مکتب

Quick Reference Notes

Week 13: Python Turtle Graphics

7th Grade Computer Science

1 Introduction

1.1 This Week's Big Question

How can coding turn mathematical patterns into beautiful art? This week, we celebrate everything you've learned by creating visual masterpieces with Python's Turtle Graphics. You'll see your loops come alive as colorful patterns on screen!

Prerequisites

Before starting this celebration week, you should be comfortable with:

- Creating and using for loops with range() (Week 9)
- Working with nested loops (Week 10)
- Combining loops with conditions (Week 11)
- Understanding how loops create patterns

1.2 What You Already Know

You've mastered the fundamental building blocks of programming. You can create loops that repeat actions, nest loops to create complex patterns, and use conditions to control program flow. You understand how to trace through code and debug when things go wrong.

1.3 What You'll Be Able to Do

By the end of this celebration week, you'll be able to:

- Use Python's Turtle Graphics to draw on screen
- Create art with code - from smiley faces to Islamic patterns!
- Apply your loop knowledge to make animated designs
- Draw the Pakistani flag, emoji faces, and creative patterns
- Share your masterpieces in our Loop Art Gallery
- See programming as a creative tool for self-expression

2 VIDEO 1: Welcome to Turtle Graphics

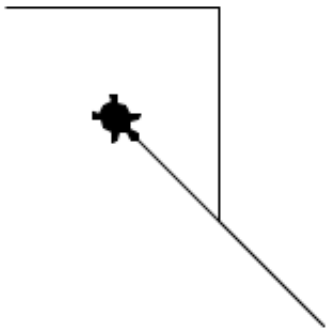
2.1 Your Digital Drawing Board

Imagine having a robot turtle with a magical pen. You're the puppeteer, and the turtle follows your every command - forward, backward, spin around! That's Python's Turtle Graphics - your canvas for digital creativity.

2.2 Let's Start Simple

First, let's see the turtle move and draw:

```
from turtle import *

# Draw a line
forward(100)

# Turn and draw another line
right(90)
forward(100)

# Let's see the turtle icon!
shape("turtle")

# Draw more lines to explore
left(45)
forward(70)
backward(140)  # Go back and forth!

done()
```



2.3 Basic Movement Commands

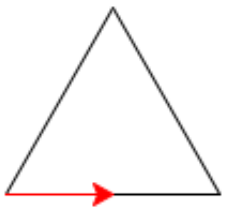
Here are your turtle's basic moves:

Command	What it does	Example
`forward(n)`	Move forward n steps	`forward(50)`
`backward(n)`	Move backward n steps	`backward(30)`
`right(degrees)`	Turn right	`right(90)`
`left(degrees)`	Turn left	`left(45)`
`speed(n)`	Set drawing speed	`speed(5)`
`shape("turtle")`	Show turtle icon	`shape("turtle")`

2.4 Drawing Our First Shape

Now let's draw a triangle using what we learned:

```
from turtle import *

speed(3)  # Slow enough to watch

# Draw a triangle
forward(100)
left(120)    # Why 120? External angle!
forward(100)
left(120)
forward(100)
left(120)    # Back to starting direction

# Add some color
color("red")
forward(50)

done()
```



Tip

Getting Started with Turtle:

- Always start with `from turtle import *`
- End with `done()` to keep the window open
- The turtle starts in the center facing right (like 3 o'clock)
- Forward moves in the direction turtle is facing
- Right and left turn the turtle (doesn't move it)

Quick Check

If the turtle is facing right and you run `right(90)`, which direction will it face? (Hint: Think clock positions!)

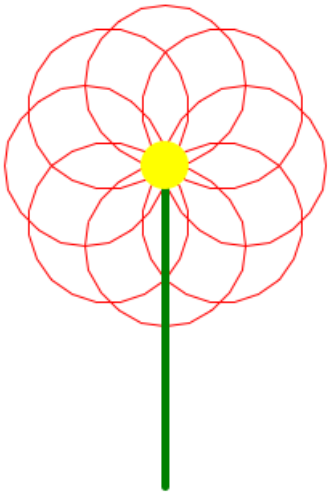
3 VIDEO 2: Loops Create Magic

3.1 From Simple Shapes to Art

Let's use loops to create a beautiful flower:

```
from turtle import *

speed(8)
color("red")
```

```
5
6  # Draw a flower with 8 petals
7  for petal in range(8):
8      circle(50)       # Draw one petal
9      right(45)        # Rotate for next petal
10
11 # Add stem
12 color("green")
13 pensize(5)
14 right(90)
15 forward(200)
16
17 # Add center
18 penup()
19 home()
20 pendown()
21 color("yellow")
22 dot(30)
23
24 done()
```

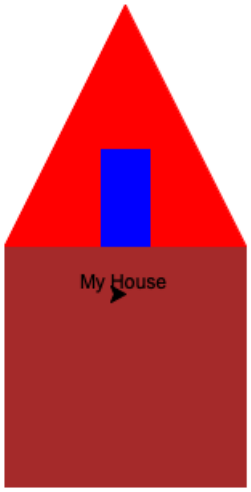


3.2 Creating a Simple House

Let's build something more complex using loops and basic shapes:

```
1  from turtle import *
2
3  speed(5)
4
5  # Draw house base
6  color("brown")
7  begin_fill()
8  for i in range(4):
9      forward(150)
10     right(90)
11 end_fill()
12
13 # Draw roof
14 color("red")
15 begin_fill()
16 goto(75, 150)     # Go to top middle
17 goto(0, 0)        # Back to start
18 goto(150, 0)      # To right corner
```

```
goto(75, 150)     # Complete triangle
end_fill()

# Draw door
penup()
goto(60, 0)
pendown()
color("blue")
begin_fill()
for i in range(2):
    forward(30)
    left(90)
    forward(60)
    left(90)
end_fill()

# Add your name
penup()
goto(75, -30)
color("black")
write("My House", align="center", font=("Arial", 12, "normal"))

done()
```



Rule

Shape Drawing Patterns:

```
# Any polygon
for i in range(sides):
    forward(length)
    right(360 / sides)

# Filled shapes
begin_fill()
# Draw your shape here
end_fill()

# Repeated patterns
for i in range(count):
    # Draw pattern
    # Move or rotate
```

Quick Check

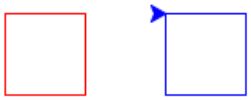
How would you modify the flower code to make 12 petals instead of 8?

4 VIDEO 3: Moving Without Drawing

4.1 The Magic of penup() and pendown()

Sometimes we want to move without drawing. That's where 'penup()' comes in:

```
from turtle import *

speed(5)

# Draw first square
color("red")
for i in range(4):
    forward(50)
    right(90)

# Move to new location WITHOUT drawing
penup()
forward(100)
pendown()

# Draw second square
color("blue")
for i in range(4):
    forward(50)
    right(90)

done()
```

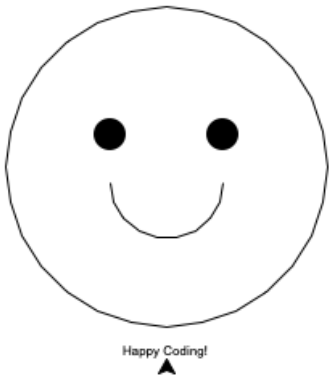


4.2 Creating a Smiley Face

Now we know enough to draw a face:

```
from turtle import *

speed(5)

# Draw face outline
circle(100)

# Move to draw left eye
penup()
goto(-35, 120)
pendown()
dot(20)

# Move to draw right eye
penup()
goto(35, 120)
pendown()
dot(20)
```

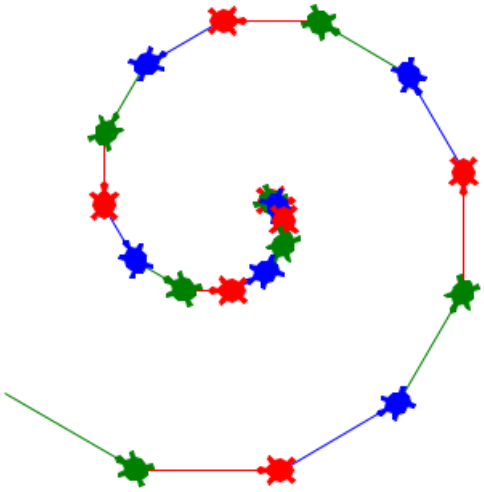
```
19
20 # Draw smile
21 penup()
22 goto(-35, 90)
23 pendown()
24 right(90)
25 circle(35, 180)  # Half circle for smile
26
27 # Write a message
28 penup()
29 goto(0, -20)
30 pendown()
31 write("Happy Coding!", align="center")
32
33 done()
```



4.3 Pattern with Stamps

Let's create a spiral using the 'stamp()' command:

```
1  from turtle import *
2
3  speed(8)
4  shape("turtle")  # Show turtle shape
5
6  # Spiral of stamps
7  for i in range(20):
8      # Rainbow colors
9      if i % 3 == 0:
10         color("red")
11     elif i % 3 == 1:
12         color("green")
13     else:
14         color("blue")
15
16     stamp()            # Leave turtle imprint
17     forward(i * 5)     # Increasing distance
18     right(30)
19
20 hideturtle()  # Hide turtle at end
21 done()
```



Tip

Animation Effects:

- Use `i` in calculations to create growth: `forward(i * 5)`
- Change colors with conditions for rainbow effects
- Combine `stamp()` with movement for trails
- Use `speed(0)` for instant drawing, `speed(1)` to watch it grow

Quick Check

In the spiral code, what happens if you change `right(35)` to `right(90)`?

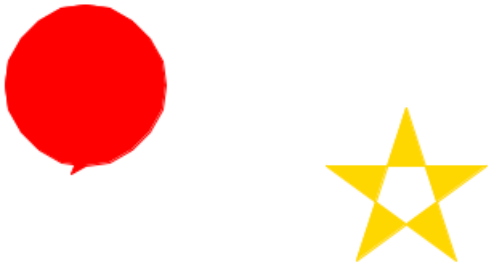
5 VIDEO 4: Advanced Patterns and Fills

5.1 Filling Shapes with Color

Let's learn to fill our shapes:

```
from turtle import *

speed(5)

# Draw a filled star
color("gold")
begin_fill()
for i in range(5):
    forward(100)
    right(144)
end_fill()

# Move and draw filled circle
penup()
goto(-150, 0)
pendown()

color("red")
begin_fill()
circle(50)
end_fill()
```

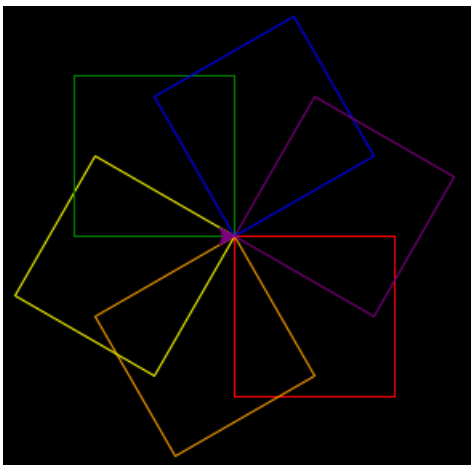
```
22
23 done()
```



5.2 Nested Loops for Complex Patterns

Let's create a mesmerizing pattern:

```
1  from turtle import *
2
3  speed(0)  # Fastest
4  bgcolor("black")
5
6  # Draw rotating squares pattern
7  for rotation in range(6):
8      for i in range(4):
9          # Different color each rotation
10         if rotation == 0:
11             color("red")
12         elif rotation == 1:
13             color("orange")
14         elif rotation == 2:
15             color("yellow")
16         elif rotation == 3:
17             color("green")
18         elif rotation == 4:
19             color("blue")
20         else:
21             color("purple")
22
23         forward(100)
24         right(90)
25
26     right(60)  # Rotate for next square
27
28 done()
```



5.3 Creating the Pakistani Flag

Let's combine everything we learned:

```
from turtle import *

speed(5)
bgcolor("black")

# Green background
color("darkgreen")
begin_fill()
for i in range(2):
    forward(300)
    left(90)
    forward(200)
    left(90)
end_fill()

# White stripe
goto(0, 0)
color("white")
begin_fill()
for i in range(2):
    forward(75)
    left(90)
    forward(200)
    left(90)
end_fill()

# Crescent moon
penup()
goto(180, 50)
pendown()
color("white")
begin_fill()
circle(50)
end_fill()

penup()
goto(190, 70)
pendown()
color("darkgreen")
begin_fill()
circle(40)
end_fill()

# Star
penup()
goto(200, 140)
pendown()
color("white")
begin_fill()
for i in range(5):
    forward(20)
    right(144)
end_fill()

# Title
penup()
goto(150, -30)
```

```
color("white")
write("Pakistan Zindabad!", align="center", font=("Arial", 16, "bold"))

done()
```

Common Error

Common Turtle Graphics Mistakes:

Forgot `penup()` before moving?

- You'll draw unwanted lines
- Always lift pen before repositioning

Shapes not filling properly?

- Must use `begin_fill()` before drawing
- Must use `end_fill()` after drawing
- Shape must be closed (end where you started)

Text not showing?

- Check text color isn't same as background
- Make sure pen is down

Challenge Yourself

Can you create:

- Your name's first letter using turtle graphics?
- A simple emoji face (happy, sad, or winking)?
- A pattern inspired by truck art?
- A night sky with stars and moon?
- Your own creative animation?

Share your best creation in our Loop Art Gallery!

Quick Check

How would you make the fireworks burst in different sizes? (Hint: use the loop variable!)

6 VIDEO 5: Summary

6.1 Celebrating Your Journey

What an incredible journey! From "Hello World" to creating digital masterpieces. You've not just learned to code - you've become a digital artist, problem solver, and logical thinker.

6.2 Your Programming Superpowers

Look at your transformation since Week 1:

Week 1: Could only...	Week 13: Can now...
Use Scratch blocks	Write Python code fluently
Make simple animations	Create complex patterns with loops
Follow instructions	Design your own algorithms
Fix syntax errors	Debug logically and systematically
Think sequentially	Think in patterns and abstractions
Solve given problems	Create original solutions

6.3 Beyond Turtle Graphics

The skills you've mastered extend far beyond drawing:

- **Loops** - Automate any repetitive task
- **Variables** - Store and manipulate any data
- **Conditions** - Make smart programs that decide
- **Debugging** - Solve problems systematically
- **Decomposition** - Break big challenges into steps
- **Creativity** - Express ideas through code

Landmark Moment

You did it! You've completed Module 1: Foundations of Programming!

From Scratch blocks to Python code, from confusion to confidence, from copying examples to creating original art - you've transformed into a programmer. You've learned not just syntax, but how to think computationally.

These skills - logical thinking, problem decomposition, pattern recognition, and systematic debugging - will serve you in mathematics, science, and life. Whether you become a doctor, engineer, artist, or entrepreneur, you now have the power to make computers work for you.

Be proud! Share your creations! Keep coding! The journey has just begun!

7 Quick Reference

Quick Reference

Turtle Graphics Command Reference:

```
# Setup
from turtle import *

# Movement
forward(distance)      # Move forward
backward(distance)     # Move backward
right(degrees)         # Turn right
left(degrees)          # Turn left
goto(x, y)             # Jump to position
home()                 # Go to (0, 0)
circle(radius)         # Draw circle
circle(radius, angle)  # Draw arc

# Pen Control
penup()                # Stop drawing
pendown()              # Start drawing
pensize(width)         # Line thickness
pencolor("color")      # Change pen color
fillcolor("color")     # Set fill color
begin_fill()           # Start fill
end_fill()             # Complete fill

# Drawing
dot(size)              # Draw filled circle
stamp()                # Leave turtle imprint
write(text)            # Write text

# Display
speed(n)               # 0=instant, 1-10 speeds
shape("turtle")        # Show turtle shape
hideturtle()           # Hide turtle
bgcolor("color")       # Background color
clear()                # Clear drawings
done()                 # Keep window open

# Fun Patterns
# Star
for i in range(5):
    forward(50)
    right(144)

# Flower
for i in range(8):
    circle(30)
    right(45)
```

8 Reflection & Gallery Preparation

As you prepare your masterpiece for the Loop Art Gallery:

1. What's the most creative thing you made with Turtle Graphics? What programming concepts made it possible?

2. Which was more challenging - planning your art or coding it? Why?
3. How did debugging turtle drawings help you understand debugging in general?
4. Look at a classmate's creation. What clever techniques did they use that you could learn from?
5. If you could combine Turtle Graphics with any other subject (math, art, science), what would you create?
6. What advice would you give to next year's students starting their programming journey?
7. How has learning to code changed the way you approach problems in other subjects?
8. What do you want to learn next in programming? What dreams do you have for your coding future?

Gallery Challenge: Create your signature piece - something that represents YOU and shows off everything you've learned. Add your name, use creative colors, include patterns, and make it memorable!
cm

Welcome to the community of programmers! Your journey continues...