# Lecture 27: Turtle Graphics

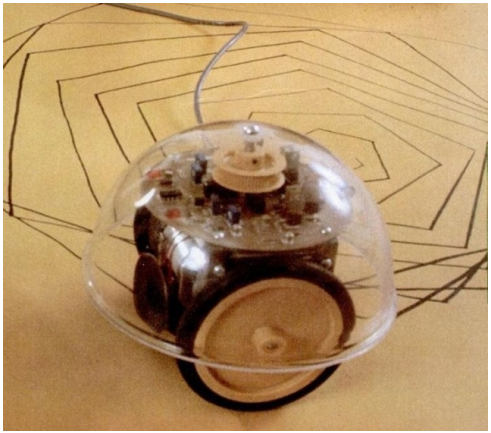
**Comp 102**

Forman Christian University

**All semester you've told Python what to think.**

**All semester you've told Python what to think.**

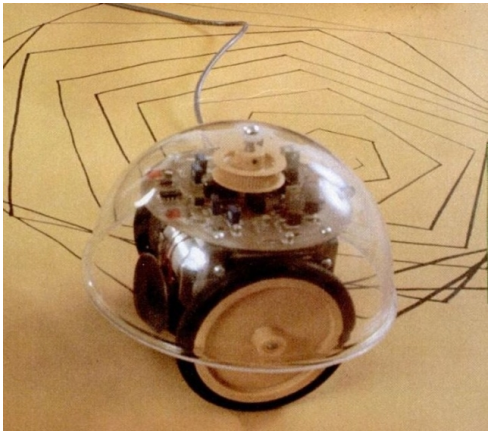
**Today you teach it to draw.**



*Papert's turtle robot, MIT 1969*

In 1969, Seymour Papert at MIT built a real robot turtle

that rolled across the floor holding a **pen**.



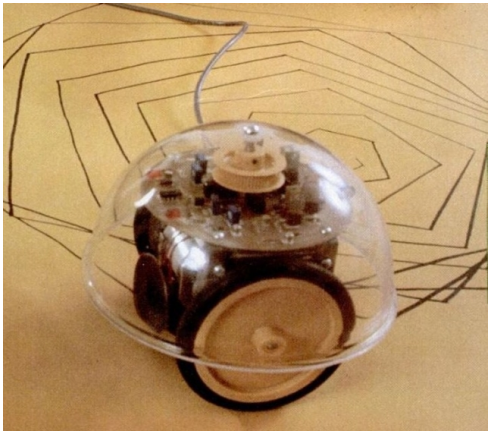
*Papert's turtle robot, MIT 1969*

In 1969, Seymour Papert at MIT built a real robot turtle

that rolled across the floor holding a **pen**.

Children gave it commands:

```
"forward 100"   "left 90"   "forward 100"
```



*Papert's turtle robot, MIT 1969*

In 1969, Seymour Papert at MIT built a real robot turtle

that rolled across the floor holding a **pen**.

Children gave it commands:

`"forward 100"` `"left 90"` `"forward 100"`

Python's `turtle` module is a direct descendant of that idea.

**Today, we become those children.**

# Today's Journey

**Blank canvas → lines → shapes → colors**

**→ functions → patterns → art**

Every concept you already know —
loops, functions, modules — just made **visual**.

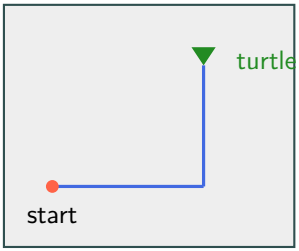
# Part 1: Meet the Turtle

# Hello, Turtle!

```
1 import turtle
2
3 t = turtle.Turtle()
4 s = turtle.Screen()
5
6 t.forward(100)
7 t.left(90)
8 t.forward(80)
```

# Hello, Turtle!

```
1  import turtle
2
3  t = turtle.Turtle()
4  s = turtle.Screen()
5
6  t.forward(100)
7  t.left(90)
8  t.forward(80)
```




`import turtle` — a **standard library module**!
`turtle.Turtle()` — creates an **object**.

# The 4 Basic Commands

- `t.forward(100)` — move forward 100 pixels

# The 4 Basic Commands

- `t.forward(100)` — move forward 100 pixels
- `t.backward(50)` — move backward

# The 4 Basic Commands

- `t.forward(100)` — move forward 100 pixels
- `t.backward(50)` — move backward
- `t.right(90)` — turn right 90°

# The 4 Basic Commands

- `t.forward(100)` — move forward 100 pixels
- `t.backward(50)` — move backward
- `t.right(90)` — turn right 90°
- `t.left(90)` — turn left 90°

# The 4 Basic Commands

- `t.forward(100)` — move forward 100 pixels
- `t.backward(50)` — move backward
- `t.right(90)` — turn right 90°
- `t.left(90)` — turn left 90°

The turtle starts at the **center** of the screen, facing **right** (East).
Movement is **relative** to the turtle's current direction.

# Lifting the Pen

- `t.penup()` — lift the pen (stop drawing)
- `t.pendown()` — lower the pen (start drawing)
- `t.goto(x, y)` — jump to a position

# Lifting the Pen

- `t.penup()` — lift the pen (stop drawing)
- `t.pendown()` — lower the pen (start drawing)
- `t.goto(x, y)` — jump to a position

Like lifting your marker off the whiteboard
to move somewhere without leaving a trail.

# You Try!

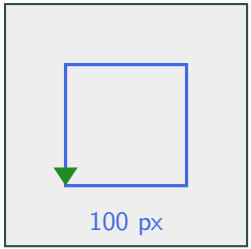
Draw a square — repeat 4 times:

```
1 t.forward(100)    # side 1
2 t.left(90)
3 t.forward(100)    # side 2
4 t.left(90)
5 t.forward(100)    # side 3
6 t.left(90)
7 t.forward(100)    # side 4
```

# You Try!

Draw a square — repeat 4 times:

```
1 t.forward(100)    # side 1
2 t.left(90)
3 t.forward(100)    # side 2
4 t.left(90)
5 t.forward(100)    # side 3
6 t.left(90)
7 t.forward(100)    # side 4
```




**7 lines** for a square?!

**There must be a better way.**

# Part 2: Loops → Shapes

# The Square, Rewritten

8 lines become **3**:

```
1 for i in range(4):
2     t.forward(100)
3     t.left(90)
```

# The Square, Rewritten

8 lines become **3**:

```
1 for i in range(4):
2     t.forward(100)
3     t.left(90)
```

Repeat 4 times: go forward, turn 90°.

**Why 90?** Because 360 ÷ 4 = 90.

# The Key Insight

$$\textbf{Turn angle} = \frac{360}{n}$$

where $n$ = number of sides.

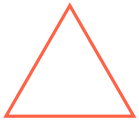
# The Key Insight

$$\textbf{Turn angle} = \frac{360}{n}$$

where $n$ = number of sides.

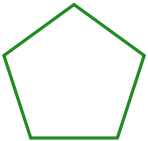
| | | |
|---|---|---|
| Triangle | $360 \div 3$ | $= 120^\circ$ |
| Square | $360 \div 4$ | $= 90^\circ$ |
| Pentagon | $360 \div 5$ | $= 72^\circ$ |
| Hexagon | $360 \div 6$ | $= 60^\circ$ |
| Octagon | $360 \div 8$ | $= 45^\circ$ |

# Shapes from One Formula

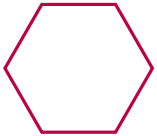


$n = 3$
120°

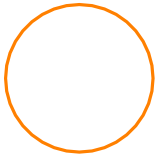
$n = 4$
90°



$n = 5$
72°



$n = 6$
60°



$n = 36$
10°

Same code, different `n` → **any shape**.

# The General Polygon

```
1 sides = 6
2 for i in range(sides):
3     t.forward(60)
4     t.left(360 / sides)
```

# The General Polygon

```
1 sides = 6
2 for i in range(sides):
3     t.forward(60)
4     t.left(360 / sides)
```

Change `sides` to 3 → triangle.
Change `sides` to 8 → octagon.

# The General Polygon

```
1 sides = 6
2 for i in range(sides):
3     t.forward(60)
4     t.left(360 / sides)
```

Change `sides` to 3 → triangle.
Change `sides` to 8 → octagon.

Change `sides` to 36 and `forward(10)` → **a circle!**

A circle is just a polygon with **many tiny sides**.

(Built-in shortcut: `t.circle(50)`)

# You Try!

**Predict:** What does this draw?

```
1 for i in range(5):
2     t.forward(80)
3     t.left(72)
```

# You Try!

**Predict:** What does this draw?

```
1 for i in range(5):
2     t.forward(80)
3     t.left(72)
```

→ **A pentagon!** $(360 \div 5 = 72)$

Now draw a **hexagon**, then change it to an **octagon**.
(Just change **two numbers**!)

# Part 3: Functions → Tools

# A Reusable Polygon Function

```
1 def draw_polygon(t, n, size):
2     for i in range(n):
3         t.forward(size)
4         t.right(360 / n)
```

# A Reusable Polygon Function

```
1 def draw_polygon(t, n, size):
2     for i in range(n):
3         t.forward(size)
4         t.right(360 / n)
```

Now **one function** draws any regular polygon:

```
draw_polygon(t, 3, 100)    # Triangle
draw_polygon(t, 6, 60)     # Hexagon
draw_polygon(t, 36, 10)    # Circle
```

# A Star Function

```
1 def draw_star(t, size, color):
2     t.pencolor(color)
3     t.fillcolor(color)
4     t.begin_fill()
5     for i in range(5):
6         t.forward(size)
7         t.right(144)
8     t.end_fill()
```

# A Star Function

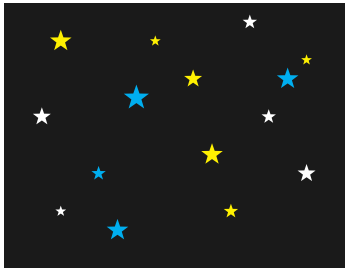
```
1  def draw_star(t, size, color):
2      t.pencolor(color)
3      t.fillcolor(color)
4      t.begin_fill()
5      for i in range(5):
6          t.forward(size)
7          t.right(144)
8      t.end_fill()
```

Why 144°? Star exterior angle: $720 \div 5 = 144$.

**You're building a toolbox.**

# Scatter Stars Across a Night Sky

```
1  import random
2  s.bgcolor("black")
3  t.speed(0); t.hideturtle()
4  colors = ["white", "yellow",
5            "cyan", "gold"]
6
7  for i in range(20):
8      t.penup()
9      t.goto(random.randint(-350,
           350),
10                random.randint(-250,
                      250))
11     t.pendown()
12     draw_star(t,
13         random.randint(10, 40),
14         random.choice(colors))
```



Every run is **different**!

# You Try!

Write a `draw_triangle(t, size)` function:

```
def draw_triangle(t, size):
    for i in range(3):
        t.forward(size)
        t.left(120)
```

Call it 3 times with sizes 50, 100, and 150.

# Part 4: Color & Style

# Painting with Code

```
1 t.pencolor("blue")          # Line color
2 t.fillcolor("yellow")       # Fill color
3 t.pensize(3)                # Line thickness
4 t.speed(0)                  # Fastest drawing
```

# Painting with Code

```
t.pencolor("blue")          # Line color
t.fillcolor("yellow")       # Fill color
t.pensize(3)                # Line thickness
t.speed(0)                  # Fastest drawing
```

## Filling a shape:

```
t.begin_fill()
# ... draw something ...
t.end_fill()
```

Everything drawn between `begin_fill()` and `end_fill()` gets filled.

# A Filled Gold Star

```
t.fillcolor("gold")
t.begin_fill()
for i in range(5):
    t.forward(150)
    t.right(144)
t.end_fill()
```

# A Filled Gold Star

```
t.fillcolor("gold")
t.begin_fill()
for i in range(5):
    t.forward(150)
    t.right(144)
t.end_fill()
```

**Neon-on-black** aesthetic:

```
s.bgcolor("black")
t.pencolor("cyan")
```

# Writing Text on the Canvas

```
1 t.penup()
2 t.goto(0, -200)
3 t.pencolor("white")
4 t.write("Hello, Turtle!",
5           align="center",
6           font=("Arial", 24, "bold"))
```

# Writing Text on the Canvas

```
1 t.penup()
2 t.goto(0, -200)
3 t.pencolor("white")
4 t.write("Hello, Turtle!",
5           align="center",
6           font=("Arial", 24, "bold"))
```

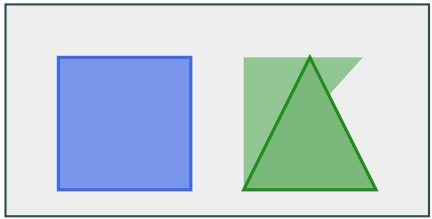
align: "left", "center", or "right"

font: (family, size, style)

# You Try!

Draw two shapes side by side:

1. A **filled blue square** on the left
2. A **filled green triangle** on the right



Hint: Use `penup()` and `goto()` to move between them.

# Part 5: Patterns

# The Secret Recipe

# The Secret Recipe

Draw a shape

↓

**Rotate a little**

↓

**Repeat**

= **Stunning patterns**

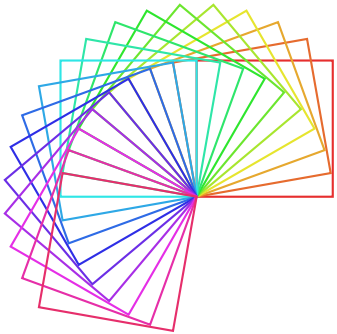
# Rosette of Squares

```
1 for i in range(36):
2     draw_polygon(t, 4,
          100)
3     t.right(10)
```

36 squares $\times$ 10° each
= 360° full circle.

# Rosette of Squares

```
1 for i in range(36):
2     draw_polygon(t, 4,
          100)
3     t.right(10)
```



36 squares × 10° each
= 360° full circle.

**3 lines of code.**
**Stunning result.**

# Add Rainbow Colors

```
colors = ["red", "orange", "yellow",
          "green", "cyan", "purple"]

for i in range(36):
    t.pencolor(colors[i % len(colors)])
    draw_polygon(t, 4, 100)
    t.right(10)
```

# Add Rainbow Colors

```
colors = ["red", "orange", "yellow",
          "green", "cyan", "purple"]

for i in range(36):
    t.pencolor(colors[i % len(colors)])
    draw_polygon(t, 4, 100)
    t.right(10)
```

`i % len(colors)` cycles through the color list!

$0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, \ldots$

# The Spirograph Effect

```
1 for i in range(200):
2     t.forward(i)
3     t.right(91)
```

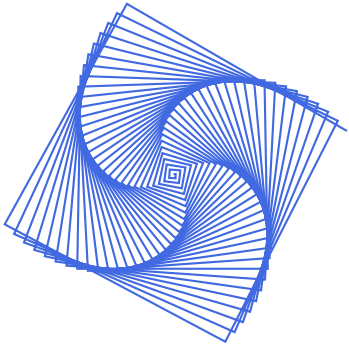
Each step gets **longer**.
Turn is **slightly off** from 90°.

# The Spirograph Effect

```
for i in range(200):
    t.forward(i)
    t.right(91)
```

Each step gets **longer**.
Turn is **slightly off** from 90°.

**Change one number, change everything:**
Try: 89, 60, 121, 175!

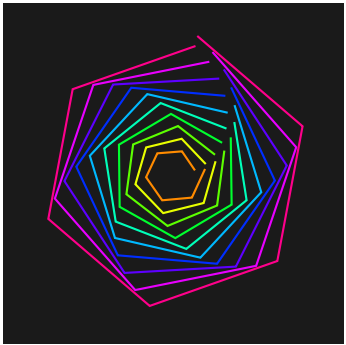


# Rainbow Spiral

```
1  import colorsys
2  s.bgcolor("black")
3  t.speed(0)
4  t.pensize(2)
5
6  for i in range(360):
7      color = colorsys.hsv_to_rgb(
8          i / 360, 1, 1)
9      t.pencolor(color)
10     t.forward(i * 0.5)
11     t.left(59)
```

# Rainbow Spiral

```
1  import colorsys
2  s.bgcolor("black")
3  t.speed(0)
4  t.pensize(2)
5
6  for i in range(360):
7      color = colorsys.hsv_to_rgb(
8          i / 360, 1, 1)
9      t.pencolor(color)
10     t.forward(i * 0.5)
11     t.left(59)
```



`colorsys.hsv_to_rgb` — hue $0 \to 1$ for rainbow.
**Why 59?** 60° = hexagon. 59° = spiral!

# You Try!

Start with this rosette:

```
for i in range(36):
    draw_polygon(t, 4, 100)
    t.right(10)
```

Now experiment:

- Change the shape (triangles? circles? stars?)
- Change the count (12? 72?)
- Change the angle (15°? 7°?)
- Add color cycling

**Make something unique!**

# Part 6: Create Your Art

# Your Turn!

Combine everything into your own artwork. Pick a challenge:

1. A **night sky** scene (moon + scattered stars + ground)
2. A **flower** with colored petals
3. An **abstract geometric pattern**
4. A **house** with roof, door, and windows
5. A row of **stars** in different colors and sizes

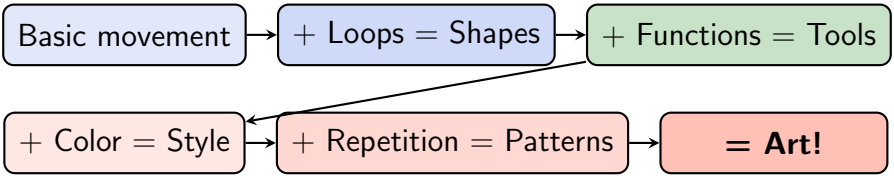
You have **10 minutes**. Use functions, loops, colors, fills!

# Starter: Night Sky Scene

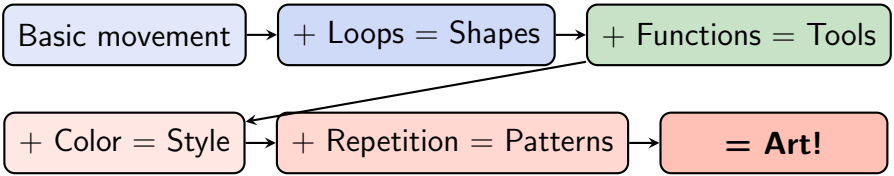
```
1  import turtle, random
2  s = turtle.Screen()
3  s.bgcolor("black")
4  t = turtle.Turtle()
5  t.speed(0); t.hideturtle()
6
7  # Moon
8  t.penup(); t.goto(250, 200); t.pendown()
9  t.fillcolor("lightyellow")
10 t.begin_fill(); t.circle(40); t.end_fill()
11
12 # Scatter stars using draw_star()
13 for i in range(25):
14     t.penup()
15     t.goto(random.randint(-380, 380),
16            random.randint(-50, 280))
17     t.pendown()
18     draw_star(t, random.randint(5, 25),
19        random.choice(["white", "yellow"]))
```

# Summary

# The Journey


Basic movement
+ Loops = Shapes
+ Functions = Tools
+ Color = Style
+ Repetition = Patterns
= Art!

# The Journey




Every concept — loops, functions, modules, objects, random — came from **earlier in the course**.
Turtle just makes it **visual**.

# Key Commands Cheat Sheet

| Category | Commands |
|---|---|
| **Move** | `forward(d)`, `backward(d)`, `goto(x,y)` |
| **Turn** | `left(a)`, `right(a)`, `setheading(a)` |
| **Pen** | `penup()`, `pendown()`, `pensize(w)` |
| **Color** | `pencolor(c)`, `fillcolor(c)` |
| **Fill** | `begin_fill()`, `end_fill()` |
| **Screen** | `bgcolor(c)`, `title(s)`, `mainloop()` |
| **Control** | `speed(n)`, `hideturtle()`, `shape(s)` |
| **Text** | `write(text, align=, font=)` |

# The Big Idea

Code isn't just about calculations.

It's a **creative tool**.

The same loops and functions you learned
for number-crunching just made **art**.

**Next time:** Tkinter — from drawing to **interactive** GUIs.

"Turtle is the artist; Tkinter is the architect."

# Practice Ideas

1. **Draw your initials** using `penup`/`pendown` between letters
2. **Polygon gallery:** concentric triangle, square, pentagon, hexagon
3. **Spiral challenge:** try angles 30, 45, 72, 89, 91, 121, 175
4. **Olympic rings:** five colored, overlapping circles
5. **Bullseye:** concentric filled circles, alternating red and white

# Questions?