12. Turtle Module (Intro)

Importing the Turtle Module

The first step in using turtle graphics is to import the `turtle` module into your Python script. You can do this by adding the following line at the beginning of your code:

```
import turtle
```

This line imports the `turtle` module, giving you access to all its functions and methods.

Creating a Turtle Object

After importing the `turtle` module, you need to create a turtle object. This object represents the virtual turtle that will draw on the canvas. You can create a turtle object by calling the `Turtle()` function:

```
t = turtle.Turtle()
```

This line creates a turtle object and assigns it to the variable `t`. You can name the variable anything you like, but it's a common convention to use a single letter for turtle objects.

Moving the Turtle

Now that you have a turtle object, you can start giving it instructions to move around the canvas. The `turtle` module provides several methods for moving the turtle, such as `forward()`, `backward()`, `left()`, and `right()`.

Here's an example that makes the turtle draw a square:

```
import turtle

t = turtle.Turtle()

# Draw a square
t.forward(100)  # Move the turtle 100 units forward
t.left(90)      # Turn the turtle 90 degrees to the left
t.forward(100)
```

```
t.left(90)
t.forward(100)
t.left(90)
t.forward(100)
t.left(90)
```

In this example, we move the turtle forward by 100 units, turn it 90 degrees to the left, and repeat these steps four times to create a square shape.

Customizing the Turtle

The `turtle` module allows you to customize the appearance and behavior of the turtle. You can change the turtle's shape, color, and speed, among other properties.

Here's an example that changes the turtle's shape and color:

```
import turtle

t = turtle.Turtle()

# Change the turtle's shape and color
t.shape("turtle")  # Set the turtle's shape to a turtle icon
t.color("green")   # Set the turtle's color to green
```

In this example, we set the turtle's shape to a turtle icon using `t.shape("turtle")` and change its color to green using `t.color("green")`.

Drawing Shapes and Patterns

One of the strengths of turtle graphics is its ability to draw various shapes and patterns with minimal code. The `turtle` module provides several methods for drawing shapes, such as `circle()`, `polygon()`, and `dot()`.

Here's an example that draws a spiral pattern:

```
import turtle

t = turtle.Turtle()

# Draw a spiral pattern
for i in range(100):
```

```
    t.forward(i * 2)
    t.left(30)
```

In this example, we use a `for` loop to repeat the following steps 100 times:

1. Move the turtle forward by a distance that increases with each iteration (`i * 2`).
2. Turn the turtle 30 degrees to the left.

This creates a spiral pattern on the canvas.

Controlling the Canvas

The `turtle` module also provides functions for controlling the canvas itself, such as setting the window size, background color, and title.

Here's an example that sets the canvas properties:

```
import turtle

# Set the canvas properties
turtle.setup(800, 600)     # Set the window size to 800×600 pixels
turtle.bgcolor("lightblue") # Set the background color to light blue
turtle.title("My Turtle Graphics")  # Set the window title

t = turtle.Turtle()

# Draw something on the canvas
#  ...
```

In this example, we use the `turtle.setup()` function to set the window size, `turtle.bgcolor()` to set the background color, and `turtle.title()` to set the window title.

Keeping the Window Open

By default, the turtle graphics window will close as soon as the script finishes executing. To keep the window open and view your drawing, you can add the following line at the end of your script:

```
turtle.done()
```

This line tells Python to keep the turtle graphics window open until you manually close it.

0. Built-in Data Types (RECOMENDED)