Overview

In this lesson, you will learn how to move a turtle around the screen using movement and rotation commands. You will also learn the difference between **moving** to a position and **teleporting** to a position, and how the turtle screen's coordinate system works.

Important Information

The turtle moves by following commands that describe **distance**, **direction**, and **position**.

Moving Forward and Backward

To move the turtle in the direction it is facing, use:

- `forward(distance)`
- `backward(distance)`

The **distance** is measured in **pixels**.

- Pixels are tiny dots on your screen.
- Larger numbers move the turtle farther.

Example:

- `forward(100)` moves the turtle forward 100 pixels.

Turning Left and Right

To rotate the turtle, use:

- `left(angle)`
- `right(angle)`

The **angle** is measured in **degrees**.

- 360 degrees is a full circle.
- 90 degrees is a quarter turn.

Example:

- `right(90)` turns the turtle 90 degrees to the right.

Shorthand Movement Commands

Turtle also provides shorter versions of the movement commands:

- `fd(distance)` → forward
- `bk(distance)` → backward
- `lt(angle)` → left
- `rt(angle)` → right

These shorthand commands behave exactly the same as the full versions.

Moving to a Specific Position (goto)

To move the turtle to a specific screen position **while drawing**, use:

- `goto(x, y)`

With `goto`:

- The turtle moves at its normal speed.
- The pen stays down by default.
- A line is drawn from the current position to the new position.

This command is useful when you want the movement itself to be visible.

Teleporting to a Position (teleport)

To instantly move the turtle to a position **without animation**, use:

- `teleport(x, y)`

With `teleport`:

- The turtle instantly appears at the new location.
- No movement animation occurs.
- No line is drawn, regardless of the pen state.

This is useful for repositioning the turtle before drawing something new.

Understanding the Screen Coordinate System

The turtle screen uses an **x–y coordinate system**:

- `(0, 0)` is the center of the screen.
- Positive **x** moves right.
- Negative **x** moves left.
- Positive **y** moves **down**.
- Negative **y** moves **up**.

This feels inverted compared to a typical math (Cartesian) graph.

Why the Y-Axis Appears Flipped on Computers

On a computer screen:

- The **x-axis** runs left to right.
- The **y-axis** runs top to bottom.
- The **z-axis** points **forward, out of the screen toward you**.

Because screens are treated as flat surfaces facing the user, coordinates are defined in a way that aligns with how pixels are stored in memory—starting at the top-left and moving downward. This makes drawing graphics faster and simpler for the computer, even though it looks flipped compared to math graphs.

Set Up

Create a new Python file called `turtle_movement.py`.

Copy, Change, Challenge

Copy

Copy and run the following code.

```python
import turtle

t = turtle.Turtle()

t.forward(100)
t.right(90)
t.forward(50)

turtle.done()
```

Watch how the turtle moves and turns.

Change

Modify the program so that:

- The turtle moves backward at least once
- Only shorthand commands (`fd`, `bk`, `lt`, `rt`) are used

Run the program again and observe the result.

Challenge

Add code that:

1. Uses `goto(x, y)` to move the turtle to a position while drawing.
2. Uses `teleport(x, y)` to instantly move the turtle to a new location.

Try both positive and negative y-values and explain why the turtle moves up or down based on how computer screens handle coordinates.