CODE KALEIDOSCOPE

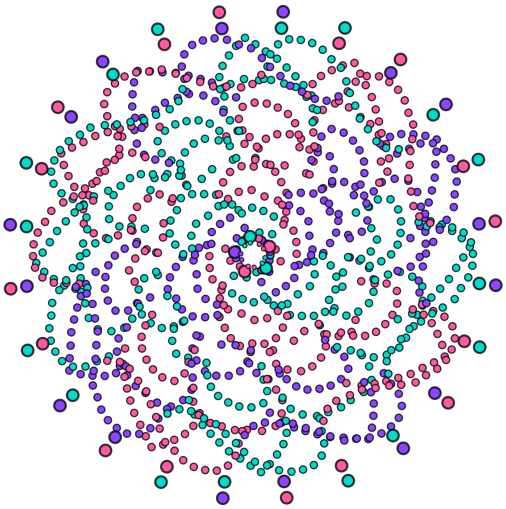
Creative Code. Colourful Thinking.

Creative Coding Student Workbook

Creative Coding with Python Turtle

Intro + 6 Lessons + Final Project

A KS3 workbook for the introduction lesson, Python Turtle lessons, final project and assessment.



https://code-kaleidoscope.com/

Inside this workbook

- ☐ PRIMM lesson notes
- ☐ WAGOLL examples
- ☐ Project portfolio
- ☐ Project homework
- ☐ Assessment
- ☐ DIRT reflection

Student Details

Student name

Class

Teacher

School

Glossary

Variable	A named value that can change.	size = 50	
Integer	A whole number.	repeats = 12	
String	Text inside quotation marks.	colour = "cyan"	
Loop	Code that repeats.	for i in range(6):	
Selection	Code that makes a choice.	if choice == 'star':	
Random	A value chosen by the program.	choice(colours)	
Function	A named reusable block of code.	def draw_star():	
Debugging	Finding and fixing errors.	Check spelling.	
Syntax	Rules for writing code correctly.	Use a colon after for.	
Input	Information given to a program.	A typed value.	
Output	What the program produces.	A Turtle drawing.	

Introduction Lesson: Exploring Code Kaleidoscope

Lesson Overview

Learning Aim: Explore Code Kaleidoscope safely and creatively, and explain how changing values affects the artwork.

Challenge: Predict, test and improve a first Code Kaleidoscope artwork.

Key Vocabulary: creative coding, variable, output, symmetry, pattern, prediction, testing, improvement

DO NOW: Spot the Pattern

Look at the pattern examples from the lesson screen.

1. What patterns can you see?
2. What do all the designs have in common?
3. How do you think a computer could create these patterns?
4. What does the word symmetry mean?

Stretch: What instructions might a computer need to create one of these designs?

PRIMM Tracker

- ☐ Predict
- ☐ Run
- ☐ Investigate
- ☐ Modify
- ☐ Make

Sketch Box: Draw your prediction

Explain Box: What happened?

Improve Box: What would you change next?

Introduction Lesson PRIMM Response

Predict: What do I think the code will do? ☐

Run: What happened when I ran it? ☐

Investigate: Which lines controlled the output? ☐

Modify: What did I change and why? ☐

Make: What did I create independently? ☐

Self-Assessment: I can...

☐ Use Turtle commands	☐ Use variables	☐ Use selection
☐ Use loops	☐ Use data types	☐ Debug code

Plenary Question

- One thing I changed was...
- One thing I noticed was...
- One key word I can explain is...
- One improvement I would make next is...

Lesson 1: From Scratch to Turtle Python

Lesson Overview

Learning Aim: Understand how Scratch commands connect to Python Turtle code.

Challenge: Draw a simple Turtle shape using commands in order.

Key Vocabulary: sequence, command, syntax, turtle

DO NOW: Scratch vs Python

Compare the Scratch and Python screenshots on the lesson page.

1. What is similar?
2. What is different?
3. Which version do you think is easier to read?
4. What do you think the Python code will do?

Stretch: Circle or list any words you recognise from Scratch.

PRIMM Tracker

- ☐ Predict
- ☐ Run
- ☐ Investigate
- ☐ Modify
- ☐ Make

Sketch Box: Draw your prediction

Explain Box: What happened?

Improve Box: What would you change next?

Lesson 1 PRIMM Response

Predict: What do I think the code will do? ☐

Run: What happened when I ran it? ☐

Investigate: Which lines controlled the output? ☐

Modify: What did I change and why? ☐

Make: What did I create independently? ☐

Self-Assessment: I can...

☐ Use Turtle commands	☐ Use variables	☐ Use selection
☐ Use loops	☐ Use data types	☐ Debug code

Plenary Question

- How is Turtle similar to Scratch?
- Why does sequence matter?
- What changed when you changed the order of commands?

Lesson 2: Variables and Interactive Art

Lesson Overview

Learning Aim: Use variables to store values that control Turtle artwork.

Challenge: Use variables to control colour, size and pen thickness.

Key Vocabulary: variable, value, assignment, size

DO NOW: Guess the Variable

```
size = 50
colour = "blue"
repeats = 12
```

1. Which variable controls the size?
2. Which variable stores text?
3. Which variable might control symmetry?
4. What might happen if repeats becomes 24?

Stretch: Write your own variable.

PRIMM Tracker

- ☐ Predict
- ☐ Run
- ☐ Investigate
- ☐ Modify
- ☐ Make

Sketch Box: Draw your prediction

Explain Box: What happened?

Improve Box: What would you change next?

Lesson 2 PRIMM Response

Predict: What do I think the code will do? ☐

Run: What happened when I ran it? ☐

Investigate: Which lines controlled the output? ☐

Modify: What did I change and why? ☐

Make: What did I create independently? ☐

Self-Assessment: I can...

- ☐ Use Turtle commands
- ☐ Use variables
- ☐ Use selection
- ☐ Use loops
- ☐ Use data types
- ☐ Debug code

Plenary Question

- Why is it useful to store values in variables?

Lesson 3: Selection and Interactive Drawing

Lesson Overview

Learning Aim: Use selection so a program can choose between different visual outcomes.

Challenge: Create an interactive drawing choice using selection.

Key Vocabulary: selection, condition, if, else

DO NOW: What Happens Next?

```
if colour == "blue":
    circle(50)
else:
    square(50)
```

1. What shape will be drawn if colour is blue?
2. What shape will be drawn if colour is red?
3. What does if mean?
4. Why is selection useful?

Stretch: Think of a real-life example of selection.

PRIMM Tracker

- ☐ Predict
- ☐ Run
- ☐ Investigate
- ☐ Modify
- ☐ Make

Sketch Box: Draw your prediction

Explain Box: What happened?

Improve Box: What would you change next?

Lesson 3 PRIMM Response

Predict: What do I think the code will do?

Run: What happened when I ran it?

Investigate: Which lines controlled the output?

Modify: What did I change and why?

Make: What did I create independently?

Self-Assessment: I can...

- ☐ Use Turtle commands
- ☐ Use variables
- ☐ Use selection
- ☐ Use loops
- ☐ Use data types
- ☐ Debug code

Plenary Question

- How is selection similar to making a choice in Scratch?

Lesson 4: Loops and Kaleidoscope Patterns

Lesson Overview

Learning Aim: Use loops to repeat commands and build symmetrical Turtle patterns.

Challenge: Create a kaleidoscope pattern using repeated Turtle commands.

Key Vocabulary: loop, repeat, iteration, range

DO NOW: Repetition Challenge

```
forward(100)
right(90)

forward(100)
right(90)

forward(100)
right(90)

forward(100)
right(90)
```

1. What shape will this draw?
2. What is repeated?
3. How many times is it repeated?
4. How could a loop make this shorter?

Stretch: Write the number of repeats needed.

PRIMM Tracker

- ☐ Predict
- ☐ Run
- ☐ Investigate
- ☐ Modify
- ☐ Make

Sketch Box: Draw your prediction

Explain Box: What happened?

Improve Box: What would you change next?

Lesson 4 PRIMM Response

Predict: What do I think the code will do? ☐

Run: What happened when I ran it? ☐

Investigate: Which lines controlled the output? ☐

Modify: What did I change and why? ☐

Make: What did I create independently? ☐

Self-Assessment: I can...

- ☐ Use Turtle commands
- ☐ Use variables
- ☐ Use selection
- ☐ Use loops
- ☐ Use data types
- ☐ Debug code

Plenary Question

- How does a loop reduce repeated code?

Lesson 5: Data Types and Generative Art

Lesson Overview

Learning Aim: Use different data types to control creative program output.

Challenge: Use data values to create controlled generative variation.

Key Vocabulary: integer, string, list, random

DO NOW: Sort the Data

```
12
"blue"
50
"teal"
100
"flower"
36
"circle"
```

1. Which values are numbers?
2. Which values are text?
3. Which values could control a Turtle design?

Stretch: What might randint(1, 10) do?

PRIMM Tracker

- ☐ Predict
- ☐ Run
- ☐ Investigate
- ☐ Modify
- ☐ Make

Sketch Box: Draw your prediction

Explain Box: What happened?

Improve Box: What would you change next?

Lesson 5 PRIMM Response

Predict: What do I think the code will do? ☐

Run: What happened when I ran it? ☐

Investigate: Which lines controlled the output? ☐

Modify: What did I change and why? ☐

Make: What did I create independently? ☐

Self-Assessment: I can...

- ☐ Use Turtle commands
- ☐ Use variables
- ☐ Use selection
- ☐ Use loops
- ☐ Use data types
- ☐ Debug code

Plenary Question

- How can data make artwork feel generative?

Lesson 6: Debugging Creative Code

Lesson Overview

Learning Aim: Debug creative Python code by reading errors, testing changes and improving output.

Challenge: Find and fix errors in Turtle code.

Key Vocabulary: debug, syntax error, logic error, test

DO NOW: Find the Bug

```
for i in range(4)
    forward(100)
    right(90)
```

1. Can you spot the mistake?
2. What error might appear?
3. How would you fix it?
4. Why is testing important?

Stretch: What shape should the corrected code draw?

PRIMM Tracker

- ☐ Predict
- ☐ Run
- ☐ Investigate
- ☐ Modify
- ☐ Make

Sketch Box: Draw your prediction

Explain Box: What happened?

Improve Box: What would you change next?

Lesson 6 PRIMM Response

Predict: What do I think the code will do? ☐

Run: What happened when I ran it? ☐

Investigate: Which lines controlled the output? ☐

Modify: What did I change and why? ☐

Make: What did I create independently? ☐

Self-Assessment: I can...

- ☐ Use Turtle commands
- ☐ Use variables
- ☐ Use selection
- ☐ Use loops
- ☐ Use data types
- ☐ Debug code

Plenary Question

- Why is testing one change at a time useful?

Final Project: Design Portfolio

Design Brief

Create an original Python Turtle artwork. Show planning, testing, debugging and improvement. Your final piece should use ideas from across the unit.

DO NOW: Design Critique

Look at the Design Critique image on the lesson page.

1. Which programming concepts can you identify?
2. What do you like about the design?
3. What would you change?
4. Which pathway will you choose: Bronze, Silver or Gold? Why?

Stretch: Sketch one improvement you could make.

Bronze Pathway - Supported

Start from a template and change colours, sizes, repeats, pen thickness and shape choices.

Silver Pathway - Guided

Combine lesson ideas to create a flower, starburst, mandala or geometric pattern.

Gold Pathway - Independent

Create your own design using variables, loops, functions and input if appropriate.

Defined End Point

- [ ] My design is original
- [ ] I used programming concepts
- [ ] I tested and improved my code
- [ ] I can explain my choices

Project Planning Sheet

My idea

Large Sketch Area

Colour palette

Variables I will use

Programming techniques

Potential problems

Development Log and Testing Log

Version 1

What worked?

What needs improving?

Version 2

What worked?

What needs improving?

Version 3

What worked?

What needs improving?

Final Evaluation

What am I most proud of?

What programming concepts did I use?

What would I improve next time?

PROJECT HOMEWORK: CREATIVE CODING CHALLENGE

Mission

Complete the activities to prove what you remember from the whole unit. Think like a code detective: look for clues, patterns and mistakes.

Keyword Detective

Circle or tick the words you can explain confidently.

☐ sequence	☐ variable	☐ input
☐ selection	☐ loop	☐ data type
☐ random	☐ function	☐ debugging
☐ syntax	☐ output	

Fill in the Gaps

A _____ stores a value.

A _____ repeats code.

_____ means fixing mistakes.

The final drawing is the program _____.

Matching Activity

1. Syntax	matches	_____	A. Choosing between outcomes
2. Selection	matches	_____	B. Rules for writing code
3. Random	matches	_____	C. A reusable named block
4. Function	matches	_____	D. A value chosen by the program

Retrieval Questions

1. Why does sequence matter?

_____

2. What does input do?

_____

3. How can variables improve art?

_____

4. What does output mean?

_____

Code Reading Task

```
size = 40
for i in range(6):
    circle(size)
    right(60)
```

What pattern will this code create? Which line is the loop? What could you change to make it larger?

End-of-Unit Assessment: Part A

Knowledge Check. Choose the best answer for each question.

1. Which statement best describes sequence?

- ☐ A. Instructions run in order
- ☐ B. A value is stored
- ☐ C. A random number is chosen
- ☐ D. A bug is fixed

2. Why would a programmer use a variable called size?

- ☐ A. To store a value that controls drawing size
- ☐ B. To make the turtle move randomly
- ☐ C. To delete an error message
- ☐ D. To change the file name

3. What does input mean in a creative coding program?

- ☐ A. Information given to the program
- ☐ B. The finished drawing
- ☐ C. A spelling mistake
- ☐ D. A repeated command

4. Which line shows selection?

- ☐ A. if choice == "circle":
- ☐ B. for i in range(6):
- ☐ C. size = 50
- ☐ D. forward(100)

5. What does a loop help you do?

- ☐ A. Repeat commands efficiently
- ☐ B. Stop all code running
- ☐ C. Turn text into an image
- ☐ D. Save a file automatically

6. Which value is an integer?

- ☐ A. 12
- ☐ B. "purple"
- ☐ C. True
- ☐ D. pencolor

7. Why might random values be useful in generative art?

- ☐ A. They create controlled variation
- ☐ B. They guarantee no errors
- ☐ C. They remove all loops
- ☐ D. They stop the turtle moving

8. What is the purpose of a function?

- ☐ A. To reuse a named block of code
- ☐ B. To make every value a string
- ☐ C. To hide syntax errors
- ☐ D. To print the worksheet

9. What should you do first when debugging?

- ☐ A. Read the error and test one change
- ☐ B. Change every line at once
- ☐ C. Delete the whole program
- ☐ D. Ignore the output

10. What is output in Turtle graphics?

- ☐ A. The drawing produced by the code
- ☐ B. The value typed by the user
- ☐ C. The name of the editor
- ☐ D. The indentation spaces

End-of-Unit Assessment: Part B

Code Reading

```
from turtle import *
colour = "purple"
repeats = 8
pencolor(colour)
for i in range(repeats):
    forward(80)
    right(45)
```

1. Identify one variable and explain what it controls.

2. Which line creates iteration? Explain how you know.

3. Predict what would change if repeats became 12.

End-of-Unit Assessment: Part C

Computational Thinking

Explain how decomposition, pattern recognition or algorithm design could help you plan a Turtle artwork.

Written explanation

Practical Design Checklist

- [ ] Used variables
- [ ] Used loops
- [ ] Used selection
- [ ] Used data types
- [ ] Used debugging
- [ ] Created original artwork

Plan your practical design task

End-of-Unit Assessment: Part D

Reflection: What went well?

Reflection: What would I improve next time?

Teacher Marking Grid

Emerging	Developing	Secure	Mastering
Needs support to identify concepts.	Uses some concepts with guidance.	Applies concepts accurately.	Combines concepts independently.

DIRT Response Sheet

DIRT = Dedicated Improvement and Reflection Time

Student Name	Date	Teacher Initials

Teacher Feedback: Highlight the Sentences That Apply

WWW - What Went Well

- [ ] You worked with focus and effort.
- [ ] You used variables effectively.
- [ ] You used selection correctly.
- [ ] You used loops / iteration effectively.
- [ ] Your code runs and produces the expected output.
- [ ] Your artwork is creative and well developed.
- [ ] You tested your code and made improvements.
- [ ] You explained your ideas clearly.

EBI - Even Better If

- [ ] Use more meaningful variable names.
- [ ] Add comments to explain your code.
- [ ] Check your indentation and syntax.
- [ ] Add input so the user can make a choice.
- [ ] Use selection to make decisions.
- [ ] Use a loop to reduce repetition.
- [ ] Break your code into functions.
- [ ] Test your code again and fix errors.

Targeted Praise

- [ ] Excellent creativity.
- [ ] Great problem solving.
- [ ] Strong debugging.
- [ ] Clear explanation.
- [ ] Good resilience.
- [ ] Improved confidence.
- [ ] Independent working.
- [ ] Well structured code.

DIRT Task: Highlight the Improvement Task for the Student

Code Improvements

- [ ] Fix errors so your code runs correctly.
- [ ] Improve the structure of your code.
- [ ] Add comments to explain your code.
- [ ] Use clearer variable names.
- [ ] Remove repeated code.
- [ ] Break your code into functions.

Concept Improvements

- [ ] Add input to make the program interactive.
- [ ] Use selection to make a choice.
- [ ] Use a loop to repeat part of your code.
- [ ] Use random to add variation.
- [ ] Use the correct data type.
- [ ] Explain the output of your program.

Design and Output Improvements

- [ ] Improve the design of your artwork.
- [ ] Add more colours, shapes or patterns.
- [ ] Make your output more detailed.
- [ ] Improve the layout on screen.
- [ ] Add an extra feature.
- [ ] Improve the user experience.

Student Response

What improvement did I make?

What was challenging?

What did I learn?