FACULTY OF COMPUTING
& INFORMATION TECHNOLOGY
KING ABDULAZIZ UNIVERSITY

FCIT
KAU

كلية الحاسبات
وتقنية المعلومات
جامعة الملك عبدالعزيز

Extra Material
Python Turtle

CPIT 110 (Problem-Solving and Programming)

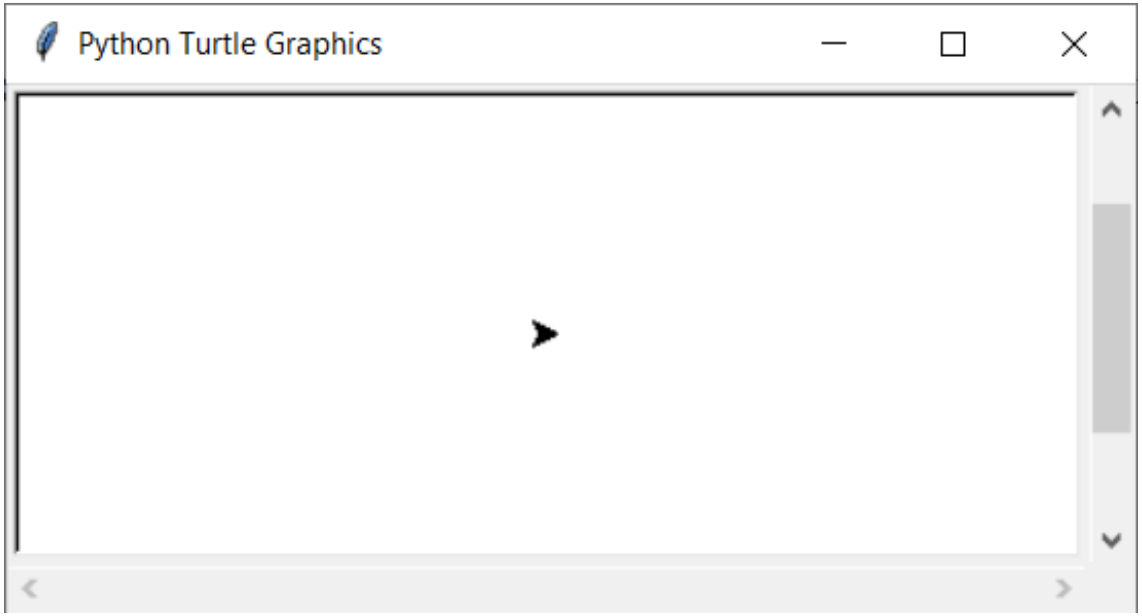
Ahmad Tayeb

Note

- This section is not included in the course syllabus (Not in exams and tests).
- This section will help you to learn programming and problem-solving skills visually.
- Enjoy ☺

Introduction to Python Turtle

- Python's turtle graphics system displays a small cursor known as a **turtle**.
- You can use Python statements to move the turtle around the screen, drawing lines and shapes.

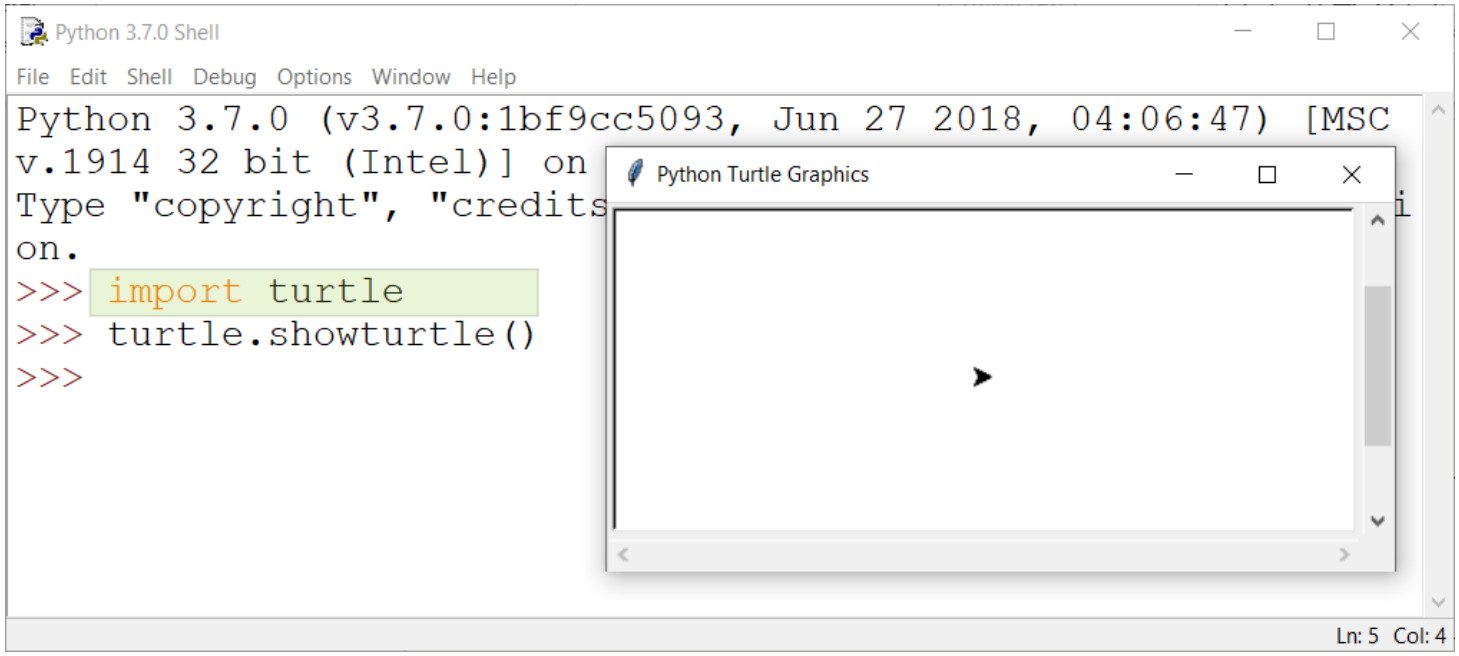


Import the Turtle Module

- To use the turtle graphics system, you must import the turtle module with this statement:

```
import turtle
```

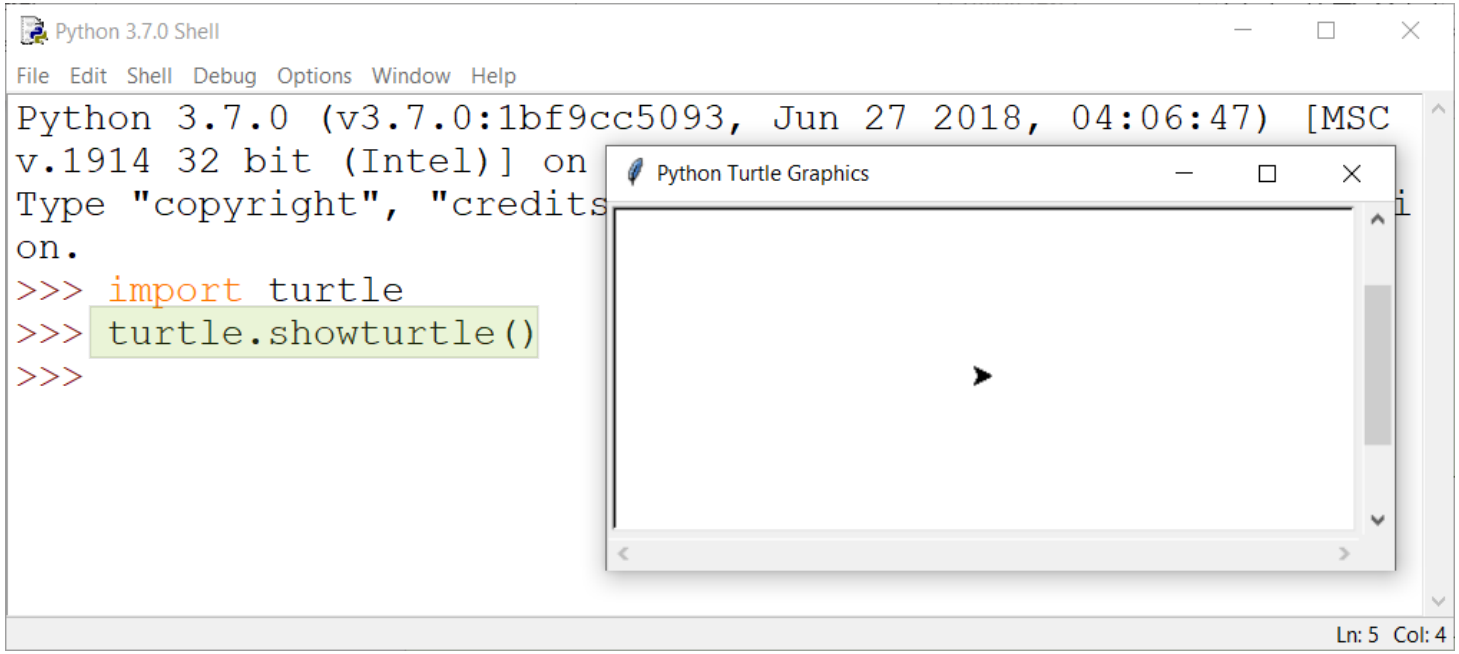
- This loads the turtle module into memory.



Showing Turtle Graphics Window

- After importing the turtle module, you can use the turtle.st() or turtle.showturtle() function to show the turtle window.

```
import turtle
turtle.showturtle()
```



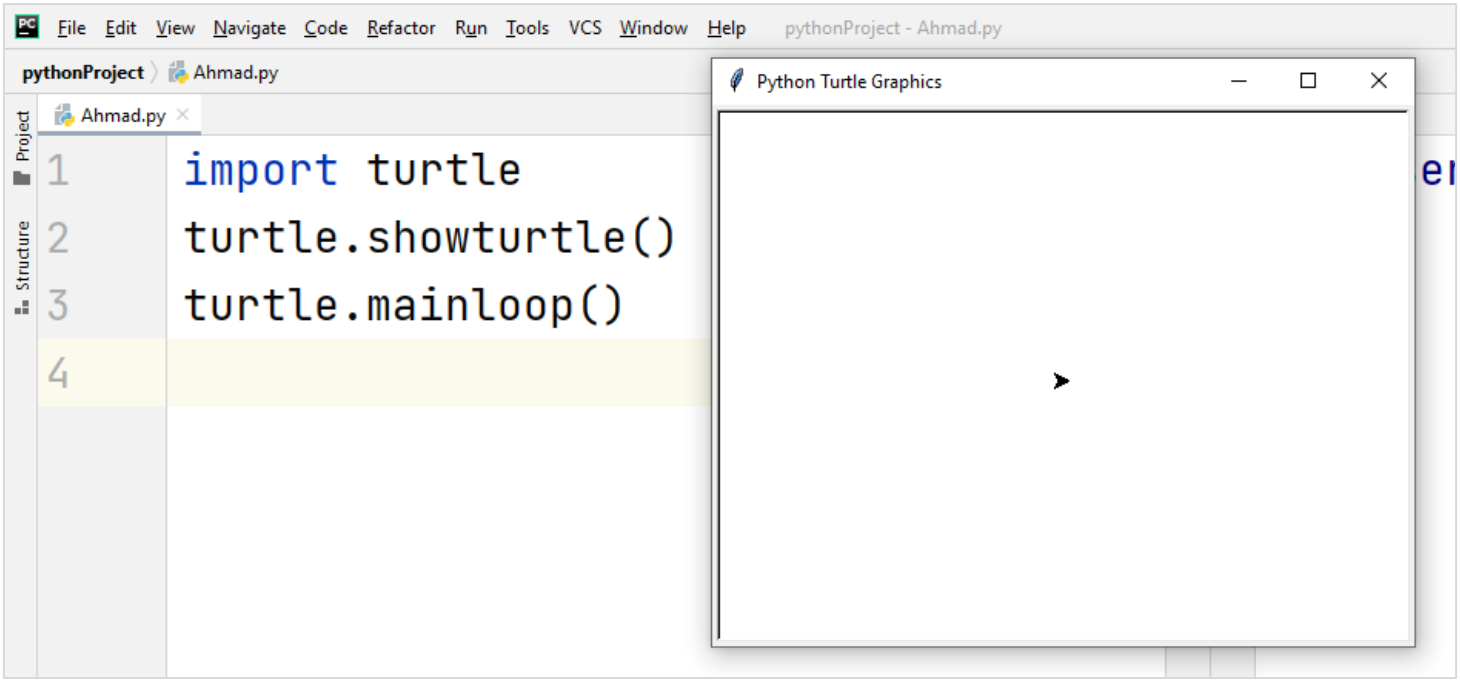
Using Turtle in Script Mode

- If you have tried to run the previous code to show the turtle window on the script mode, **not** interactive mode, you will get the turtle window closed instantly.
- To prevent the window from closing instantly after executing statements, call the turtle.mainloop() function at the end of your program.
- turtle.mainloop() will tell the turtle window not to close by itself and to wait the user to close the window manually.
- Also, you can use the turtle.done() function instead of turtle.mainloop() because they are equivalent.

```
import turtle
turtle.showturtle()
turtle.mainloop()
```

Using Turtle in Script Mode

- **Don't use** the turtle.mainloop() function in the interactive mode, for example, in IDLE, because this will prevent you from execute other statements until you close the window.



Moving the Turtle Forward

- You can use the turtle.forward(distance) or turtle.fd(distance) function to move the turtle forward by the specified distance, in the direction the turtle is headed.
- Example:

```
1  import turtle
2  turtle.showturtle()
3  turtle.forward(100)
4  turtle.mainloop()
```

Run

=

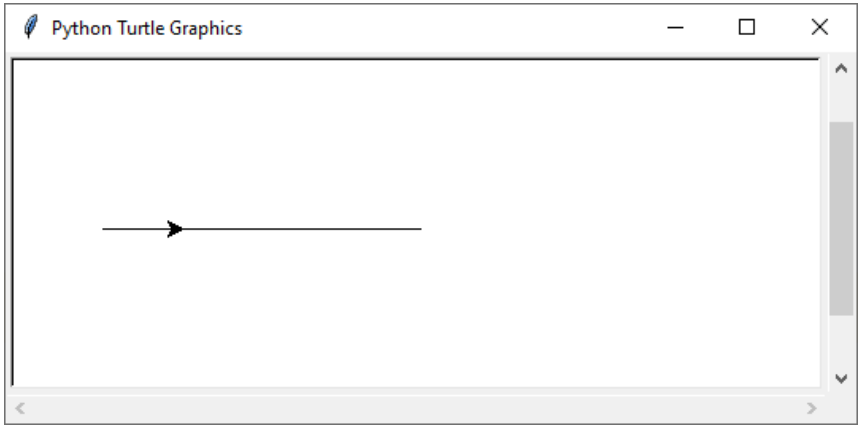
```
1  import turtle
2  turtle.st()
3  turtle.fd(100)
4  turtle.done()
```

Run

Moving the Turtle Backward

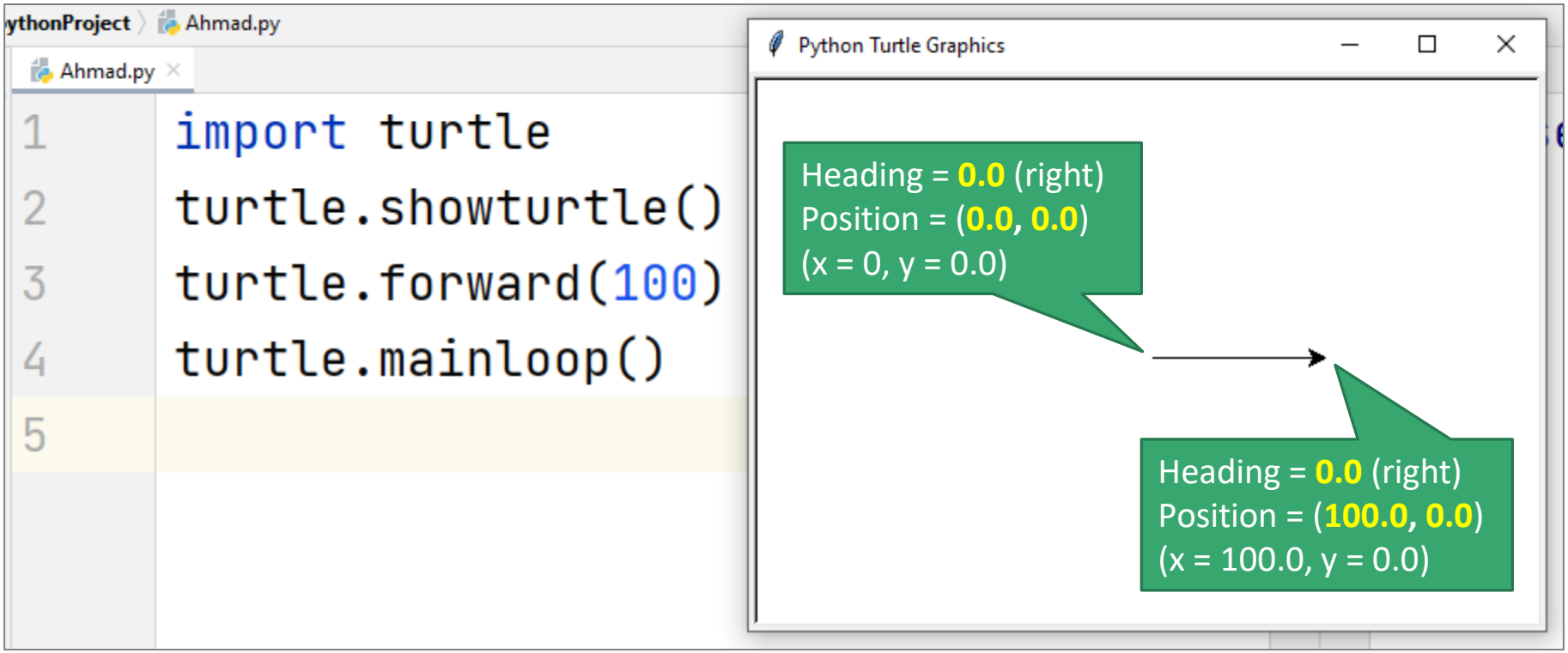
- You can use the turtle.backward(distance) or turtle.back(distance) or turtle.bk(distance) function to move the turtle backward by the specified distance, in the **opposite** direction the turtle is headed.
- Example:

```
1  import turtle
2  turtle.showturtle()
3  turtle.backward(200)
4  turtle.forward(50)
5  turtle.mainloop()
```



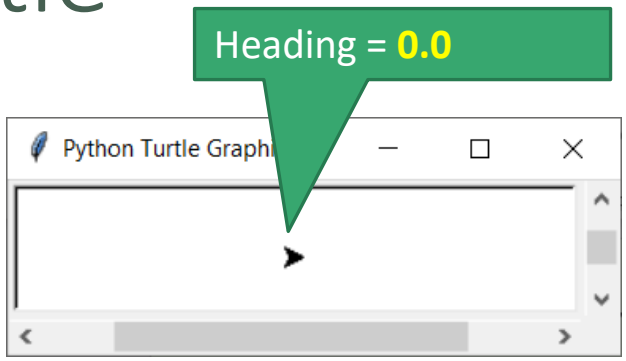
Turtle's Initial Heading and Positions

- By default, the turtle is pointing to the **right** at the center (heading = **0.0**), and its position is (x = **0.0**, y = **0.0**).



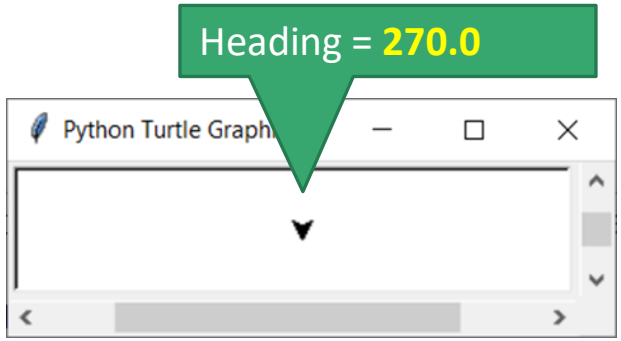
Turning the Turtle

- The turtle's initial heading is **0** degrees (east ⇨).



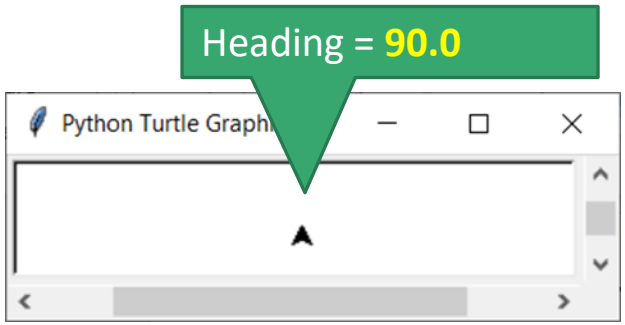
- Use the turtle.right(angle) or turtle.rt(angle) function to turn the turtle **right** (↻) by angle degrees.

```
turtle.right(90)
```



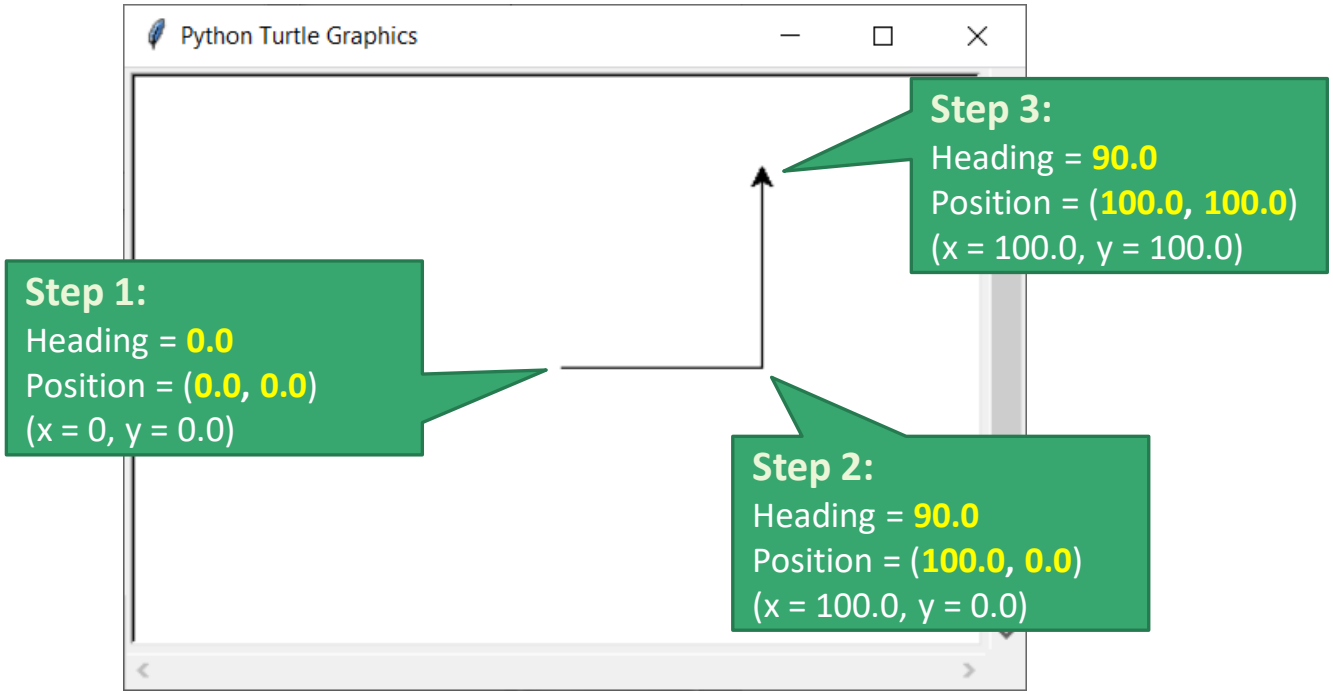
- Use the turtle.left(angle) or turtle.lt(angle) function to turn the turtle **left** (↺) by angle degrees.

```
turtle.left(90)
```



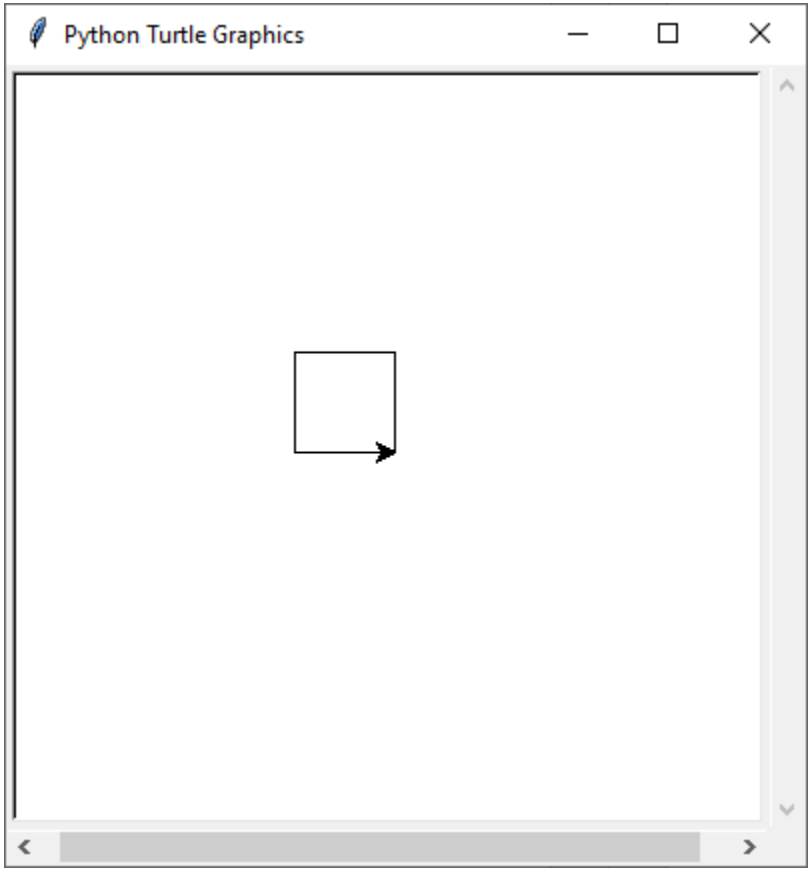
Turning the Turtle
Example #1

```
1  import turtle
2  turtle.showturtle()
3  turtle.forward(100)
4  turtle.left(90)
5  turtle.forward(100)
6  turtle.done()
```



Turning the Turtle
Example #2

```
1  import turtle
2  turtle.st()
3  turtle.left(90)
4  turtle.forward(50)
5  turtle.left(90)
6  turtle.forward(50)
7  turtle.left(90)
8  turtle.forward(50)
9  turtle.left(90)
10 turtle.forward(50)
11 turtle.done()
```

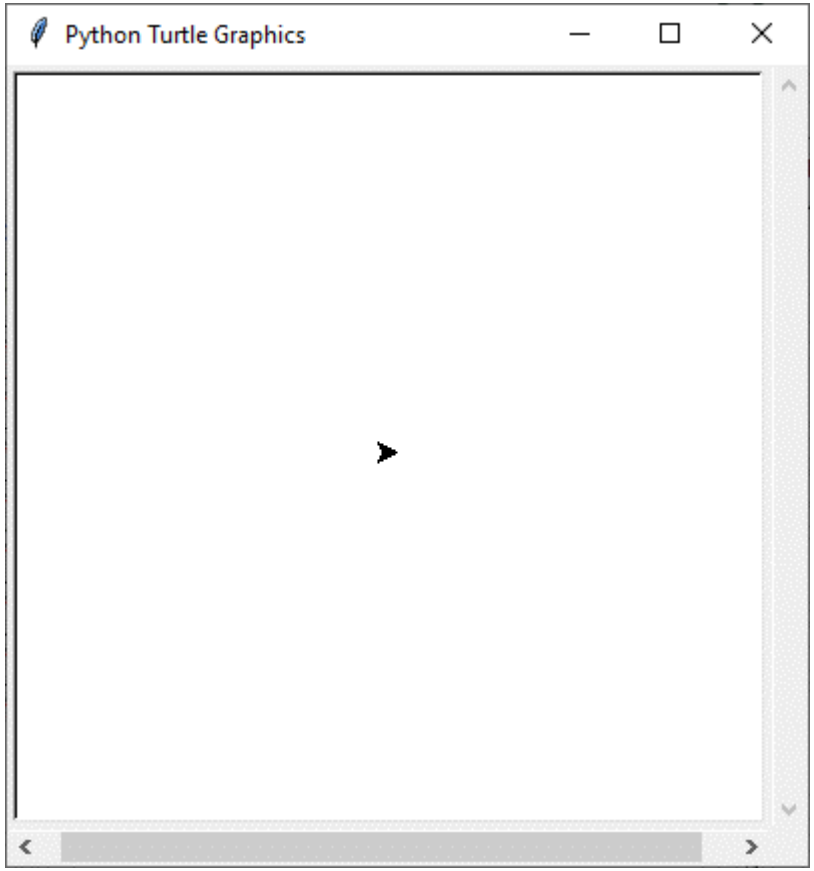


Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```

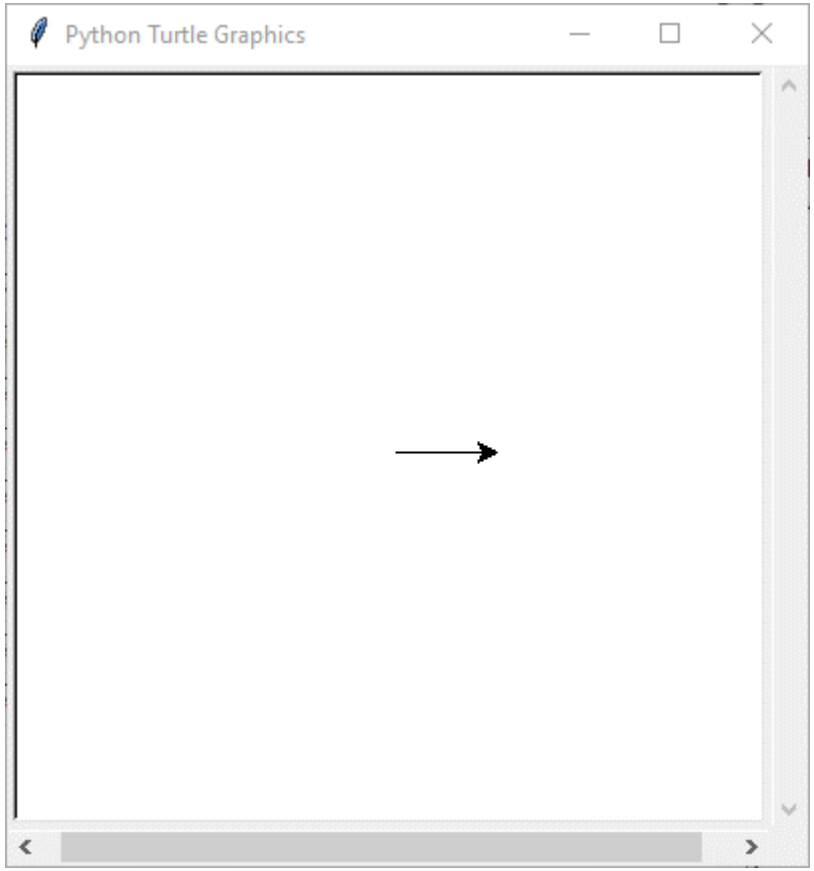
Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



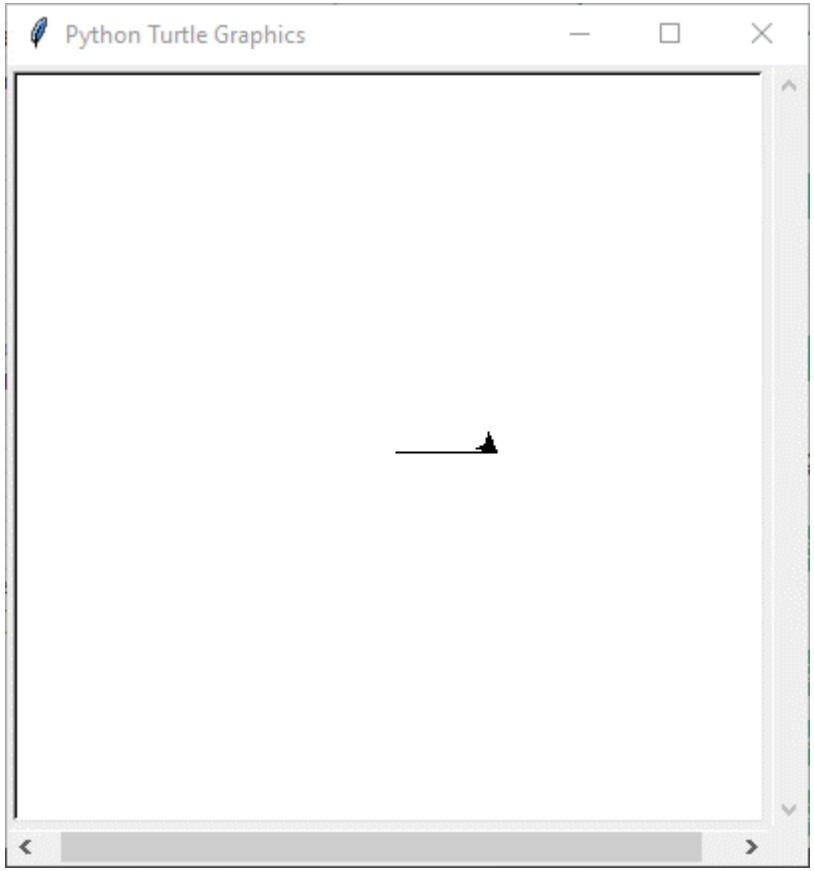
Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



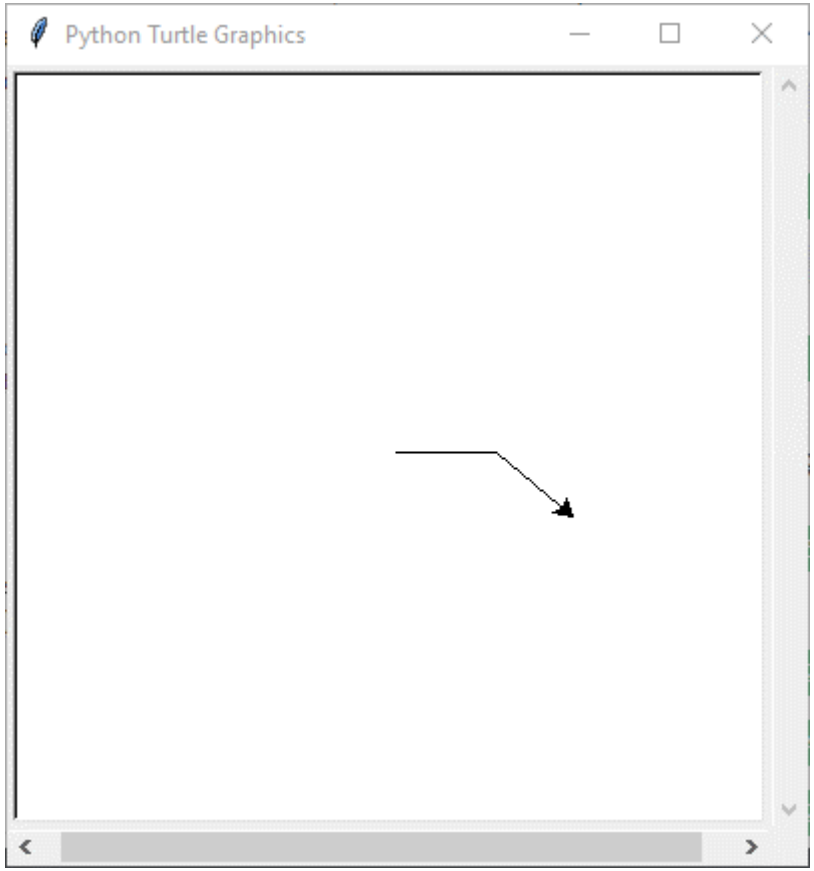
Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



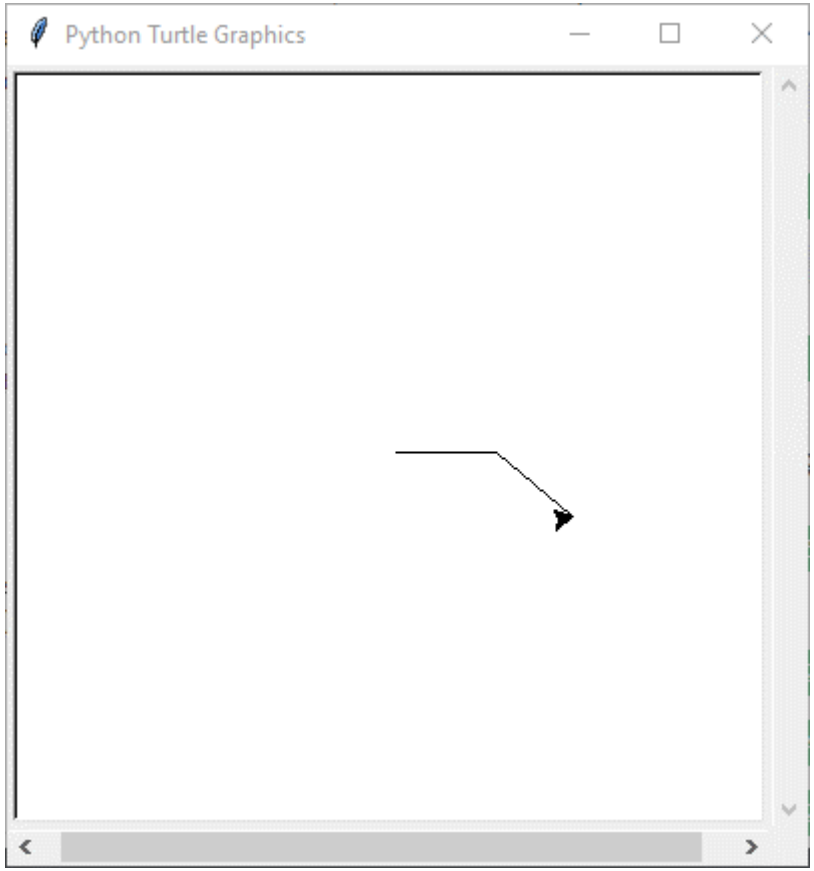
Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



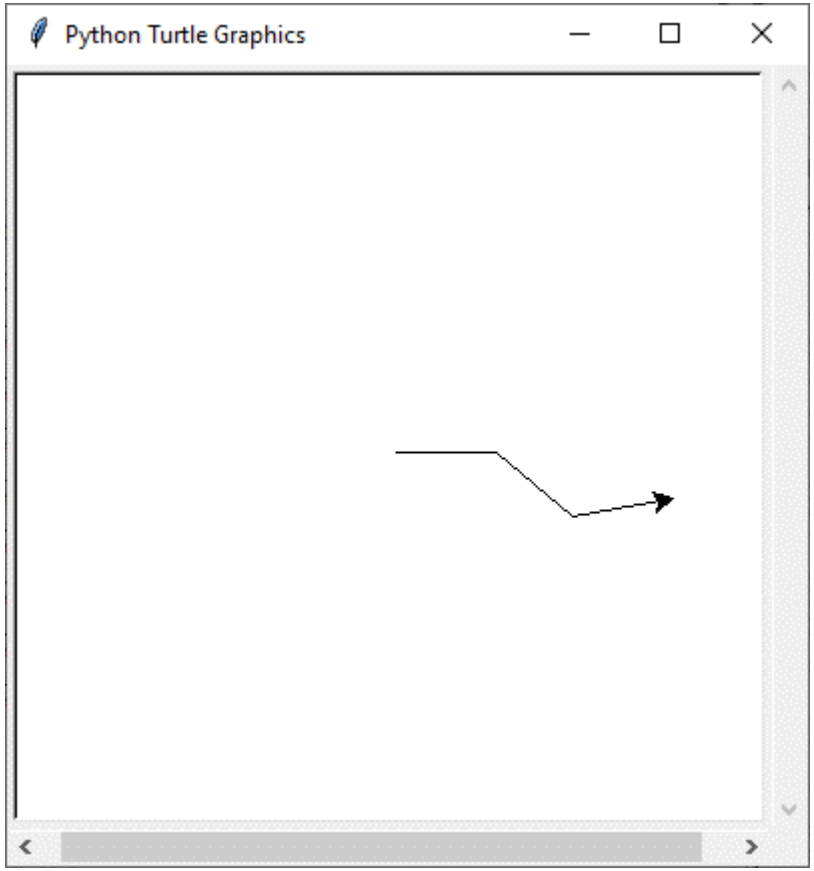
Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



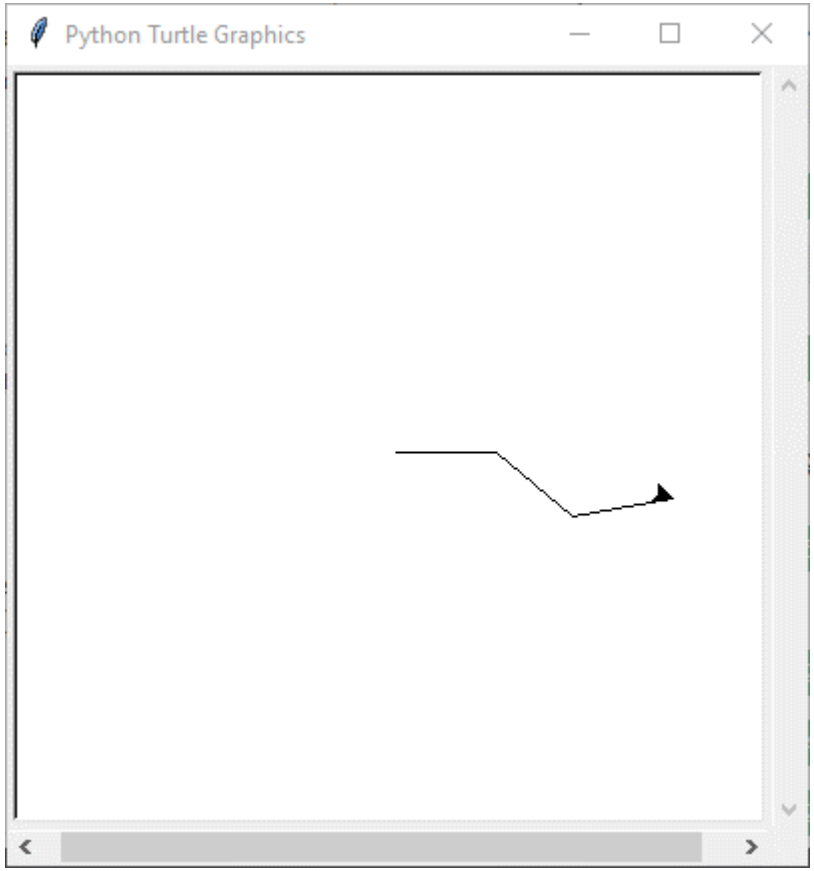
Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



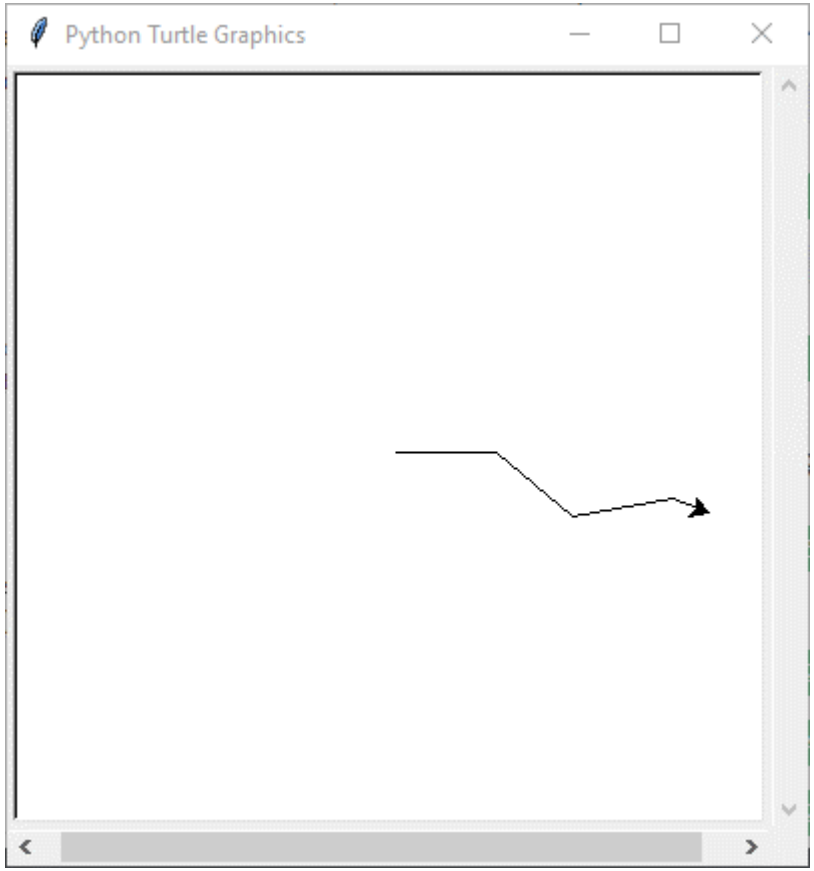
Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



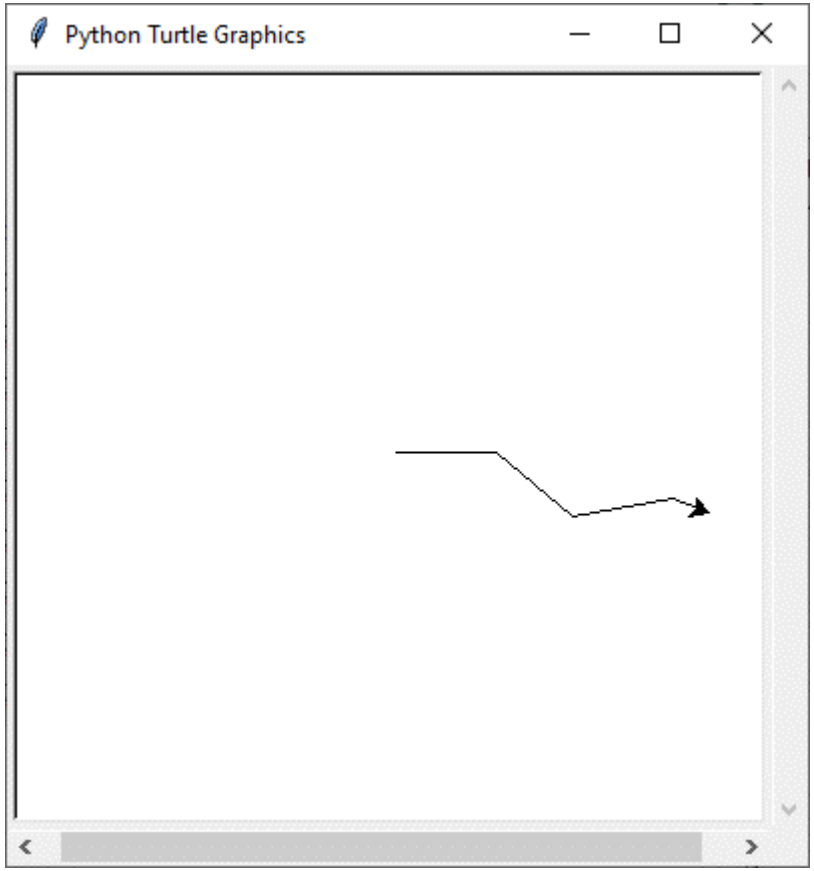
Trace Turning the Turtle

```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



Trace Turning the Turtle

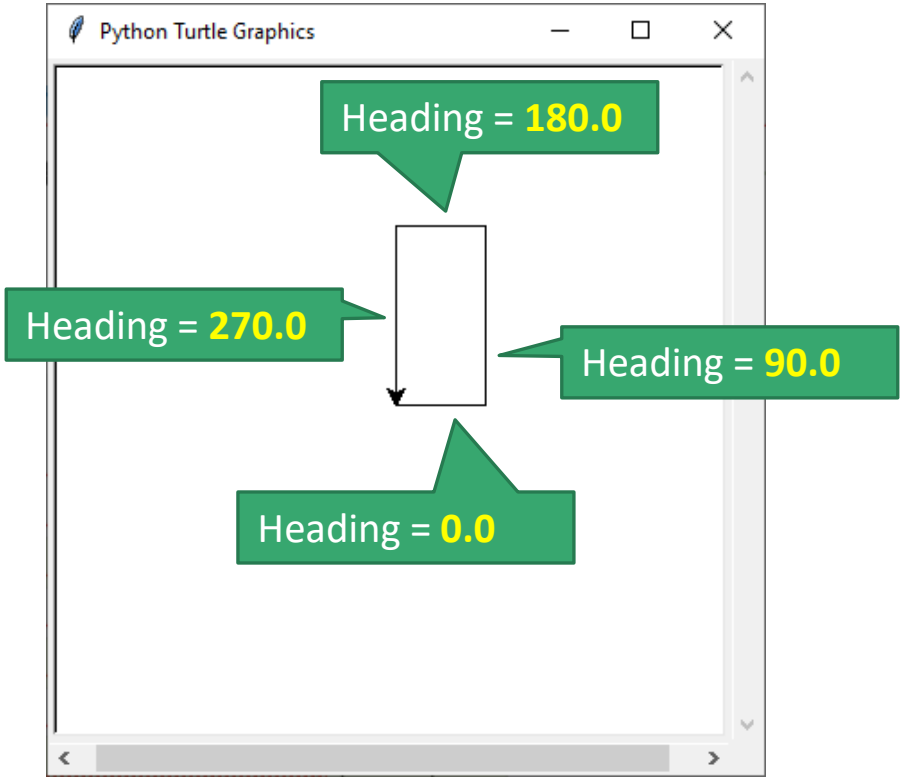
```
1  import turtle
2  turtle.st()
3  turtle.forward(50)
4  turtle.right(40)
5  turtle.forward(50)
6  turtle.left(50)
7  turtle.forward(50)
8  turtle.right(30)
9  turtle.forward(20)
10 turtle.done()
```



Setting the Turtle's Heading

- Use the turtle.setheading(angle) or turtle.seth(angle) function to set the turtle's heading to a specific angle.
- Common directions in degrees:
  0 - east 180 – west
  90 - north 270 - south
- Example:

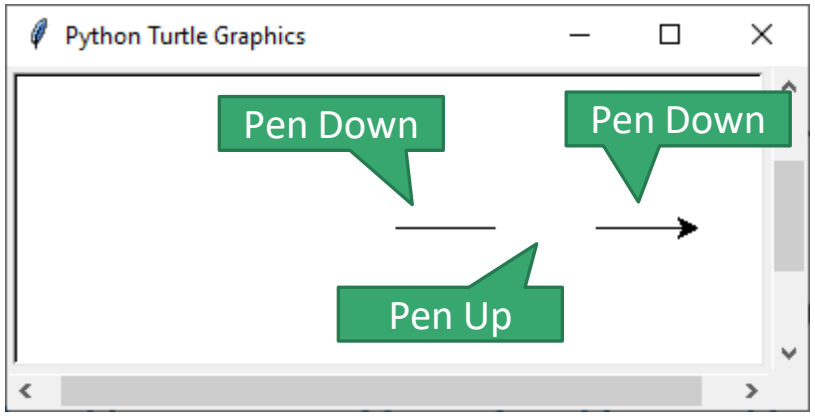
```
1  import turtle
2  turtle.showturtle()
3  turtle.forward(50)
4  turtle.setheading(90)
5  turtle.forward(100)
6  turtle.setheading(180)
7  turtle.forward(50)
8  turtle.setheading(270)
9  turtle.forward(100)
10 turtle.done()
```



Setting the Pen Up or Down

- When the turtle's pen is down, the turtle draws a line as it moves. By default, the pen is down.
- When the turtle's pen is up, the turtle does not draw as it moves.
- Use the turtle.penup() or turtle.pu() or turtle.up() function to raise (☛) the pen.
- Use the turtle.pendown() or turtle.pd() or turtle.down() function to lower ( ) the pen.

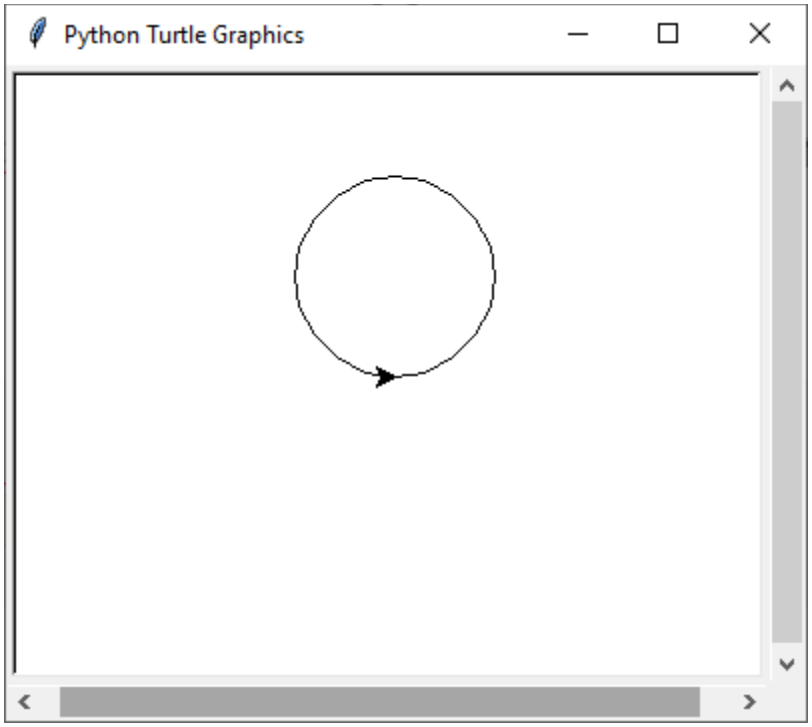
```
1  import turtle
2  turtle.showturtle()
3  turtle.forward(50)
4  turtle.penup()
5  turtle.forward(50)
6  turtle.pendown()
7  turtle.forward(50)
8  turtle.done()
```



Drawing Circles

- Use the turtle.circle(radius) function to draw a circle with a specified radius.
- Example:

```
1  import turtle
2  turtle.showturtle()
3  turtle.circle(50)
4  turtle.done()
```



turtle.circle(…) Function From Python Turtle Documentation

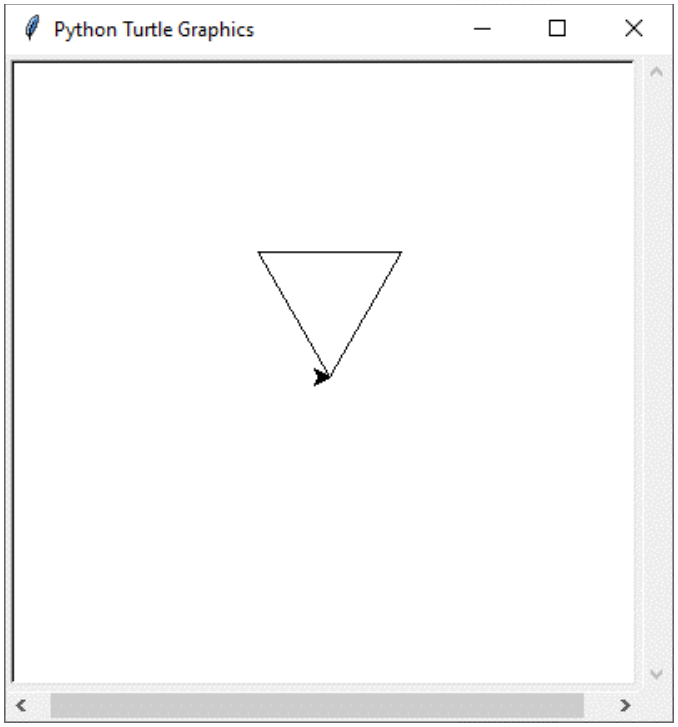
turtle.circle(radius, extent=None, steps=None)

- Draw a circle with given radius. The center is radius units left of the turtle.
- extent – an angle – determines which part of the circle is drawn. If extent is not given, draw the entire circle (360). If extent is not a full circle, one endpoint of the arc is the current pen position.
- Draw the arc in counterclockwise direction if radius is positive, otherwise in clockwise direction.
- The direction of the turtle is changed by the amount of extent.
- As the circle is approximated by an inscribed regular polygon, steps determines the number of steps to use. If not given, it will be calculated automatically. May be used to draw regular polygons.

Drawing Triangles

- You can use the turtle.circle(radius, step=3) function to draw a triangle with a specified radius.
- Example:

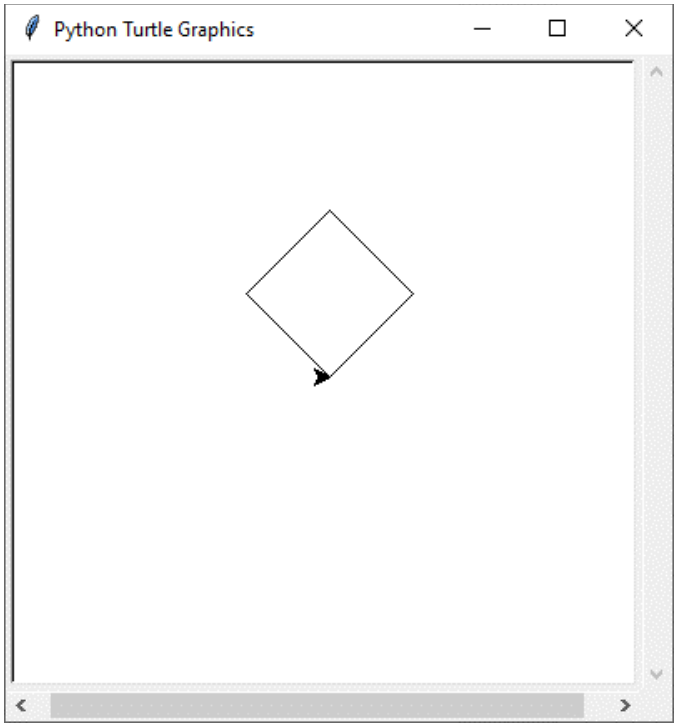
```
1  import turtle
2  turtle.showturtle()
3  turtle.circle(50, step=3)
4  turtle.done()
```



Drawing Diamonds

- You can use the turtle.circle(radius, step=4) function to draw a diamond.
- Example:

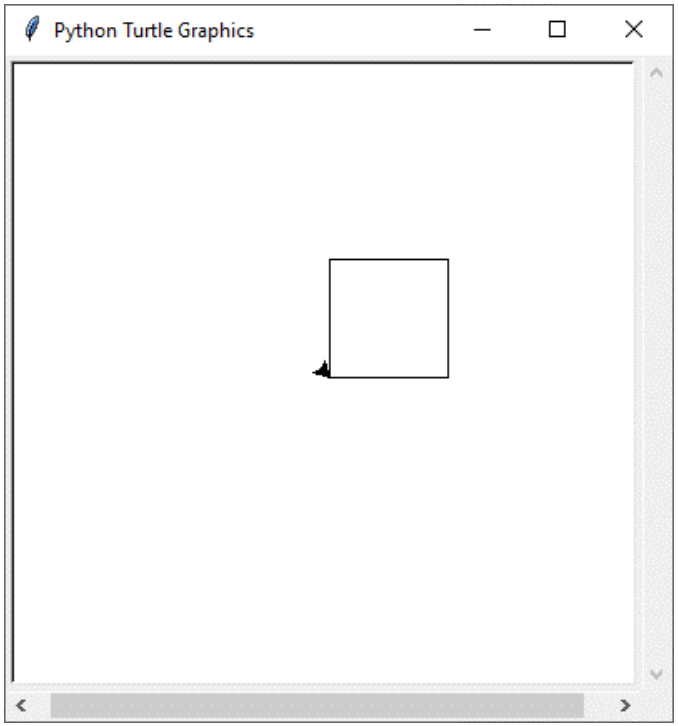
```
1  import turtle
2  turtle.showturtle()
3  turtle.circle(50, step=4)
4  turtle.done()
```



Drawing Squares

- You can use the turtle.circle(radius, step=3) function to draw a square **after** adjusting the heading of the turtle.
- Example:

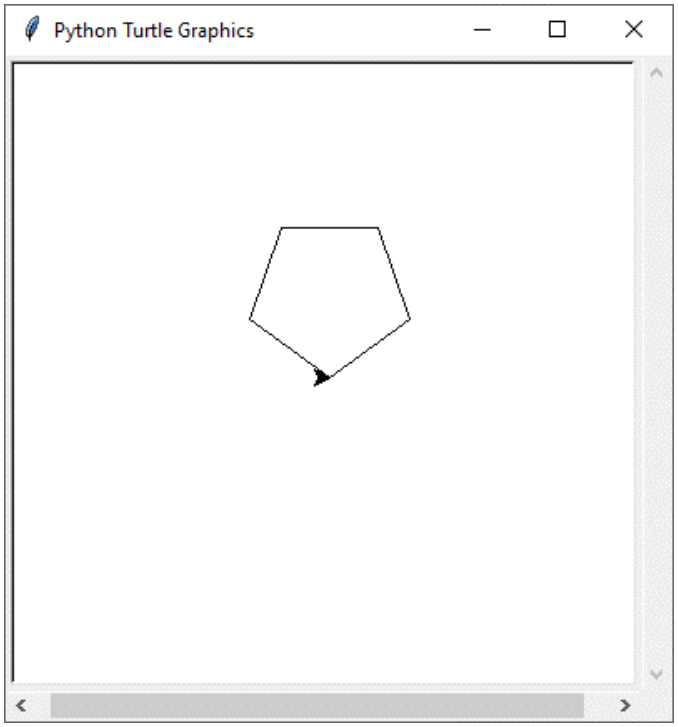
```
1  import turtle
2  turtle.showturtle()
3  turtle.right(45)
4  turtle.circle(50, steps=4)
5  turtle.done()
```



Drawing Pentagons

- You can use the turtle.circle(radius, step=3) function to draw a pentagon.
- Example:

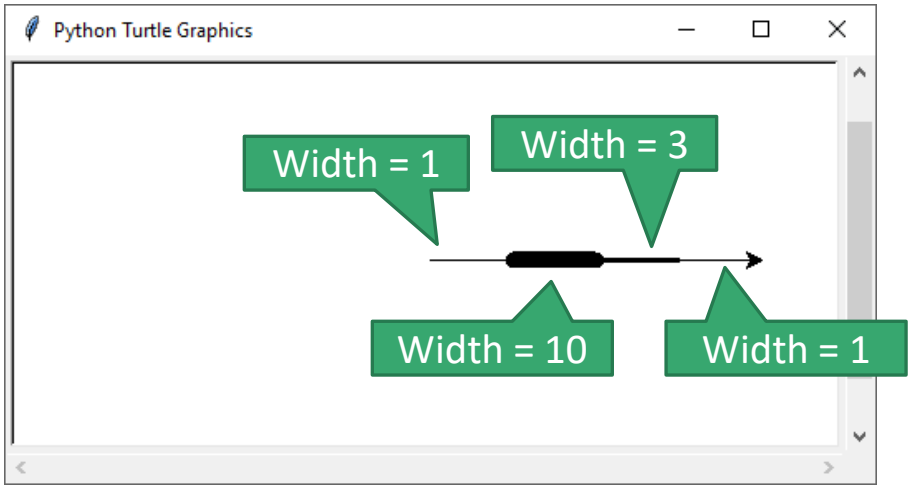
```
1  import turtle
2  turtle.showturtle()
3  turtle.circle(50, steps=5)
4  turtle.done()
```



Changing the Pen Size

- Use the turtle.pensize(width) or turtle.width(width) function to change the width of the turtle's pen (line thickness), in **pixels**.
- Example:

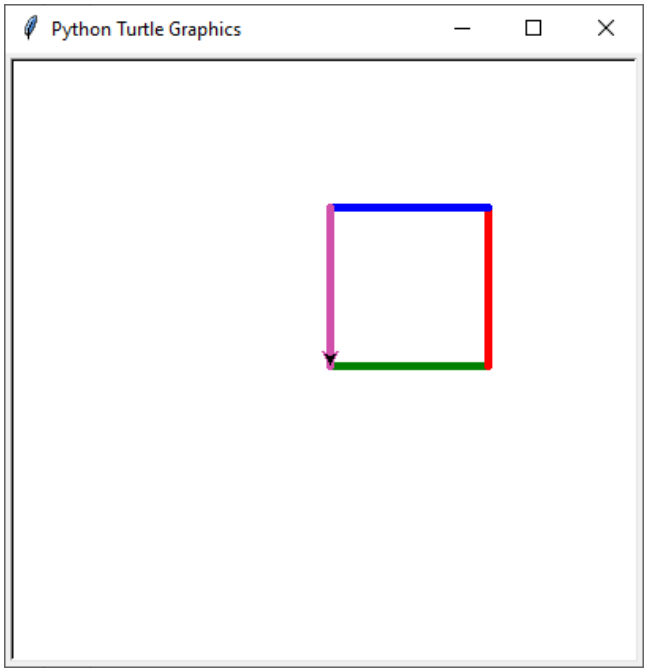
```
1  import turtle
2  turtle.showturtle()
3  turtle.forward(50)
4  turtle.pensize(10)
5  turtle.forward(50)
6  turtle.pensize(3)
7  turtle.forward(50)
8  turtle.pensize(1)
9  turtle.forward(50)
10 turtle.done()
```



Changing the Drawing Color

- Use the turtle.pencolor(color_string) function to change the turtle's drawing color.
- color_string can be the name of the color (ex: red, green, blue, …) or color hex code (ex: #ff5733, #8d33ff, …).

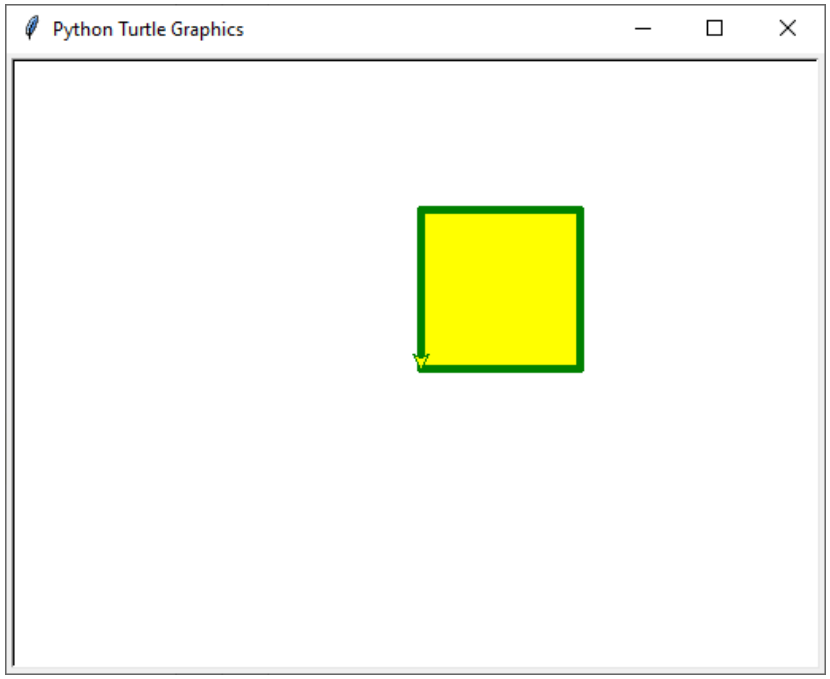
```
1  import turtle
2  turtle.pencolor("green")
3  turtle.pensize(5)
4  turtle.forward(100)
5  turtle.pencolor("red")
6  turtle.left(90)
7  turtle.forward(100)
8  turtle.pencolor("blue")
9  turtle.left(90)
10 turtle.forward(100)
11 turtle.pencolor("#d150ab")
12 turtle.left(90)
13 turtle.forward(100)
14 turtle.done()
```



Changing the Filling Color

- Use the turtle.fillcolor(color_string) function to change the turtle's filling color.
- You have to use turtle.begin_fill() before drawing a shape to be filled and turtle.end_fill() to fill the shape drawn after the last call to begin_fill().

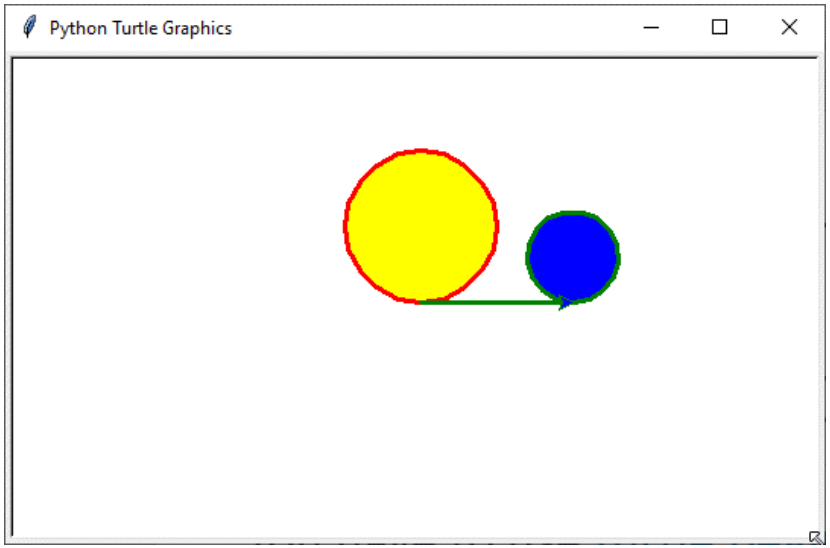
```
1  import turtle
2  turtle.fillcolor("yellow")
3  turtle.begin_fill()
4  turtle.pencolor("green")
5  turtle.pensize(5)
6  turtle.forward(100)
7  turtle.left(90)
8  turtle.forward(100)
9  turtle.left(90)
10 turtle.forward(100)
11 turtle.left(90)
12 turtle.forward(100)
13 turtle.end_fill()
14 turtle.done()
```



Changing the Drawing and Filling Color

- You can use the turtle.color(pen_color, filling_color) function to change the turtle's drawing (pen) and filling color in one statement.
- You have to use turtle.begin_fill() before drawing a shape to be filled and turtle.end_fill() to fill the shape drawn after the last call to begin_fill().

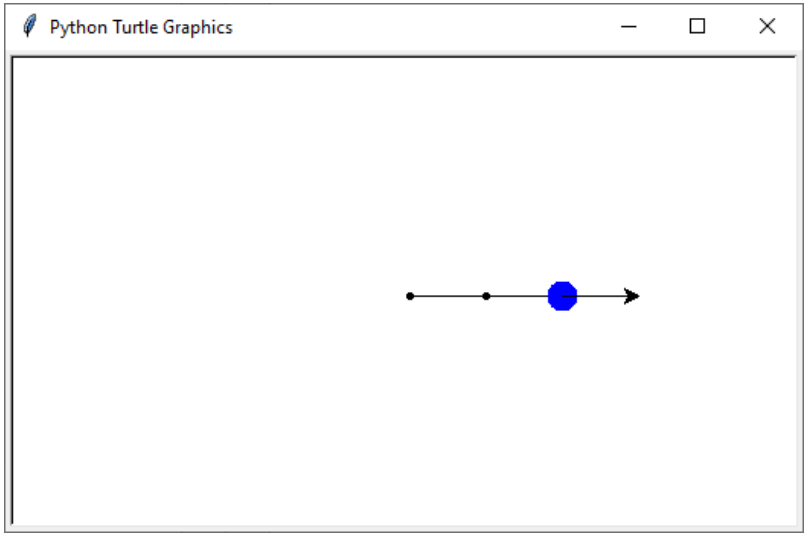
```
1  import turtle
2  turtle.pensize(3)
3  turtle.color("red", "yellow")
4  turtle.begin_fill()
5  turtle.circle(50)
6  turtle.end_fill()
7  turtle.color("green", "blue")
8  turtle.forward(100)
9  turtle.begin_fill()
10 turtle.circle(30)
11 turtle.end_fill()
12 turtle.done()
```



Drawing Dots

- You can use the turtle.dot(size=None, color=None) function to draw a circular dot with diameter size, using color.
  - If size is not given, the maximum of pensize+4 and 2*pensize is used.
  - If color is not given, the current pen color is used.
- Example:

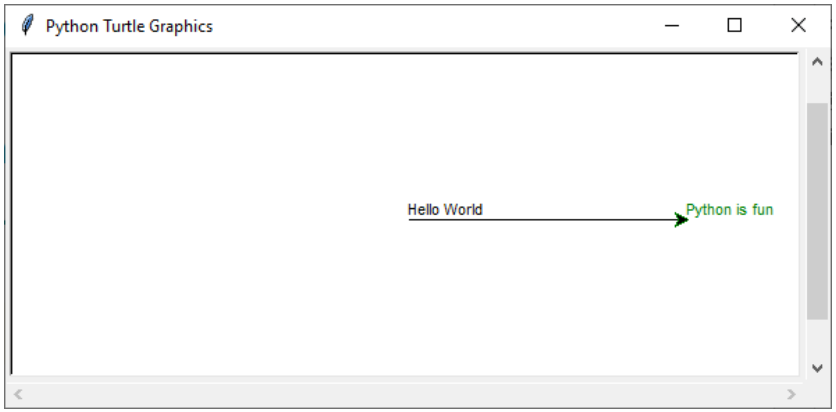
```
1  import turtle
2  turtle.dot()
3  turtle.forward(50)
4  turtle.dot()
5  turtle.forward(50)
6  turtle.dot(20, "blue")
7  turtle.forward(50)
8  turtle.done()
```



Displaying Text

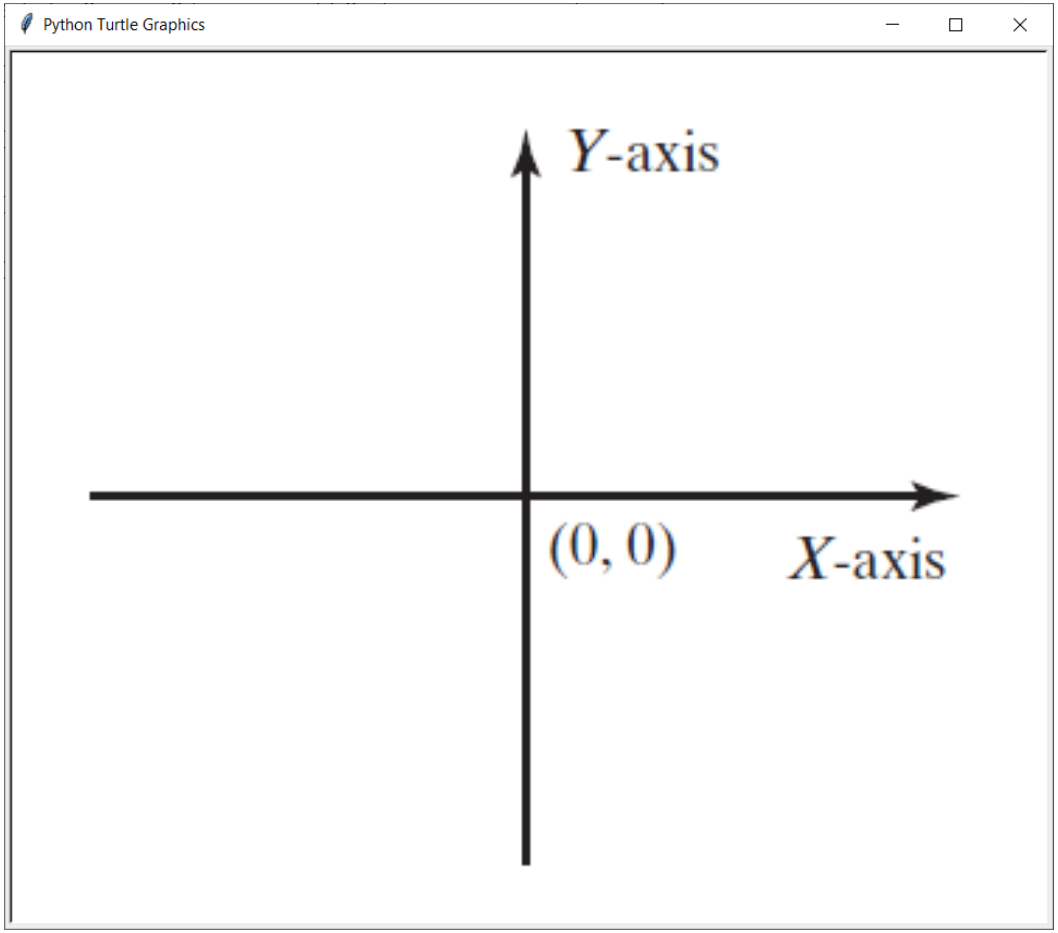
- Use turtle.write(string, move=False, align="left", font=("Arial", 8, "normal")) function to write string at the current turtle position according to align ("left", "center" or "right") and with the given font.
- If move is True, the pen is moved to the bottom-right corner of the text. By default, move is False.

```
1  import turtle
2  turtle.write("Hello World")
3  turtle.forward(200)
4  turtle.pencolor("green")
5  turtle.write("Python is fun")
6  turtle.done()
```



Working with Coordinates

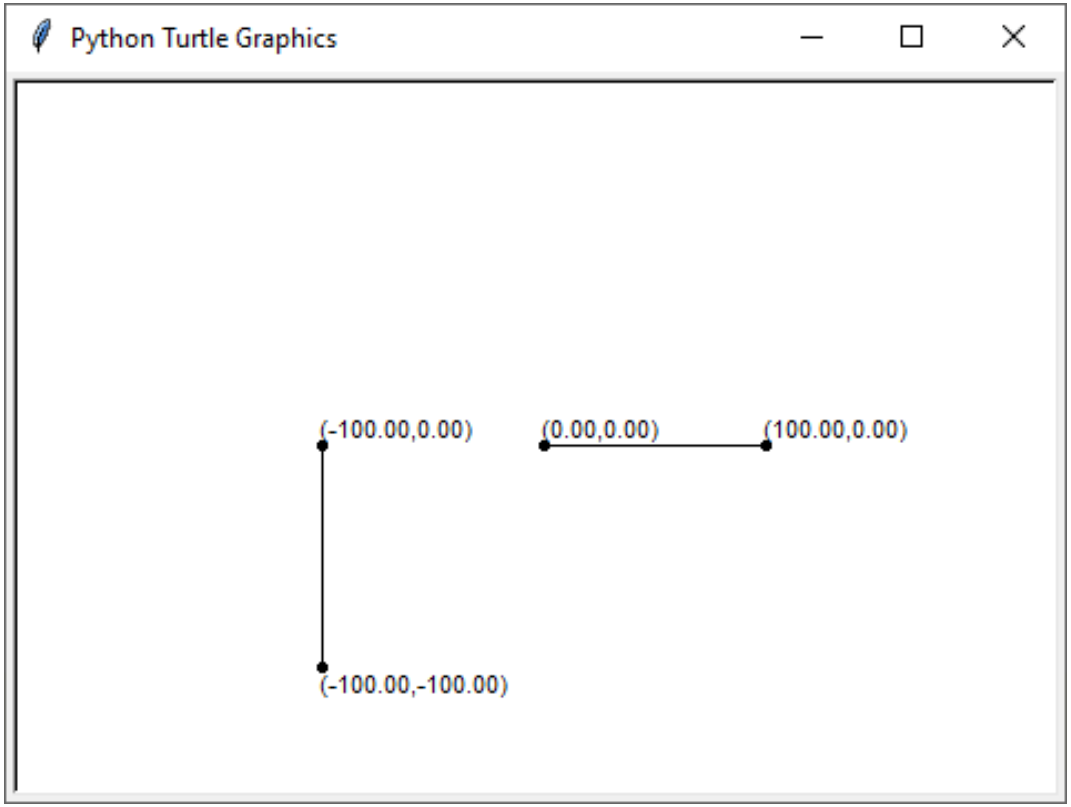
- The turtle uses Cartesian Coordinates.



Working with Coordinates
Example of Coordinates

- You can use turtle.pos() function to get the current position of the turtle.

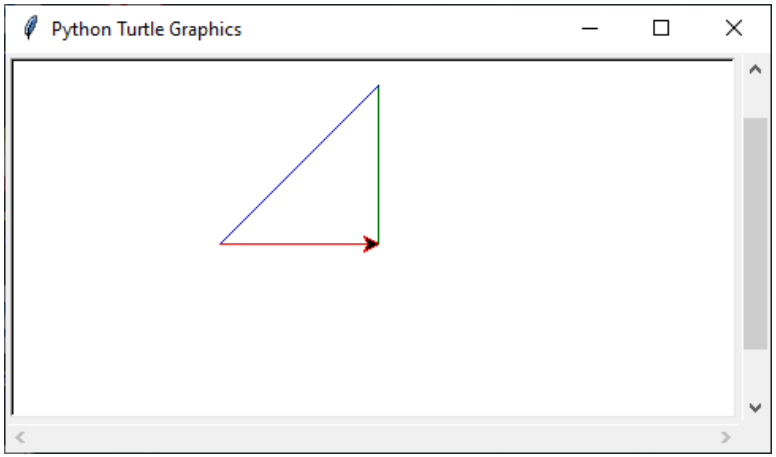
```
1  import turtle
2  turtle.write(turtle.pos())
3  turtle.dot()
4  turtle.forward(100)
5  turtle.dot()
6  turtle.write(turtle.pos())
7  turtle.penup()
8  turtle.backward(200)
9  turtle.pendown()
10 turtle.dot()
11 turtle.write(turtle.pos())
12 turtle.right(90)
13 turtle.forward(100)
14 turtle.dot()
15 last_pos = turtle.pos()
16 turtle.penup()
17 turtle.forward(15)
18 turtle.write(last_pos)
19 turtle.hideturtle()
20 turtle.done()
```



Moving the Turtle to a Specific Location

- You can use the turtle.goto(x, y) or turtle.setpos(x, y) or turtle.setposition(x, y) function to move the turtle to a specific location (x and y position).
- You can also use turtle.setx(x) function to set the turtle's x coordinate and leave y coordinate unchanged.
- You can also use turtle.sety(y) function to set the turtle's y coordinate and leave x coordinate unchanged.

```
1  import turtle
2  turtle.pencolor("green")
3  turtle.goto(0, 100)
4  turtle.pencolor("blue")
5  turtle.goto(-100, 0)
6  turtle.pencolor("red")
7  turtle.goto(0, 0)
8  turtle.done()
```



Changing the Turtle's Speed

- You can use the turtle.speed(speed=None) function to set the turtle's speed to an integer value in the range 0, 1, 2, …, 9, 10 .
  - If no argument is given, return current speed.
  - If input is a number greater than 10 or smaller than 0.5, speed is set to 0.
  - Speed strings are mapped to speed values as follows:
    - "fastest": 0 "fast": 10 "normal": 6
    - "slow": 3 "slowest": 1
  - Speeds from 1 to 10 enforce increasingly faster animation of line drawing and turtle turning.
- Note: speed = 0 means that no animation takes place. forward/back makes turtle jump and likewise left/right make the turtle turn instantly.

```
turtle.speed(3)
turtle.speed("fast")
```