The Apex Project

Turtle Programming(Turtle, variables)

Beginner level first lesson

Table of Contents

[Hour 1: 3](#_Toc153277479)

[Understanding the python library: 3](#_Toc153277480)

[Turtle Programming: 3](#_Toc153277481)

[Let’s Draw with a Turtle! 4](#_Toc153277482)

[Drawing shapes: 7](#_Toc153277483)

[Square: 7](#_Toc153277484)

[Triangle: 10](#_Toc153277485)

[Stairs: 13](#_Toc153277486)

[Hexagon: 16](#_Toc153277487)

[Cross: 19](#_Toc153277488)

[This will be an excellent time to take a short break before starting the next hour's lesson. 22](#_Toc153277489)

[Hour 2: 23](#_Toc153277490)

[Green Turtle: 23](#_Toc153277491)

[**Drawing with a Green Turtle:** Let's Make a Cool Shape! 23](#_Toc153277492)

[Starburst: 26](#_Toc153277493)

[**Create a Starburst with a Red Turtle!** 26](#_Toc153277494)

[Yellow Star: 29](#_Toc153277495)

[**Drawing a Yellow Star in the Night Sky!** 29](#_Toc153277496)

[**Drawing Rainbow Stairs with Turtle!** 32](#_Toc153277497)

[Pink Circle: 35](#_Toc153277498)

[**Drawing a Pink Circle with Our Turtle Friend!** 35](#_Toc153277499)

[Flower: 38](#_Toc153277500)

[**Draw a Flower with a Turtle!** 38](#_Toc153277501)

[Lesson Recap: 41](#_Toc153277502)

[At the end of the lesson, make sure you complement the students and let them know they have done great, promising that the next lesson will be even more interesting, challenging, and fun. 41](#_Toc153277503)

Beginner level first lesson

Tutor

# Hour 1:

### Understanding the python library:

Before you begin your Python programming, you need to understand what a **library** is. In the non-computer world, a library is a place where different types of books are stored. You can access these books anytime, take whatever information you need from them, and return them to the same place.

In the computer world, a library works similarly. By definition, a **library** is a set of essential functions and methods that you can access to make your programming more accessible. The Python turtle library contains all the methods and functions needed to create your images. To access a Python library, you need to import it into your Python environment like this:

import turtle

Now that you have just welcomed a turtle into your digital world. The turtle is part of a graphical library, like an artist ready to create. To showcase this artist's work, you need a stage or a canvas, known in Python as the 'screen'. To set this stage, you'll use a variable.

Think of variables as boxes in the world of programming. Like a box is used to store items, a variable stores data or information you might need later in your code. When you give a value to a variable, it's like placing something inside this box for the first time - this is called initializing the variable. The interesting part? The contents of this box, or the variable's value, can change as your program runs, adapting to your coding story.

Now, to bring the turtle's canvas to life, you'll initialize a variable for the screen. Here's how you can do it:

s = turtle.getscreen()

A different window ought to appear and open:

### Turtle Programming:

Learning how to move the turtle in the desired direction is the first step in programming using the Python turtle module. You'll then discover how to personalise both your turtle's surroundings and itself. Lastly, you will pick up a few more commands that will allow you to carry out a few unique tasks.

Transferring the Turtle

Four directions are available for a turtle to move in:

* Forward
* Backward
* Left
* Right

The turtle travels in the direction it is facing, either forward or backwards. By slightly rotating t.left() t.right(), you can alter this direction. Try each of these instructions in the following manner:

t.right(90)  
t.forward(100)  
t.left(90)  
t.backward(100)

It is also possible to use shortened version of the command movements.

t.rt() # instead of t.right()  
t.fd() # instead of t.forward()  
t.lt() # instead of t.left()  
t.bk() # instead of t.backward()

### Let’s Draw with a Turtle!

Hey there, young coder! Today, we will use a turtle (not a real one, but a fun computer turtle!) to draw a cool shape. We’ll tell our turtle exactly where to go and what to do. Ready? Let’s get started!

1. **Meet Your Turtle**: First, we must bring our turtle to the screen. We do this by telling Python we want to use a Turtle. This is like picking your favourite toy out of the toy box.

import turtle  
  
# Create a turtle object  
t = turtle.Turtle()

1. **Setting Up**: Now, we will give our turtle some rules. We’ll tell it how far to walk and how much to turn. We’ll put these rules into something called variables. Think of them like instructions you write on a sticky note for the turtle to remember.

**Walking the Distance**: We have a rule called distance that tells our turtle how many steps to take. We set this to 100, like saying, “Take 100 turtle steps forward.”

# Define variables  
distance = 100

1. **Turning Around**: Next, we have an angle. This tells our turtle to turn a certain amount. We’ve told our turtle to turn 90 degrees, like saying, “Turn to your right until you’re facing the way your right hand is pointing.”

# Define variables  
angle = 90

1. **On the Move**: Now, we’ll put these rules into action. We say t.forward(distance) to make our turtle walk forward that many steps. It’s like saying, “Walk the number of steps on the sticky note labelled ‘distance’.”

# Use variables to move the turtle  
t.forward(distance) # Move forward by 'distance' units

1. **Right Turn**: We walk forward and tell our turtle to t.right(angle). This is like telling our turtle, “Turn to your right the number of degrees on the sticky note labelled ‘angle’.”

t.right(angle) # Turn right by 'angle' degrees

1. **Moving Again**: We want our turtle to walk again, so we repeat, telling it to go forward by the distance.

t.forward(distance) # Move forward by 'distance' units again

1. **What Happens?**: When we run this code, our turtle will draw a straight line, turn to the right, and then draw another line the same length. It’s like drawing a big “L” on the ground.

import turtle  
  
# Create a turtle object  
t = turtle.Turtle()  
  
# Define variables  
distance = 100  
angle = 90  
  
# Use variables to move the turtle  
t.forward(distance) # Move forward by 'distance' units  
t.right(angle) # Turn right by 'angle' degrees  
t.forward(distance) # Move forward by 'distance' units again  
  
turtle.done()

## Drawing shapes:

### Square:

**Let’s Draw a Square with Our Turtle Friend!**

Hello, young explorer! Today, we will be artists and use our computer to draw a square. But guess what? We’ll have a little turtle helper to draw for us. Let’s get started!

1. **Calling Our Turtle**: First things first, we need our turtle. When we write

t = turtle.Turtle()

,it’s like calling our turtle to come out and get ready to draw.

1. **Ready to Draw**: Our turtle is on the screen with its pen ready. It’s waiting for our instructions to start drawing.
2. **First Line**: We tell our turtle, t.forward(100). This means, “Hey turtle, walk forward 100 steps.” The turtle moves straight and draws a line as it goes.
3. **Turn Right**: After the first line, we need to turn to start the next side of the square. We say, t.right(90). This is like telling our turtle, “Turn to your right, a quarter turn.”
4. **Draw More Lines**: We repeat these steps three more times. Each time, our turtle walks another 100 degrees and then turns right. It’s like walking along the edges of a big square on the ground.
5. **Completing the Square**: After the fourth line, our turtle ends up where it started. Guess what? You’ve just drawn a square!
6. **All Done**: We finish with turtle.done(), which tells our turtle, “Great job! We’re finished drawing.”

**Let’s Think**:

* A square has four sides that are all the same length. That’s why we told our turtle to move forward 100 steps four times.
* We also turned 90 degrees each time to make those sharp square corners.

**Try This**:

* What if you change the number in t.forward(100) to a bigger or smaller number? What happens to your square?
* What if you turn a different amount, like t.right(45)? Can you still make a square?

**You Did It!** You’re a fantastic turtle whisperer and artist! You told the turtle exactly how to draw a perfect square. What other shapes might you want to draw? Give them a try!

A black arrow on a white background

Description automatically generated

Triangle:

**Drawing a Triangle with Our Turtle Pal!**

Hey there, young coder! Are you ready for a new drawing adventure? With the help of our computer turtle, we're going to draw a triangle. A triangle has three sides and three corners. Let's start!

1. **Meet Our Turtle**: First, we bring our turtle to the screen. When we write t = turtle.Turtle(), we're getting our turtle ready to draw.
2. **Start Drawing**: Our turtle is all set with its pen. Now, it's waiting to hear from us. What should it draw? A triangle!
3. **First Side of the Triangle**: We tell our turtle t.forward(100). This means, "Hey turtle, can you walk straight for 100 steps?" As the turtle moves, it draws a line behind it. That's the first side of the triangle!
4. **Turning Time**: Now, we need to turn to start the next side. We say, t.right(120). This tells our turtle to turn slightly above a right angle (90 degrees). Imagine turning to look a bit backwards – that's 120 degrees!
5. **Second Side**: We draw the second side just like the first. Again, we say t.forward(100). The turtle walks another 100 steps straight ahead, tempting as it goes.
6. **Turn Again**: It's turn time! Another t.right(120) to get our turtle facing the right way for the last side of our triangle.
7. **Last Side**: Finally, we draw the third side with t.forward(100). The turtle marches another 100 steps.
8. **All Done!** And just like that, we've got a triangle! We finish with turtle.done(), which is our way of saying, "Good job, turtle! We're drawn."

**Let's Think About It**:

* A triangle has three sides. That's why we instructed the turtle three times to move forward and turn.
* We turned 120 degrees each time, not 90 like in a square. That's because the corners of a triangle are wider.

**Try This**:

* What happens if you change the number in t.forward(100)? Make it 50 or 150.
* Can you make the turtle turn a different amount, like 90 degrees? What shape does it make, then?

**Great Work!** You're becoming a real turtle-commanding artist! You've drawn a triangle with the turtle. What other shapes can you make? Try experimenting to see what you can create!

A black arrow in a white background

Description automatically generated

Stairs:

**Let's Draw Stairs with Our Turtle Friend!**

We're going on a cool adventure with our computer turtle. We'll teach it to draw a set of stairs. Yes, stairs! Just like the ones you might climb every day. Ready to start? Here we go!

1. **Please bring in the Turtle**: First, we need our turtle. We do this with t = turtle.Turtle(). This is like saying, "Hey turtle, come out and let's draw!"
2. **Drawing the First Step**: Now, let's draw our first step. We tell our turtle t.forward(50). This means, "Please move forward 50 steps." Our turtle moves straight ahead and draws a line.
3. **Turning Upwards**: After drawing the base of the step, we need to go up. We tell our turtle, t.left(90). This is like telling our turtle to turn left and face upwards.
4. **Going Up the First Step**: Now we go up! We repeat t.forward(50). The turtle climbs up, drawing the rising part of the step.
5. **Turning to the Next Step**: We need our turtle to turn right to start the next step. We say t.right(90). It's like telling the turtle, "Turn your head right and get ready for the next step."
6. **Drawing the Second Step**: Like before, we draw the base of our second step with t.forward(50).
7. **Turning and Climbing Again**: We turn left to go up the next step with t.left(90) and climb with t.forward(50).
8. **One More Step**: We're still going! Let's turn right with t.right(90) and draw the base of our last step with t.forward(50).
9. **Finishing Up**: We've drawn three steps! Great job! We finish with turtle.done(), telling our turtle, "That's enough, thank you!"

**Think About It**:

* We drew the stairs by making the turtle walk straight and then turn. When we turn left, we go up; when we turn right, we go across.

**Try This**:

* Can you add more steps to your stairs? Just repeat the pattern of going straight and turning.
* What happens if you change the number 50 to a bigger number? Do the steps get bigger?

**You're Amazing!** You've just drawn stairs with a turtle! You're becoming a fantastic turtle guide and a great artist. What else can you draw? Try making your patterns!

A black arrow on a white background

Description automatically generated

Hexagon:

**Drawing a Hexagon with Our Turtle Buddy!**

Hey there, young coding friends! Today, we're going to do something extraordinary. We're going to teach our computer turtle to draw a hexagon. What's a hexagon, you ask? It's a shape with six sides, like a honeycomb! Let's start drawing!

1. **Call Our Turtle**: First, we must bring our turtle to help us draw. We do this by writing t = turtle.Turtle(). This is like saying, "Hello, turtle, let's make some art!"
2. **Starting the Hexagon**: Our turtle is ready with its pen. Now, it's waiting to hear from us. Let's draw the first side of our hexagon. We tell our turtle, t.forward(70). This means, "Please walk forward 70 steps."
3. **First Turn**: After the first line, we must turn our turtle to start the next side. We say, t.right(60). This tells our turtle to turn right just a bit. Imagine looking slightly to your right – that's 60 degrees!
4. **Drawing More Sides**: We keep telling our turtle to go forward and then turn. We do these five more times to make all six sides of our hexagon. We say t.forward(70) and then t.right(60) each time.
5. **Complete the Hexagon**: Our turtle will have made a perfect hexagon after the sixth side. It looks like a flat bee's honeycomb or a cool badge!
6. **All Done!** We've drawn a hexagon! To finish, we write turtle.done(), like telling our turtle, "Great work, we've finished our drawing!"

**Think About It**:

* A hexagon has six sides, all the same length. We told our turtle to move forward 70 steps six times.
* After each side, we turn 60 degrees. This makes the corners of our hexagon.

**Try This**:

* What happens if you change the t.forward(70) number to something else? 100 or 50?
* Can you make the turtle turn a different amount, like 90 degrees? What shape do you get, then?

**Fantastic Job!** You're becoming a master at guiding the turtle to draw shapes! What other shapes can you create with your turtle? Try different turns and distances to see what you can make!

A black arrow in a white background

Description automatically generated

Cross:

**Fun with a Blue Turtle: Drawing cross**

Hey, excellent young coders! Today, we will create a super cool drawing on the computer using our friend, the turtle. We will make it even more exciting by adding a beautiful colour - blue! Let's dive in and start our colourful coding adventure!

1. **Bring Out Your Turtle**: First, let's call our turtle to the screen. We do this with t = turtle.Turtle(). It's like inviting our turtle out to play and draw.
2. **Choosing a Color**: Before we start drawing, let's pick a colour for our turtle. We create a 'colour name' by writing colour = 'blue'. It's like choosing a crayon from a box of paints. Then, we tell our turtle to use this colour by saying t.color(colour). Now, our turtle's pen is dipped in blue ink!
3. **Let's Start Drawing**: Our turtle is ready with its blue pen. We start by saying t.forward(50), which tells our turtle, "Walk forward 50 steps." It leaves a lovely blue trail behind.
4. **Making Turns**: After walking a bit, it's time to turn. We use t.left(90) and t.right(90) to tell our turtle how to ride. Imagine your turtle is at a crossroads and decides to turn left or right.
5. **Creating a Cross Pattern**: We keep telling our turtle to walk and turn. This makes a cool Cross pattern, like lightning bolts or a bumpy road in a cartoon!
6. **Finishing the Masterpiece**: We follow all these steps to create our Cross pattern. After the last line, we've made a super cool pattern on the screen.
7. **Time to Rest**: After all that drawing, our turtle needs a break. We write t.hideturtle() to let our turtle go off and rest. And turtle.done() is like saying, "Our drawing is complete, and it's awesome!"

**Think and Try**:

* We chose blue by setting colour = 'blue', but what if you try other colours? Maybe red, green, or yellow? Just change the colour name!
* Experiment with the pattern. What happens if you add more t.forward() and turns? Can you make a more giant Cross?

**You're Amazing!** Wow, you're not just a coder but an artist, too! You've created a beautiful Cross pattern with our turtle friend. What other creative drawings can you come up with? Have fun exploring!

A black and white background

Description automatically generated

import turtle  
  
#  
t = turtle.Turtle()

colour = 'blue'  
t.color(colour) # Set the turtle's color to blue  
  
t.forward(50)  
t.left(90)  
t.forward(50)  
t.right(90)  
t.forward(50)  
t.left(90)  
t.forward(50)  
  
# Finish drawing  
t.hideturtle() # Hide the turtle arrow  
turtle.done()

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

## **Hour 2:**

### **Green Turtle:**

### **Drawing with a Green Turtle:** Let's Make a Cool Shape!

Hello, young coding explorers! We’re going on an artistic adventure with our computer turtle. We'll draw a neat shape and paint our screen green with our turtle's help. Ready to start? Here we go!

1. **Call the Turtle to Play**: First, we must prepare our turtle. We do this by writing t = turtle.Turtle(). This is like inviting our turtle out to have some drawing fun.
2. **Pick a Color**: It's time to choose a colour for our drawing. We want to make our turtle's pen green, so we write a variable colour = "green" – that's like picking a green crayon. Then we tell our turtle, t.color(colour), like dipping the turtle's pen in green paint.
3. **Let's Begin Drawing**: Now that our turtle is ready with its green pen, it's time to draw. We start by turning our turtle a bit with t.left(40). It's like telling our turtle to look slightly to the left before it starts walking.
4. **Drawing a Square, but a Little Slanted**: We'll draw a square, but it'll look slightly slanted because of our first turn. We tell our turtle t.forward(100) to walk forward 100 steps and start drawing our green line.
5. **Turning and Drawing More**: After drawing the first line, we turn our turtle left with t.left(90) and walk forward again with t.forward(100). We do these three more times to make our square.
6. **Finish the Square**: We kept telling our turtle to turn left and walk forward until we had all four sides of our square. It will look like a diamond because of the way we started!
7. **All Done with Our Drawing**: We've drawn a cool, slanted square (or a diamond)! Now, it's time to rest. We tell our turtle to hide with t.hideturtle(), and we're all finished with turtle.done().

**Think and Try**:

* Why does our square look slanted? It's because we started by turning our turtle 40 degrees. What happens if you start with t.left(90)?
* Try changing the colour = "green" to another colour. What if it's blue, red, or yellow?

**Fantastic Work!** You did an excellent job! You guided your turtle to draw a cool shape and learned about turning and walking forward. What other shapes can you draw? Try starting with a different turn and see what happens!

A black arrow in a white background

Description automatically generated

Starburst:

### **Create a Starburst with a Red Turtle!**

Hey there, young coders! We will use our computer turtle to draw something exciting – a starburst! And we'll make it in a bright red colour. Let's see how we can guide our turtle to create this cool picture.

1. **Meet Our Turtle**: First, we bring out our turtle. We write t = turtle.Turtle() to get our turtle ready for some artistic fun.
2. **Choosing Our Color**: We want our drawing to be red, so we'll tell our turtle that. First, we pick the colour by creating a variable colour = 'red'. Then, we tell our turtle to use this colour with t.color(colour). It's like giving our turtle a red pen.
3. **Starting Our Starburst**: Now, let's start making our starburst. We first turn our turtle a bit by saying t.left(80). It's like telling our turtle to look slightly towards the left before starting.
4. **Drawing the Starburst Rays**: We will draw the rays of our starburst now. We tell our turtle t.forward(60) to move forward 60 steps. This makes one ray of our starburst.
5. **Turning and Drawing More Rays**: We'll turn our turtle and draw another ray after each ray. We use commands like t.right(70) and t.left(160) to turn our turtle in different directions. Each time we turn, we also move forward to draw another ray.
6. **Keep Going**: We repeat the pattern of turning and moving forward. This creates the beautiful, spiky rays of our starburst.
7. **Finishing Our Starburst**: We continue following the steps until we've drawn all the rays of our starburst. It will look like a big, bright, red sun or a firework!
8. **All Done!** We've created a fantastic starburst! We tell our turtle to hide because it's done a great job. We write t.hideturtle(), and then turtle.done() to finish our drawing.

**Think About It**:

* Our starburst looks cool because we keep turning and drawing in a pattern. Each ray is made by moving forward and then turning.
* We chose red for our starburst, but what if you pick a different colour? Try changing colour = 'red' to another colour like 'blue' or 'yellow'.

**Great Job!** You're an incredible turtle artist! You've made a bright, spiky starburst. What other patterns can you make? Try changing the angles or the length of the steps to create new designs!

A black arrow in a white background

Description automatically generated

import turtle  
  
t = turtle.Turtle()  
colour = 'red'  
t.color(colour) # Set the turtle's color to green  
  
  
t.left(80)  
t.forward(60)  
t.right(70)  
t.forward(60)  
t.left(160)  
t.forward(60)  
t.right(70)  
t.forward(60)  
t.left(160)  
  
# Finish drawing  
t.hideturtle() # Hide the turtle arrow  
turtle.done()

### Yellow Star:

### **Drawing a Yellow Star in the Night Sky!**

Hi there, young space explorers! We will use our computer turtle to draw a bright yellow star in the night sky. Let's start our cosmic art adventure!

1. **Setting Up the Night Sky**: First, let's create our night sky. We do this by setting the background colour to black. We create a variable bgCoulor = "black" and set the turtle.bgcolor( bgCoulor). It's like painting our computer screen black, like the night sky!
2. **Call Our Turtle to Draw**: Now, let's prepare our turtle. We write t = turtle.Turtle() to bring our turtle out. Our turtle is our artist today!
3. **Pick a Star Color**: Stars shine bright in the sky, right? Let's make our star yellow. We write colour = "yellow" to choose our star's colour. Then, we say t.color(colour) to give our turtle a yellow pen.
4. **Drawing the First Ray of the Star**: Let's start drawing our star. We move our turtle forward by writing t.forward(100). This draws the first ray of our star, a straight yellow line.
5. **Turning to Draw More Rays**: After the first ray, we must turn our turtle to draw the next one. We use t.right(144) to turn our turtle. It's like telling our turtle, "Turn a bit more than a right angle to get ready for the next ray."
6. **Creating a Star**: We keep moving forward and turning to make our star. We repeat the on and correct turn pattern four more times. Each time, we're adding another ray to our star.
7. **Completing Our Star**: Our star takes shape as we follow these steps. After the last turn, we have a beautiful five-pointed star!
8. **Finishing Our Space Art**: We're done! Our star is shining bright in the night sky. We tell our turtle to hide because it did a great job. We write t.hideturtle(), then turtle.done() to finish.

**Think and Try**:

* Our star is yellow because we chose that colour. What happens if you change colour = "yellow" to another colour like 'red' or 'blue'?
* Why does our star look perfect? It's because we turn our turtle by 144 degrees each time, which is the ideal angle for a five-pointed star.

**You Did It!** Wow, you just made a star! You're not just a coder but an artist, too! What other cool drawings can you make with your turtle? Try drawing a moon or a planet!

A black arrow in a white background

Description automatically generated

import turtle  
# Create turtle object 't'  
t = turtle.Turtle()  
bgCoulor = "black"  
turtle.bgcolor(bgCoulor)  
colour = "yellow"  
t.color(colour) # Set the turtle's color to blue  
  
# Start drawing the star  
t.forward(100) # Draw the first line of the star  
t.right(144) # Turn the turtle by 144 degrees to the right  
  
t.forward(100) # Repeat the pattern to complete the star  
t.right(144)  
  
# Finish drawing  
t.hideturtle() # Optionally hide the turtle arrow  
turtle.done()

**Drawing Rainbow Stairs with Turtle!**

Hello, young artists and coders! We will draw some magical stairs, each step in a different colour and with a thicker line than the last. Ready to make some art with coding? Let's jump right in!

1. **Meet Our Turtle Artist**: First, we need our turtle. We write t = turtle.Turtle(): Get our turtle ready for drawing. Think of the turtle as your paintbrush.
2. **Setting Up Our Paintbrush**: Before we start, we need to set our paintbrush size. We do this by creating a variable called pensize and assigning it to 1 (which means a thin line).
3. **Choosing Our First Color**: We will make our first stair step blue. We create a variable named colour and set it to 'blue'. Then, we tell our turtle to use this colour and our pensize by writing t.color(colour) and t.pensize(pensize).
4. **Drawing the First Step**: Let's draw our first step! We move our turtle forward by 50 degrees with t.forward(50), then turn it left to prepare for the next step.
5. **Changing Colors and Making the Brush Thicker**: For each new step, we will revise the colour and make our brush a bit thicker.
   * We changed our colour variable to a new colour, like 'red'.
   * We add to our pensize variable to make the line thicker. We write pensize+1, which means "add 1 to our original pensize."
   * We then tell our turtle to use the new colour and pensize.
6. **Creating a Rainbow Staircase**: We keep changing colours and increasing our pensize for each step. We use colours like 'red', 'green', 'yellow', and 'purple', making each step a different colour and slightly thicker than the last.
7. **Finishing Our Colorful Stairs**: After drawing all our steps with different colours and thicknesses, we've created a beautiful rainbow staircase!
8. **Great Job!** Our drawing is done, and it looks fantastic! We tell our turtle to hide with t.hideturtle() and finish with turtle.done().

**Think and Try**:

* Our staircase is colourful because we keep changing the colour variable. What other colours can you try?
* The lines get thicker because we add to the pensize each time. What happens if you make the lines even thicker?

**You're Awesome!** You've just created a rainbow staircase with coding! You learned how to change colours and make lines thicker. What other cool drawings can you create with these new skills?

A black arrow in a white background

Description automatically generated

import turtle  
  
t = turtle.Turtle()  
pensize = 1  
colour = 'blue'  
  
# Drawing stairs  
t.color(colour)  
t.pensize(pensize)  
t.forward(50)  
t.left(90)  
colour = 'red'  
t.color(colour)  
t.pensize(pensize+1)  
t.forward(50)  
t.right(90)  
colour = 'green'  
t.color(colour)  
  
t.hideturtle() # Optionally hide the turtle arrow  
turtle.done()

Pink Circle:

### **Drawing a Pink Circle with Our Turtle Friend!**

Hello, young coding artists! We will use our computer turtle to draw a pretty pink circle! Let's learn how to guide our turtle to create this lovely shape.

1. **Meet Your Turtle**: First, we need to get our turtle ready. We write t = turtle.Turtle() to bring our turtle out for drawing. Imagine this turtle as a fun drawing robot!
2. **Setting Up Our Pen**: We have a unique tool called a 'pen' that our turtle uses to draw. We will set the size of this pen to 1, which means a nice, thin line. We do this by creating a 'pensize' variable and assigning it to 1.
3. **Picking a Color**: Now, let's choose a colour for our drawing. We want our circle to be pink, so we create a variable called 'colour' and set it to 'pink'. Then we tell our turtle to use this colour for drawing by saying t.color(colour).
4. **Drawing a Circle**: It's time to draw! We tell our turtle to draw a circle with t.circle(90). This means, "Hey turtle, can you draw a circle that's 90 steps around?" Watch as our turtle draws a beautiful pink circle!
5. **All Done!** And just like that, we've drawn a pink circle! Isn't it pretty? We've created art with coding!

**Think and Try**:

* We chose pink for our circle, but what if you want a different colour? Try changing colour = 'pink' to another colour like 'blue' or 'green'.
* The number 90 in t.circle(90) is the size of our circle. What happens if you change it to a bigger number, like 120, or a smaller one, like 60?

**Fantastic Job!** You're an amazing turtle artist! You've learned how to draw a circle and change its colour. What else can you draw with your turtle? Try a different shape or colour next time!

A black arrow in a white background

Description automatically generated

Flower:

### **Draw a Flower with a Turtle!**

Hey there, young space adventurers! We will use our computer turtle to draw a beautiful flower with bright blue dots at each point. We'll learn how to use a new tool called .dot to make our flower even more magical. Ready to start? Let's go!

1. **Prepare Your Turtle**: First up, we need our turtle. We bring it out by writing t = turtle.Turtle(). Think of it as your little robot that loves to draw.
2. **Choosing a Color**: Let's pick a color for our star. We create a variable colour = “blue” t.color(colour) to give our turtle a blue pen. It's like telling our turtle, "Your pen is blue now, so let's make some blue magic!"
3. **Making a Dot**: Before drawing the flower, we will create a variable dotSize = 60 use t.dot(dotSize) to make a big blue dot. The number 60 tells our turtle how big it is to make the dot. It's like stamping a big blue spot to start our star.
4. **Drawing the Flower**: We start making our star by moving our turtle forward with t.forward(55). This marks a line for one arm of our flower.
5. **Adding More Dots**: After each line, we add another big dot with t.dot(dotSize). This creates a beautiful point on each end of our flower's arms.
6. **Turning to Make More Arms**: To make the next arm of the flower, we turn our turtle slightly by writing t.right(144). This is like telling our turtle, "Turn a bit, and let's draw the next part of our flower."
7. **Completing the Star**: We repeat drawing a line, adding a dot, and then turning to create each arm of our flower. We do this until we have five components, each with a bright blue dot at the end.
8. **Admiring Our Flower**: We've drawn all the arms of our flower, and it looks fantastic! It's a beautiful blue star with sparkling points.
9. **Finishing Up**: Our drawing is complete, and it's time for our turtle to rest. We write t.hideturtle() to let our turtle take a break, and turtle.done() to finish our drawing.

**Try This Out**:

* What happens if you change the colour in t.color("blue") to something else, like "red" or "green"? Can you make a rainbow flower?
* You can play with the size of the dots and the length of the lines. What happens if you make the dots bigger or the lines longer?

**Great Work!** You're a fantastic space artist! You've created a starry sky with your turtle. What other space wonders can you draw? A moon or a planet? Have fun exploring the universe with your turtle!

A screenshot of a computer

Description automatically generated

import turtle  
  
# Create turtle object 't'  
t = turtle.Turtle()  
colour = "blue"  
t.color(colour) # Set the turtle's color to blue  
dotSize = 60  
# Draw a dot at the starting point and the first line  
t.dot(dotSize)  
t.forward(55)  
t.dot(dotSize)  
  
# Turn and draw the second line  
t.right(144)  
t.forward(55)  
t.dot(dotSize)  
  
# Turn and draw the third line  
t.right(144)  
t.forward(55)  
t.dot(dotSize)  
  
# Turn and draw the fourth line  
t.right(144)  
t.forward(55)  
t.dot(dotSize)  
  
  
# Finish drawing  
t.hideturtle() # Optionally hide the turtle arrow  
turtle.done()

### Lesson Recap:

### At the end of the lesson, make sure you complement the students and let them know they have done great, promising that the next lesson will be even more interesting, challenging, and fun.

**Lesson Recap: Our Turtle Drawing Adventure!**

Hey there, super coders! Today, we had a fantastic time with our turtle friend, learning to draw fantastic shapes and patterns. Let's take a moment to look back at what we've created and learned:

1. **Meeting Our Turtle**: We started by calling out our turtle with t = turtle.Turtle(). This was like inviting a new friend to join us in our drawing adventure.
2. **Drawing Lines and Shapes**: We learned how to move our turtle to draw lines. Remember how we made the turtle move forward and turn to create different shapes? We drew lines, squares, triangles, and even stairs!
3. **Adding Color to Our Drawings**: We gave our turtle different coloured pens to make our drawings vibrant and fun. We used t.color(colour) to change colours and create our art pop!
4. **Using the .dot() Function**: We discovered how to use t.dot() to add beautiful dots to our drawings. This added a special touch to our star, making it look like it was shining bright.
5. **Creating a Star**: One of the coolest things we did was drawing a star! We used forward movements and right turns to make each point of the star. And those dots at the end of each line made it look extra special.
6. **Experimenting with Pensize**: We also played with the thickness of our turtle's pen using t.pensize(pensize). This made some lines thicker than others, adding an exciting twist to our drawings.
7. **Exploring and Having Fun**: We used our imagination to explore and create. We learned that coding with Turtle Graphics is not just about instructions; it's about having fun and being creative!

**What's Next?** You've done a fantastic job today! You're now on your way to becoming a Turtle Graphics expert. Remember, there's so much more you can do with your turtle. Try experimenting with different shapes, colours, and sizes. Next time, we can create a whole new world with our turtle!

**Great Work, Everyone!**: Thanks for joining in on this coding adventure. You're all fantastic artists and coders. I can't wait to see what we'll create in our next lesson!