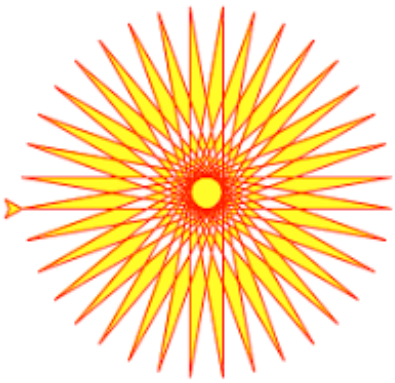
Geographic Art with Python's Turtle Package

Introduction

In this project, we'll be using a Python library called `turtle` to create some fun and creative geographic art! The `turtle` module allows us to control a virtual "turtle" that moves around the screen, drawing lines and shapes as it goes. We can use the turtle to draw anything from simple shapes like squares and circles to more complex designs like mountains, rivers, and even landscapes.



Think of the turtle as your own little artist that you get to program. By telling the turtle how far to move, which direction to turn, and when to change colors or start filling in shapes, you'll be able to create your own picture.

For this project, you'll be designing and drawing geographic features. You can get creative with elements like:

- **Mountains**: Draw a mountain range using triangles.
- **Rivers**: Use curves to create flowing water.
- **The Sun**: Draw a bright sun using circles.
- **Clouds**: Create fluffy clouds with multiple small circles.

Along the way, we'll be practicing core Python skills like using loops, functions, and variables, and you'll learn how to use `turtle` to bring your ideas to life. By the end of the project, you'll have your own piece of geographic art!

Let's get started and watch what your turtle can do!

Python review/introduction

Before we can start drawing with our turtle, we need to begin with an introduction to Python if you are new or a refresher if you are a returning member. You will first need to make a free account on replit (https://replit.com/), which is where we will do all of our coding.

Once you have done this, select "+ Create Repl" with a Python template, and start trying some of the code examples below! To run your code, press this button, and the result will show up in the console on the right:

Variables and data types

```
# Variables store values
width = 200
height = 300
color = "blue"
```

Loops

```
# A for loop example to draw a square
for i in range(4):
    print(i)
```

Functions

```
# Defining a function to draw a square
def draw_square():
    for i in range(4):
        turtle.forward(100)
        turtle.right(90)
```

Conditionals

```
x = 10
if x > 5:
    print("x is greater than 5")
```

Introduction to Turtle

Some example `turtle` code:

```python
import turtle

# Create a turtle object
pen = turtle.Turtle()

# Move the turtle forward by 100 units
pen.forward(100)

# Turn the turtle right by 90 degrees
pen.right(90)

# Move the turtle forward again
pen.forward(100)

# Keep the window open
turtle.done()
```

In the `turtle` module, you control a turtle that moves around the screen to draw shapes. The turtle always has a direction it's facing (like north, east, south, or west), and you can tell it how far to move and when to turn. Here are some basic movement commands to get you started:

1. **Moving Forward and Backward**:
   - `pen.forward(distance)`: Moves the turtle forward by the specified `distance` in the direction it's currently facing.
   - `pen.backward(distance)`: Moves the turtle backward by the specified `distance` without changing the direction it's facing.
2. **Turning Left and Right**:
   - `pen.right(angle)`: Turns the turtle to the right by a specified number of degrees (clockwise).
   - `pen.left(angle)`: Turns the turtle to the left by a specified number of degrees (counterclockwise).

How Turtle Uses Angles and Coordinates

Angles:

Turtle uses angles to control the direction of movement. The turtle starts by facing to the right (0 degrees). Turning right or left changes the angle the turtle faces:

- **Right Turn**: A positive angle (e.g., `pen.right(90)`) turns the turtle clockwise.
- **Left Turn**: A positive angle (e.g., `pen.left(90)`) turns the turtle counterclockwise.

The turtle moves based on its current direction:

- If the turtle faces 0 degrees, it moves to the right.
- After turning right by 90 degrees (to face 90 degrees), the turtle moves downward.
- A left turn of 45 degrees would turn it 45 degrees counterclockwise from its current direction.

Coordinates:

The turtle starts at the center of the screen at coordinates (0, 0), which is known as the origin. The screen works like a grid:

- Moving **right** increases the turtle's x-coordinate (positive x).
- Moving **left** decreases the x-coordinate (negative x).
- Moving **up** increases the y-coordinate (positive y).
- Moving **down** decreases the y-coordinate (negative y).

You can also move the turtle directly to specific coordinates using `pen.goto(x, y)`. This moves the turtle to the position `(x, y)` on the grid without worrying about angles.

```
pen.penup()  # Lift the pen so it doesn't draw while moving
pen.goto(100, 200)  # Move the turtle to the coordinates (100, 200)
pen.pendown()  # Put the pen down to start drawing
```

Drawing simple shapes using loops and angles

Drawing a square:

```
for _ in range(4):
    pen.forward(100)
    pen.right(90)
```

Drawing a circle:

```
pen.circle(50)  # Draw a circle with radius 50
```

Building the Geographic Art

1. Designing the Art

Brainstorm geographic elements to include:

- Mountains (using triangles)
- A river or ocean (using curves)
- The sun (using a circle)

2. Drawing Mountains (Triangles)

- Use loops to draw repetitive shapes like mountains.

```
def draw_triangle():
    for _ in range(3):
        pen.forward(100)
        pen.left(120)  # Turning to make a triangle

# Draw multiple triangles for a mountain range
for _ in range(5):
    draw_triangle()
    pen.forward(120)  # Move to the next spot for the next triangle
```

3. Drawing a River (Curved Lines)

- Create smooth curves using the `circle()` function.

```
pen.setheading(270)  # Point the turtle downwards
pen.circle(50, 180)  # Draw an arc (half of a circle) to create a wave
```

4. Drawing a Sun

- Use a filled circle to represent the sun.

```
pen.color("yellow")  # Set color to yellow
pen.begin_fill()  # Start filling the circle with color
pen.circle(50)  # Draw a circle for the sun
pen.end_fill()  # End filling
```

Finishing and Enhancing the Art

1. Adding More Details

- Try adding more features like clouds, trees, or other landscape elements using loops and functions.

```
def draw_cloud():
    for _ in range(3):
        pen.circle(20)  # Draw a small circle
        pen.right(60)  # Move to the next circle position

pen.penup()  # Lift pen to move without drawing
pen.goto(-150, 150)  # Move to a new location
pen.pendown()  # Put pen down to draw
draw_cloud()
```

2. Experimenting with Colors and Sizes

- Try changing the line color, thickness, and fill to make the artwork more colorful and dynamic.

```
pen.pensize(3)  # Increase line thickness
pen.color("blue")  # Change color to blue
pen.forward(100)
```

Now try to create a landscape of your own! Happy coding ☺