Code Library

Tiny programs you can copy, paste, and run. One library at a time, then mix them together.

How to use this handout

Each example has three parts: **what it does**, **the code**, and **a tweak to try**. Copy any code block into the Trinket editor on the lesson page, then press **Run**. Change a number, change a colour, change a word - see what happens. That's how you learn.

Want to try your own idea? Ask an AI like Claude or ChatGPT for something like *'Write me a short Python turtle program that draws a flower'*, then paste the code into the Trinket editor and click Run. Reading other people's code is one of the fastest ways to get better.

TURTLE - the drawing toolbox

Turtle moves a little pen around the screen. You tell it: go forward, turn left, pick a colour. With just a few lines you can draw shapes, spirals, even tiny houses.

1. Draw a line (the simplest turtle program)

Bring in the toolbox, make a turtle, send it forward.

```
import turtle

t = turtle.Turtle()
t.forward(100)
t.left(90)
t.forward(100)
turtle.done()
```

Try changing 100 to 200 and watch the line grow. Change 90 to 45 - what shape do you start to make?

2. Draw a square (using a loop)

A square is four sides and four 90-degree turns. A loop saves us from typing the same thing four times.

```
import turtle

t = turtle.Turtle()
for i in range(4):
    t.forward(100)
    t.left(90)
turtle.done()
```

Change range(4) to range(5) and left(90) to left(72) - you've got a pentagon!

3. Colourful square spiral

Pick four colours and cycle through them. Each loop, draw a bit further than the last.

```
import turtle

screen = turtle.Screen()
screen.bgcolor("black")

t = turtle.Turtle()
t.pensize(4)
t.speed(0)

colours = ["cyan", "magenta", "yellow", "lime"]
for i in range(40):
    t.color(colours[i % 4])
    t.forward(i * 5)
    t.right(90)

t.penup()
t.goto(0, -100)
t.color("white")
t.write("Coded by YOUR NAME", align="center", font=("Arial", 18, "bold"))
screen.mainloop()
```

This is the spiral from class. Try changing right(90) to right(91) and watch the spiral curve.

4. Mini house

A square + a triangle = a tiny house. Each shape is a small loop.

```
import turtle

screen = turtle.Screen()
screen.bgcolor("skyblue")

t = turtle.Turtle()
t.pensize(3)

# the body
t.fillcolor("burlywood")
t.begin_fill()
for i in range(4):
    t.forward(150); t.left(90)
t.end_fill()

# the roof
t.penup(); t.goto(0, 150); t.setheading(0); t.pendown()
t.fillcolor("red")
t.begin_fill()
for i in range(3):
    t.forward(150); t.left(120)
t.end_fill()

t.hideturtle()
screen.mainloop()
```

Add a door or a window of your own - they're just more shapes!

RANDOM - the surprises toolbox

The *random* library is tiny but powerful. It picks numbers, colours, and items from a list so your programs feel different every time.

5. Roll a dice

Pick a random whole number between 1 and 6, just like a real dice.

```
import random

roll = random.randint(1, 6)
print("You rolled a", roll)
```

Run it five times. Different number every time, right? That's randomness.

6. Random name picker

Give random.choice() a list and it grabs one item at random. Great for picking teams or daily winners.

```
import random

players = ["Ada", "Grace", "Linus", "Hedy", "Alan"]
winner = random.choice(players)
print("Today's winner is:", winner)
```

Add your own friends' names to the list, then run it.

PYGAME - the games toolbox

Pygame opens a window, draws shapes and images, plays sounds, and listens to the keyboard. Every game we build in this course uses pygame.

7. Open a window

The very first pygame program: open a black 600x400 window with a title.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("My First Pygame Window")

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

pygame.quit()
```

The 'while running' loop keeps the window open until you click the close button.

8. Draw a coloured square

Fill the background, draw a blue rectangle, flip the screen so we can see it.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Hello Shapes")

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    screen.fill((255, 255, 255))                    # white background
    pygame.draw.rect(screen, (0, 0, 255), (250, 150, 100, 100))  # blue square
    pygame.display.flip()

pygame.quit()
```

Colours are (Red, Green, Blue), 0-255. Try (255, 0, 0) for red, (0, 255, 0) for green.

9. A square that moves on its own

Same as #8, but we keep adding to x so the square slides right.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Moving Square")

x, y = 50, 180
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    screen.fill((255, 255, 255))
    pygame.draw.rect(screen, (0, 0, 255), (x, y, 50, 50))
    x += 2
    if x > 600:
        running = False
    pygame.display.flip()

pygame.quit()
```

This is the heart of every game: change a number a tiny bit, redraw, repeat.

MIXING LIBRARIES - turtle + random

The fun starts when you combine toolboxes. Turtle for drawing, random for surprises = a different artwork every time you run it.

10. Random colour spiral

Turtle draws the spiral. Random picks a new colour every loop.

```
import turtle
import random

screen = turtle.Screen()
screen.bgcolor("black")

t = turtle.Turtle()
t.pensize(2)
t.speed(0)

colours = ["red", "orange", "yellow", "lime", "cyan", "magenta", "white"]
for i in range(120):
    t.color(random.choice(colours))
    t.forward(i * 2)
    t.right(59)

screen.mainloop()
```

Change right(59) to right(91), then to right(120). Each angle makes a totally different pattern!

Where these libraries take us next

Galaxy Defender uses **pygame** (window + shapes + keyboard) and **random** (to scatter enemies in random places). You've just seen the building blocks. Over the next 9 lessons we stack them together to make a full space shooter.

Challenge: pick any example above, change three things, run it, see what happens. That's the secret of every coder - small experiments, fast feedback.