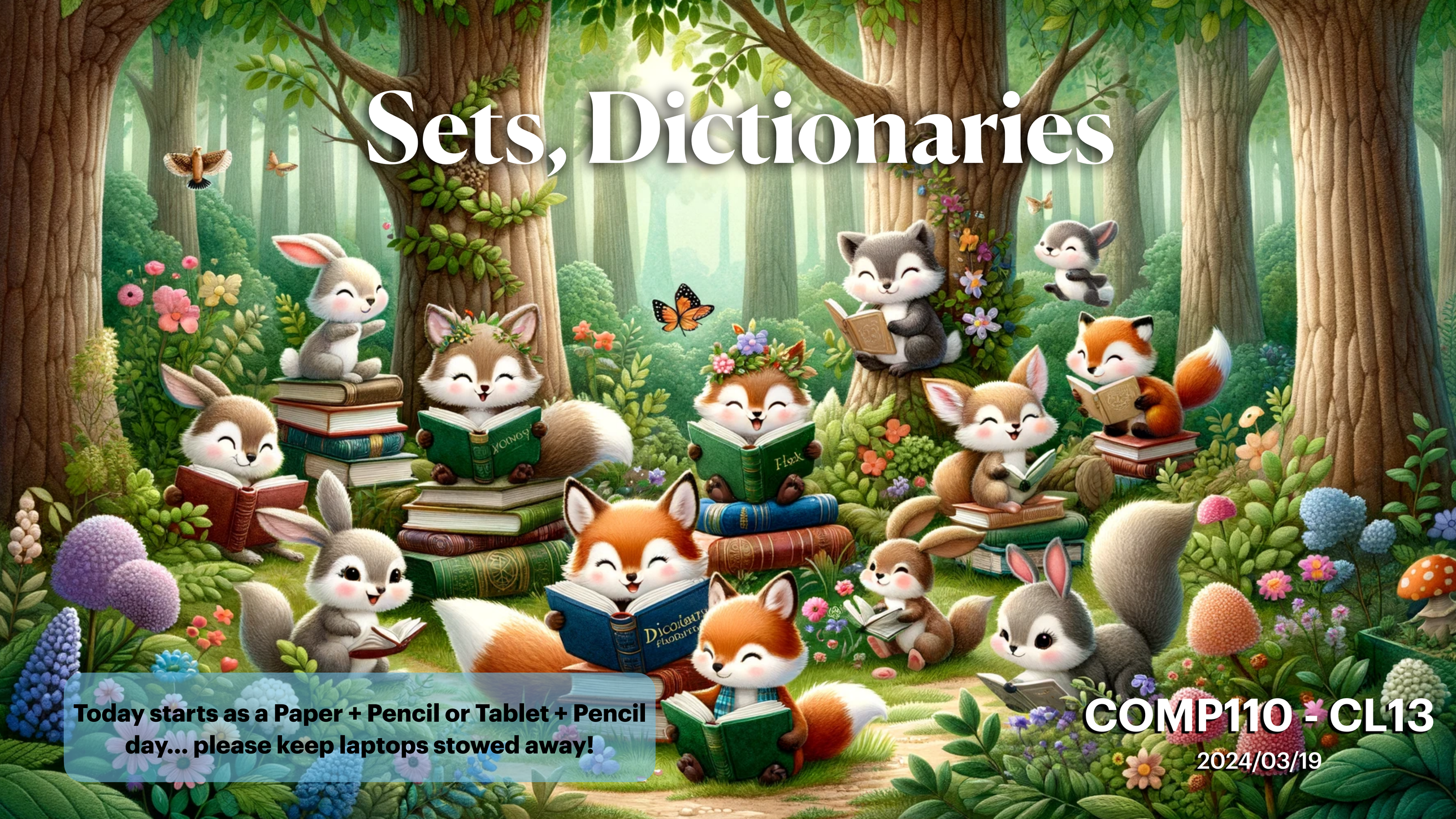
Sets, Dictionaries



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

COMP110 - CL13

2024/03/19

Warm-up: Diagram the Following Program

```
1   def intersection(a: list[str], b: list[str]) -> list[str]:
2       result: list[str] = []
3
4       idx_a: int = 0
5       while idx_a < len(a):
6           idx_b: int = 0
7           found: bool = False
8           while not found and idx_b < len(b):
9               if a[idx_a] == b[idx_b]:
10                  found = True
11                  result.append(a[idx_a])
12              idx_b += 1
13          idx_a += 1
14
15      return result
16
17
18  foo: list[str] = ["a", "b"]
19  bar: list[str] = ["c", "b"]
20  print(intersection(foo, bar))
```

Follow-on Questions:

1. How many times was line 9 evaluated?
2. If neither a or b contained any common elements, how many times would line 9 evaluate generally? Express in terms of len(a) and len(b).

```python
1  def intersection(a: list[str], b: list[str]) -> list[str]:
2      result: list[str] = []
3
4      idx_a: int = 0
5      while idx_a < len(a):
6          idx_b: int = 0
7          found: bool = False
8          while not found and idx_b < len(b):
9              if a[idx_a] == b[idx_b]:
10                 found = True
11                 result.append(a[idx_a])
12             idx_b += 1
13         idx_a += 1
14
15     return result
16
17
18 foo: list[str] = ["a", "b"]
19 bar: list[str] = ["c", "b"]
20 print(intersection(foo, bar))
```

Follow-on Questions:

1. How many times was line 9 evaluated?
2. If neither a or b contained any common elements, how many times would line 9 evaluate generally? Express in terms of len(a) and len(b).

A word of caution about 'in' with lists...

When in doubt, avoid it!

```python
def intersection(a: list[str], b: list[str]) -> list[str]:
    result: list[str] = []

    idx_a: int = 0
    while idx_a < len(a):
        if a[idx_a] in b:
            result.append(a[idx_a])
        idx_a += 1

    return result


foo: list[str] = ["a", "b"]
bar: list[str] = ["c", "b"]
print(intersection(foo, bar))
```

Each time you use `in` with a list, it is linearly searching each element to check for existence.

Sets

Worst...

```python
def intersection(a: list[str], b: list[str]) -> list[str]:
    result: list[str] = []

    idx_a: int = 0
    while idx_a < len(a):
        if a[idx_a] in b:
            result.append(a[idx_a])
        idx_a += 1

    return result
```

Orders of magnitude better...

```python
def intersection(a: list[str], b: set[str]) -> set[str]:
    result: set[str] = set()

    idx_a: int = 0
    while idx_a < len(a):
        if a[idx_a] in b:
            result.add(a[idx_a])
        idx_a += 1

    return result
```

Suppose **a** and **b** each had 1,000,000 elements, the worst case difference here is approximately **1,000,000** operations versus 1,000,000**2 or ***1,000,000,000,000*** operations.

Dictionaries

Homework

- EX04 - List Utils - Due Today at 11:59pm
- EX05 - Dictionary Utils - Due Monday 3/25 at 11:59pm
- RD00 - Ethical Algorithms - Due Friday 3/29 at 11:59pm