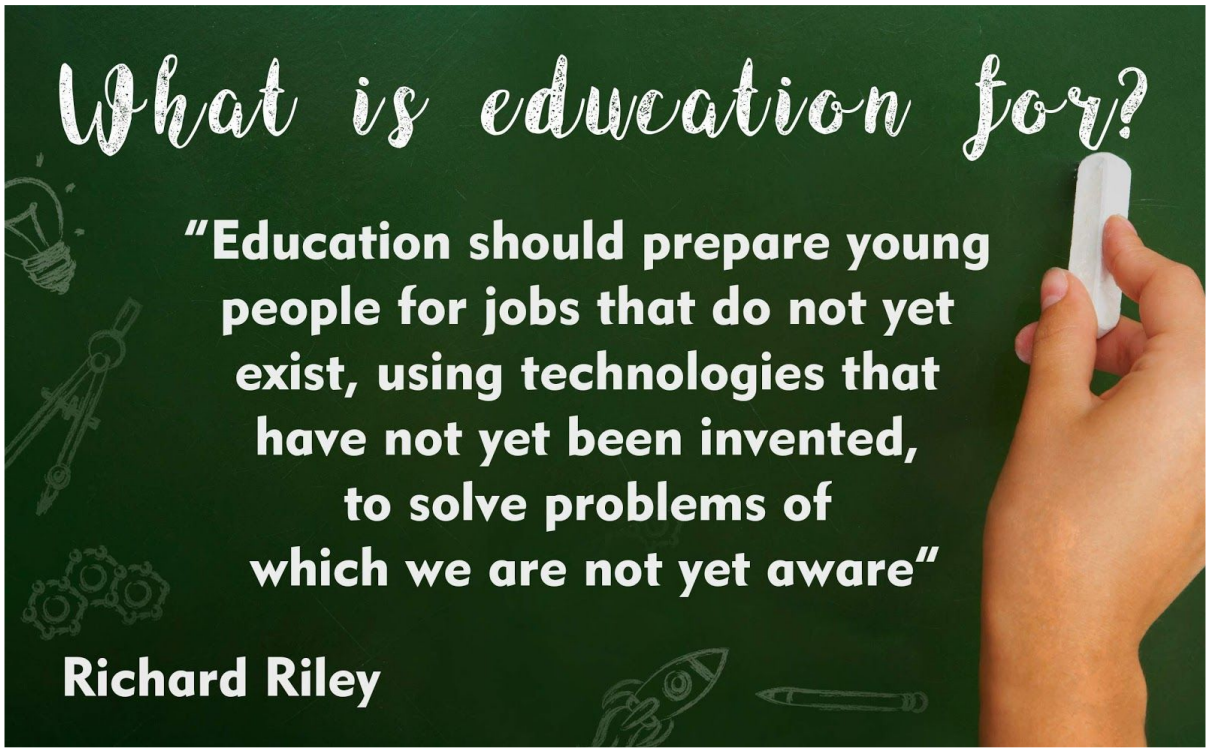
Coding Kids can help 40

**Appendix A - Teaching Python to Beginners** **A**



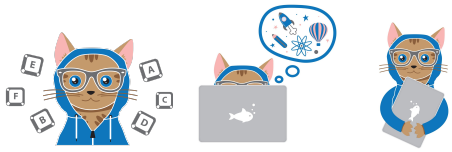
# Benefits of Learning to Code




Coding, also known as computer programming, is a process of writing commands that tell computers to execute and complete certain tasks. These commands, or instructions, are computer programs. We use computer programming languages to communicate our instructions to the computer. As our world becomes increasingly dependent on technology, coding is a vital skill for children to learn.

Teaching coding introduces a new dimension of learning in a school environment. Coding is a fun and engaging tool that encourages a growth mindset and creates an authentic learning experience. In addition, technological advancement impacts industries and is changing the nature of jobs, so it is critical for children to develop the technical skills that will prepare them for the workforce. Digital literacy, and specifically computational thinking, is on its way to becoming a basic and vital skill in an increasingly digital world.

“Growth mindset” is based on the belief that a person’s abilities and intelligence develop from effort and perseverance and are not fixed. Learning to code, unlike a traditional academic environment, encourages a growth mindset. In a traditional academic environment, a student is given a mark based on the number of correct (and incorrect) answers which encourages students to feel that their intelligence is fixed and based on the results of their assessments.

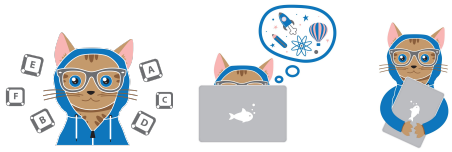


Learning to code is based on a process of trial and error. A mistake does not determine the student's mark; rather, the mistake is a stepping stone to finding the correct answer. Whether you are a professional programmer or a beginner, trial and error is the foundation of your learning process. It does not matter that you do not know the correct answer now; what matters is that you persevere through failed attempts and continue to make your way towards finding the correct answer.

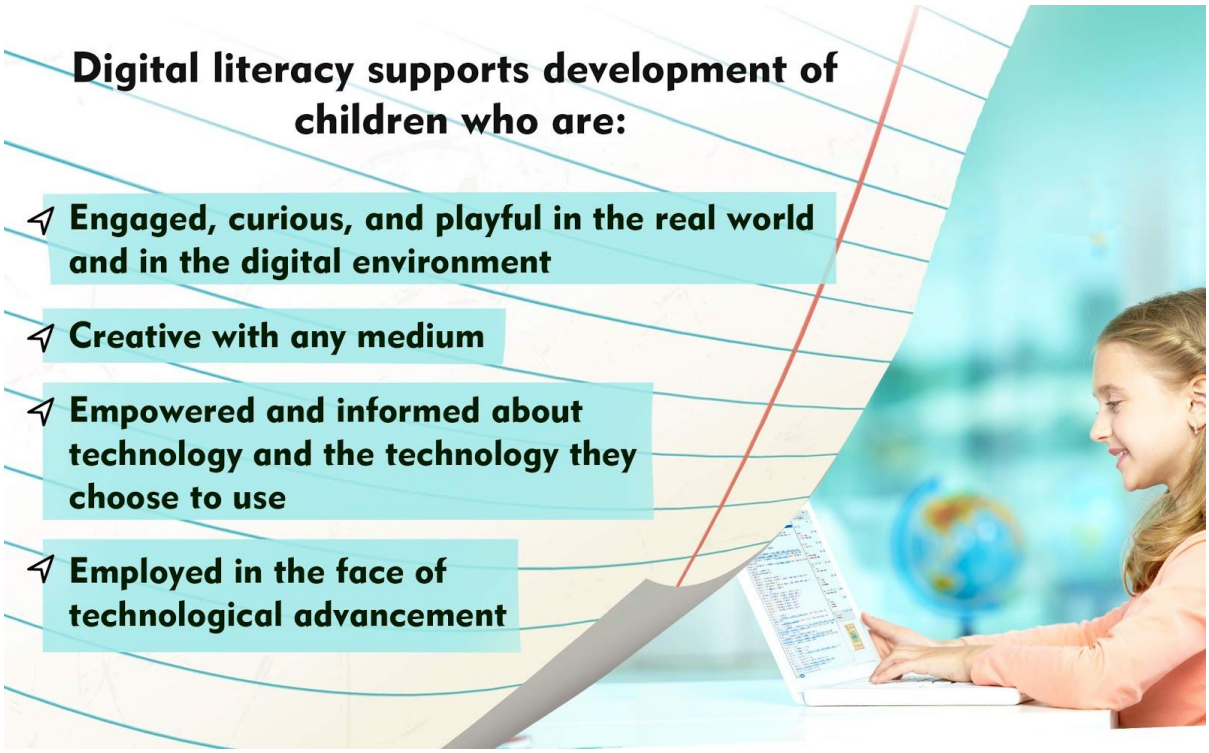

Coding class in Brisbane

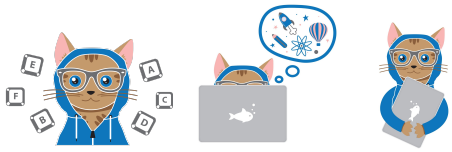
Coding is a fun and helpful tool that creates an authentic learning experience. Rather than researching about the theory and history of computer programming and computational thinking as an academic task (e.g. by writing a school essay), students can actually write their own code and create their own computer programs to achieve specific tasks. Students are able to personalise projects and develop digital solutions to problems that are meaningful to them. Learning to code can easily be project based. You can start the learning process by building simple programs that create animations, experiment with geometries, and develop simple computer games.

Coding and robotics needs to become an integral part of the school curriculum as technological advancement changes both industry and the nature of jobs and with artificial intelligence becoming a greater factor in our world. There are increasing signs that artificial intelligence has already been invented and could replace or significantly alter the jobs of doctors, lawyers, accountants, journalists, psychologists and music composers. As Marc Andreessen, an American entrepreneur, investor, and software engineer, once said, "Software is eating the world."



Teaching children by using a learning process founded on growth mindset, authentic learning and digital literacy can support their optimal development. Learning coding and robotics requires learning from trial and error which supports development of a growth mindset. Creating digital solutions gives students an authentic learning experience by building real world projects with real impacts. Digital literacy and specifically technology creation focuses on teaching empowering skills which allow students to create, innovate and express themselves.





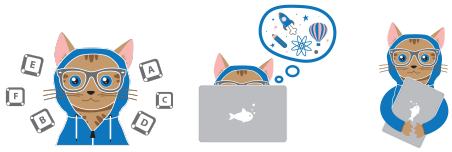
# 1. Learning Progression

Teachers who are considering the Digital Technologies curriculum or wonder how to introduce coding and robotics at school often ask where to start. That will be covered in the next section. The more important question is, what are we aiming for?  Where will the the learning pathway take our students?.


Technology creation vs Technology consumption

Digital literacy includes skills in both technology consumption and technology creation. These two terms refer to very different skillsets but are often combined under the umbrella term of “digital literacy.” “Technology consumption” refers to the use digital devices such as smart phones, tablets and computers, and the use of software such as spreadsheets or presentation slides. “Technology creation”, on the other hand, is the knowledge of how to create digital or technological solutions - for instance, developing a computer program or mobile app.

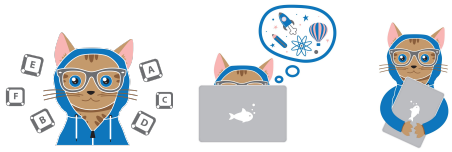
Digital literacy education based on technology consumption is outdated. Teaching children, that is digital natives, how to create powerpoint slides or word processing software is becoming more irrelevant. The trend for software development is to create user-friendly software, otherwise a competitor software will become the preferred one of choice. The need to learn how to use software will become more obsolete over time. This is even more true for “digital natives”, children who were born into a digitally connected world to whom digital devices are a natural means of communication. “Digital immigrants” are adults who grew up in a world where connected devices either did not exist or were not ubiquitous.



Digital immigrants require more thorough training to learn and adopt new technology, however digital natives have grown up in a device rich environment.

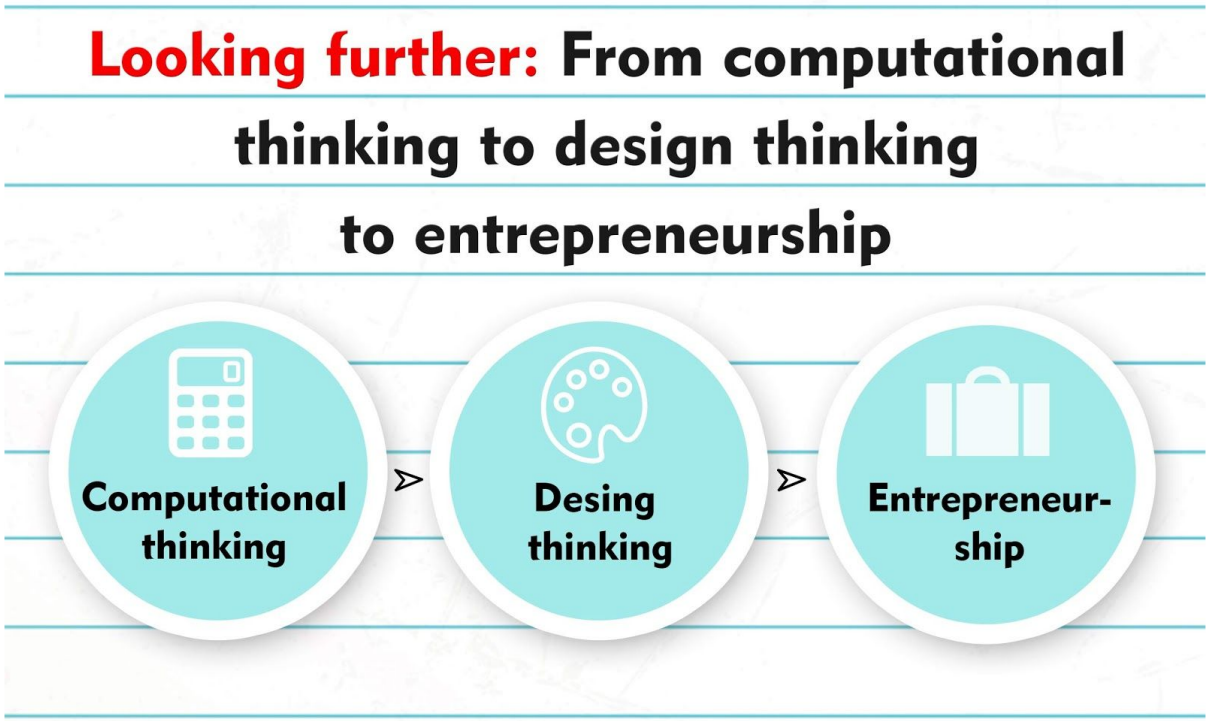

Technology creation vs Technology consumption

Coding and robotics (the use of computer-controlled machines that can sense their environment and make real-time decisions to perform manual tasks), and more broadly, computational thinking (a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science), can and should be integrated into current subjects: science, mathematics, English, humanities and social sciences, and the arts.



The progression of learning associated with coding and robotics can be summarised into three areas:

1. Computational thinking.
2. Design thinking.
3. Entrepreneurship.

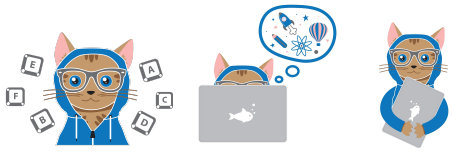



Looking further: From computational thinking to design thinking to entrepreneurship

Computational thinking is a problem solving process that involves defining a problem and then outlining a solution in a series of ordered steps that a human or machine can effectively carry it out. An example is the use of an algorithm to solve a problem. Computational thinking supports computer programming and software development, but it can also be used as a problem solving approach across disciplines including mathematics, sciences and the humanities.

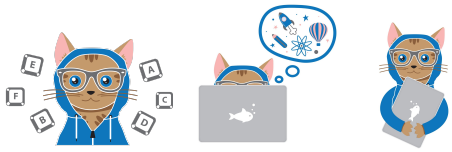
Design thinking is a robust process of problem solving that focuses on creating a better future and finding desirable solutions for the end user. The process involves gaining an understanding of the problem from the user's perception, and it draws on logic, imagination, intuition, iterative development, and systematic reasoning to explore possibilities of what could be. Design thinking then uses these possibilities to create desired outcomes that benefit the end user.

Entrepreneurship brings these puzzle pieces together to extend problem solving, develop solution into a marketable product, and create a viable business model as a means of delivering solutions to the market place.



Digital literacy, in terms of technology creation, encourages confidence in experimenting, innovating, and inventing. It starts from a technical perspective that is developed by building simple computer games and animations and grows into an entrepreneurial perspective by solving community problems and making the product or solution broadly accessible.

The new Digital Technologies subject aims to teach students digital literacy in the context of the current subjects. Digital Technologies is not a stand alone subject but rather it is to be integrated into existing curriculum such as English, and humanities and social sciences.



# 2. Unplugged Activities

Unplugged activities - activities that do not involve technology - can teach students computational thinking. Let's look at two examples:

1. Place the flag on the target.
2. Sorting data.

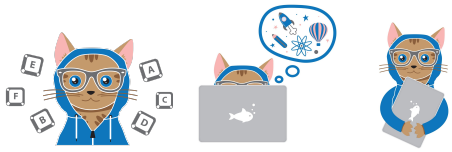
## Place the Flag on the Target

This activity achieves Content Descriptor: Years 3 and 4, Digital Technologies Processes and Production Skills: Define simple problems and describe and follow a sequence of steps and decisions (algorithms) needed to solve them. The objective and context can be changed so that the learning outcome is integrated with a current subject, e.g. English or humanities and social sciences.

This is a group activity for groups with 2-4 members. Each group of students receives a pack of cards with instructions. Examples of instructions on each card would be "pick up flag", "walk __ steps", or "do the chicken dance".

Example instructions for a pack of 12 cards


The goal of the activity is to use the instruction cards to create a sequence of steps that achieve the task of taking a flag from the group's home base to a specified target location in the room; after placing the flag at the target, the team members must then return to the home base.



Once the groups think they have created an effective sequence of actions that will allow them to complete the task, they demonstrate the instructions by acting them out and can then see whether the instructions are sufficient to achieve the task.


Diagram of an unplugged group activity: Place the flag on the target

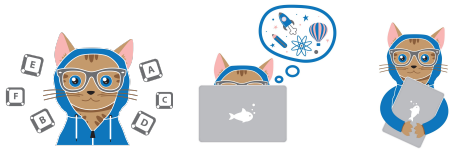
## Sorting Data

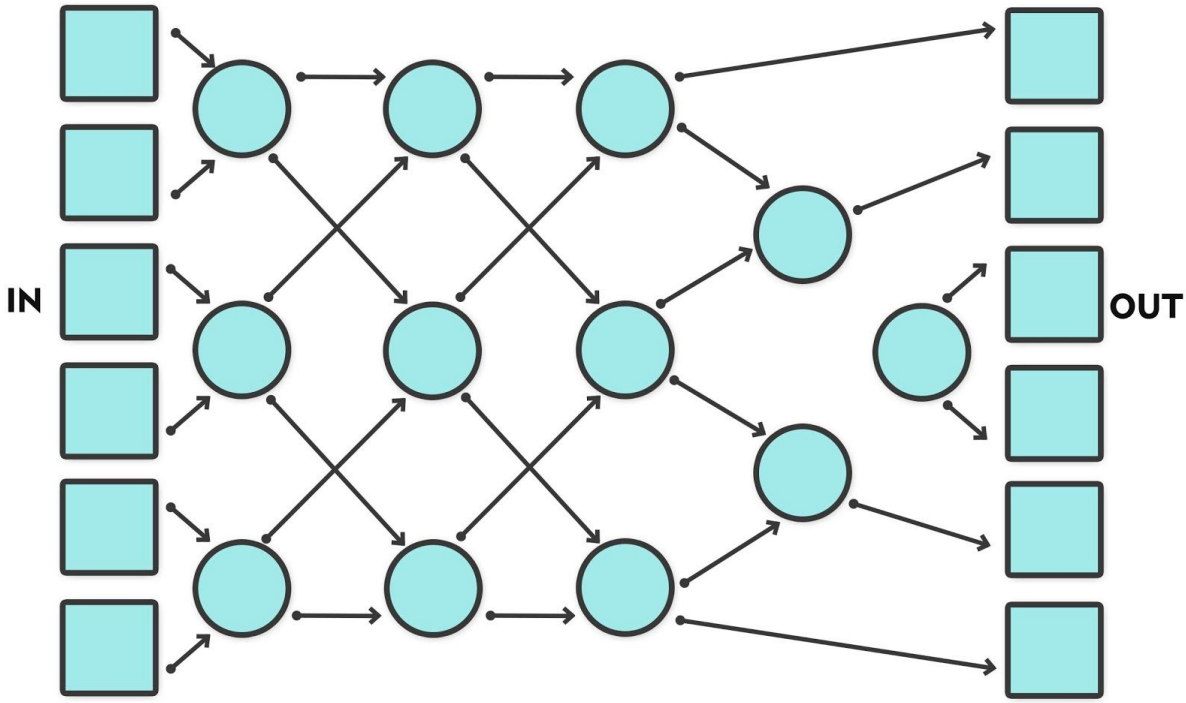
This activity achieves Content Descriptor: Years 5 and 6, Digital Technologies Processes and Production Skills: Design, modify and follow simple algorithms represented diagrammatically and in English involving sequences of steps, branching, and iteration (repetition). This activity is best aligned with the maths subject curriculum.

This is a group activity that can start with a group of six students; with more organisation and coordination, it can also work with much larger groups, for instance, 20 students.

Procedure for a group of six students:

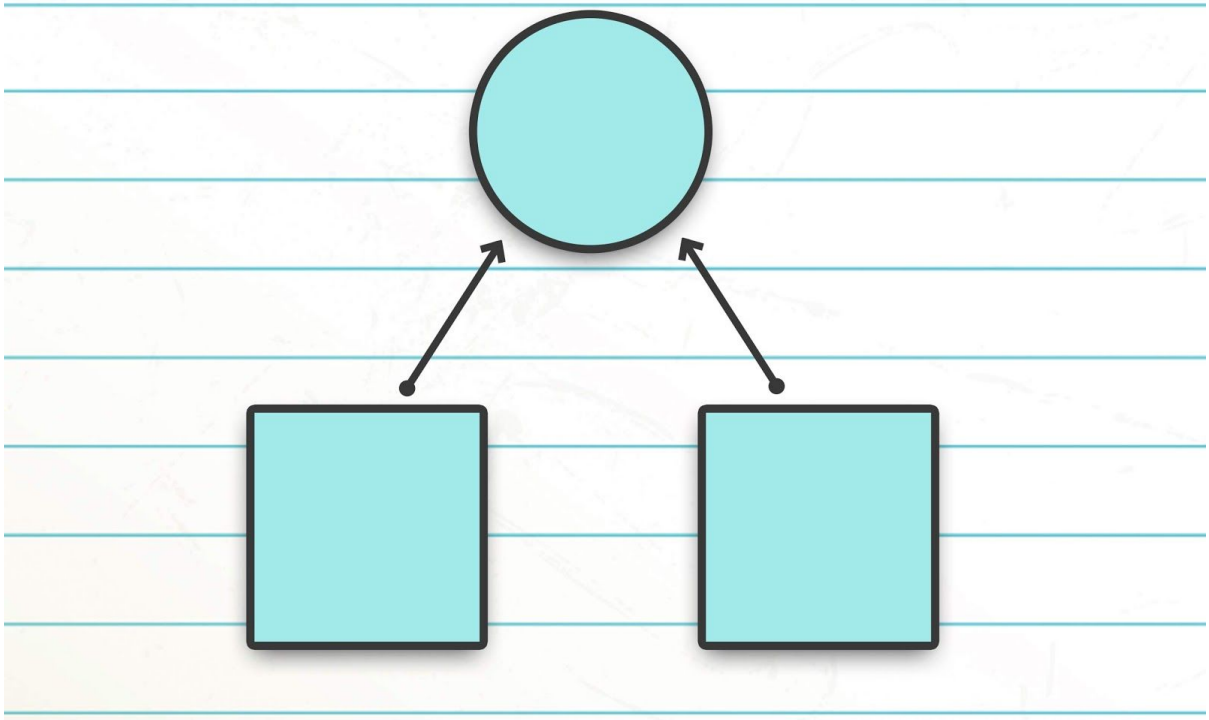
1. Each number from one to six is represented by a single student.
2. The students stand in a line in random positions.
3. A sorting network is drawn on the floor to show that each pair of students/numbers will step forward to meet at a node in the network. (Students will pair off from left to right.) At the node, a decision is made. The lower number steps forward and to the right (of the camera view) and the higher number steps forward and to the left (of the camera view).
4. Step 3 iterates until students reach the opposite end of the sorting network (see diagram on next page).
5. The six numbers are sorted from highest to lowest (left to right).






A sorting network to sort six numbers is drawn on the floor (Credit: http://csunplugged.org/sorting-networks/)


Students begin by lining up in random order at the input end of the sorting network (Credit: http://csunplugged.org/sorting-networks/)



Each pair of numbers steps forward to a node in the network. A decision is made at the node (Credit: http://csunplugged.org/sorting-networks/)


A pair of numbers meet at the node. The lower number will step forward and to the right (of the camera view) and the higher number will steps forward and to the left (of the camera view). (Credit: http://csunplugged.org/sorting-networks/)

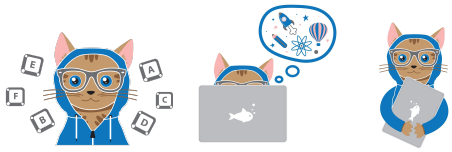

The lower number has stepped forward and to the right (of the camera view) and the higher number has stepped forward and to the left (of the camera view). Each pair of numbers follows the same process. (Credit: http://csunplugged.org/sorting-networks/)


After several iterations, the numbers are finally sorted by highest to lowest from left to right. (Credit: http://csunplugged.org/sorting-networks/)

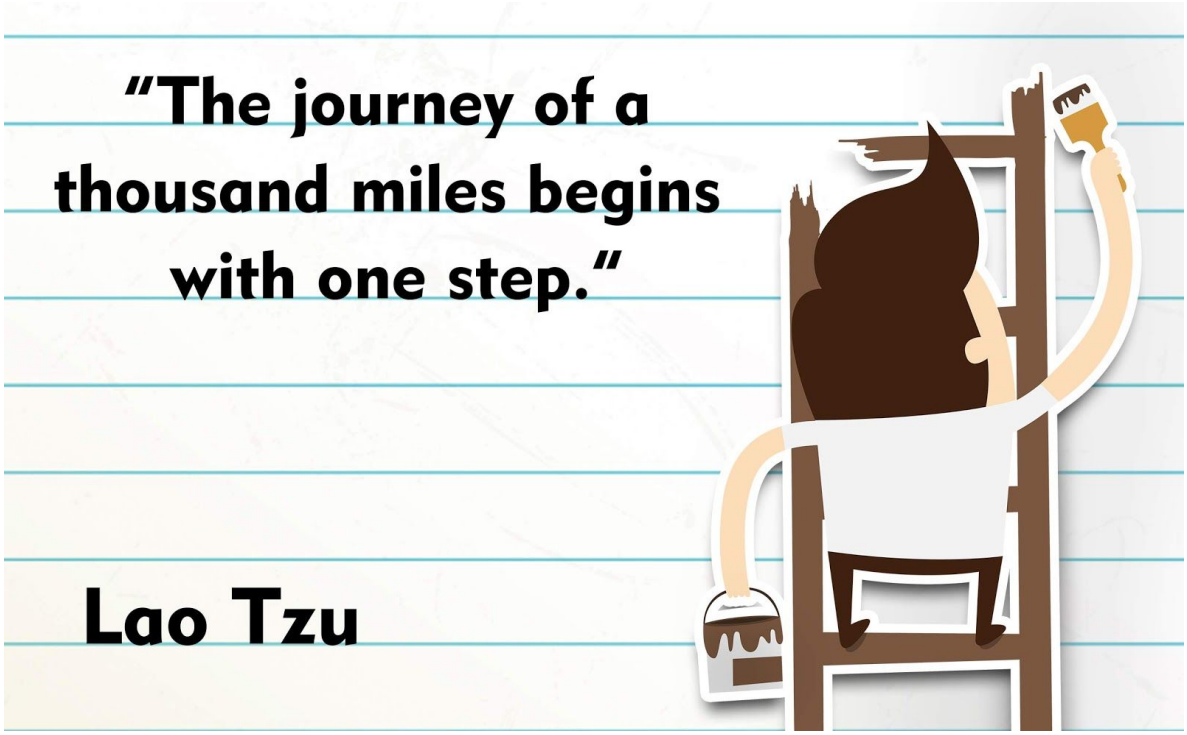

Try the same process with larger numbers to add more of a challenge. (Credit: http://csunplugged.org/sorting-networks/)

Try the same process with a greater number of students to sort (Credit: http://csunplugged.org/sorting-networks/)

Watch this video to see how it works: https://youtu.be/30WcPnvfiKE



# 3. Coding and Robotics




"The journey of a thousand miles begins with one step" - Lao Tzu

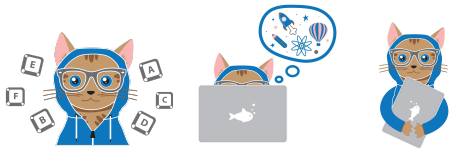
Coding or computer programming is the giving a computer a set of instructions, or commands, so that it can solve a computing problem and execute a task as an executable computer program. Computer programming is used to develop software, websites, web apps, mobile apps and much more.

Robotics is the design, construction and application of machines, that is robots, that can operate automatically. The machines can sense their environment or use data to make real-time decisions and execute complex tasks. Robots are programmable by a computer.

These are skills and tools which can be integrated into the current school subjects such as sciences and English.

As a starting point to integrate coding and robotics at school, I would begin with the following tools:

- Scratch (https://www.scratch.mit.edu), a free web-based platform for primary school students.
- Python (https://www.python.org/downloads/), a general purpose, readable computer programming language that is a great language to start beginners with for example high school students and possibly year 5 or 6 students. To learn to program with Python you will need to download the Python package in the link above.
- Unplugged activities which teach computational thinking.

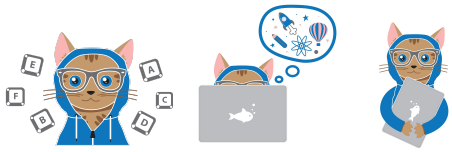


If I were to incorporate robotics into current school subjects I would start with:

- Beebots, bee-like robots which are controlled by simple arrow buttons and suit Foundation - Year 2 (https://www.bee-bot.us/).
- mBots are educational robots based on Arduino Uno and use a Scratch-like drag and drop coding platform suited for Years 3-6. (http://www.makeblock.com/product/mbot-robot-kit)
- Lego Mindstorms EV3 are complex robots with near endless accessories to extend its functionality and problem solving capability (https://www.lego.com/en-us/mindstorms) and suits Years 5-6 and high school students

As discussed in an article published by the Association for Psychological Science, “more than 80,000 App Store apps are described as being educational or learning-based, however, there are currently no science-based standards to guide this determination.” This has left parents and teachers wondering how to tell the educational apps from the “educational” apps. (Read the article here: http://bit.ly/29Y2n10.) My recommendation is to stick to a handful of core technologies which are versatile tools in delivering desired digital literacy outcomes and are easily integrated into current subjects. Tablets and apps are not necessarily a part of this suite of core technologies, where the learning outcomes tend to be limited.

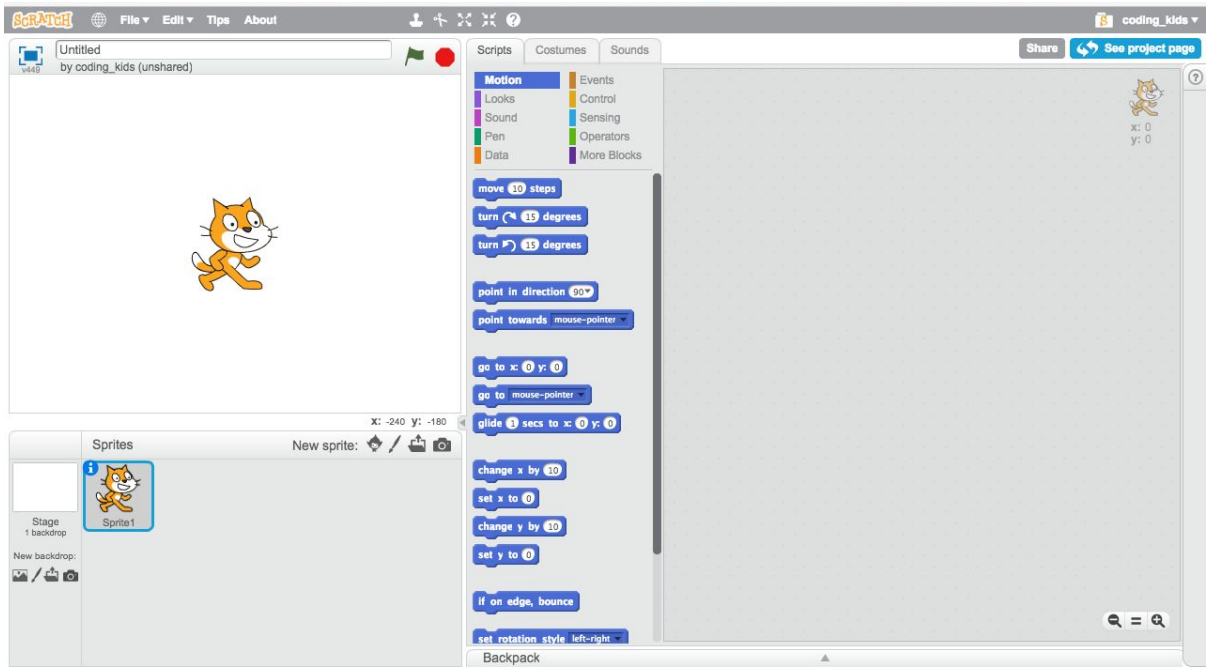
There are many other options that you can use to introduce coding and robotics into your school, but I find that these platforms are the most engaging and flexible, and they also provide best educational value. The platforms are versatile, can achieve comprehensive learning outcomes as outlined in the Digital Technologies curriculum and can be integrated into almost any subject.



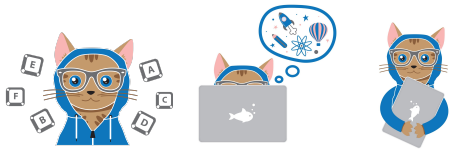
# 4. Scratch

Scratch (www.scratch.mit.edu) is a drag and drop coding platform that was developed by the Massachusetts Institute of Technology (MIT) to teach children computer programming concepts. Unlike textual programming languages, Scratch allows students to get straight into computational thinking concepts without having to  struggle with syntax and typing.

Let's build two simple games and one animation in Scratch to get an introduction to computer programming using this language.




Screenshot of the Scratch website (www.scratch.mit.edu).



# Leap over the Frog Game


Leap over the Frog game

Leap over the Frog is the first game that we will build using Scratch, and you can view the finished product here: https://scratch.mit.edu/projects/96060964/.

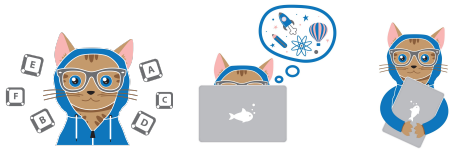
In building this game, we achieve Content Descriptor for Years 5 and 6, Digital Technologies Processes and Production Skills: Design, modify and follow simple algorithms represented diagrammatically and in English involving sequences of steps, branching, and iteration (repetition).

Before we start, let's clarify a new term. A "sprite" is a character or object in the programme.

The objective of the game is to avoid the frog for as long as possible. If the player, that is the cat, touches the frog then it is game over.

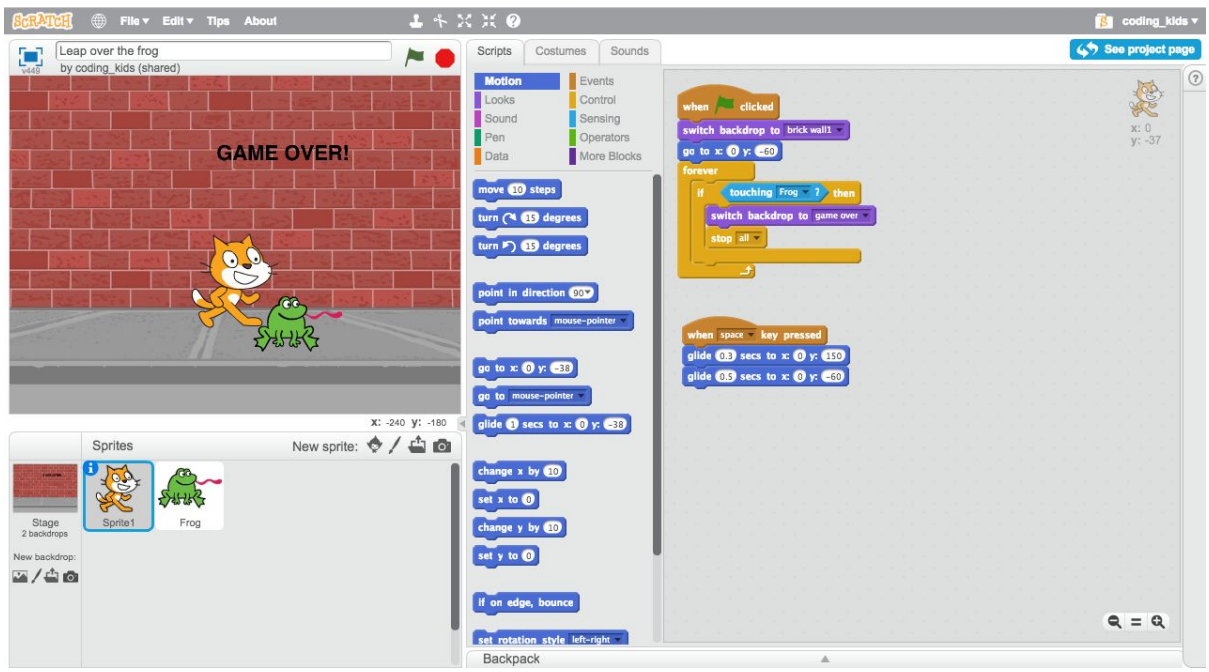
There are four parts to the Leap over the Frog game:

1. The game starts with the "brick wall" backdrop.
2. The cat sprite jumps when the player presses the space key.
3. The frog sprite repeatedly moves from right to left until the game ends.
4. When the cat touches the frog, the "game over" backdrop is triggered.



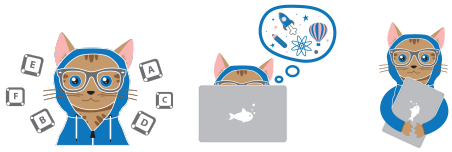
In building this game, we learn:

1. How to break the game into parts and create a series of steps. (Problem solving)
2. How to test our code at each step along the way. We should not proceed to the next step until we test our code to make sure that the program behaves in the way that we expect it to. (Testing code)
3. How to locate and move sprites using the XY coordinates, which include positive and negative numbers. (Cartesian coordinates)
4. How to write the program to run the code by pressing the space key; that is, we want the cat to jump when we press the space key. (Event blocks and sequencing)
5. How to move a sprite up and down; in other words, we learn how to make the cat jump. (Cartesian coordinates and “glide” block)
6. How to make two backdrops: the “brick wall” backdrop and the “game over” backdrop. (Graphic design)
7. How to design the game so that it starts with the “brick wall” backdrop. (Game initialisation)
8. How to program the frog to repeatedly move from right to left until the end of the game. (Looping)
9. How to trigger the “game over” screen when the cat touches the frog. (Decision making / branching and sequencing)




Writing the code to program the cat

Here is the YouTube video on how to make this game: https://youtu.be/fn0kgY16mOM



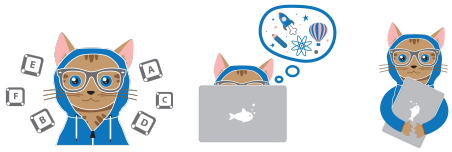
# Forest of Danger Game


Forest of Danger game

Forest of Danger is the second game we will build in Scratch. It is more complex than the previous game. You can view the finished version here: https://scratch.mit.edu/projects/112230999/.

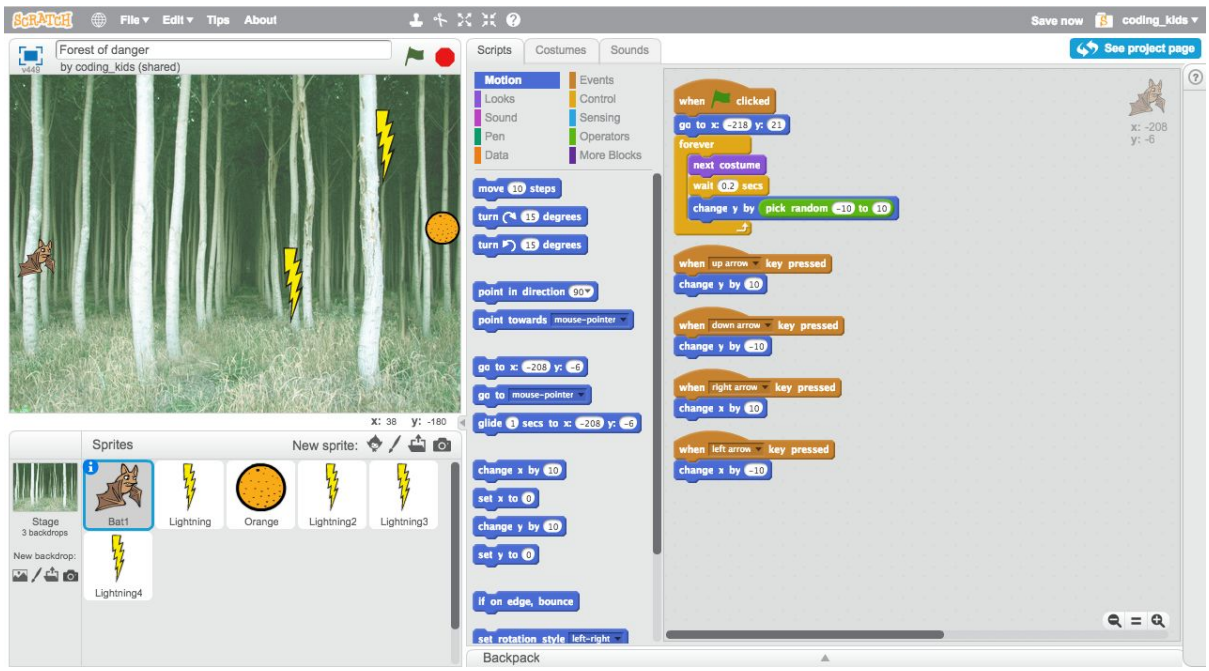
There are seven parts to the game:

1. The bat sprite is controlled with arrow keys.
2. The bat's (player's) goal is to reach the orange.
3. The "forest" backdrop appears at the start of the game and during game play.
4. The "You win" backdrop appears when the player reaches the orange.
5. The "Game over" backdrop appears when the player loses the game by touching a lightning bolt.
6. The first lightning bolt sprite. There are four lightning bolts in the game. One is programmed and tested to ensure that it behaves as expected. Once we are confident with the first lightning bolt sprite then we can duplicate it.
7. Duplicate the lightning bolt sprite three times to get a total of four lightning bolts.

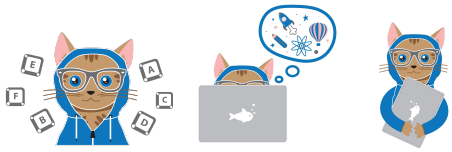


In building this game, we learn:

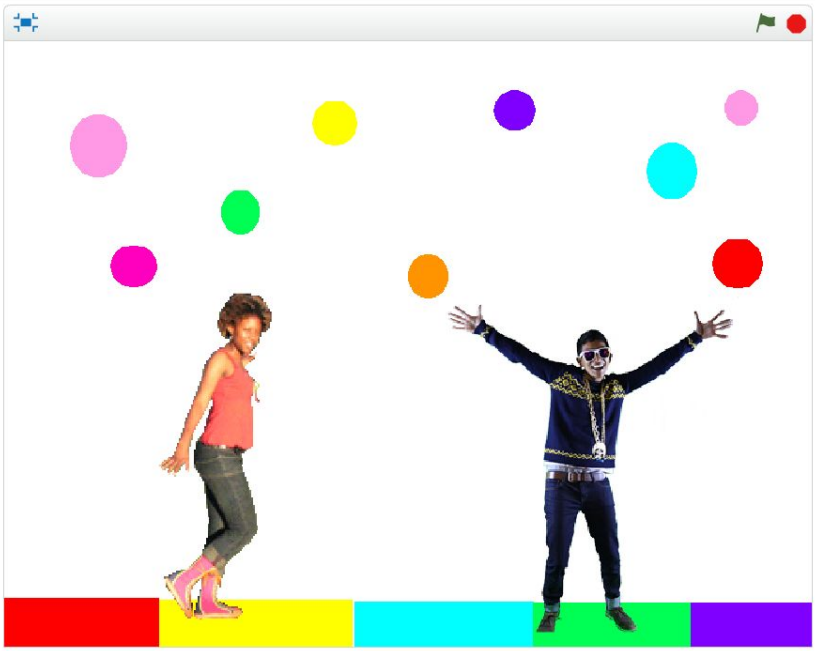
1. How to break the game into parts and create a series of steps. (Problem solving)
2. How to test our code at each step along the way. We should not proceed to the next step until we test our code to make sure that the program behaves in the way that we expect it to. (Testing code)
3. How to control the bat sprite with arrow keys by using the XY coordinates. (Cartesian coordinates)
    a. Up - Y axis in the positive direction
    b. Down - Y axis in the negative direction
    c. Right - X axis in the positive direction
    d. Left - X axis in the negative direction
4. How to animate the bat sprite. (Animation)
5. How to add the original “forest” backdrop.
6. How to create the “You win” backdrop. (Graphic design)
7. How to create the “Game over” backdrop. (Graphic design)
8. How to trigger the “You Win” backdrop when the bat touches the orange. (Decision making/branching and sequencing)
9. How to create a repeatedly falling lightning sprite. (Looping and XY coordinates)
10. How to trigger “Game over” when the bat touches the lightning. (Decision making/branching and sequencing)
11. How to duplicate the first lightning sprite and change the duplicate sprites’ XY coordinates. (XY coordinates)




Code for the bat sprite



# Dance Party Animation



Dance Party animation made in Scratch

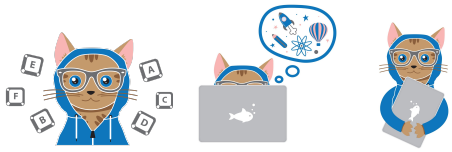
Like coding games, coding animations is a fun way to teach a variety of valuable skills. You can view the finished version of the Dance Party animation (which is discussed below) here: https://scratch.mit.edu/projects/101060127/.

There are three parts to the Dance Party animation:

1. The dancing sprites are animated.
2. The dance floor background  is designed and animated.
3. Music is added.

In building this animation, we learn:

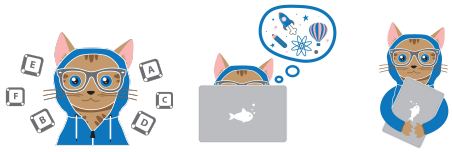
1. How to identify sprites that can be animated. (Recognise that animations are made from a series of still images)
2. How to animate sprites. (Create animations from a series of costumes)
3. How to design a backdrop.
4. How to animate a backdrop. (Create animations from a series of backdrops)
5. How to add looping music. (Looping and wait commands)
6. Why we should separate audio (sounds) and visual (images) forever loops (repeating actions). (Looping, audio data, visual data)



## Tips on Delivering these Classes to Students

1. Explore the interface before creating your first game
    a. Experimenting with various code blocks
    b. Explore the sprite and backdrop libraries
    c. Explore the XY coordinates by experimenting with motion blocks, e.g. what is the difference between “go to” blocks and “glide” blocks.
2. Break down the game or animation into a series of steps before starting to code. Write down the steps. Tick each step as it is completed. Emphasise the importance of working on one step at a time and not proceeding to the next step until code has been tested and behaves as expected.
3. Allow students to guess and experiment to find a solution that works. There are different ways of solving a problem.
4. Give students the opportunity to personalise their Scratch project. Examples of this are use or different sprites or additional features such as a multi-player game or keeping score.

Connect with Emily, Coding Kids, for more details on upcoming PD events.
Follow us on YouTube for more “how to make” videos:
https://www.youtube.com/channel/UCyq9eq4A9B5wH1n6-9NlAow/videos

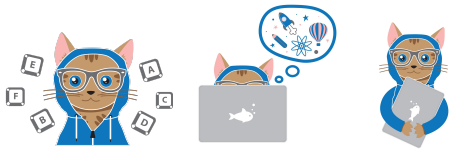


# 5. Python

Python is a general-purpose textual programming language that is very readable and makes a great starting point for beginner programmers. Python is great for beginners because it is easily readable, that is it is as close to the English language as a computer programming language can get, and it comes with Turtle graphics which is a visual way of introducing programming. The Python package can be downloaded for free and is required so that you can use Python to write programs.

Learning to program with Python can be integrated into any of the current subjects. Students can write interactive adventure stories where the reader can select options, which is a digital version of “Choose Your Own Adventure” books. Maths and science can be explored using Python. Scientific data can be processed, analysed and graphically represented using Python. Python can be used to experiment with geometries and create physics engines to explore projectiles and gravity effects. Digital solutions solving community issues can be developed in the context of humanities and social sciences.

The image on the following page  displays a Python file. Programmers can write code and save it in a file. This file can be saved, edited and run. The code includes commands which tell the computer what actions to execute and how.



IDLE File Edit Format Run Options Window Help 100% Sun 1:27 pm

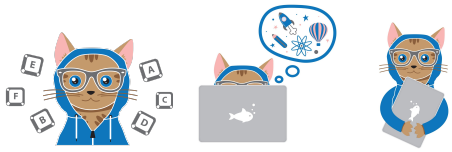
RockPaperScissors.py - /Users/emilydelapena/Documents/Python/Python #1/RockPaperScissors.py (3.5.2)

```
#RockPaperScissors

#Learn: while true, string+string

#1. List
#2. Randomly choose from list
#3. Player chooses
#4. Computer choice displays
#5. Display who wins

import random

RockPaperScissors_list=["r", "p", "s"]
player_wins=['pr','sp','rs']
player_score=0
comp_score=0

while True:
    computer = random.choice(RockPaperScissors_list)
    #print(computer) #delete later

    player = input("Let's play Rock, Paper, Scissors. Press 'r' for rock, 'p' for paper and 's' for scissors. 3, 2, 1...")

    if player=='q':
        print("Player score:",player_score, "Computer score:", comp_score)
        break

    if player==computer:
        print("Draw")
    elif player+computer in player_wins:
        print("You win!")
        player_score=player_score+1
```

Ln: 18 Col: 0

Learn to build fun computer games like Rock, Paper, Scissors with Python.

See Appendix A for details on beginner level projects that help students learn Python fundamentals.



# 6. Lego Mindstorms

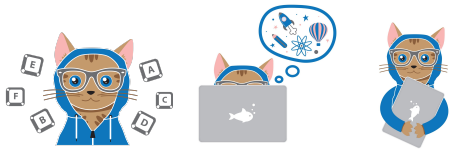
Lego Mindstorms robots are a great introduction in coding for robotics. A robot is a combination of software and hardware that can sense information from its environment and use that information to make real-time decisions, e.g. navigating a terrain.

Students can learn how to write software and build a robot that can make its way to a target by navigating an obstacle course. This activity teaches learning from trial and error in a tactile environment.

In order to program a Lego Mindstorm robot to navigate various characteristics in an obstacle course, students must learn about the robot's sensors. The first sensor to understand and work with is the colour sensor, which will be the subject of the line follow program. The colour sensor is a sensor that distinguishes between eight different colours. It also serves as a light sensor by detecting reflected light intensities. The line follow program is a program that allows the robot to follow say, a black line on a white background like a train follows a rail track.

The Lego Mindstorms EV3 is the third generation robotics kit in Lego's Mindstorms line

The robots are programmed using the EV3 software which is free to download on the Lego Mindstorms website (https://www.lego.com/en-us/mindstorms/downloads/download-software).



The software allows students to program the robot in a drag and drop coding environment, like Scratch. Coding blocks are dragged into the workspace and connected together into a workflow for the robot.

The challenges below demonstrate the types of skills that a robot should be able to execute depending on the student's programming abilities. Most challenges are based on a line follow program.

## Easy Challenges

1. Follow a straight line (black line on white background).
2. Follow a gentle curve.
3. Follow a hairpin turn, e.g. 7cm radius 180 degree turn.
4. Follow a right angle (90 degree) corner.

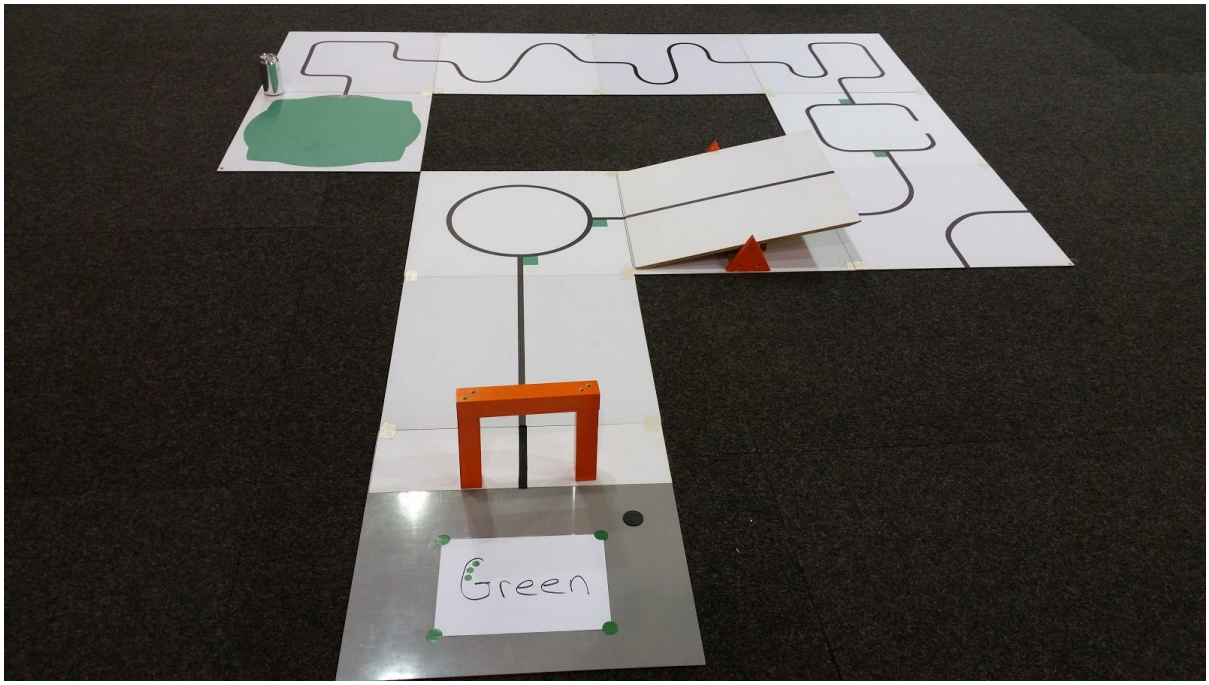
## Intermediate Challenges

5. Go over speed bumps (5mm high).
6. Make a decision as to which turn to take at a fork in the road (with green square indicators).
7. Continue on a dashed line.
8. Continue with a gap in the line.
9. Go up and down ramps, including a seesaw.

## Advanced Challenges

10. Detect an obstacle (such as a water tower) and go around it to connect back onto the line.
11. Identify when it is in a green area.
12. Detect a 375 ml aluminium can wrapped in aluminium foil.
13. Push or pull the aluminium can out of the green area.

These challenges were seen on the "rescue fields" at the 2016 Robocup Junior Competition in Queensland in 2016. A "rescue field" is an obstacle course that the competition robots had to navigate. The start position is the orange gate and the end position of the obstacle course is the green patch or the "contaminated spill zone". Once the robot reached the green patch it had to locate a silver aluminium can, or the rescue target, that was located on the green patch and then push the aluminium can out of the green patch and into "safety", that it outside of the contaminated spill zone.




A “rescue field” at the 2016 Robocup Junior (Queensland) Rescue Competition

# 7. Timetables and Group Sizes

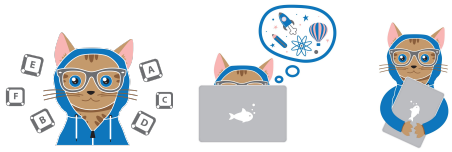
## Fitting coding and robotics into a school timetable

The new Digital Technologies subject will be integrated into current subjects. The learning outcomes in the new Digital Technologies subject are grouped into year groups:

- Foundation - Year 2
- Year 3-4
- Year 5-6.

This means that the learning outcomes in Digital Technologies can be spread across multiple years, for example all the outcomes outlined in Foundation to Year 2 can be spread across the three year levels. It is not required to repeat each Digital Technologies outcome in each year level from Foundation to Year 2.

Digital Technologies is not a stand alone subject, it is to be integrated into the current subjects. This means that a single class activity can achieve learning outcomes from both English and Digital Technologies. One activity can achieve many outcomes simultaneously.



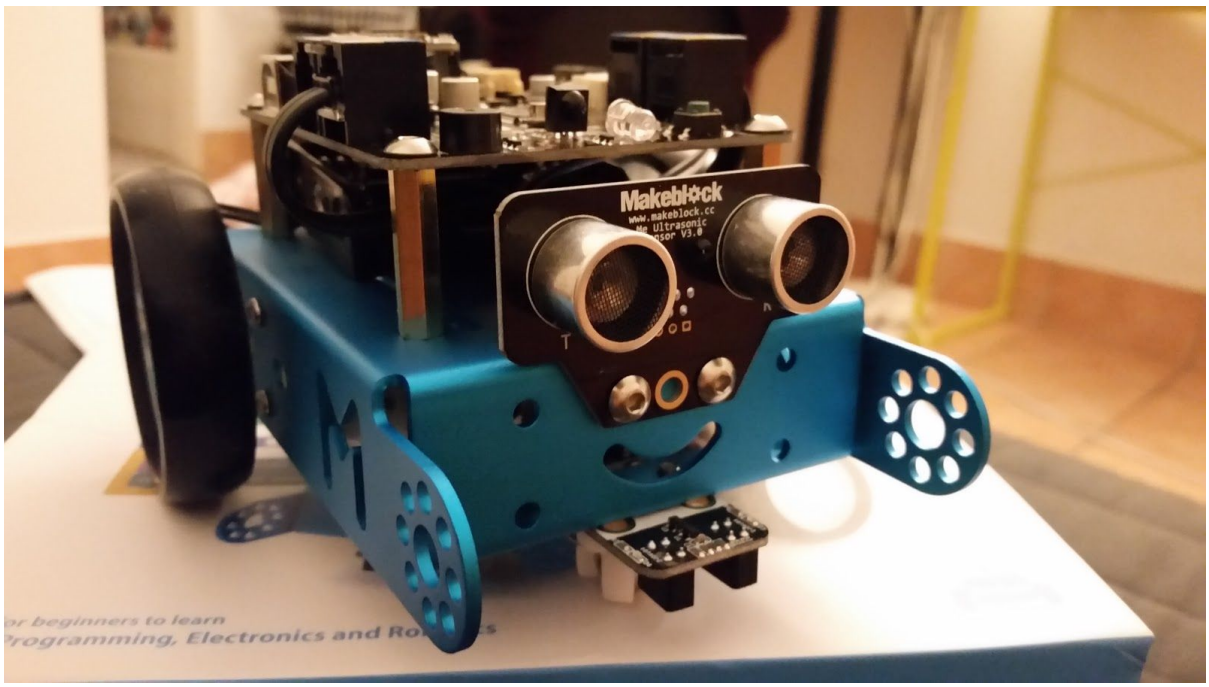
# 8. Other Gadgets to Use

Use of hardware, whether robots or circuitry, supports tactile learning. Circuitry tools, such as littleBits or Makey Makey, allow students to explore and experiment with hardware to create inventions. Circuitry is a basic building block of computers. The on and off switch in a simple circuit is the basis for binary data and programming computers.

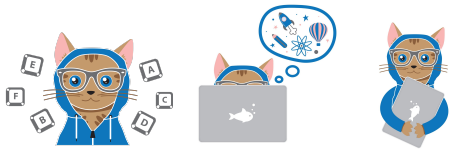
Careful selection of the right tools and technologies that are versatile across learning outcomes and subjects can make it cost effective to deliver the Digital Technologies subject. Digital Technologies does not have to be an expensive exercise for schools. Here are some versatile technologies for Digital Technologies learning outcomes.

**mBot**
mBot is an educational robot that is programmed using mBlock software. mBlock is based off of Scratch, which allows students who are familiar with Scratch to easily onboard onto programming an mBot. Mbot can be used to build a digital solution for an environmental or English project.

**mBot activity**
Programme mBot to navigate an obstacle course. mBot can be programmed to follow lines, detect and avoid walls, and detect and avoid tabletop edges.

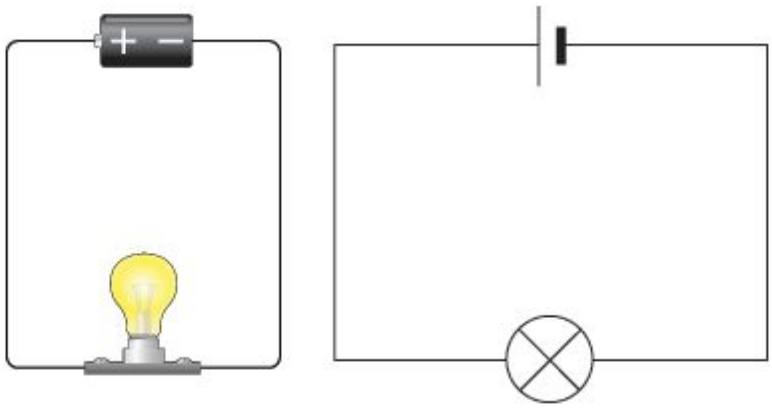



An assembled mBot



**littleBits**

littleBits is an electrical kit that allows students to invent using electrical circuit modules with buzzers, LEDs, buttons, light sensors, sound sensors, DC motors, and other accessories. Students can create digital solutions in science and social science projects.

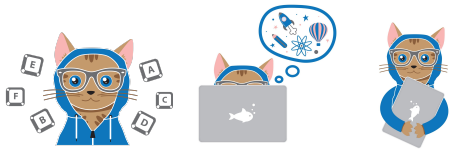


A simple circuit

The image on the left is a drawing of a simple circuit consisting of a light bulb and a battery.

The image on the right is a schematic drawing of the circuit on the left using standard representations of a battery and a light bulb commonly used in electronics.


Modules in a littleBits Base Kit



**littleBits activity**

Introduction to littleBits—build simple circuits using:

- A battery power module, a button, and an LED.
- A battery power module, a slider, and a bar graph.
- A battery power module, a light sensor, and a sound buzzer.

After building the simple circuits above students can experiment and build their own invention.

**Makey Makey**

Makey Makey is an input/output device that uses circuitry. Examples of an input device are a mouse and a keyboard. A mouse and keyboard is used to input information into a computer. For example if you want to type a report you use the keyboard to input letters and punctuation.

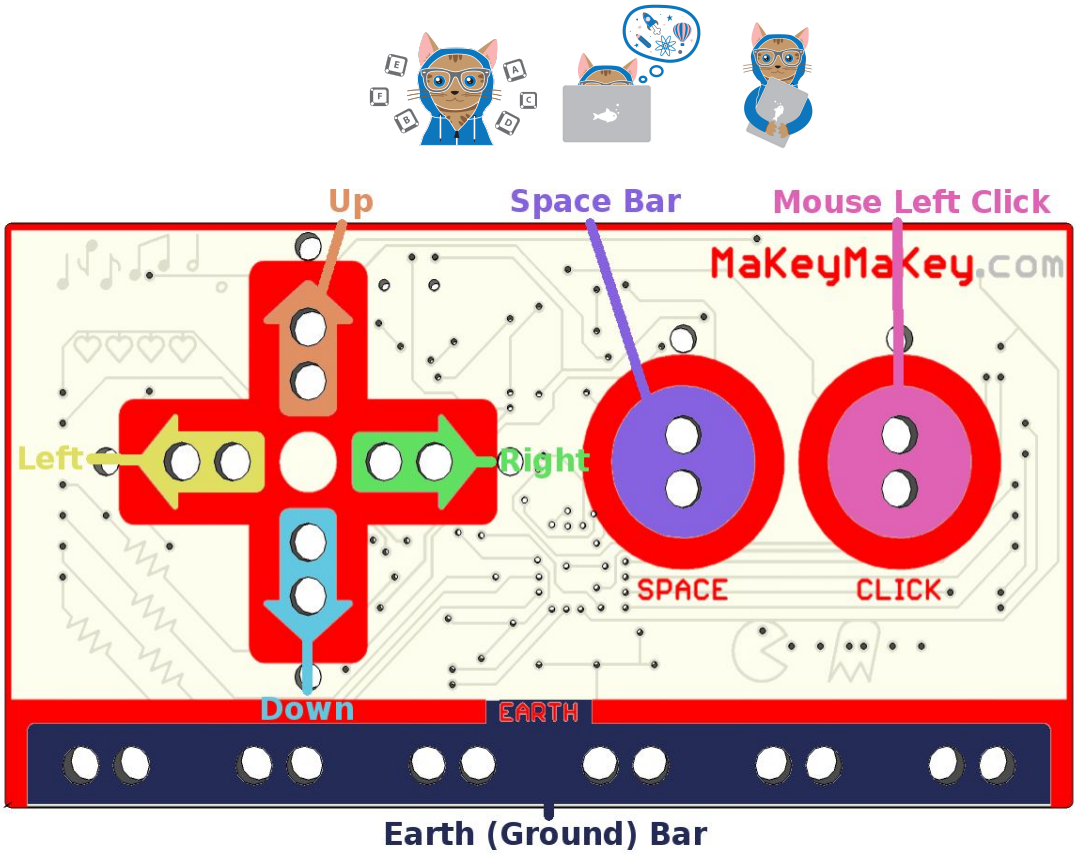
A Makey Makey is an input device but it only has a limited number of inputs (see images below):

1. Up, down, left and right arrow keys.
2. Space bar.
3. W, a, s, d, f & g keys.
4. Mouse movements: up, down, left and right.
5. Mouse clicks: left click and right click.

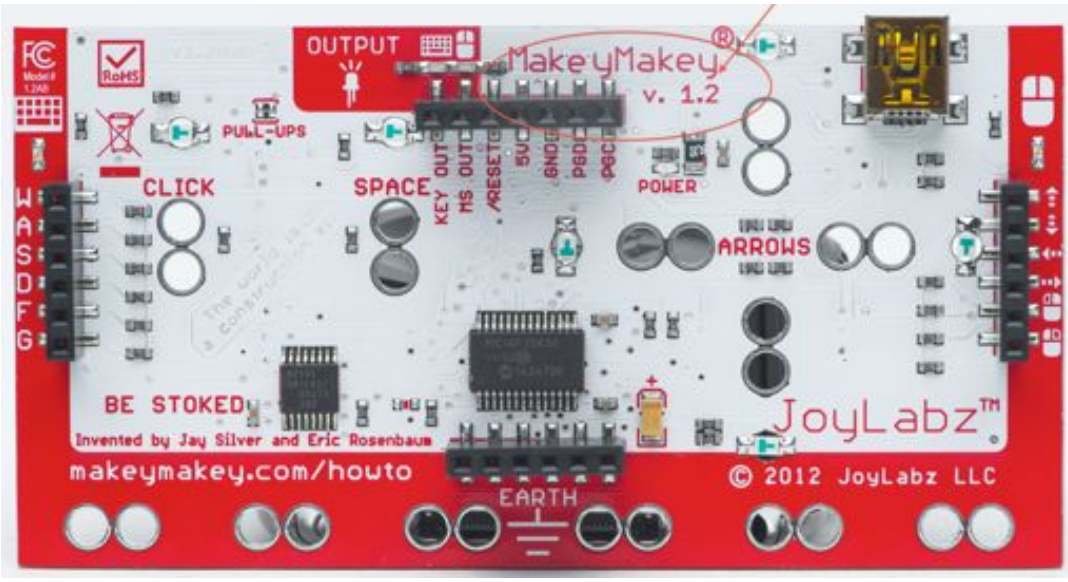
By connecting the Makey Makey to the computer it can integrate with Scratch which means you can develop a program in Scratch and create the input hardware with Makey Makey.

By connecting the Makey Makey inputs e.g. the arrow keys with conductive materials, e.g. marshmallows, bananas, aluminium foil, and buckets of water you can create your own input hardware. The Makey Makey inputs are connected to conductive materials using alligator wires.

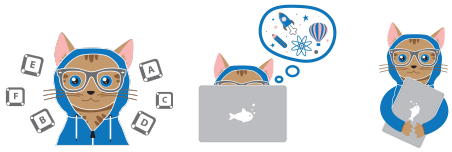
The use of Makey Makey and Scratch can allow student to create digital solutions in English, science and social science projects.

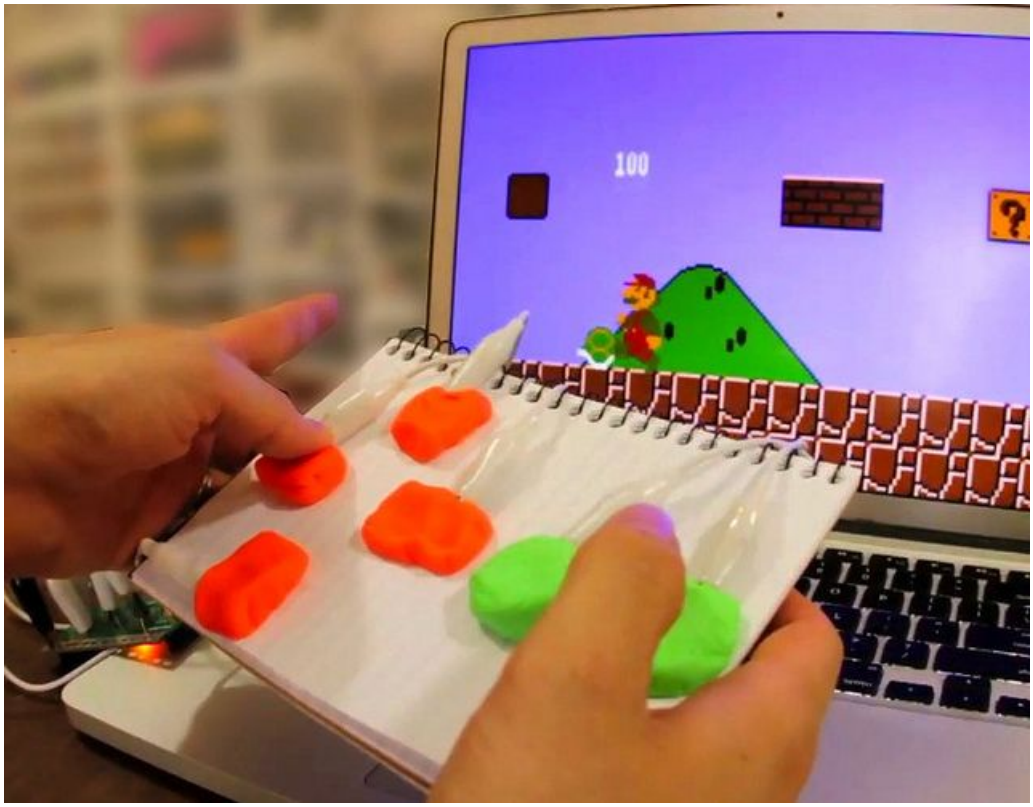



Front of the Makey Makey




Back of the Makey Makey





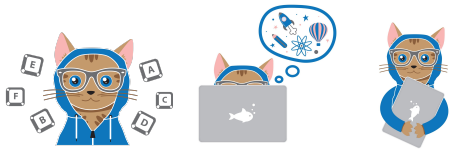

Connecting a Makey Makey to “Play dough” as an input device to play a computer game

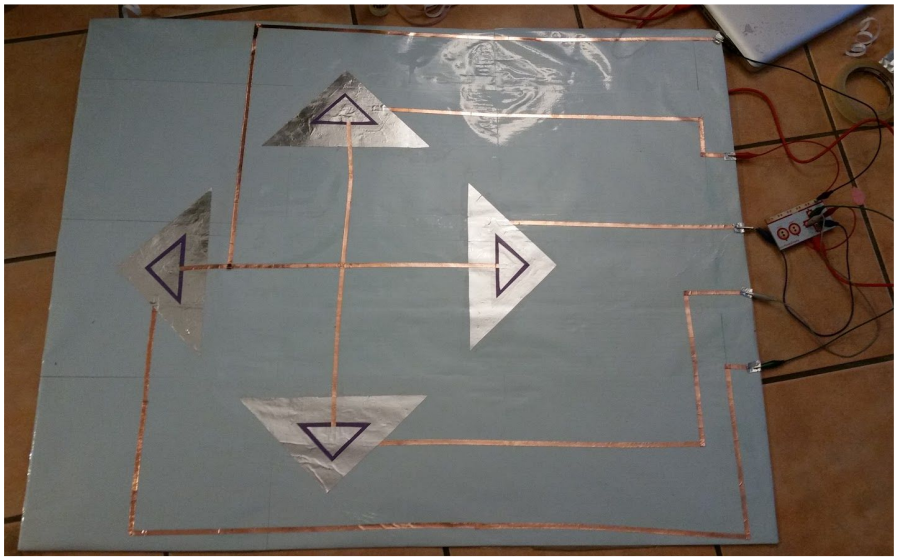
**Makey Makey activity**
Build a game controller (arrow keys, space key, and left mouse click) using Makey Makey and conductive materials. Connect the Makey Makey to the computer using the usb cable that comes with the Makey Makey pack. Then use the controller to play a game you built in Scratch.

For example, you could build a game similar to *Dance Dance Revolution* by creating the game programme in Scratch and then creating an input device using buckets of water or pads of aluminium foil as the arrow key inputs to control the game. Here is a *Dance Dance Revolution-style* game, programmed with Scratch, that can be played using a Makey Makey controller: https://scratch.mit.edu/projects/116737710/.

A dance mat can be built with aluminium foil arrows glued onto cardboard. Connect the aluminium foil arrows to the Makey Makey arrow inputs and connect the player to the Makey Makey ground.

The dance mat in the following image is made from cardboard, aluminium foil, sticky tape, book cover adhesive, copper tape (can be replaced with aluminium foil and sticky tape).

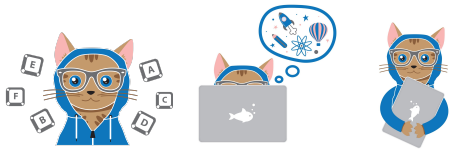




A homemade dance mat


Dance Dance Revolution Game

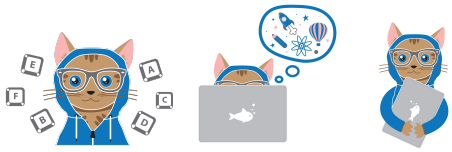
Just like a dance mat was built using  the Dance Dance Revolution-style game, a piano mat or an operation-style game can be made using household materials like aluminium foil, sticky tape and cardboard.



Build a piano mat using Scratch and Makey Makey


Build an Operation game using Scratch and Makey Makey

Or simply build your own digital invention.



# 9. Coding, Robotics, and Invention Competitions

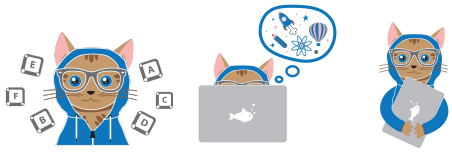
Competitions are a great way to excite students into exploring and playing with coding and robotics. At competitions student teams are often scheduled to compete or present their inventions at set times in front of a panel of judges. When not competing, students can enjoy the day by watching and learning from other teams. Teachers can best prepare their students by focusing on feedback as a means for learning, whether it is feedback from robots testing the terrain or from judges.

Competing gives students the opportunity to work with technology in a competitive environment, develop creative problem solving skills and experience team work under pressure. The best performing teams have team members that are resilient in the face of challenges and continue having a go all the way to the end. They tend to also have spent the most of the year preparing.

There are many competitions available in Australia, but here are a few of my favourites:

- Robocup Junior (http://www.robocupjunior.org.au/).
- ICT Young Explorers (http://www.youngictexplorers.net.au/).
- First Lego League (https://firstaustralia.org/programs/first-lego-league/).
- Coolest Projects (http://coolestprojects.org/).

Student team booths at Young ICT Explorers, Brisbane 2016



# 10. School Tech Excursions in South East Queensland

Including school excursions in your coding and robotics program will engage your students on another level and give them an opportunity to see fun, tactile, and real life applications of the technology they are learning in the classroom. Below are some practical ideas for excursions in South East Queensland.

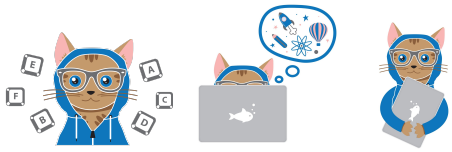
**Holoverse** (http://holoverse.com.au/)
Located in Southport, Gold Coast, Holoverse is a virtual reality adventure. The free school tour includes both a virtual reality adventure experience and a one-hour lecture on the technology behind it.

**Robotic Vision** (http://roboticvision.org/)
Robotic Vision, located in the Brisbane Central Business District, is a research centre focused on building robots that can “see”. Sue Keay, Robotic Vision’s COO, gives students a tour of the research facility. She also introduces the robots that are learning to “see” and complete tasks that are normally done by humans, allowing students to observe the robots in action.

**National Science Week** (https://www.scienceweek.net.au/)
National Science Week is an Australian government initiative that occurs every August. The initiative celebrates science and technology with thousands of events throughout the country that are designed for people of all ages. Check the website for a listing of events in your local area to find excursions that would be a good match for your curriculum.
Location: Nationwide

**Fun Palace**
(2016 event: http://www.slq.qld.gov.au/whats-on/calevents/general/learning/fun-palace-2016)
Fun Palace is a festival for child scientists and artists that is held every October at the State Library of Queensland and at various regional Queensland libraries. Activities include coding and invention workshops.

**World Science Festival Brisbane** (http://www.worldsciencefestival.com.au/)
World Science Festival Brisbane is an annual week-long celebration and exploration of science that occurs each March. Workshops for children include programming robots and making inventions.



Robotic Vision

Holoverse

# 11. Where to Find Help

There are many wonderful resources that will help you to integrate coding, robotics, and computational thinking into your classroom and curriculum:

1. Former school students who are currently studying computer science, information technology (IT) or mechatronics.
2. Local university and Technical and Further Education (TAFE) students in computer science, IT, game design, web design, education.
3. Non-profit organisations such as CoderDojo or Code Club Australia.
4. Various online resources such as “Programming in Scratch”, a free online course from edX.org. (https://www.edx.org/course/programming-scratch-harveymuddx-cs002x-1)

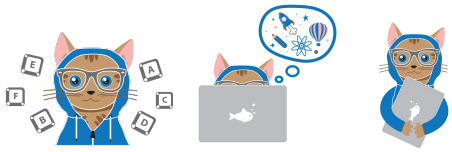
## Coding Kids can help

Coding Kids can help in various ways. We offer:

- In-school coding classes.
- After-school coding clubs.
- School holiday code camps.
- Professional development workshops.
- Tutoring in Auslan and various other sign languages.

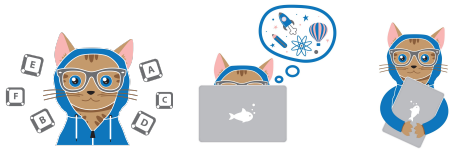
Contact Emily de la Pena at Coding Kids for more information:

- hello@codingkids.com.au
- 0449 162 677



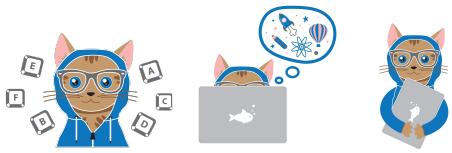
Find Coding Kids at these Queensland schools and libraries:

| | | |
|---|---|---|
| Ipswich Junior Grammar School | Cannon Hill Anglican College | Ashgrove State School |
| Belmont State School | Browns Plains State School | Bulimba State School |
| Camp Hill State School | Hendra State School | Corinda State School |
| Leichhardt State School | Oakleigh State School | Norman Park State School |



| | | |
|---|---|---|
| Seven Hills State School | St Joseph's, Kangaroo Point | St Brendan's, Moorooka |
| Gympie Library | Noosa and Cooroy Libraries | Queensland College of Teachers |
| State Library of Queensland | Homeschool groups (South Brisbane, Ipswich & Yatala) | |

# Appendix A - Teaching Python to Beginners



# Table of Contents


**Download Python** **3**

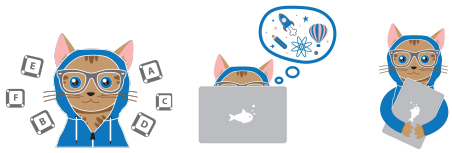
**1. Hello World** **4**

**2. Drawing with Turtle** **5**

- Import 5
- Naming Turtle 6
- Forward and Turn 7
- Reset 7

**3. Variables** **8**

- Data: numbers and strings 8
- Calculator 9
- Strings 10



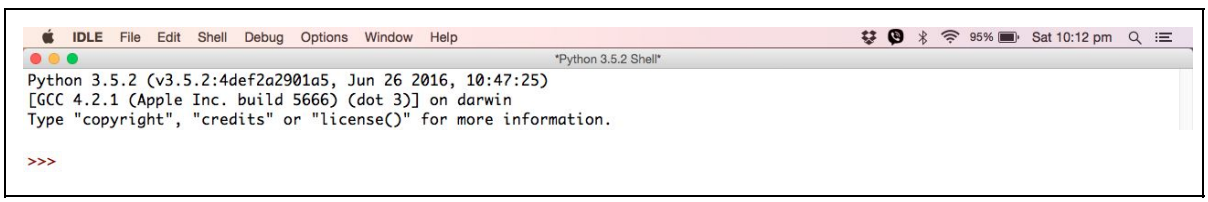
# Download Python

You can download Python for Windows, Linux/UNIX or Mac OS X by following this link: https://www.python.org/downloads/

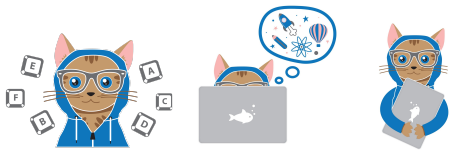
Once Python is installed, you can use it to complete three lessons that will provide an introduction to computer programming.

To begin the first lesson, open an IDLE window. IDLE will be in the folder where the Python package was downloaded and saved to. (The IDLE program is part of the Python install.) You should see a new open window; this is called the Python shell. A shell is just a window where you can type commands and then run them to see what happens.

There are three  “greater than” symbols (>>>) in the Python Shell (see image below). This is called the “prompt.” When you see the prompt, it means that the computer is waiting for you to type a command that will tell it what to do.




Python shell (window): “>>>” is called the prompt

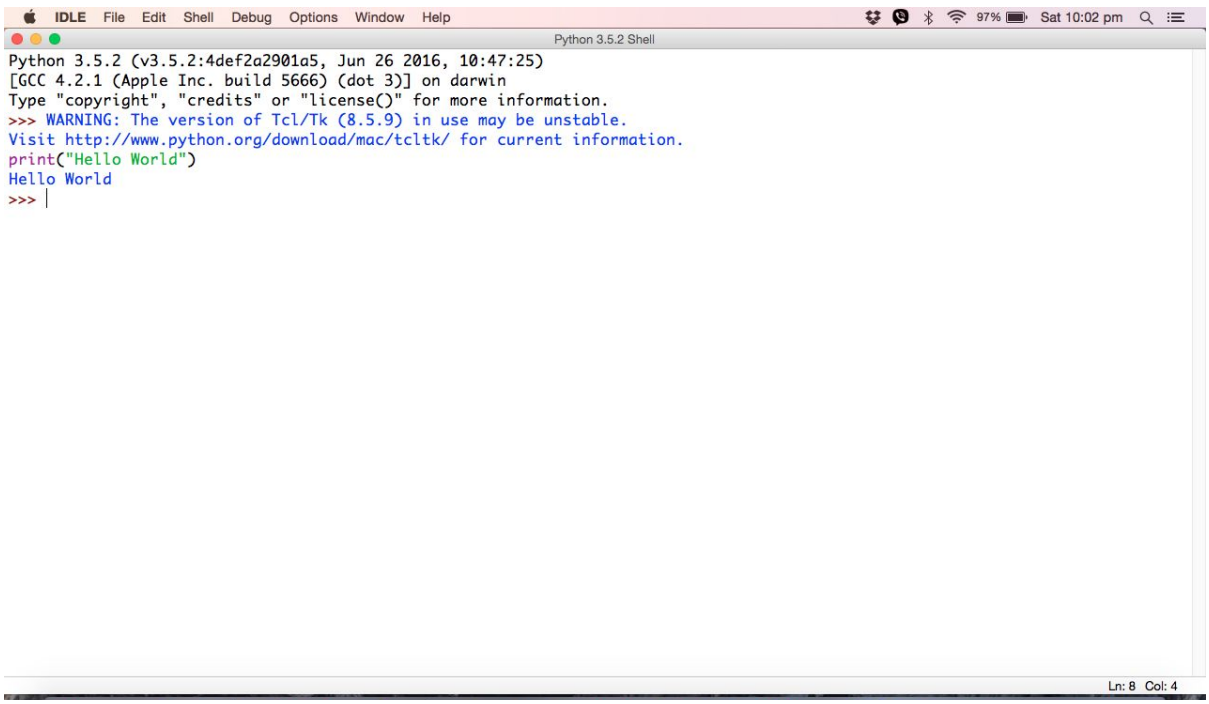


# 1. Hello World

There is a longstanding tradition that the first program you ever run when you learn a new computer language should generate the output “Hello World”. Let’s continue with this tradition.

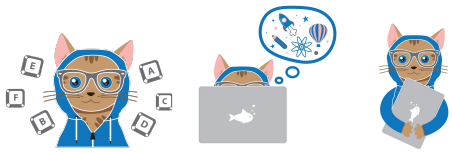
Let’s try our first command, which is an instruction that tells the computer what to do.
In the Python shell, type:
>>> print(“Hello World”)

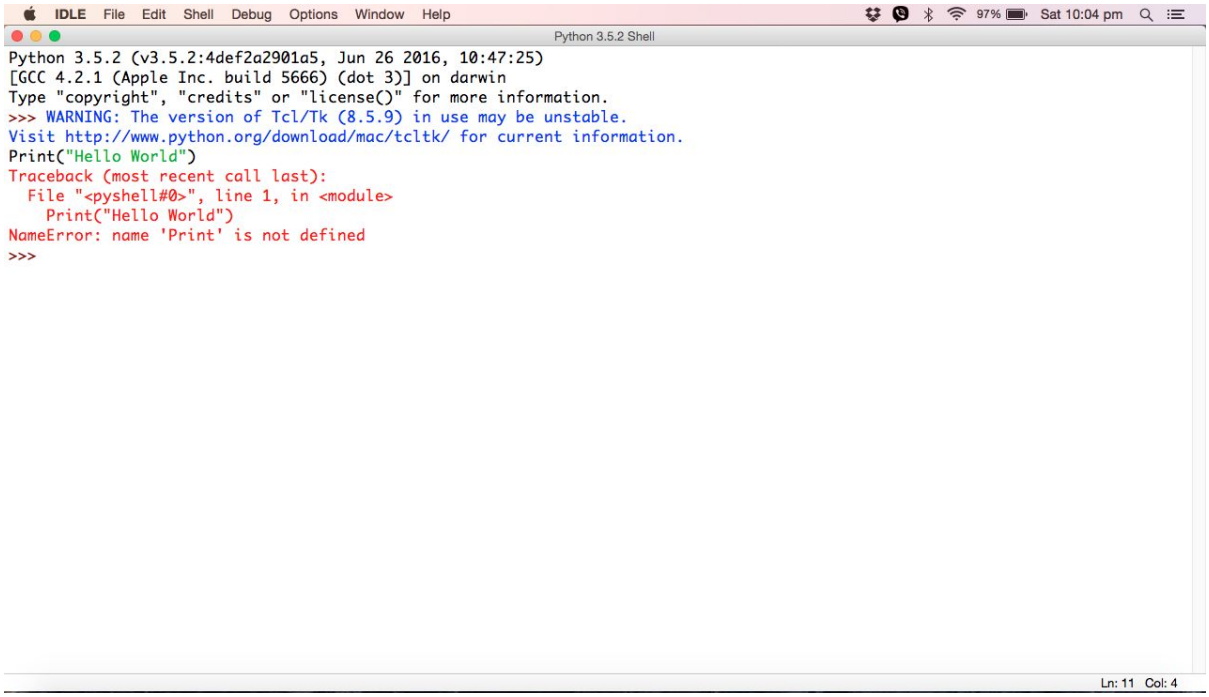
In the Python shell, when we are ready to tell the computer that we have finished with a line of instruction, we press the “enter” key. Press it to see what happens. This is what you should see:




Remember, in order to get the desired result, you need to type the command correctly. The computer is not smart, so it cannot guess as to what you intended for it to do when you type the command incorrectly. If you make a mistake the computer will just be confused and it will display an error message. Check that you use the close quotation marks and brackets, that is brackets and quotation marks always come in pairs, the open and close brackets and the open and close quotation marks. Additionally, commands are case sensitive. Notice that we use all lowercase letters in our “print” command.

If we accidentally type print with an uppercase “P” then an error message appears. The last line of the error message gives us a clue: “NameError: name ‘Print’ is not defined”. Python does not recognise “Print” but it does recognise the command “print” with a lowercase “p”.






What did we do here? We gave the computer a command, “print”. That means that we told the computer to display something onto the screen.

Most of the commands will look something like this: a command (e.g. print) and then something inside of brackets (e.g. “Hello World”).

>>> print(“Hello World”)

Congratulations! You are now a computer programmer. You have given your computer an instruction and your computer has successfully executed the instruction.

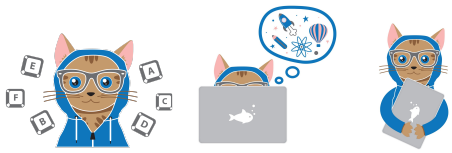
Everything you will learn from now on is based on the concept of commands. Moving forward, we will combine commands to create programs that give the computer instructions in order to make it do what we want it to do.

## 2. Drawing with Turtle

Turtle graphics module is available with Python. It is an easy way to introduce programming to children because it is visual and students can create computer programs that draw shapes with various colours. The visual feedback allow students to easily identify errors and fix them.

### Import

Let’s go back to our Python Shell. To get Turtle to appear, we will need to import the Turtle graphics module which is included in the Python install. Let’s use the import command:
>>> import turtle



You're going to use the import command quite a bit when you use Python. Import tells the computer what kind of commands you're going to need for your program. In this case, we want it to load (import) the commands for working with Turtle.

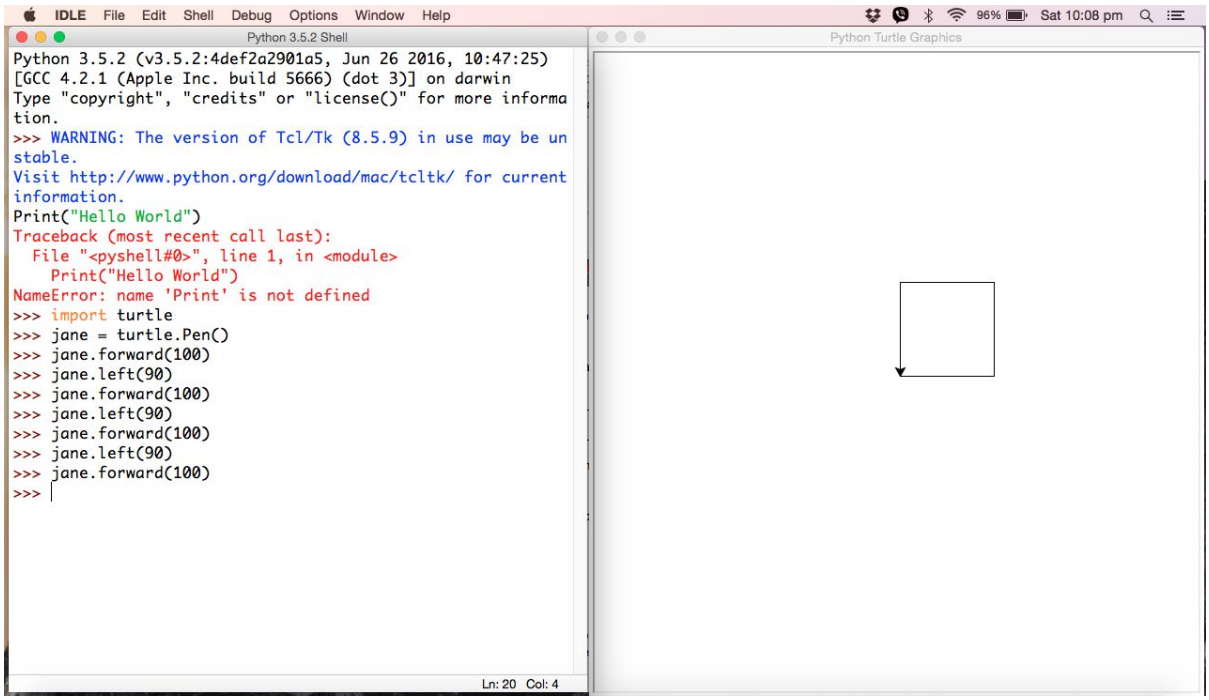
## Naming Turtle

Now that we have imported the Turtle commands, let's name turtle. Here is the command for that:
>>> jane =  turtle.Pen()

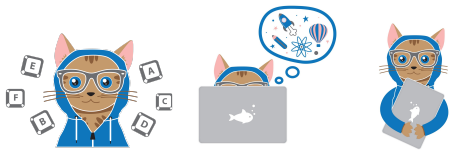
"turtle.Pen()" is the command that names my turtle Jane. Now all the turtle commands will be preceded with "jane".

Also, did you notice that I used a capital "P" for the "pen" portion of the command? If you use a lowercase "p", it will not work correctly.

Executing this command has opened a Turtle window. Let's arrange the windows so that we can see both the Python shell and the Turtle window at the same time (see image below). We will be typing commands into the Python shell to control the turtle we created, and we'll be able to see how these commands control the turtle in the Turtle window.




The Python shell is the left window and the Turtle window is on the right. You can also read the code to instruct turtle to draw a square that is 100 pixels by 100 pixels in size.



## Forward and Turn

Let's tell the turtle, Jane, to move forward.
>>>jane.forward(100)

Before we press "enter", what do you think that 100 means?

- Is it centimetres?
- Millimetres?
- Perhaps a unit for speed?
- It is actually pixels. Because Jane lives in a computer screen, we measure distance in pixels.

In other words, "100" refers to a number of pixels. We are telling Jane the turtle to move forward 100 pixels when we type "jane.forward(100)" into the prompt.

Now we want to Jane to turn left. We use jane.forward() for jane to move forward. What might we type if we want Jane to turn left?
>>> jane.left(90)

Previously we told jane to move forward 100 pixels. Now we want Jane to turn left 90. But 90 what? It doesn't make sense to measure a turn in pixels since pixels measure distance. In this case, "90" refers to degrees. We are telling Jane to turn left 90 degrees.

If we tell Jane to move forward after turning, she will move in the direction she now faces. Have a go:
>>> jane.forward(100)

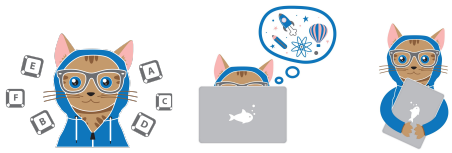
Let's try repeating these commands a few more times to make Jane move in a square.

Now try changing colours and try drawing a circle. Can you guess what the Python commands might be?

## Reset

What if we want to clear everything and start again with empty screen?
>>> jane.reset()

This clears the screen and places Jane in her original position at the centre of the screen, facing towards the right.



# 3. Variables

## Data: numbers and strings

The concept of variables is important in programming. You can think of a variable as a label. It is a shortcut for pointing to something that we want to say. Here is an example:
>>> score = 10

I have typed a command to tell the computer that when I type “score,” I really mean “10”. Now if I say print “apples”:
>>> print(score)

The computer prints the number 10 on the screen. This is because we have previously told the computer that “score” equals 10. But this is not permanent. I can change what the variable “score” points to. Now, if I want “apples” to point to the number 15, how might I do that?
>>> score = 15

Now print “score”.
>>> print(score)

You should see  the number 15 printed on your screen.

Now try this using different variables—for instance, “lives” and “health” (useful variables when designing game dashboards).

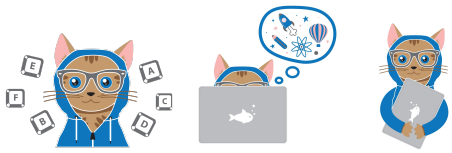
Numbers are not the only thing that can be stored in a variable. For example:
>>> winner = “Sam”
>>> print(winner)

You can usually name variables to be whatever you like, but it is always good to use a name that is relevant so that you or another programmer reading your code. This ensures that your code is easy to understand.

For example, let’s say you are building a game where the player has to collect gems. You may then create a variable called “gems” that points to the number of gems the player has collected.

>>> gems = 20

If you decide to call your variable “cakes” instead of “gems,” you may later get confused as to why the variable that stores the number of collected gems is called “cakes”. Or, even worse, you may be reading the code and wonder why there is a “cakes” variable when there are  no cakes in the game. Therefore, it is best practice for the variable name to reflect the meaning of the information.



## Calculator

Did you know that you can use the Python shell as a calculator?

We can add numbers:
>>> 3 + 12
15

>>> 20 - 2
18

>>> 5 * 2
10

(We use an asterisk in programming as a multiplication sign. This is because we use “x” to represent other things, e.g. x-coordinates.)

>>> 50 / 2
25.0

We use a forward slash to indicate division in programming.

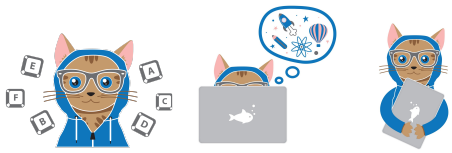
We can even use variables in calculations. First let’s define a few variables.
>>> Apples = 15

>>> Bananas = 2

>>> Apples * 2
30

We can also create variables with numbers and use them in calculations as follows:
>>> Apples + Bananas
17

We can use math operations as part of more complex algorithms using variables and displayed graphically.



## Strings

Information stored by a computer is called data. We have already learnt that variables can store information (data) such as numbers. Variables are not just limited to storing numbers, however; they can store other types of data as well, including text.

Remember our first command, when we printed Hello World:
>>> print(“Hello World”)

“Hello World” is typed in quotation marks in the command. We call text inside quotation marks a “string”. A string can consist of multiple characters, letters, or numbers—whatever we can type.  We place quotes around the string so that the computer knows that the string is not a command.

We can try this:
>>> fruit = “orange”
>>> print(fruit)
orange

But what happens if I leave the quotation marks out?
>>> fruit = orange

I get an error message because the computer thinks that “orange” is a variable; in other words, the computer expects “orange” to point to stored data. However, we never created a command to make “orange” point to anything. Therefore, the computer doesn’t understand what we mean and is unable to execute the command.

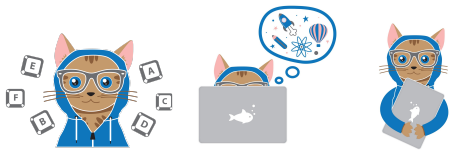
Let’s have a closer look at how strings work. Strings can be added together.
>>> hello = “hi” + “there”
>>> print(hello)
hithere

What happened?

If we don’t include a space, the computer won’t know to add a space. Let’s add a space this time:
>>> hello = “hi” + “ ” + “there”
>>> print(hello)
hi there

Strings are really just a bunch of characters. That’s why they’re called strings.

Let’s try this now:
>>> friend = “E” + “r” + “i” + “n”
>>> print(friend)
Erin



We have just completed three exercises in programming with Python:

1. Hello World
2. Drawing with Turtle
3. Variables.

Now that you know the basics of the Python language, you are ready to explore and write your own little programs.