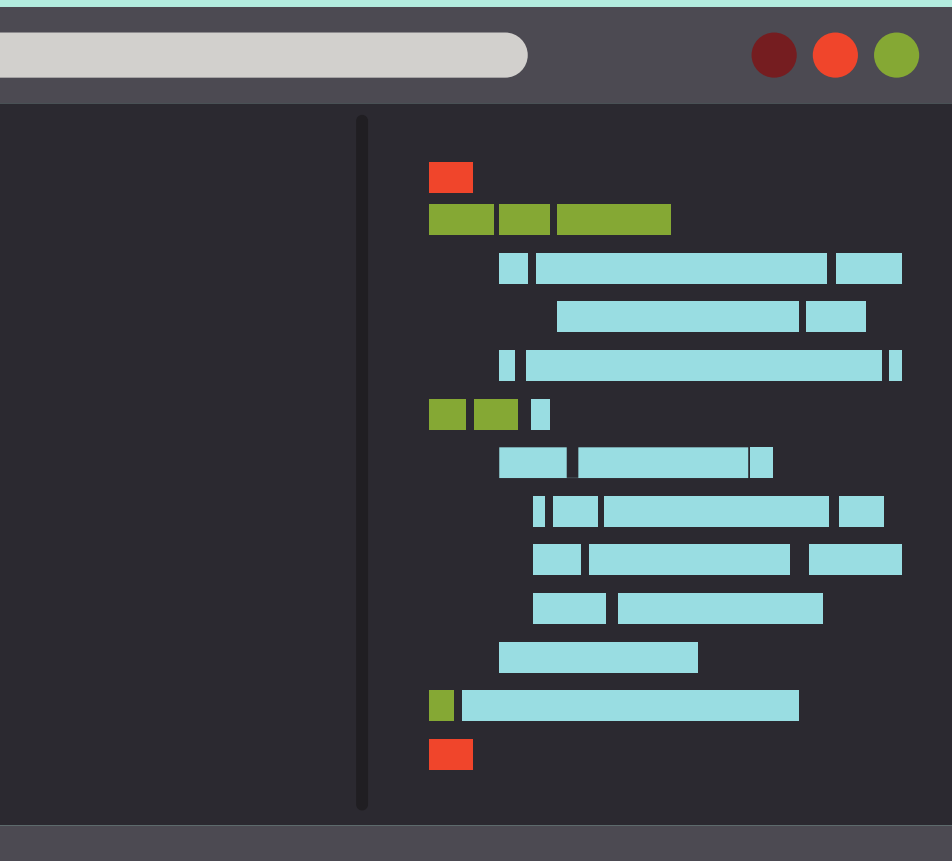



YOUNG MINDS

# DIGITAL LEARNING MASTERCLASSES



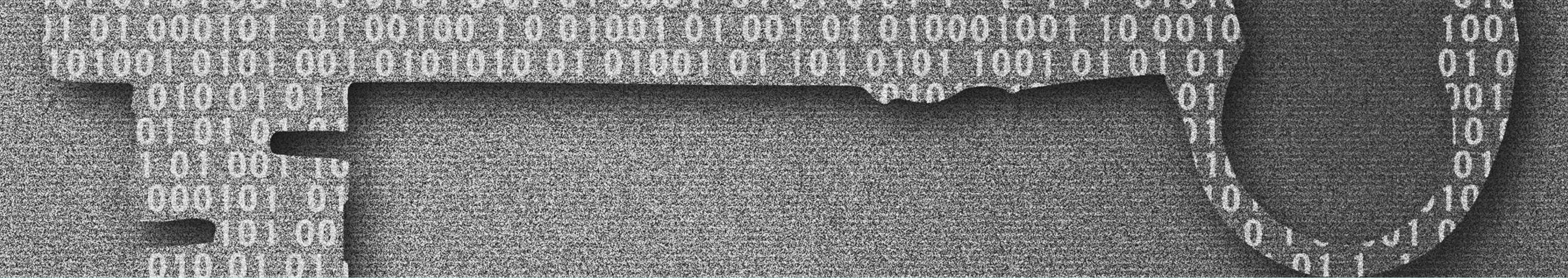
Games & Projects for Students Aged 7-18



# Digital Learning Today

## WHY START NOW?

Many people think that coding is an impossible skill to learn, or that it's something best left to geeky geniuses. This is just a myth. Anyone can learn to code, and indeed, more people should learn to code. Mark Zuckerberg and Bill Gates recently launched a video to get more kids into coding - if they think it's a good idea, so do we!



# What Skills Can Coding Teach You?

**THINKING  
COMPUTATIONALLY AND  
CREATIVELY**

Learning to code has a plethora of benefits for children - not only can it be hugely enjoyable and creative, but learning to code helps students think critically in a structured way, teaching them to think computationally to approach a wide variety of problems and solve them. This isn't just a great skill for software engineers, mathematicians and logicians it's useful for everyone and students will see benefits across a wide range of subjects. Coding is a great display of initiative & intelligence - tech is everywhere, everything is being automated and there is no better time to stay ahead of the curve.

# PROGRAMMING BENEFITS 101

## IT'S EXTREMELY ENGAGING

Often people don't realise how fun it is until they actually have a go at coding something. This is because the goal of coding is to build something that solves a problem, and this idea of building things is fundamental to why coding is fun - think of it like solving a logic puzzle (Sudoku), or building with lego.

## THINKING COMPUTATIONALLY

Programming languages are deterministic, and often require you to build up larger features out of smaller individual functions. This means that over time, you naturally end up thinking in a more computational way. This aids problem-solving efficiency and allows students to abstract away from specific problems & find general solutions that apply in other situations.

## THINKING CREATIVELY

Working out how to solve a problem requires creativity. Programming is the same - there are many ways to approach and solve the same problem. No two programs are written the same way; in fact, experienced coders even seem to develop their own style of coding.

You can also use code creatively when making art. A huge range of art is created digitally - think about every Pixar film you've ever seen.

## HIGHER EDUCATION & CAREERS

If you want to attend an elite university, you should expect to have in-person interviews, where you will be asked about your skills and interests, such as coding. Coding opens up a huge range of career opportunities. The McKinsey Global Institute even reports that by 2030, 800 million jobs worldwide will have been replaced by robotic automation. You could be the person who programs all the robots.



FROM THEORY TO PRACTICE



# Programming Masterclasses

PROJECTS & GAMES

# Coding a Maze

This activity helps beginners to think like a programmer. The task starts by challenging the student to write a series of commands ('left', 'right', 'forward' etc) to guide a lego figure (or similar) step by step through a maze, which can be built from anything but lego works well. Students quickly realise that it's a bit tedious to command the figure to move forward 7 times in a row, it's much easier to say "Do this next command 7 times." This naturally introduces the concept of 'for' or 'while' loops. The progression of this task challenges the student to write a general set of instructions to solve the maze put in front of them. Maze variations ensure the task can be tailored to any age or experience.

If the student tries to implement these general set of instructions on another maze it is unlikely to be able to solve the maze. As an extension task, more experienced students can be prompted to think about how they could adapt their set of instructions to solve any maze, no matter the complexity or size.

This task aims to develop the programmer's 'way of thinking' and allows students of any age to learn coding concepts without needing to learn a text-based programming language. In addition, this task helps younger children to think from a point of reference different than their own. The child's left may not be the same as the figure-in-the-maze's left, even the youngest children can learn this valuable skill of switching reference frame.

# Morse Code

Morse code is one of the simplest and most versatile methods of telecommunication in existence. It's been used for more than 160 years (longer than any other electronic encoding system!) Morse code encodes text by replacing letters with dots and dashes. Typically, these dots and dashes are transmitted using electric current but they can also be generated by turning a light on and off, tapping an object or blowing on a whistle making it a useful mode of communication in an emergency.

This task requires the student to develop a program that automatically converts English text input by a user into Morse Code dots and dashes and vice versa. As an extension exercise the program could be further developed to play the encrypted Morse code text over the speakers of a computer.

During this task the student will learn various fundamentals of Python programming:

- Working with 'strings'
- Conditional Execution
- For' Loops
- Defining and calling functions
- 'Print' statements

This task would suit any programmer from complete beginner to intermediate as there is plenty of scope to simplify or make more complex depending on the student.



# Ceasar CIPHER

The Ceasar cipher is one of the earliest known and simplest of ciphers and is a type of substitution cipher where the letters of the alphabet are shifted by a certain number of places down the alphabet. The number of places the letters are shifted is known as the key and can be from 1 to 25. For example, a shift of 1 would mean that A is shifted to B, and B to C...etc.

The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, here is a Caesar cipher using a left rotation or 'key' of three places:


Plain:     ABCDEFGHIJKLMN**OP**QRSTUVWXYZ  
Cipher:    XYZABCDEFGHIJKLMN**OP**QRSTUVWXYZ

When encrypting, a person looks up each letter of the message in the "plain" line and writes down the corresponding letter in the "cipher" line.

Plain text:  THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG  
Cipher text: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Deciphering is done in reverse, with a right shift of 3.





The Caesar cipher is named after Julius Caesar, who used the same shift of three to protect messages of military significance. While Caesar and his friends used two rotating disks containing the letters of the alphabet to encrypt and decrypt their letters, this task requires the student to develop a Python or R program to carry out the process. The program will prompt a user for a piece of text to be encrypted and the number of places to shift each letter otherwise known as the 'key'. The program will return an encrypted version of the text which can only be decrypted by someone who has the cipher key. The program will also be able to decrypt the response to ensure the reply is readable.

During this task the student will learn various fundamentals of Python programming:

- Working with 'strings'
- Conditional Execution
- 'For' Loops
- Defining and Calling Functions
- 'Print' Statements

This task would suit any programmer from complete beginner to intermediate as there is plenty of scope to simplify or make more complex depending on the student.

# Rock - Paper - Scissors

## HUMAN VS COMPUTER

In this task, the student will develop a Python program which can play Rock - Paper - Scissors (RPS) against a human player. They will utilise the fact that humans struggle to be truly random to ensure the program will beat a human opponent more than half the time. The program you create must implement the rules of RPS and provide the framework for the game to be played against a human player. In addition, the program must keep a record of the previous selections and use the information to make a weighted choice in an attempt to defeat the human opponent.

While completing this task, the student will learn various fundamentals of Python programming:

- Importing packages
- Working with values and variables
- Conditional Execution
- Loops
- 'Print' statements

This task would suit a student with some python programming experience or a beginner looking for something to work on alongside learning the basics.

# FizzBuzz

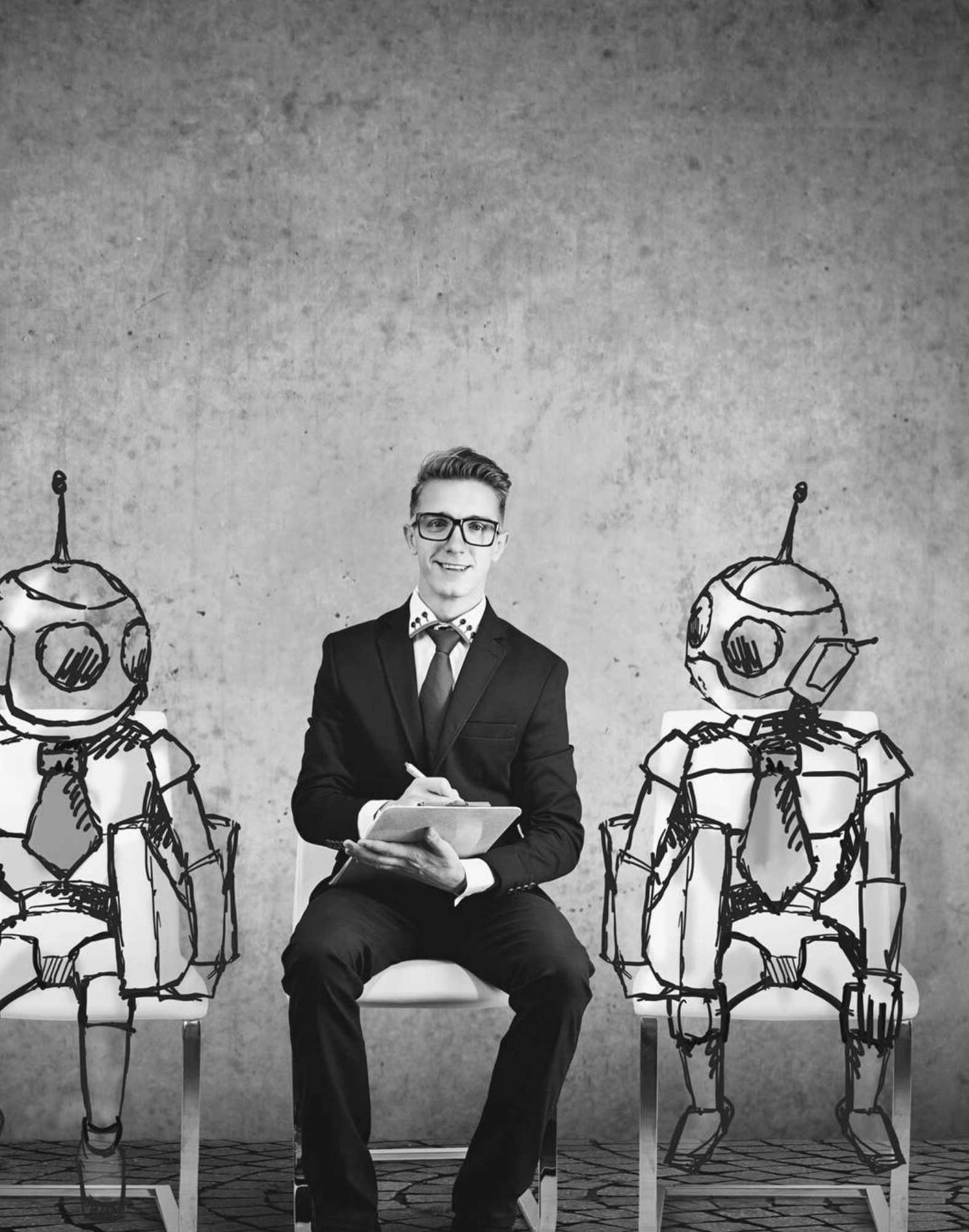
## THE TRUE TEST OF A PROGRAMMER

FizzBuzz is a game for at least 2 people. In its simplest form the players take it in turns to count from 1 to 100, but multiples of three are replaced with 'Fizz', multiples of five are replaced with 'Buzz' and multiples of both three and five are replaced with 'FizzBuzz'. This task challenges the student to develop a program to replace one human player. The program they create must implement the rules of FizzBuzz, provide the framework for the game to be played against a human player and check the answers input by the human opponent during the game. In addition, the program must keep a record of the 'high score' - the highest number achieved by the human opponent playing the game without an error.

While completing this task, the student will learn various fundamentals of Python programming:

- Importing packages
- Working with values and variables
- Conditional Execution
- Loops
- 'Print' statements

This task suits a student with some python or R programming experience or a beginner looking for something to work on alongside learning the basics.



11-18

INTERMEDIATE+

# Go Fish

Go Fish is a classic card game for at least two players. The rules can vary slightly, but generally each player is dealt nine cards and aims to complete sets by requesting cards from the opposition. On their turn, a player asks their opponent for a given rank (e.g. such as three kings). If the opponent has any cards of the named rank they must hand over all such cards and the player can ask again. If the opponent has no cards of the named rank, the requester draws a card from the deck and their turn ends. Whenever a player completes a set, it is removed from their hand. The game ends when every book is complete and the player with the most books wins.

Creating a python program to enable a user to play against a computer at 'Go Fish' is a fairly complex task. It involves implementing the rules of the game, the game prompts and interfaces as well as keeping track of the cards in the remaining deck and the hands of the players.

While completing this task, the student will build on the fundamentals of Python programming:

- Importing packages
- Defining and calling functions
- Working with values, variables and classes
- Conditional Execution
- 'While' and 'For' Loops
- 'Print' statements

This task suits a competent student with significant Python experience.



# Battleships

11-18

INTERMEDIATE+

## A LOGICAL AND CRITICAL THINKING GAME

Battleships is a two player board game. Each player positions ships on a 9 x 9 grid hidden from view of the other player. The players take it in turns to suggest positions of the ships on the other players board, if they hit one of the opponent's ships they are allowed to go again, if they miss their goes is over. Both players keep a record of their hits and misses and the game is over when one player has exposed all of the other player's ships.

The student will develop a program that allows a computer to play against a human. As well as building the game framework, the program must take input coordinates as selected by the human, output 'hit' or 'miss' as well as keep a record of the previous hits.

During this task the student will learn various fundamentals of Python programming:

- Working with 'strings' and 'integers'
- Conditional Execution
- 'For' Loops
- Defining and calling functions
- Print' statements