Python Turtle (functions)

Beginner level Forth lesson

Table of Contents

[What's a Function? 🤔🌈 2](#_Toc153367507)

[**Anatomy of a Function (Its Parts) 🤖🔍** 2](#_Toc153367508)

[**Example: 🌟✨** 2](#_Toc153367509)

[Creating an Interactive Turtle Drawing! 🐢🌟 3](#_Toc153367510)

[Creating an Interactive Turtle Rainbow Drawing! 🐢🌈 5](#_Toc153367511)

[🌈 Starting Your Turtle Town Adventure! 🏠🐢 7](#_Toc153367512)

[**🌟 Congratulations!** 9](#_Toc153367513)

[This will be an excellent time to take a short break before starting the next hour's lesson. 11](#_Toc153367514)

[🏓 Making a Ping Pong Game with Python Turtle 🐢 12](#_Toc153367515)

[**🌟 Now You're Ready!** 16](#_Toc153367516)

[🌟 Coding Adventure Recap with Python Turtle 🐢🏓 18](#_Toc153367517)

[**🚀 Reflecting on Your Coding Journey** 18](#_Toc153367518)

[**Keep Experimenting and Having Fun! 🎨👾** 18](#_Toc153367519)

## What's a Function? 🤔🌈

Imagine you have a robot friend who loves to help you with tasks. You can teach this robot to do something special, like draw a picture, solve a math problem, or even dance a little jig. In the world of coding, this robot friend is called a "function"!

A function in programming is like a mini-adventure story that your computer can follow. It's instructions that tell your computer exactly what to do to complete a specific task, like drawing a shape or adding numbers. The best part? Once you teach your computer how to do this task, it can do it repeatedly without you having to repeat the instructions!

**Anatomy of a Function (Its Parts) 🤖🔍**

Let's break down what makes up a function using our robot friend analogy:

1. **Name of the Function**: This is like the robot's name. It's how we tell our computer which instructions we want to use. For example, if our function draws a star, we might name it draw\_star.
2. **Parameters (Sometimes)**: These are like special instructions or choices you give your robot. For instance, if your function is about drawing shapes, a parameter might be the shape's size. Not all parts need these, but they're helpful when giving your robot different options.
3. **The Instructions**: This is the heart of the function. It's a list of steps that the robot (or your computer) will follow to complete the task. Just like how you follow the steps in a recipe.
4. **Return (Sometimes)**: After the function finishes its task, it might give something back to you. This could be an answer to a math problem, a drawing, or even a "Task Complete!" message. Not all functions return something, but when they do, it's like getting a little gift back from your robot friend!

**Example: 🌟✨**

Let's say we have a function named draw\_star. When you tell your computer to use draw\_star, it follows the instructions to draw a star on the screen. If you ask it to draw a big or small star, those are your parameters – like telling your robot to draw a star "this big" or "that small".

So, a function is like having a super smart robot friend on your computer, ready to help you whenever needed. You need to tell it what to do, and it's ready to go on an adventure of tasks and creativity! 🤖💫🎨

## Creating an Interactive Turtle Drawing! 🐢🌟

1. **Start Your Canvas:**
   * Let's get our drawing tools ready! First, type import turtle to bring in the Turtle graphics.
   * Set up your drawing area with screen = turtle.Screen() and give your artwork a title using screen.title("Python Turtle Stars").
2. **Bring Your Turtle to Life:**
   * Create your turtle artist: star\_turtle = turtle.Turtle(). This turtle will draw all your cool stars.
   * Make your turtle's moves easy to follow by setting its speed: star\_turtle.speed(3).
3. **Draw the First Star:**
   * It's time to create a star! We'll use a particular function, draw\_star(size), that takes a length and draws a star for us.
   * To draw our first star, let's type draw\_star(50). This will make a star with a size of 50.
4. **Move and Draw More Stars:**
   * Let's draw more stars in different places!
   * Lift your turtle's pen using star\_turtle.penup(), move it to a new location with star\_turtle.goto(x, y) (replace x and y with numbers), then put the pen down with star\_turtle.pendown().
   * Now, draw another star with draw\_star(size) using a different dimension—for example, draw\_star(30).
5. **Fill the Sky with Stars:**
   * Keep moving your turtle and drawing stars of different sizes. Remember to pick up and put down the pen each time you move to a new spot!
6. **Close Your Art Gallery:**
   * Once you've filled your sky with beautiful stars, it's time to admire your work!
   * Your drawing will stay open until you click on the window. To close it, click anywhere inside the window.
7. **Experiment and Have Fun:**
   * You can change the size of the stars, move them to different places, or even change the colour of your turtle's pen with star\_turtle.color("colorname").
   * Try different things and see what amazing starry skies you can create!

🌟 **Congratulations, young artist!** You've learned how to create a beautiful starry sky using Python Turtle. Experiment with different colours, sizes, and positions to make your sky more magical! 🌌🎨

A screenshot of a computer

Description automatically generated

## Creating an Interactive Turtle Rainbow Drawing! 🐢🌈

1. **Start Your Canvas:**
   * Let's begin our colourful adventure! Type import turtle to bring the Turtle graphics into your project.
   * Set up your drawing area with screen = turtle.Screen() and give your rainbow masterpiece a title using screen.title("Python Turtle Rainbow").
2. **Bring Your Turtle to Life:**
   * Create your artistic turtle: rainbow\_turtle = turtle.Turtle(). This turtle will paint the rainbow in the sky.
   * Adjust your turtle's drawing speed for a better view of the rainbow creation: rainbow\_turtle.speed(2).
3. **Prepare Your Rainbow Colors:**
   * Rainbows have many colours! We'll use a list of colours to draw each part of the rainbow.
   * Define the rainbow colours with rainbow\_colors = ["red", "orange", "yellow", "green", "blue", "indigo", "violet"].
4. **Create a Function to Draw Rainbow Arcs:**
   * We need a particular function to draw each coloured arc of the rainbow.
   * Let's make a function named draw\_semi\_circle(radius, colour). This function will draw a semi-circle with a given radius and colour.
5. **Paint the Rainbow:**
   * Start with a radius (size) for the innermost arc. Let's say radius = 50.
   * Use a loop to draw each semi-circle of the rainbow. For each colour in rainbow colours, your turtle will draw an arc and then prepare to draw the next one.
6. **Watch the Rainbow Come to Life:**
   * Run your code and watch your turtle artist create a beautiful rainbow, one colour at a time.
   * Each arc will be bigger than the last, making a whole rainbow.
7. **Finish Your Art:**
   * After the rainbow is complete, your turtle can take a rest. Hide it using rainbow\_turtle.hideturtle().
   * To close your artwork, click on the turtle window. That's when screen.exitonclick() comes into play.
8. **Experiment and Have Fun:**
   * Try changing the colours or the size of the rainbow by adjusting the rainbow\_colors list and the radius value.
   * What happens if you add more colours or change the order of the colours?

🌈 **Congratulations, young artists!** You've just created a vibrant rainbow using Python Turtle. This project combines coding skills with your creativity, making learning fun and colourful. Keep experimenting to see all the different rainbows you can create! 🎨🐢

A screenshot of a computer

Description automatically generated

## 🌈 Starting Your Turtle Town Adventure! 🏠🐢

1. **Setting Up Your Drawing Environment**

* First, we need to prepare our drawing tools. We're bringing in the Turtle graphics library and some randomness for colours.
* import turtle import random

1. **Creating the Drawing Window**

* Next, we set up the space where our Town will be drawn.
* screen = turtle.Screen() screen.title("Python Turtle Town")

1. **Bringing Your Turtle to Life**

* Now, we will introduce our turtle artist, who will draw.
* town\_turtle = turtle.Turtle() town\_turtle.speed(3)

1. **Preparing Colors and Drawing Functions**

* We create a list of colours and define functions to draw squares and triangles to make up our houses.
  + colours = ["red", "green", "blue", "yellow", "purple"]
  + def draw\_square(size): for \_ in range(4): Town\_turtle.forward(size) town\_turtle.right(90)
  + def draw\_triangle(size): for \_ in range(3): town\_turtle.forward(size) Town\_turtle.left(120)

1. **Functions to Build Houses and Grass**

* We have special functions to create houses and grass.
  + def draw\_house(x, y, size, house\_color, roof\_color): town\_turtle.color(house\_color)  
    town\_turtle.begin\_fill()  
    town\_turtle.penup()  
    town\_turtle.goto(x, y)  
    town\_turtle.pendown()  
    draw\_square(size)  
    town\_turtle.end\_fill()  
      
    # Draw the roof of the house  
    town\_turtle.color(roof\_color)  
    town\_turtle.begin\_fill()  
    town\_turtle.penup()  
    town\_turtle.goto(x, y) # Move to the top of the square  
    town\_turtle.pendown()  
    draw\_triangle(size)  
    town\_turtle.end\_fill()
  + def draw\_grass(x, y, width, height): Town\_turtle.color("green")  
    town\_turtle.begin\_fill()  
    town\_turtle.penup()  
    town\_turtle.goto(x, y)  
    town\_turtle.pendown()  
    for \_ in range(2):  
     town\_turtle.forward(width)  
     town\_turtle.right(90)  
     town\_turtle.forward(height)  
     town\_turtle.right(90)  
    town\_turtle.end\_fill()
* **Explanation**:
  + draw\_house: This tells our turtle how to draw a house with a specific colour and size.
  + draw\_grass: This function draws green grass around our houses.

1. **Setting Up and Drawing the Town**

* We defined how big the houses should be and how much space should be between them, and then we drew the Town.

house\_size = 50 space\_between\_houses = 20 grass\_margin = 5

* + def draw\_town\_setup(house\_index, house\_size, space\_between\_houses, grass\_margin): # Instructions to set up each house house\_x = house\_index \* (house\_size + space\_between\_houses) - (3 \* house\_size + 2.5 \* space\_between\_houses)  
      
    # Draw grass under the house  
    draw\_grass(house\_x - grass\_margin,  
     -house\_size - grass\_margin \* 10,  
     house\_size + 2 \* grass\_margin,  
     grass\_margin)  
      
    # Draw the house with randomly chosen colors  
    draw\_house(house\_x,  
     -house\_size,  
     house\_size,  
     random.choice(colors), # Random color for the house  
     random.choice(colors)) # Random color for the roof

for i in range(6): draw\_Town\_setup(i, house\_size, space\_between\_houses, grass\_margin)

* **Explanation**:
  + house\_size, space\_between\_houses, grass\_margin: These decide our Town layout.
  + draw\_town\_setup: This function builds each part of our Town, one house at a time.
  + for i in range(6): draw\_Town\_setup(i, house\_size, space\_between\_houses, grass\_margin)

We're building 6 houses in our Town.

1. **Wrapping Up Your Town**

* After the drawing was done, our turtle artist hid, and we kept the window open to admire our Town.
* **Code**:
* pythonCopy code
* town\_turtle.hideturtle() screen.exitonclick()
* **Explanation**:
  + town\_turtle.hideturtle(): Our turtle takes a rest after all the hard work.
  + screen.exitonclick(): This lets us keep looking at our Town until we click the window.

**🌟 Congratulations!**

You've just built a whole Turtle Town using code! Each part of this code worked like magic to create colourful houses, green grass, and fun. Feel free to play around with the colours, house sizes, or even the number of houses to see what new Towns you can create! 🎨🏡🐢

A screenshot of a computer

Description automatically generated

# This will be an excellent time to take a short break before starting the next hour's lesson.

## 🏓 Making a Ping Pong Game with Python Turtle 🐢

1. **Setting Up the Game Basics**

* First, we need to get ready for our game. We bring in the Turtle graphics library and set some basic rules.
* import turtle # Constants PLAYER\_A\_SCORE = 0 PLAYER\_B\_SCORE = 0 WINDOW\_WIDTH = 800 WINDOW\_HEIGHT = 600 PADDLE\_SPEED = 0 PADDLE\_STRETCH\_WID = 5 PADDLE\_STRETCH\_LEN = 1 PADDLE\_MOVE\_STEP = 90 BALL\_MOVE\_X = 3 BALL\_MOVE\_Y = 3
* **Explanation**:
  + We import the turtle module to use Turtle graphics.
  + We set constants like window size, paddle speed, and ball movement speed. These are like the rules of our game.

1. **Creating the Game Window**

window = turtle.Screen()  
window.title('Ping Pong')  
window.bgcolor('black')  
window.setup(width=WINDOW\_WIDTH, height=WINDOW\_HEIGHT)

* **Explanation**:
  + We made a new window for our game, named it 'Ping Pong', and chose 'black' for the background colour.

1. **Displaying the Score**

* We create a Turtle object to show the players' scores on the screen.
  + pen = turtle.Turtle() pen.speed(PADDLE\_SPEED) pen.color('blue') pen.penup() pen.hideturtle() pen.goto(0, 260)
  + def display\_score(): pen.clear() pen.write("Player A: {} Player B: {}".format(PLAYER\_A\_SCORE, PLAYER\_B\_SCORE), align="center", font=('Arial', 24, 'normal')) display\_score()
* **Explanation**:
  + The pen is like a little robot that writes the score at the top of the screen.
  + display\_score is a function that updates and shows the current score.

1. **Creating the Ball and Paddles**

* We make a ball for the game and two paddles for the players.
  + ball = turtle.Turtle() ball.shape('circle') ball.color('orange') ball.penup() ball.goto(5, 5)
  + # Initialize Right Paddle
  + right\_paddle = turtle.Turtle() right\_paddle.speed(PADDLE\_SPEED) right\_paddle.shape('square') right\_paddle.color('white') right\_paddle.shapesize(stretch\_wid=PADDLE\_STRETCH\_WID, stretch\_len=PADDLE\_STRETCH\_LEN) right\_paddle.penup() right\_paddle.goto(350, 0)
  + # Initialize Left Paddle
  + left\_paddle = turtle.Turtle() left\_paddle.speed(PADDLE\_SPEED) left\_paddle.shape('square') left\_paddle.color('white') left\_paddle.shapesize(stretch\_wid=PADDLE\_STRETCH\_WID, stretch\_len=PADDLE\_STRETCH\_LEN) left\_paddle.penup() left\_paddle.goto(-350, 0)
* **Explanation**:
  + The ball is orange and round, like a small sun.
  + right\_paddle and left\_paddle are the paddles players use to hit the ball. They are white squares.

1. **Moving the Paddles**

* When players press keys, we write functions to move the paddles up and down.
  + def moveRightPadleUp(): y = right\_paddle.ycor() y = y + 90 right\_paddle.sety(y)
  + def moveRightPadleDown(): y = right\_paddle.ycor() y = y - 90 right\_paddle.sety(y)
  + def moveLeftPadleUp(): y = left\_paddle.ycor() y = y + 90 left\_paddle.sety(y)
  + def moveLeftPadleDown(): y = left\_paddle.ycor() y = y - 90 left\_paddle.sety(y)
* **Explanation**:
  + These functions change the position of the paddles. sety changes the Y-coordinate (up and down position) of the paddles.

1. **bSetting Up Keyboard Controls**

* We tell the game to listen to keyboard presses and move the paddles accordingly.
* **Code**:
* pythonCopy code
* window.listen() window.onkey(moveLeftPadleUp, 'w') window.onkey(moveLeftPadleDown,'s') window.onkey(moveRightPadleUp, 'Up') window.onkey(moveRightPadleDown, 'Down')
* **Explanation**:
  + window.listen() prepares the game to respond to key presses.
  + window.onkey() sets which keys ('w', 's', 'Up', 'Down') control the paddles.

1. **Making the Ball Move and Bounce**

* We create a function to make the ball move and bounce off the edges and paddles.
  + def move\_ball():
  + global PLAYER\_A\_SCORE, PLAYER\_B\_SCORE, BALL\_MOVE\_X, BALL\_MOVE\_Y

* + # Move the ball
  + ball.setx(ball.xcor() + BALL\_MOVE\_X) ball.sety(ball.ycor() + BALL\_MOVE\_Y) # Border checking and paddle collision
  + # Top and bottom  
    if ball.ycor() > 290 or ball.ycor() < -290:  
     BALL\_MOVE\_Y \*= -1
  + # Left and right  
    if ball.xcor() > 390:  
     ball.hideturtle()  
     ball.goto(0, 0)  
     ball.showturtle()  
     BALL\_MOVE\_X \*= -1  
     PLAYER\_A\_SCORE += 1  
     display\_score() elif ball.xcor() < -390:  
     ball.hideturtle()  
     ball.goto(0, 0)  
     ball.showturtle()  
     BALL\_MOVE\_X \*= -1  
     PLAYER\_B\_SCORE += 1  
     display\_score()
  + # Paddle collision  
    if (ball.xcor() > 340 and ball.xcor() < 350) and (  
     ball.ycor() < right\_paddle.ycor() + 50 and ball.ycor() > right\_paddle.ycor() - 50):  
     ball.setx(340)  
     BALL\_MOVE\_X \*= -1

if (ball.xcor() < -340 and ball.xcor() > -350) and (  
 ball.ycor() < left\_paddle.ycor() + 50 and ball.ycor() > left\_paddle.ycor() - 50):  
 ball.setx(-340)  
 BALL\_MOVE\_X \*= -1

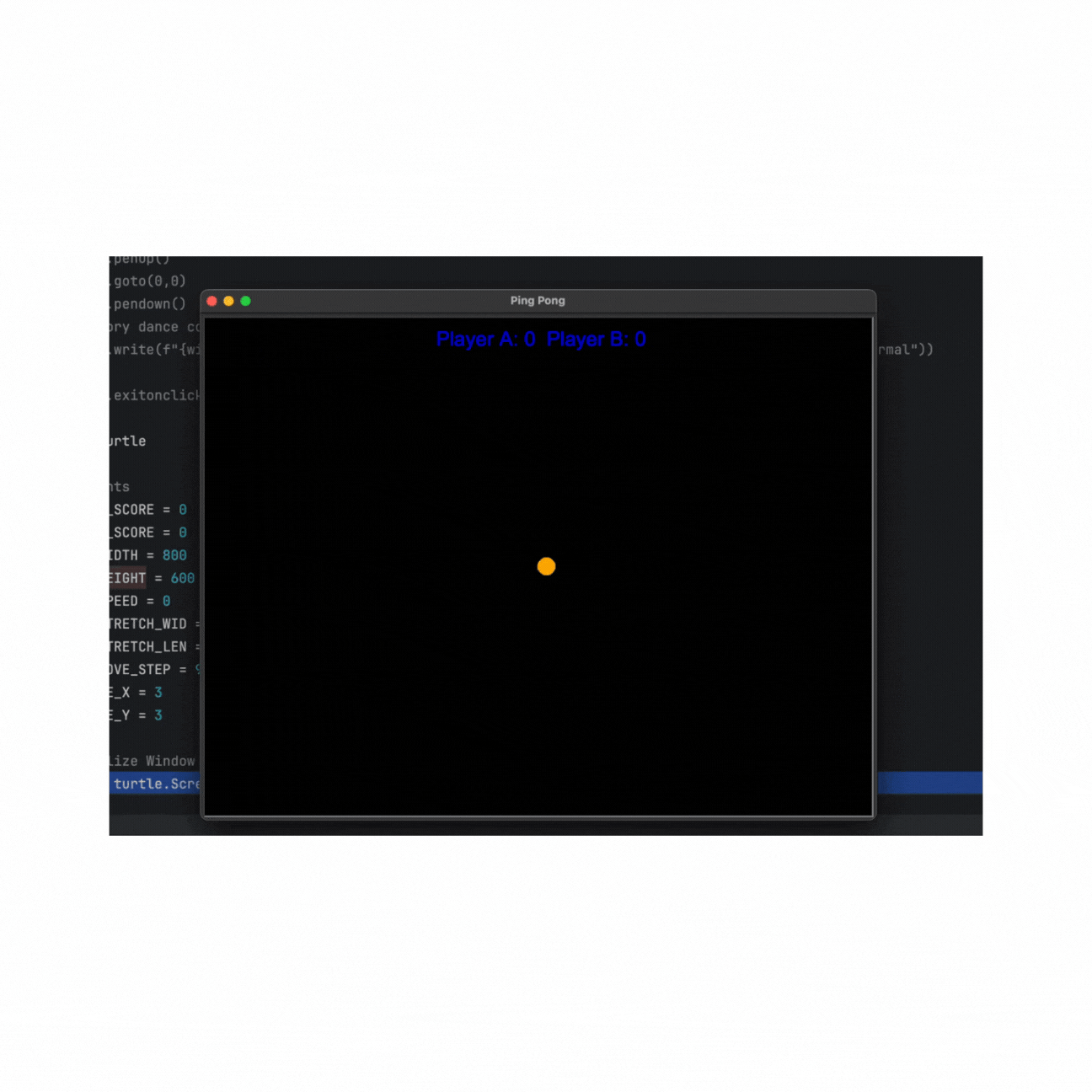
* + window.update() move\_ball()
* **Explanation**:
  + move\_ball() makes the ball move around the screen. setx and sety change the ball's position.
  + The ball bounces off the edges and paddles, and the score updates when it passes a paddle.

1. **Ending the Game**

* Finally, the game ends when someone clicks on the window.
* pythonCopy code
* window.exitonclick()
* **Explanation**:
  + window.exitonclick() ends the game when you click on the window.

**🌟 Now You're Ready!**

With this code, you've created a Ping Pong game! Each part of the code works together to make a fun and interactive game. You can try playing it with a friend. Can you keep the ball bouncing and score points? Have fun and explore more with coding! 🎉👩‍💻👨‍💻🏓



### 🌟 Coding Adventure Recap with Python Turtle 🐢🏓

**Congratulations, Brilliant Coders! 🎉**

In today's lesson, you ventured further into the world of coding with Python Turtle and achieved some incredible feats! Let's look back at what you've mastered:

1. **Discovering Functions**:
   * You've learned how to create and use functions in Python. Functions are like your personal coding helpers – you tell them what to do once, and they can do it repeatedly whenever you ask!
   * **Key Concept**: Understanding how to write functions and why they're useful in programming.
2. **Exploring Turtle Coordinates (xcor(), ycor())**:
   * You explored using xcor() and ycor() with Python Turtle. These functions help you find where your Turtle is on the screen, which is handy for making more precise movements and drawings.
   * **Key Concept**: Become familiar with coordinates and how they help position the Turtle.
3. **Learning about shapesize()**:
   * With shapesize(), you discovered how to change the size of your Turtle, making it bigger or smaller. This added a new level of creativity to your drawings and games.
   * **Key Concept**: Use shapesize() to adjust the appearance of your Turtle.
4. **Creating a Ping Pong Game**:
   * The highlight was building your very own Ping Pong game! You used all the skills you learned to make a game where you can control paddles and bounce a ball. How cool is that?
   * **Key Concept**: Combining functions, coordinate understanding, and size adjustments to create an interactive and fun game.

**🚀 Reflecting on Your Coding Journey**

In this lesson, you've not only learned new coding skills but also how to apply them in a fun and practical way. Remember, each line of code you write is a step towards becoming a more skilled coder. The Ping Pong game is a testament to your hard work and creativity!

**Keep Experimenting and Having Fun! 🎨👾**

These new skills are just the beginning. Try adding new features to your Ping Pong game. You could keep score or change the ball's colour. There's so much more to explore and create.

Great work, everyone! You're making amazing progress on your coding adventure. Keep practising, keep playing, and who knows what incredible games or projects you'll develop next! 🐢💻🎈🏓