**UNIT 1: Creative Computing for All**

**Lesson 1.1 – Algorithms**

**Activity 1.1.3 Fun with Flowers**

Consider how you could make your circle program simpler and easier to maintain. In what ways could it change? What if you wanted to allow a user to specify the number of circles that it draws?

| I could make the circle program simpler and easier to maintain by grouping things into methods/functions that are then called in the program, as then the number of times each function (e.g.: draw\_circle) is used could be decided by the user through input(). |
| --- |

STEP 6: Run the program in the code editor. Describe what you see.

| The turtle draws an equilateral triangle with side length 100 px. |
| --- |

You will need to reference the algorithms you write in the next few steps, up to and including step 10. To help you remember the algorithms, record the code you write.

STEP 7: Use a while loop to draw a square. Copy and paste your program in the box below.

| import turtle as trtl  painter = trtl.Turtle()  def draw\_polygon(sides, sidelength = 100):  i = 0  while (i < sides):  painter.forward(sidelength)  painter.right(360/sides)  i+=1  draw\_polygon(4)  wn = trtl.Screen()  wn.mainloop() |
| --- |

STEP 8: Use a while loop to draw an octagon. Copy and paste your program in the box below.

| import turtle as trtl  painter = trtl.Turtle()  # new starting location so the entire  # octagon is visible  painter.penup()  painter.goto(-50,150)  painter.pendown()  def draw\_polygon(sides, sidelength = 100):  i = 0  while (i < sides):  painter.forward(sidelength)  painter.right(360/sides)  i+=1  draw\_polygon(8)  wn = trtl.Screen()  wn.mainloop() |
| --- |

STEP 9: Write an algorithm that loops to draw a circle of circles. Iterate 18 times, turn 20 degrees for every 20 pixels forward, and stamp the turtle.

Copy and paste your program in the box below.

| import turtle as trtl  painter = trtl.Turtle()  painter.shape("circle")  painter.penup()  def draw\_polygon(sides, sidelength = 100):  i = 0  while (i < sides):  painter.forward(sidelength)  painter.right(360/sides)  i+=1  painter.stamp()  draw\_polygon(18, 20)  wn = trtl.Screen()  wn.mainloop() |
| --- |

STEP 11: Modify it so that the flower has larger petals and half as many as the original. Be sure to maintain even spacing between petals.

Copy and paste your program in the box below.

| import turtle as trtl  painter = trtl.Turtle()  painter.speed(0)  # stem  painter.color("green")  painter.pensize(15)  painter.goto(0, -150)  painter.setheading(90)  painter.forward(100)  # leaf  painter.setheading(270)  painter.circle(20, 120, 20)  painter.setheading(90)  painter.goto(0, -60)  # rest of stem  painter.forward(60)  painter.setheading(0)  # draw flower function  def draw\_flower(petals = 9, petal\_size = 4):  petals\_drawn = 0  painter.penup()  painter.goto(20,180)  painter.shape("circle")  painter.color("darkorchid")  painter.turtlesize(petal\_size)  while (petals\_drawn < petals):  painter.right(360/petals)  painter.forward(petal\_size\*15)  painter.stamp()  petals\_drawn += 1  draw\_flower()  wn = trtl.Screen()  wn.mainloop() |
| --- |

What is the purpose of the modulus?

| The modulo confines a value to be within a set range from 0 to the modulus, which is useful for iterating through a finitely-sized list without the need for additional logic to determine if the index is outside the bounds of the array. Additionally, the modulus is used in cryptography, forming the basis of Diffie-Hellman key exchange and RSA. |
| --- |

STEP 19: Create the program using modulus so that instead of drawing the tower in gray, it alternates colors every three floors.

Copy and paste your program in the box below.

| import turtle as trtl  painter = trtl.Turtle()  painter.speed(0)  painter.pensize(5)  # starting location of the tower  x = -150  y = -150  # height of tower and a counter for each floor  num\_floors = 63  floor = 0  # iterate  while floor < num\_floors:  # set placement and color of turtle  painter.penup()  painter.goto(x, y)  if (floor % 6 < 3):  painter.color("gray")  else:  painter.color("blue")  y = y + 5 # location of next floor  floor = floor + 1    #draw the floor  painter.pendown()  painter.forward(50)  wn = trtl.Screen()  wn.mainloop() |
| --- |

In your own words define a decidable problem.

| A decidable problem will always produce a predictable output |
| --- |

In your own words define an undecidable problem.

| An undecidable problem’s output can usually not be predicted/determined by an algorithm. |
| --- |

STEP 22: Explain why the program will terminate if you used -1 as your input number. How could you fix this program?

| If -1 is the input number, the program will terminate because -1 does not satisfy the condition i >= 0 in the while loop. This could be fixed by changing the while loop to while true: and removing the loop body. |
| --- |

CONCLUSION

Describe a scenario in which the mod operator could be useful outside of this course.

| The modulo operator is useful for iterating through a list/array without accidentally calling an index out of the range of the array by setting the modulo to the length of the array. |
| --- |

What are the benefits of reusing existing code?

| Reusing existing code saves time, as it is not necessary to rewrite the code, and makes the code more readable, as calling a named function is more readable than copying and pasting code. |
| --- |