CPU

COMP 110

Final Exam Prep +

De-stressing with Recursive Art

Congratulations, you made it to LDOC!

This semester, we've covered a LOT:

- Objects
- Data types
- Expressions
- Functions
- Memory diagrams
- Boolean expressions
- Conditionals
- Scope

- User input
- While loops
- For loops
- Importing modules
- Lists
- Unit tests
- Dictionaries
- Object-oriented prog.

- Classes and methods
- Recursive structures
- Recursive functions
- Importing and reading files

This is no small feat!

Final Exam

Saturday, Dec 7 at 8–11am

in Hamilton 100 or Genome G100 or G200

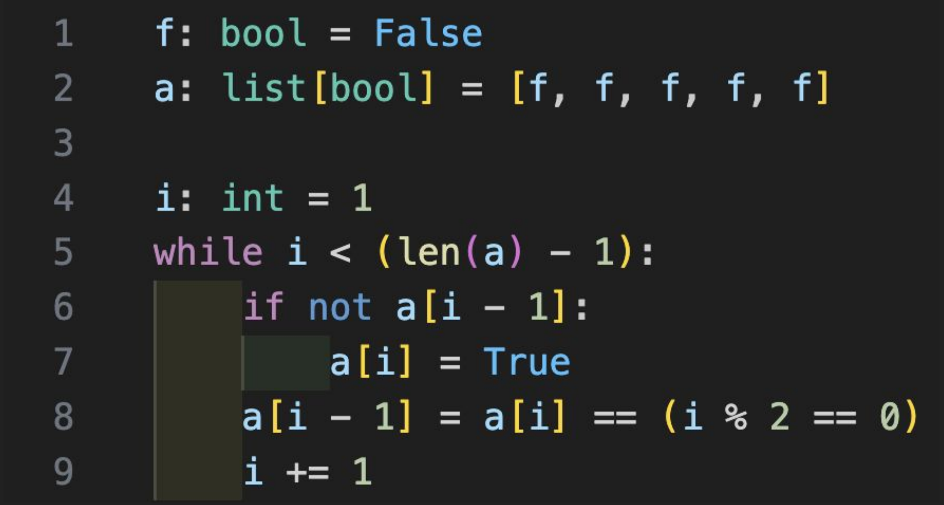
- Similar format to quizzes, but ~2x longer
  - Multiple choice, short answer, memory diagrams
  - Cumulative (any concept in the course is fair game)
    - Concepts from each quiz on the final exam
- Look out for an email about your room and seat assignment ***today***!

Makeup on Dec 8 at 12–3pm in Sitterson (SN) 014

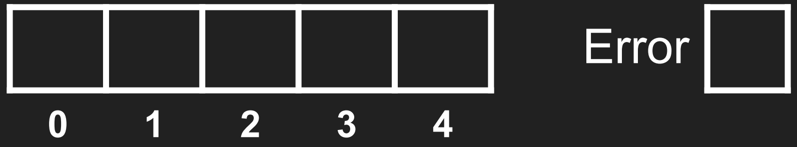
One last practice question

Please trace this code that modifies a boolean list, `a`. You will *completely shade in the squares of items whose value is assigned* `True`. If an error occurs during the evaluation of the loop, fill in the `Error` box and **stop evaluating**. If any item's value was assigned `True` prior to the error, keep its value shaded.

You can assume `a` is initialized with 8 `False` elements, as shown below.



```python
f: bool = False
a: list[bool] = [f, f, f, f, f]

i: int = 1
while i < (len(a) - 1):
    if not a[i - 1]:
        a[i] = True
    a[i - 1] = a[i] == (i % 2 == 0)
    i += 1
```

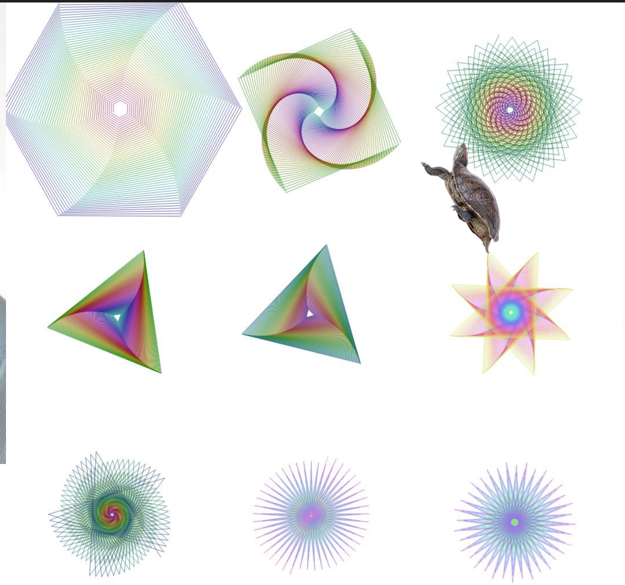


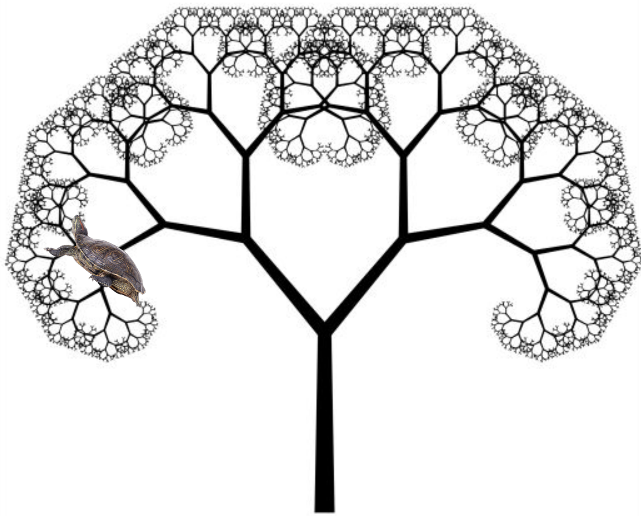
Time to de-stress: recursive art with `turtle` graphics!

Python library that lets us draw shapes on a virtual canvas

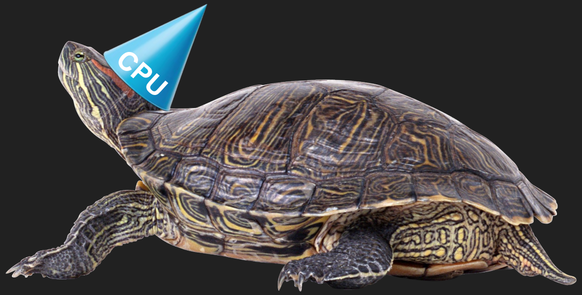
Imagine dipping a virtual turtle’s tail in (non-toxic) paint and directing the turtle around a virtual canvas!

lots of recursion!





Getting started with `turtle` graphics



Steps to get us started:

1. Create a new folder in your workspace called **‘art’**
2. Inside that folder, create a new file called **`turtle.py`**
3. In your browser, navigate to:

go.unc.edu/turtle

4. Select all the code on that page (ctrl+A or command+A) and copy it (ctrl+C or command+C)
5. Paste the code into your `turtle.py` file (ctrl+V or command+V). Then, save it!
6. Also in your **‘art’** folder, create a new file called **`flower.py`**

Code-along: Type the following into **flower.py:**

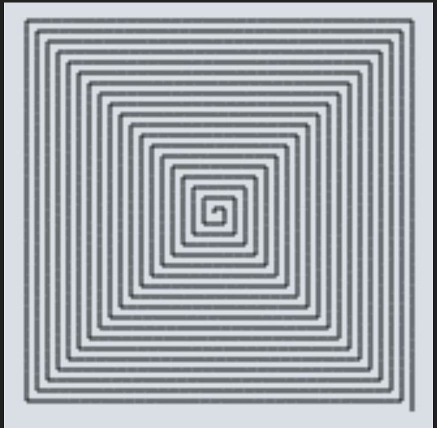
```
"""Turtle art!"""

from .turtle import Turtle
from math import pi

__template__ = "https://24ss2.comp110.com/static/turtle/"

def main() -> Turtle:
    t: Turtle = Turtle()
    t.setSpeed(0.25)

    t.forward(150)
    t.left(pi / 2.0)

    t.forward(140)
    t.left(pi / 2.0)

    return t
```

Before you run the code, what shape do you think the Turtle will draw?

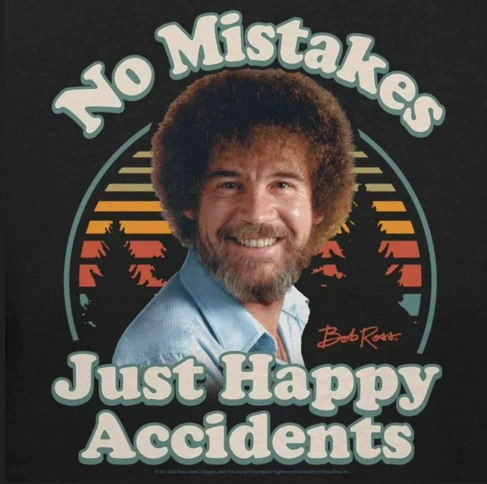
Your turn! Write a loop to draw a shape

- Write a while loop (don’t forget a counter variable!) that, inside of the loop:
  - Turns the **Turtle t** left by **pi/2.0**
  - Moves the **Turtle t** forward by 150, 148, 146, and so on, until 0 (not moving forward at all)
  - Update your variable so that it moves toward the loop’s terminating condition
- Once you’re finished, try running it. What shape do you see?

- A spiral!!
- Try increasing the speed to 10 or 100 once you have it working. Additionally, try playing with the angle the turtle is turning to develop different spirals.



Next: Drawing happy, little trees!

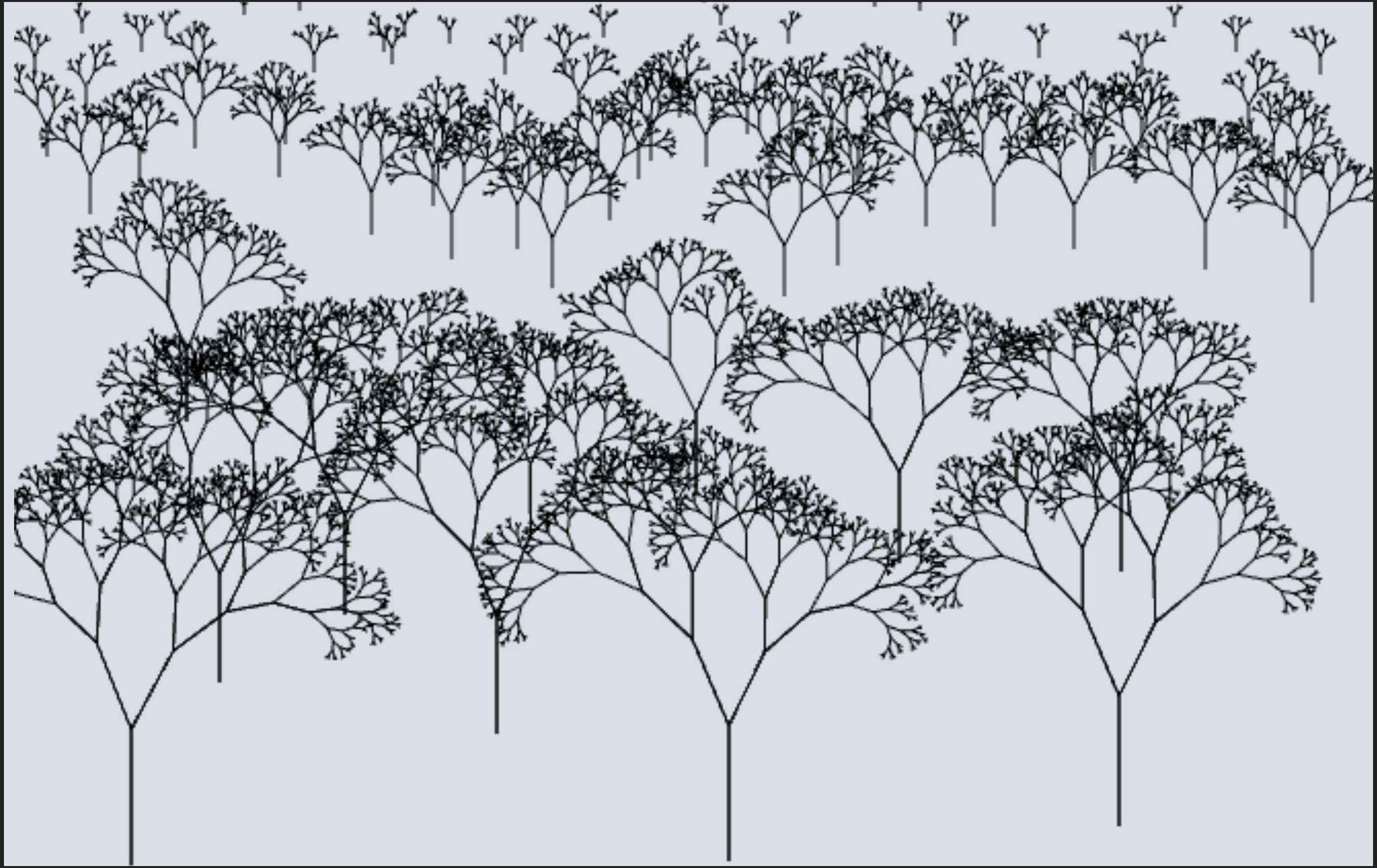


```python
"""Some happy, little trees!"""

from .turtle import Turtle
from math import pi
from random import random

__template__ = "https://24ss2.comp110.com/static/turtle/"

DEGREE: float = -pi / 180.0  # Constant


def main() -> None: ...


def click(x: float, y: float) -> Turtle:
    """Moves turtle to wherever we click on the canvas + draws line!"""
    t: Turtle = Turtle()
    t.moveTo(x, y)
    t.turnTo(90 * DEGREE)
    t.forward(100)
    return t
```

TODO: Complete the **branch** function to make the trees a little happier (add branches)!

```python
def branch(t: Turtle, length: float, angle: float) -> None:
    t.turnTo(angle)
    t.forward(length)
    # TODO: if length is greater than 10:
    # THEN call the branch function again (recursion!)
    #   The turtle argument of the call should be t, the same object
    #   The length argument of the call should be 75% of length's value
    #   The angle is angle + 30 * DEGREE
    t.turnTo(angle + pi)
    t.forward(length)
```

Seeing the forest for the trees



Thank you for a great semester!

❤️, your COMP110 Team

P.S. Please complete your Course Evaluation;

it helps us improve the course!