Ricardo Rosa

The Apex Project

Turtle Python decision making (if, elif, else)

Biginner level lesson 2

Table of Contents

[Hour 1: 3](#_Toc153364746)

[Understanding the goto Method and the X-Y Axis 3](#_Toc153364747)

[Creating a Turtle's Interactive Story Adventure (decision making)! 6](#_Toc153364748)

[This will be an excellent time to take a short break before starting the next hour's lesson. 9](#_Toc153364749)

[Hour 2: 10](#_Toc153364750)

[Creating a Turtle Race Game in Python!: 10](#_Toc153364751)

[Hello, young coders! Today, we're going to have an exciting turtle race. You'll get to guess who wins, and we'll learn about random, shape="turtle", penup(), pendown(), randint(), goto, and the == operator. Ready? Let's dive in! 10](#_Toc153364752)

[1. **Setting Up Our Race**: 10](#_Toc153364753)

[ **Importing Libraries**: We start with importing turtles and import random ones. This gives us tools to create turtles and generate random numbers. 10](#_Toc153364754)

[ **Preparing the Screen**: We create our race track with screen = turtle.Screen() and name our race using screen.title("Turtle Race"). 10](#_Toc153364755)

[2. **Creating Our Racers**: 10](#_Toc153364756)

[ **Turtle Shapes**: We created two turtles named turtle1 and turtle2 with turtle.Turtle(shape="turtle"). This makes them look like cute turtles! 10](#_Toc153364757)

[ **Coloring Our Turtles**: We color turtle1 blue and turtle2 red using turtle1.color("blue") and turtle2.color("red"). 10](#_Toc153364758)

[3. **Positioning the Turtles**: 10](#_Toc153364759)

[ **Lifting the Pen**: We lift their pens using penup() so they don't draw while going to the start line. 10](#_Toc153364760)

[ **Moving to the Start Line**: We use goto(-200, 20) for turtle1 and goto(-200, -20) for turtle2. The goto method moves the turtle to a specific position on the screen. The first number is for left/right, and the second is for up/down. 10](#_Toc153364761)

[ **Putting the Pen Down**: With pendown(), they're ready to race and draw their path. 10](#_Toc153364762)

[4. **Guess Who Wins**: 10](#_Toc153364763)

[ **Your Guess**: A pop-up will ask you to guess the winner. We use screen.textinput to collect your guess. Type 'blue' or 'red' and press Enter. 10](#_Toc153364764)

[5. **Start the Race**: 10](#_Toc153364765)

[ **Moving Forward**: We move each turtle a random distance with random.randint(1, 500). This decides how many steps each turtle takes. 10](#_Toc153364766)

[6. **Who's the Fastest?**: 10](#_Toc153364767)

[ **Checking Positions**: We check who went further using turtle1.xcor() and turtle2.xcor(). The xcor() function tells us the turtle's position along the x-axis. 10](#_Toc153364768)

[ **Declaring the Winner**: We compare your guess with the winner using the == operator. If your guess matches the winner, we print a congratulatory message. If not, we tell you who won. 11](#_Toc153364769)

[7. **Wrapping Up**: 11](#_Toc153364770)

[ **Ending the Game**: Finally, we end the race with turtle.done(). 11](#_Toc153364771)

[Think and Try: 11](#_Toc153364772)

[ What happens if you type a different colour for your guess? 11](#_Toc153364773)

[ Experiment with the numbers in randint(1, 500). How does it change the race? 11](#_Toc153364774)

[Great job! You've learned how to create a turtle race game and discovered new programming concepts. What other cool games can you think of using these ideas? Let your creativity flow! 11](#_Toc153364775)

[Creating Your Custom Shapes with Python Turtle! 12](#_Toc153364776)

[Creating a Colourful House with Python Turtle! 14](#_Toc153364777)

[Lesson Recap: 18](#_Toc153364778)

[Our Turtle Coding Quest! 18](#_Toc153364779)

Beginner level second lesson

Tutor

# Hour 1:

## Understanding the goto Method and the X-Y Axis

**Objective**: Learn how the goto method moves the turtle to different points on the screen using the x and y axes.

#### Exercise Steps:

1. **Introduction to the X-Y Axis**:
   * Explain that the screen is like a map, with an x-axis (left and right) and a y-axis (up and down). The centre of the screen is where x=0 and y=0.
   * Show a simple diagram of the x-y axis.
2. **Starting with Turtle**:
   * Start by creating a turtle object with t = turtle.Turtle().
3. **First Movement - Going Up**:
   * Ask the students to move the turtle up by 100 steps. They should use the goto method with (0,100).
   * Example Code: t.goto(0, 100)
   * This moves the turtle to a point directly above the centre (0 on the x-axis and 100 on the y-axis).
4. **Moving to the Right**:
   * Next, move the turtle to the right using goto(100, 100).
   * Example Code: t.goto(100, 100)
   * This will take the turtle to a point 100 steps right of the centre and still 100 steps up.
5. **Going Down**:
   * Now, ask them to bring the turtle down. Use goto(100, 0).
   * Example Code: t.goto(100, 0)
   * The turtle will move straight down to be level with the centre but still 100 steps to the right.
6. **Returning to the Center**:
   * Finally, have the turtle return to the centre (0,0) using goto(0, 0).
   * Example Code: t.goto(0, 0)
   * The turtle comes back to its starting point.
7. **Discussion**:
   * Discuss how changing the numbers in goto(x, y) moves the turtle to different places. Larger x values move it right, and smaller (or negative) x values move it left. Larger y values move it up, and smaller (or negative) y values move it down.
8. **Experiment**:
   * Please encourage students to try their goto commands to see where the turtle ends. They can use positive and negative values for x and y.

#### Conclusion:

This exercise helps students visualize and understand the concept of coordinates and movement on a 2D plane, which is fundamental in graphics programming and mathematics.

A screenshot of a computer

Description automatically generated

## Creating a Turtle's Interactive Story Adventure (decision making)!

Hey there, super coders! Today, we're embarking on an exciting adventure with our turtle friend, where we'll learn about making choices in coding. Get ready to guide our turtle through a story with your decisions!

1. **Prepare the Stage**: First, let's set up our coding stage. We write import turtle to bring in the Turtle library, then screen = turtle.Screen() to get our screen ready, and t = turtle.Turtle() will call our turtle artist to the stage. Think of it like opening the curtain for a play!
2. **Making Choices**: In this adventure, our turtle will choose to go left or right. We'll ask you, the coder, to make this choice. We use option = screen.textinput("Choose a Path", "Does the turtle go 'left' or 'right'?").lower() to get your answer.
   * **What It Does**: This function is like asking a question and waiting for an answer. When we use screen.textinput("Choose a Path", "Does the turtle go 'left' or 'right'?"), a small window pops up, asking the user to make a choice.
   * **Input Values Explained**:
     + The first input, "Choose a Path", is the title of the pop-up window.
     + Our question is the second input, "Does the turtle go 'left' or 'right'?". The user can type their answer here.
3. **Turning Left or Right**: Our turtle will turn and move based on your choice. If you choose 'left', our turtle turns left with t.left(90) and moves forward. If you choose 'right', it turns right with t.right(90) and moves forward. We use t.forward(100) to move our turtle 100 steps forward in either direction.
4. **Revealing the Outcome**: We'll reveal what happens after the turtle moves. If it went left, maybe it found a treasure! We use t.write("The turtle found a treasure!", font=("Arial", 16, "normal")). If it went right, perhaps it met a new friend! We write a message using the same t.write() function.
   * **What It Does**: t.write is like our turtle holding a pen and writing a message on the screen. It's used to display text where the turtle is.
   * **Input Values Explained**:
     + The first input, like "The turtle found a treasure!", is the message we want to write.
     + The second input, font=("Arial", 16, "normal"), tells us about the font style, size, and type. In this case, we're using the Arial font, size 16, and the style is normal (not bold or italic).
5. **What if it's a Wrong Choice?**: Sometimes, we might not make the right choice. If the choice is neither 'left' nor 'right', we'll tell you it's not valid and to try again with t.write("Oops! That's not a valid choice. Try again.", font=("Arial", 16, "normal")).
6. **Wrapping Up the Adventure**: After the adventure, we use turtle.done() to let our program know we're done with our story.

Think and Try:

* What other choices can you add to the story? Maybe 'forward' or 'backward'?
* Try adding more outcomes based on different decisions. What fun things can the turtle discover?

You're Amazing! You've just guided a turtle through an interactive story using coding! You learned about making decisions with if-else statements. What other adventures can you create with these new skills? Let your imagination guide you!

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.

# Hour 2:

## Creating a Turtle Race Game in Python!:

### Hello, young coders! Today, we're going to have an exciting turtle race. You'll get to guess who wins, and we'll learn about random, shape="turtle", penup(), pendown(), randint(), goto, and the == operator. Ready? Let's dive in!

### **Setting Up Our Race**:

### **Importing Libraries**: We start with importing turtles and import random ones. This gives us tools to create turtles and generate random numbers.

### **Preparing the Screen**: We create our race track with screen = turtle.Screen() and name our race using screen.title("Turtle Race").

### **Creating Our Racers**:

### **Turtle Shapes**: We created two turtles named turtle1 and turtle2 with turtle.Turtle(shape="turtle"). This makes them look like cute turtles!

### **Coloring Our Turtles**: We color turtle1 blue and turtle2 red using turtle1.color("blue") and turtle2.color("red").

### **Positioning the Turtles**:

### **Lifting the Pen**: We lift their pens using penup() so they don't draw while going to the start line.

### **Moving to the Start Line**: We use goto(-200, 20) for turtle1 and goto(-200, -20) for turtle2. The goto method moves the turtle to a specific position on the screen. The first number is for left/right, and the second is for up/down.

### **Putting the Pen Down**: With pendown(), they're ready to race and draw their path.

### **Guess Who Wins**:

### **Your Guess**: A pop-up will ask you to guess the winner. We use screen.textinput to collect your guess. Type 'blue' or 'red' and press Enter.

### **Start the Race**:

### **Moving Forward**: We move each turtle a random distance with random.randint(1, 500). This decides how many steps each turtle takes.

### **Who's the Fastest?**:

### **Checking Positions**: We check who went further using turtle1.xcor() and turtle2.xcor(). The xcor() function tells us the turtle's position along the x-axis.

### **Declaring the Winner**: We compare your guess with the winner using the == operator. If your guess matches the winner, we print a congratulatory message. If not, we tell you who won.

### **Wrapping Up**:

### **Ending the Game**: Finally, we end the race with turtle.done().

### Think and Try:

### What happens if you type a different colour for your guess?

### Experiment with the numbers in randint(1, 500). How does it change the race?

### Great job! You've learned how to create a turtle race game and discovered new programming concepts. What other cool games can you think of using these ideas? Let your creativity flow!

A screenshot of a computer

Description automatically generated

### Creating Your Custom Shapes with Python Turtle!

Hey, young coders! We will draw some cool shapes with Python Turtle, and you can decide their size and type! Let's create a circle or a square with your specifications. Follow these steps to start:

1. **Setting Up the Canvas**:
   * **Start by Launching Turtle**: We begin with import turtle. This brings in our Turtle toolkit.
   * **Prepare Your Screen**: We set up our drawing screen with screen = turtle.Screen() and give our project a title using screen.title("Size and Shape Challenge").
2. **Creating Your Turtle Artist**:
   * **Bring the Turtle to Life**: We create a turtle named t with t = turtle.Turtle(). Think of 't' as your paintbrush!
3. **Choosing Your Brush Size**:
   * **Decide on the Thickness**: A box asking for a size number (1-10) will pop up. Type in a number and press Enter. This number will decide how thick your turtle's lines will be.
4. **Picking a Shape to Draw**:
   * **Select Your Shape**: Another box will ask you to choose a shape. Type '1' for a circle or '2' for a square, then press Enter.
5. **Customizing Your Shape**:
   * **For a Circle**: If you chose a circle, you'll be asked for the radius (how big around the circle is). Type in a number for the radius and press Enter.
   * **For a Square**: If you chose a square, you'll be asked for the side length (how long each side of the square is). Type in a number for the side length and press Enter.
6. **Drawing Time**:
   * **Watch Your Turtle Create**: Your turtle will start drawing based on your choices. For a circle, it'll use the radius you gave. For a square, it'll use the side length you provided to make each side of the square.
7. **Finishing Your Art**:
   * **Admire Your Work**: You can see the shape you created once the turtle is drawn. It's all based on your input!
8. **Closing the Canvas**:
   * **End the Program**: After admiring your shape, the program ends with turtle.done().

Think and Try:

* What happens if you choose a large number for the size or the radius/side length?
* Can you modify the program to draw other shapes?

You're doing great! You just learned how to create shapes based on your inputs. Experiment with different sizes and shapes to see what you can create!

A screenshot of a computer

Description automatically generated

### Creating a Colourful House with Python Turtle!

Hey there, coding adventurers! Ready to draw a beautiful house with a roof, a door, and a window? Follow these steps to make your very own cosy house using Python Turtle. We'll also add colours and details like a door handle and a window cross!

**Explore Colours with Fill**: The begin\_fill() and end\_fill() functions work together to fill a shape with colour. When you call begin\_fill(), Turtle starts remembering its path. After you draw the shape and call end\_fill(), Turtle fills the entire area enclosed with the colour you've set with fillcolor(). What new colours can you try, and which parts of the house could they fill?

1. **Set Up Your Canvas**:
   * Open your Python environment and import Turtle with import turtle.
   * Create your drawing screen with screen = turtle.Screen() and give your artwork a title like "Draw a House".
2. **Meet Your Turtle Artist**:
   * Make a new turtle named 't' using t = turtle.Turtle(). This Turtle will be your brush for painting the house.
   * Set your Turtle's drawing speed to slow with t.speed(1). This makes it easier to watch your house being drawn.
3. **Drawing the House Base**:
   * Move your Turtle to the starting point with t.penup() and t.goto(0, 0).
   * Get ready to draw and fill the base of the house with colour. Use t.pendown() t.fillcolor("brown") for a lovely wooden look, and begin the fill with t.begin\_fill().
   * Draw a square by moving the Turtle backwards and turning right to make each side with t.backward(200) and t.right(90).
   * Complete the square and fill it with t.end\_fill().
4. **Adding the Roof**:
   * Change the roof colour with t.fillcolor("orange") and start the fill with t.begin\_fill().
   * Draw two diagonal lines to make a pointy roof. Move your Turtle forward with t.forward(141.4) and turn with t.left(90).
   * Finish the roof and fill the colour with t.end\_fill().
5. **Creating the Door**:
   * To draw the door, move the Turtle to where the door starts with t.penup() and t.goto(-60, -100).
   * Start coloring the door blue with t.fillcolor("blue") and t.begin\_fill().
   * Make a rectangle by moving forward and turning left to draw each side with t.forward() and t.left(90).
   * Finish the door and fill it with blue using t.end\_fill().
6. **Adding the Door Handle**:
   * Put the handle on the door with t.penup(), t.goto(-20, -150), and t.pendown().
   * Draw a small dot for the handle using t.dot(10).
7. **Drawing the Window**:
   * Move your Turtle to its position for the window with t.penup() and t.goto(-60, 0).
   * Start filling in the window with blue using t.fillcolor("blue") and t.begin\_fill().
   * Draw a square for the window with t.forward(50) and t.left(90) for each side.
   * Complete the window and fill it with t.end\_fill().
8. **Making the Window Cross**:
   * Draw the horizontal part of the cross by moving to t.penup(), t.goto(-60, -25), and t.pendown(). Then go forward with t.forward(50).
   * For the vertical part, move your Turtle up with t.penup(), t.goto(-85, 0), and t.pendown(). Then draw the line with t.left(90) and t.forward(50).
9. **Show Off Your House**:
   * When you're all done, hide the Turtle with t.hideturtle() to get a clear view of your house.
   * Keep your masterpiece on the screen with Turtle.done().

Think and Try:

* **Experiment with the House Size**: What happens if you change the numbers when setting the house size, the door, or the window? Try using more significant numbers for t.forward() commands. How does this affect the overall look of your house?
* **Play with the Roof Shape**: Can you change the shape of the roof? What if you make the top flatter or much steeper? Adjust the lines' angles and length to see how it changes.
* **Add More Features**: Could you add a garden or a chimney to your house? Use the exact steps you learned to create new shapes. Draw a rectangle for a chimney or circles for flowers.

You're doing wonderfully! Today, you've learned how to draw a house with a Python Turtle, fill shapes with colour, and change the size and angles to modify your drawing. Experiment with different colours, conditions, and sizes to see what else you can create!

A screenshot of a computer

Description automatically generated

Lesson Recap:

### Our Turtle Coding Quest!

Bravo, coding champions! What a journey we've embarked on in the world of Python Turtle. We've brought imagination to life on our screens with each line of code. Let's take a stroll down memory lane and recall the incredible tools we've added to our coding toolbox:

1. **Decision-Making with if and else**: We've become digital decision-makers, using if and else statements to guide our turtle through choices and adventures, creating different paths and outcomes based on our inputs.
2. **Mastering the goto Method**: With the goto method, we've learned to move our turtle to exact spots on our canvas, understanding how the x and y coordinates chart our turtle's journey across the screen.
3. **Navigating the X-Y Axis**: We've explored the Cartesian plane, using the X-Y axis to position our turtle precisely and create structured drawings.
4. **We were communicating with turtle.write**: Our turtle became a storyteller, using turtle.write to send messages and label parts of our drawings, enhancing our understanding of the text in graphics.
5. **We are interacting with screen.textinput**: We discovered how to interact with users, asking for their input to influence our turtle's path and the shapes it draws.
6. **Playing with Randomness**: With randint(), we've introduced the spice of randomness into our projects, making each code run a unique display.
7. **Understanding Boolean Operators**: We've delved into boolean logic, using operators like <, >, ==, and != to compare values and make decisions in our code.
8. **Changing Turtle Shape**: We've personalized our turtle, changing its shape to match our artistic vision and giving our programs a fun twist.
9. **Colouring with begin\_fill() and end\_fill()**: We've filled our shapes with beautiful colours, using begin\_fill() and end\_fill() to paint the inside of the figures we draw, bringing vibrancy to our creations.

You've all done a spectacular job, and I'm thrilled with the progress each of you has made. Remember, each command you learn is like a new colour on your palette, ready to paint whatever worlds your mind can dream up.

Great Work, Everyone! Hats off to you for your hard work and creativity! You're turning into a true coding artist, and I'm anticipating where your newfound skills will take you. Keep practising; soon, you'll be coding up a storm that'll dazzle everyone who sees it.

I'm looking forward to our next lesson, which promises to be even more exciting, challenging, and fun!

Keep reviewing these concepts, try combining them in new ways, and don't hesitate to experiment with your ideas. Your creativity is the limit!