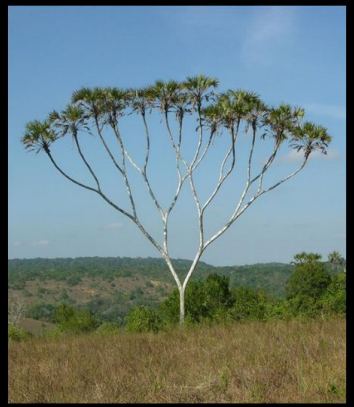
Intro to Recursive Structures & Processes

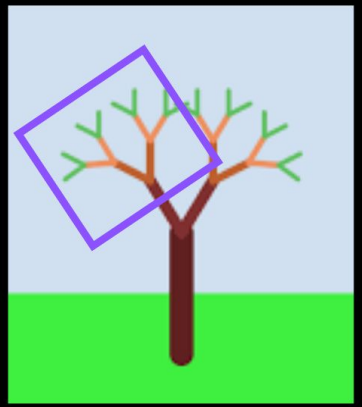
Note:
Today's LS (Recursive Structures) will be due *tomorrow* at 11:59pm

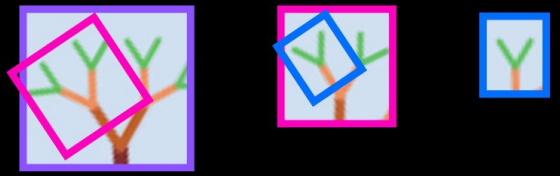
Recursion: defining an operation/object in terms of itself

A real-world phenomenon! Examples:

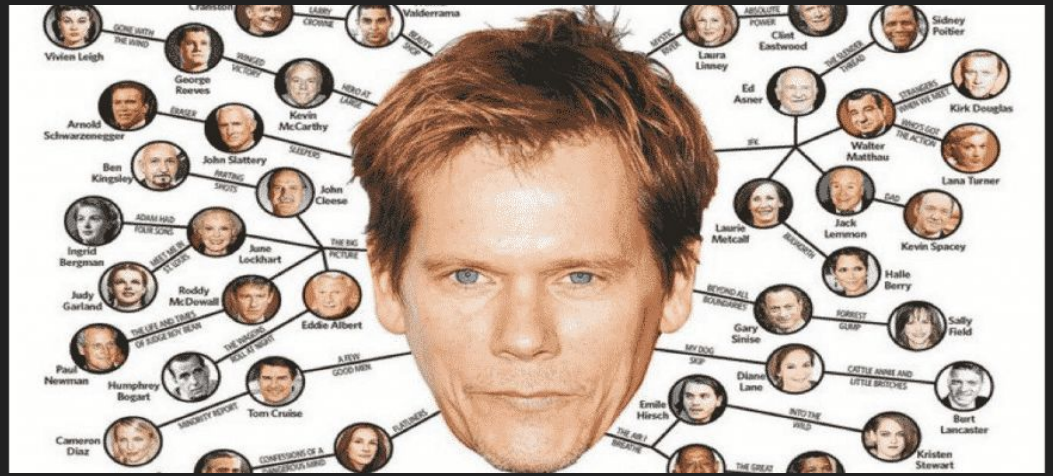
- **You** have **parents**, who have **parents**, who have **parents**, who have **parents**, who…
  … were the **first humans**
- A **tree** has **branches**, which have **branches**, which have **branches**, which…
  … have **leaves**





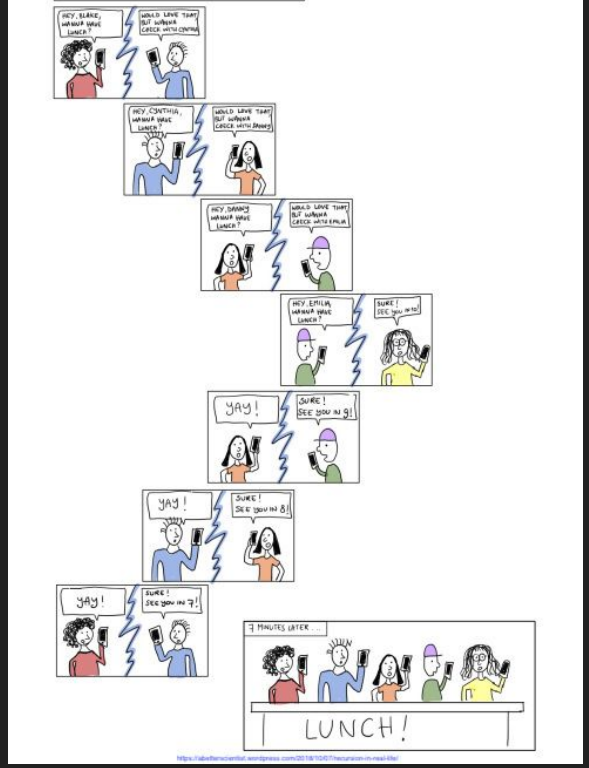


Different recursive structures for different purposes



Six degrees of Kevin Bacon

graph/network



Coordinating plans before 3-way calls were possible

linked list

Memory diagram

```python
from __future__ import annotations  # Ignore for now!

class Node:
    value: int
    next: Node | None

    def __init__(self, val: int, next: Node | None):
        self.value = val
        self.next = next

# Note: There are no errors!
two: Node = Node(2, None)
one: Node = Node(1, two)
# We'll extend this diagram shortly, leave room
```

Let's write a recursive function called `sum`!

```python
from __future__ import annotations  # Ignore for now!

class Node:
    value: int
    next: Node | None

    def __init__(self, val: int, next: Node | None):
        self.value = val
        self.next = next

# Note: There are no errors!
two: Node = Node(2, None)
one: Node = Node(1, two)
# We'll extend this diagram shortly, leave room
```

Write a function called `sum` that adds up the `value`s of all `Node`s in the linked list.

Diagramming the `sum` function call