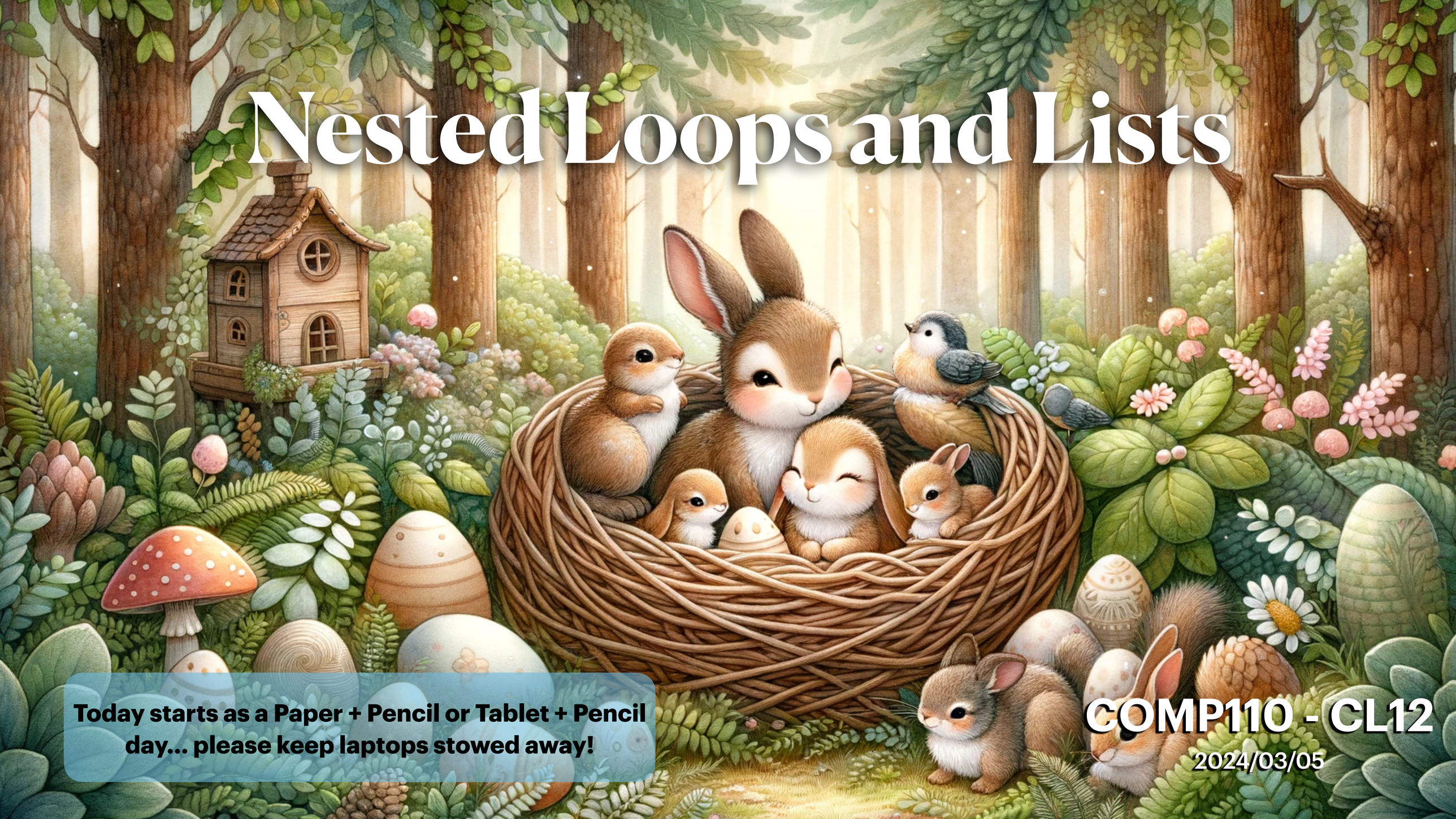
Nested Loops and Lists



Today starts as a Paper + Pencil or Tablet + Pencil day... please keep laptops stowed away!

COMP110 - CL12

2024/03/05

Warm-up: Diagram the Following Program

```
1    xs: list[int] = [10, 20, 30]
2    i: int = 2
3    xs[i] = xs[i - 1]  # A
4    print(xs)
5    i = i - 1
6    xs[i - 1] = xs[i]  # B
7    print(xs)
```

Follow-on Questions:

1. Describe line with comment A in English
2. Describe line with comment B in English

Relative Reassignment Operators

Trace a Memory Diagram

```
1   def triangle(n: int) -> None:
2       i: int = 1
3       line: str
4       while i <= n:
5           line = ""
6           while len(line) < i:
7               line += "*"
8           print(line)
9           i += 1
10
11
12  triangle(2)
```

Insertion Sort Algorithm Intuition

Goal: Move items in the list back to their correctly sorted position one-by-one.

1. Start with Current Index i at index 1
2. Hold current index's value aside in Current Value x
3. Compare Current Value with the value before it, if exists
    1. Current value less than previous? Copy/"shift" previous value forward one index. Repeat until no more previous values or previous value is at most current value.
    2. Assign / "**Insert**" Current Value to the last index that was shifted forward. This is its correctly sorted position up to the Current Index!
    3. Add One to Current Index, Go to Step 2
4. Once current index >= len(list), done!

Current Value **`i`** is:

Current Value **`x`** is:

0	1	2	3	4
40	10	30	20	50

Try it out!

Goal: Move items in the list back to their correctly sorted position one-by-one.

1. Start with Current Index i at index 1
2. Hold current index's value aside in Current Value x
3. Compare Current Value with the value before it, if exists
    1. Current value less than previous? Copy/"shift" previous value forward one index. Repeat until no more previous values or previous value is at most current value.
    2. Assign / "**Insert**" Current Value to the last index that was shifted forward. This is its correctly sorted position up to the Current Index!
    3. Add One to Current Index, Go to Step 2
4. Once current index >= len(list), done!

Current Value **`i`** is:

Current Value **`x`** is:

0	1	2	3	4
50	40	20	30	10

Tracing Insertion Sort

```python
def sort(xs: list[int]) -> None:
    """Sort with the insertion sort algo."""
    N: int = len(xs)  # Number of items
    i: int = 1  # "current index" starts at 2nd index
    x: int  # The "current value"
    si: int  # The search "shift index"
    while i < N:
        x = xs[i]  # store current value
        si = i
        while si > 0 and x < xs[si - 1]:
            xs[si] = xs[si - 1]  # Shift item forward
            si -= 1
        xs[si] = x
        i += 1
        print(xs)


values: list[int] = [40, 10, 30]
sort(values)
print(values)
```

Diagramming Nested Lists

```
row_0: list[int] = [1, 2]
row_1: list[int] = [2, 4]
row_2: list[int] = [3, 6]

table: list[list[int]] = [row_0, row_1, row_2]


mystery: _________________ = table[0]

print(row_2[1])
print(table[2])
print(table[2][1])
```

Nested List Notes

Diagramming Nested Lists

```
1   def mul_table(height: int, width: int) -> list[list[int]]:
2       rows: list[list[int]] = []
3       row_i: int = 1
4       while row_i <= height:
5           col_i: int = 1
6           row: list[int] = []
7           while col_i <= width:
8               row.append(row_i * col_i)
9               col_i += 1
10          rows.append(row)
11          row_i += 1
12      return rows
13
14
15  print(mul_table(3, 2))
```

```python
def mul_table(height: int, width: int) -> list[list[int]]:
    rows: list[list[int]] = []
    row_i: int = 1
    while row_i <= height:
        col_i: int = 1
        row: list[int] = []
        while col_i <= width:
            row.append(row_i * col_i)
            col_i += 1
        rows.append(row)
        row_i += 1
    return rows


print(mul_table(3, 2))
```