



7. # Draws an egg figure.

```
def main():
    print("  _____")
    print(" /           \\")
    print("/           \\")
    print("-\"'-\"'-\"'-\"'-\"'-\"'-")
    print("\\           /")
    print(" \\_____ /")
```

8. # Draws several egg figures.

```
def main():
    draw_egg()
    draw_egg()
    draw_bottom()
    draw_top()
    draw_line()
    draw_bottom()

def draw_egg():
    draw_top()
    draw_bottom()
    draw_line()

def draw_top():
    print("  _____")
    print(" /           \\")
    print("/           \\")

def draw_bottom():
    print("\\           /")
    print(" \\_____ /")

def draw_line():
    print("-\"'-\"'-\"'-\"'-\"'-\"'-")
```

9. # Draws two rocket ship figures side-by-side.

```
def main():
    print_top()
    print_square()
    print_label()
    print_square()
    print_top()

def print_top():
    print("  /\ \      /\ \")
    print(" /  \ \    /  \ \")
    print("/    \ \  /    \ \")

def print_square():
    print("+-----+ +-----+")
    print("|       | |       |")
    print("|       | |       |")
    print("+-----+ +-----+")

def print_label():
    print("|United| |United|")
    print("|States| |States|")
```

10. # This program prints a college "fight song" with verses and repetition.

# The code uses functions for structure and to remove redundancy.

```
def main():
    go_team_go()
    print()
    big_verse()
```

```

big_verse()
go_team_go()

def go_team_go():
    print("Go, team, go!")
    print("You can do it.")

def big_verse():
    go_team_go()
    print("You're the best,")
    print("In the West.")
    go_team_go()
    print()

```

11. # This program prints a pattern of starry figures.

```

def main():
    print_figure1()
    print()
    print_figure2()
    print()
    print_figure3()

def print_figure1():
    print_horizontal_bar()
    print_x()

def print_figure2():
    print_horizontal_bar()
    print_x()
    print_horizontal_bar()

def print_figure3():
    print(" *")
    print(" *")
    print(" *")
    print_figure1()

def print_horizontal_bar():
    print("*****")
    print("*****")

def print_x():
    print(" * *")
    print(" *")
    print(" * *")

```

12. # This program prints a pattern of figures such as eggs and stop signs.

# The code uses functions for structure and to remove redundancy.

```

def main():
    egg()
    print()
    egg()
    line()
    print()
    stop_sign()
    line()
    print()

def egg():
    egg_top()
    egg_bottom()

def egg_top():
    print(" _____")

```

```

print(" /      \\")
print("/      \\")

def egg_bottom():
    print("\\      /")
    print(" \\_____/")

def stop_sign():
    egg_top()
    print("|  STOP  |")
    egg_bottom()

def line():
    print("+-----+")

```

## Chapter 2

```

1. number = 1
   increment = 3
   for i in range(1, 11):
       print(str(number) + " ", end="")
       number = number + increment
       increment = increment + 2
   print() # to end the line

2. n1 = 1
   n2 = 1
   print(str(n1) + " " + str(n2) + " ", end="")
   for i in range(3, 13):
       n3 = n1 + n2
       n1 = n2
       n2 = n3
       print(str(n2) + " ", end="")
   print()

3. for i in range(1, 5):
    print("*" * 5)

4. for i in range(1, 6):
    print("*" * i)

5. for i in range(1, 8):
    print(str(i) * i)

6. for i in range(1, 7):
    print("|      ", end="")
    print()
    for i in range(1, 7):
        for j in range(1, 11):
            print(j % 10, end="")
    print()

7. COUNT = 6
   INNER_COUNT = 7
   def main():
       for i in range(1, COUNT + 1):
           print("|", end="")
           print(" " * (INNER_COUNT - 1), end="")
       print()

       for i in range(1, COUNT + 1):
           for j in range(1, INNER_COUNT + 1):

```

```

        print(j % INNER_COUNT, end="")
    print()

```

```

main()

```

```

8. def print_design():
    for line in range(1, 6):
        print("-" * (-1 * line + 6), end="")
        print(str(2 * line - 1) * (2 * line - 1), end="")
        print("-" * (-1 * line + 6))

```

```

9. def main():
    for line in range(1, 7):
        print("\\" * (2 * line - 2), end="")
        print("! " * (-4 * line + 26), end="")
        print("/" * (2 * line - 2))

```

```

main()

```

```

10. SIZE = 4

```

```

def main():
    for line in range(1, SIZE + 1):
        print("\\" * (2 * line - 2), end="")
        print("! " * (-4 * line + (4 * SIZE + 2)), end="")
        print("/" * (2 * line - 2))

```

```

main()

```

```

11. # Draws a resizable window figure with for loops

```

```

# and a constant.

```

```

COUNT = 3

```

```

def main():
    draw_line()
    for i in range(2):
        for j in range(COUNT):
            draw_bar()
        draw_line()

```

```

# Draws a horizontal line: +===+===+

```

```

def draw_line():
    print("+", end="")
    print("=" * COUNT, end="")
    print("+", end="")
    print("=" * COUNT, end="")
    print("+")

```

```

# Draws a single line of bars: |   |   |

```

```

def draw_bar():
    print("|", end="")
    print(" " * COUNT, end="")
    print("|", end="")
    print(" " * COUNT, end="")
    print("|")

```

```

main()

```

```

12. def main():

```

```

    for line in range(1, 6):
        print("/" * (-4 * line + 20), end="")
        print("*" * (8 * line - 8), end="")
        print("\\" * (-4 * line + 20))

```

```

main()

```

```

13. SIZE = 5
    def main():
        for line in range(1, SIZE + 1):
            print("/" * (-4 * line + 4 * SIZE), end="")
            print("*" * (8 * line - 8), end="")
            print("\\" * (-4 * line + 4 * SIZE))

    main()

```

## Chapter 3

1.
 

```

def print_numbers(max):
    for i in range(1, max + 1):
        print("[", i, "]", end="")
    print() # to end the line of output

```
2.
 

```

def print_powers_of_2(max):
    for i in range(0, max + 1):
        print(2 ** i, end=" ")
    print() # to end the line of output

```
3.
 

```

def print_powers_of_n(base, exp):
    for i in range(0, exp + 1):
        print(base ** i, end=" ")
    print() # to end the line of output

```
4.
 

```

def print_square(small, big):
    amount = big - small + 1
    for i in range(0, amount):
        for j in range(0, amount):
            print((j + i) % amount + small, end="")
        print()

```
5.
 

```

def larger_abs_val(n1, n2):
    return max(abs(n1), abs(n2))

```
6.
 

```

def quadratic(a, b, c):
    determinant = b * b - 4 * a * c
    root1 = (-b + math.sqrt(determinant)) / (2 * a)
    root2 = (-b - math.sqrt(determinant)) / (2 * a)
    print("First root =", root1)
    print("Second root =", root2)

```
7.
 

```

def last_digit(num):
    return abs(num) % 10

```
8.
 

```

def area(radius):
    answer = math.pi * radius ** 2
    return answer

```
9.
 

```

def pay(salary, hours):
    if hours < 8:
        return salary * hours

```

```

else:
    regular_pay = salary * 8
    overtime_pay = (1.5 * salary) * (hours - 8)
    return regular_pay + overtime_pay

```

10.

```

def sphere_volume(r):
    return 4.0 * math.pi * r * r * r / 3.0;

```

11.

```

def pad_string(s, length):
    spaces = ""
    for i in range(0, length - len(s)):
        spaces += " "
    return spaces + s

```

12.

```

def vertical(string):
    for i in range(0, len(string)):
        print(string[i])

```

13.

```

def main():
    panel = DrawingPanel(220, 150, "yellow")
    panel.set_color(Color.BLUE);
    panel.fill_oval(50, 25, 40, 40, "blue")
    panel.fill_oval(130, 25, 40, 40, "blue")
    panel.fill_rect(70, 45, 80, 80, "red")
    panel.draw_line(70, 85, 150, 85, "black")

```

```

main()

```

14.

```

def main():
    panel = DrawingPanel(450, 150, "yellow")
    draw_figure(panel, 50, 25)
    draw_figure(panel, 250, 45)

def draw_figure(panel, x, y):
    panel.fill_oval(x, y, 40, 40, "blue")
    panel.fill_oval(x + 80, y, 40, 40, "blue")
    panel.fill_rect(x + 20, y + 20, 80, 80, "red")
    panel.draw_line(x + 20, y + 60, x + 100, y + 60)

```

```

main()

```

15.

```

def main():
    panel = DrawingPanel(320, 180)
    draw_face(panel, 10, 30)
    draw_face(panel, 150, 50)

def draw_face(panel, x, y):
    panel.draw_oval(x, y, 100, 100, "black")
    panel.fill_oval(x + 20, y + 30, 20, 20, "blue")
    panel.fill_oval(x + 60, y + 30, 20, 20, "blue")
    panel.draw_line(x + 30, y + 70, x + 70, y + 70, "red")

```

```

main()

```

16.

```

def main():
    panel = DrawingPanel(520, 180)
    for i in range(5):

```

```
draw_face(panel, 10 + i * 100, 30)
```

```
def draw_face(panel, x, y):
    panel.draw_oval(x, y, 100, 100, "black")
    panel.fill_oval(x + 20, y + 30, 20, 20, "blue")
    panel.fill_oval(x + 60, y + 30, 20, 20, "blue")
    panel.draw_line(x + 30, y + 70, x + 70, y + 70, "red")

main()
```

17.

```
def main():
    panel = DrawingPanel(200, 200, "white")
    show_design(panel)

def show_design(panel):
    for i in range(1, 5):
        x = i * 20
        y = i * 20
        w = (10 - 2 * i) * 20
        h = (10 - 2 * i) * 20
        panel.draw_rect(x, y, w, h, "black")

main()
```

18.

```
def main():
    show_design(300, 100)

def show_design(width, height):
    panel = DrawingPanel(width, height, "white")
    for i in range(1, 5):
        x = i * width // 10
        y = i * height // 10
        w = (10 - 2 * i) * width // 10
        h = (10 - 2 * i) * height // 10
        panel.draw_rect(x, y, w, h, "black")

main()
```

19.

```
def main():
    panel = DrawingPanel(110, 110)
    for i in range(10):
        panel.draw_rect(5, 5 + 10 * i, 10 + 10 * i, 10)

main()
```

20.

```
# version 2
def main():
    panel = DrawingPanel(110, 110)
    for i in range(10):
        panel.draw_rect(5, 5 + 10 * i, 100 - 10 * i, 10)

main()

# version 3
def main():
    panel = DrawingPanel(110, 110)
    for i in range(10):
        panel.draw_rect(95 - 10*i, 5 + 10*i, 10 + 10*i, 10)

main()
```