31.03.21 07:08

CSCI 120
Introduction to CompSci and Programming I
Lec 6: Class II & Tree I

Jetic Gū

Overview

- Focus: Python Programming
- Architecture: von Neumann
- Core Ideas:
    1. Binary Tree using Class

P0
Review

Python Class

- Custom data structures
- Class have methods and attributes, shared between its objects
- Native classes: e.g. `list`, `dict`, `str`

Review

P1
Tree

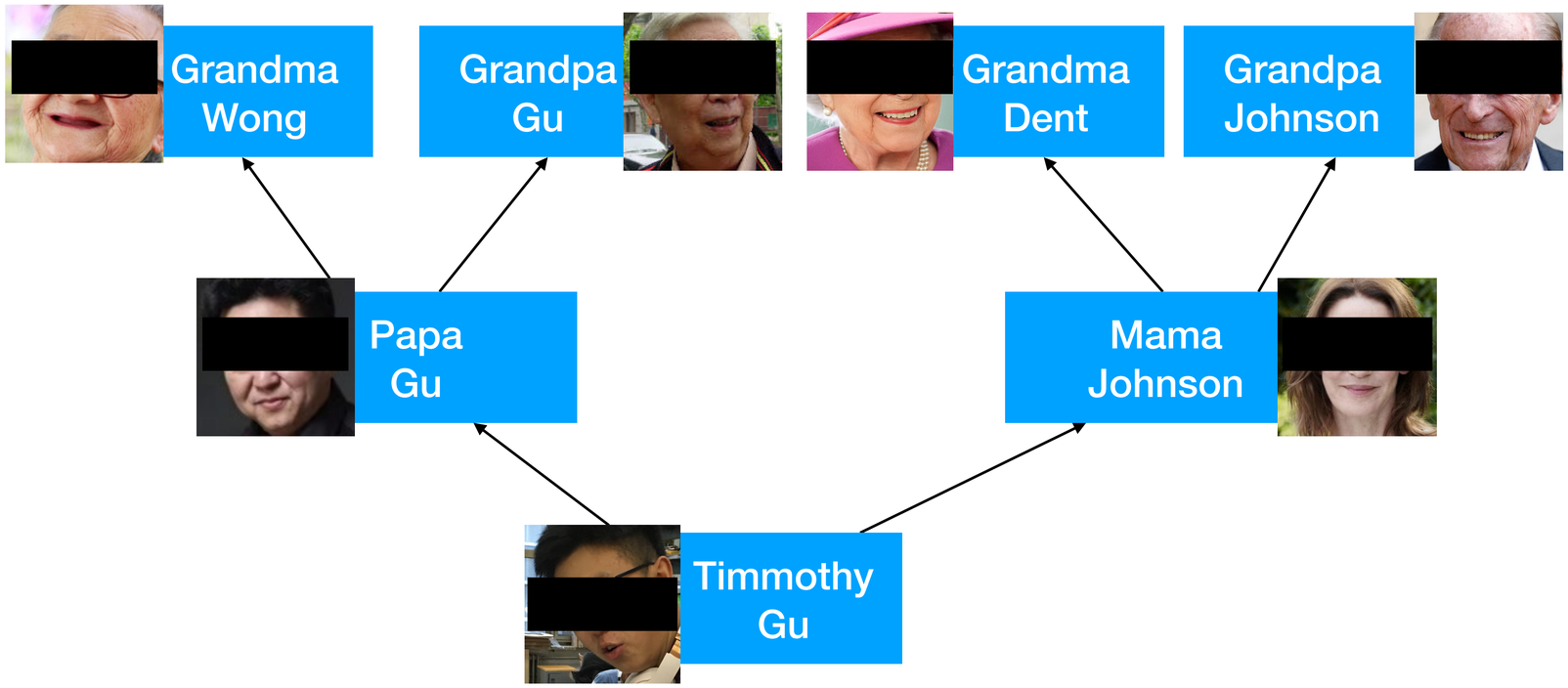
Binary Tree

Summary

P1
Tree

A Family Tree Problem

- Let’s say you want to store information regarding some relatives

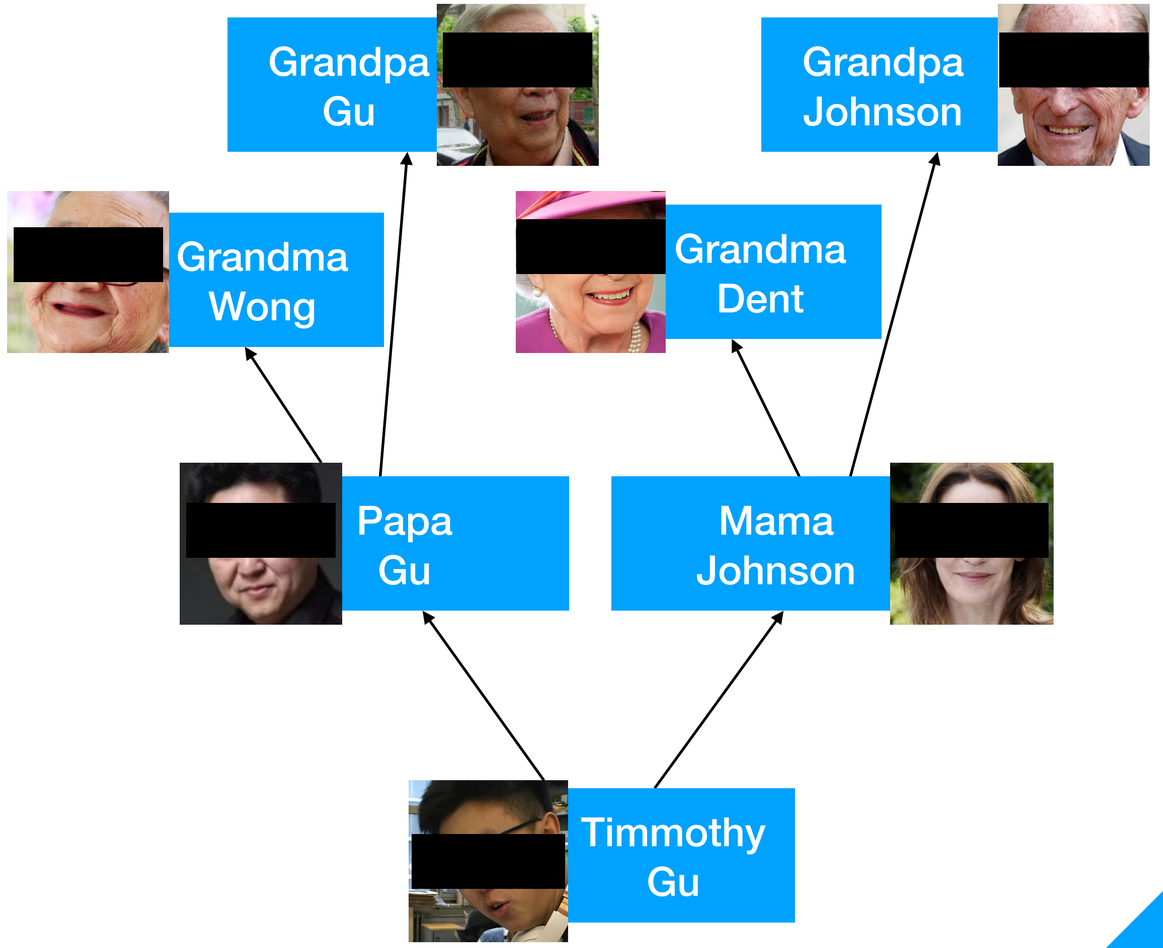


Example

P1
Tree

A Family Tree Problem

```
class Family:
  def __init__(self):
      self.name = "NoName"
      self.mama = None
      self.papa = None
```

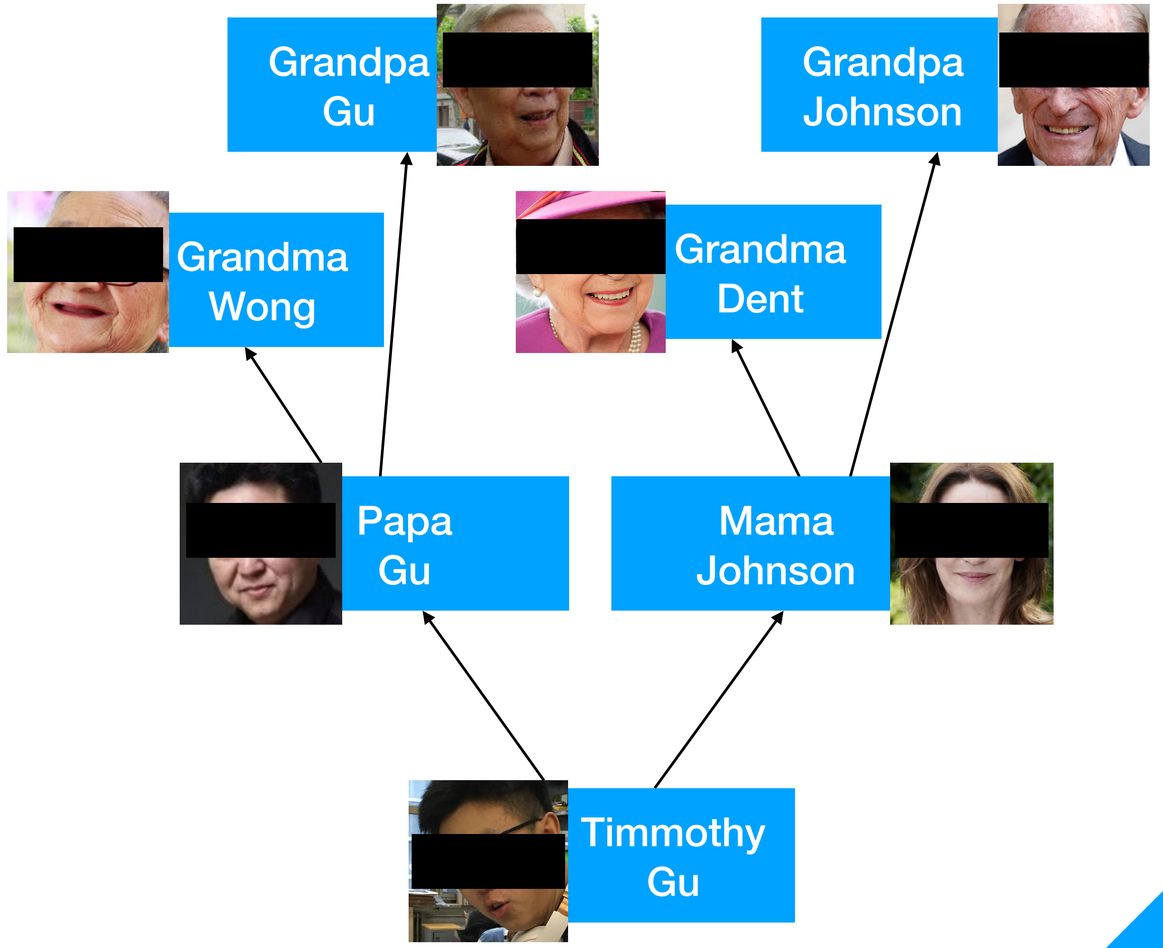


Example

P1
Tree

A Family Tree Problem

```
class Family:
  def __init__(
      self,
      name="NoName"
      mama=None,
      papa=None):
    self.name = name
    self.mama = mama
    self.papa = papa
```

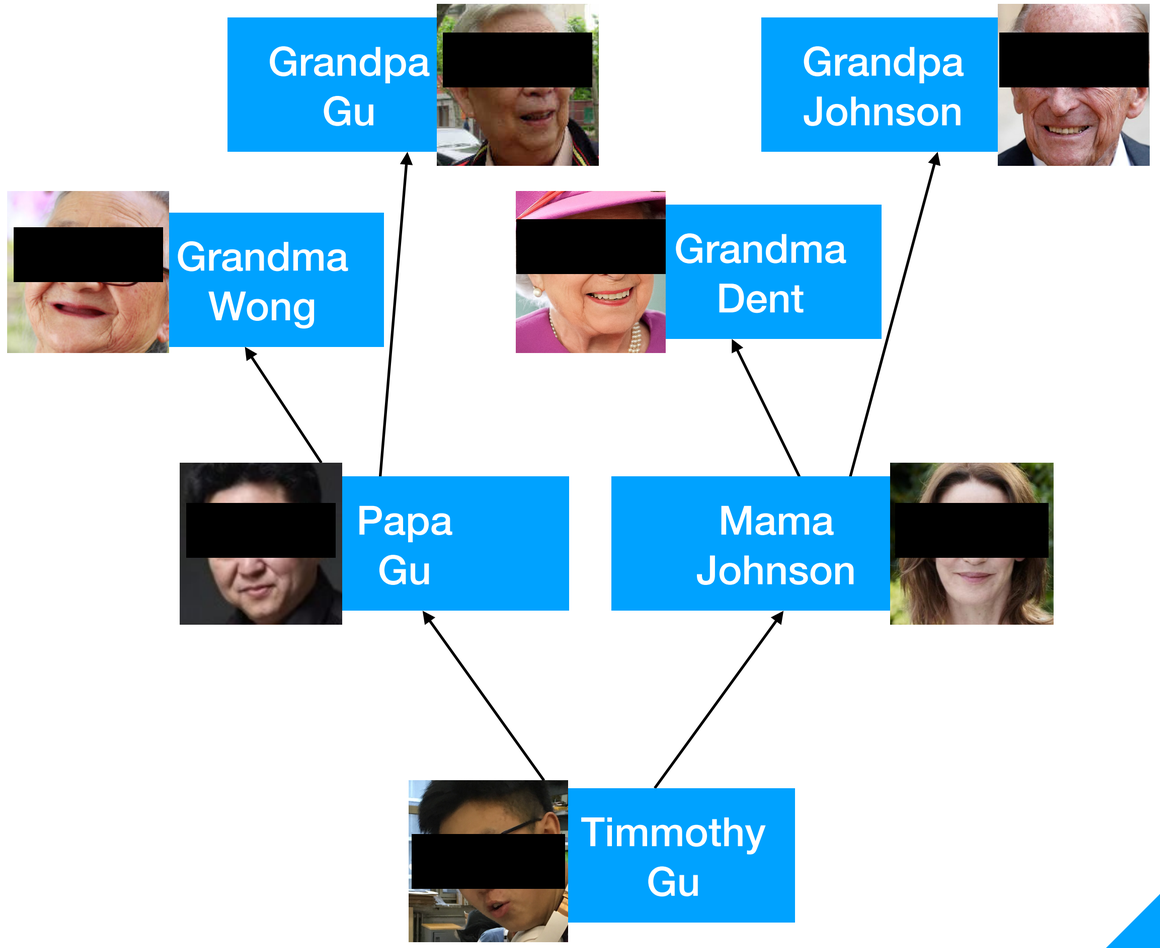


Example

A Family Tree Problem

```
class Family:
  def __init__(...):
    ...
  def intro(self):
    print("my name is...")
    print("my parents
    are...")
```

- Can we print the family tree in a better way?

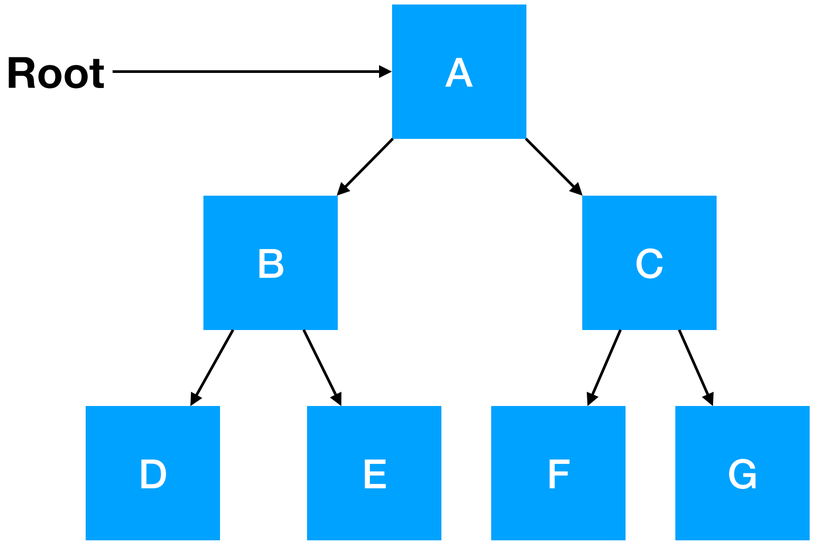


Example

Definition

- A Binary Tree (BT) is a tree in which each node contains at most two child nodes(left child and right child).

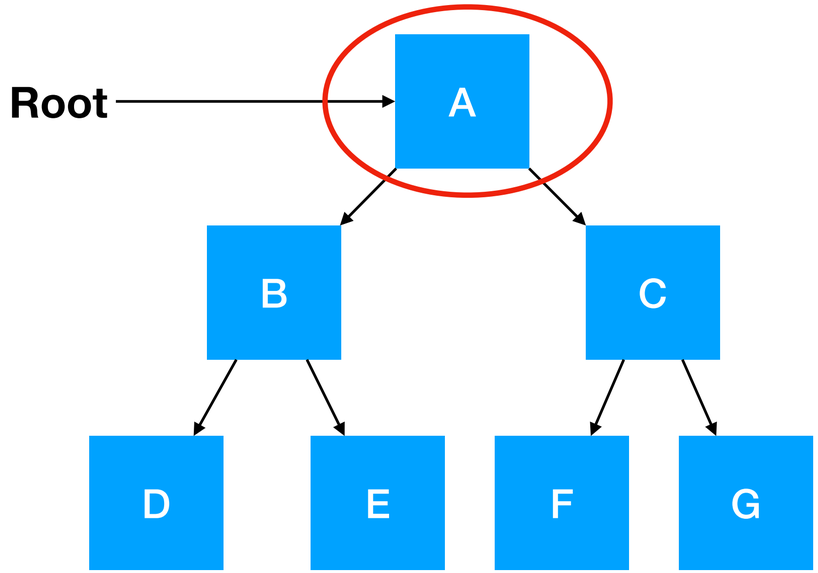
- Binary Trees are the simplest trees in Graph Theories, and has a lot of applications



Concept

Definition

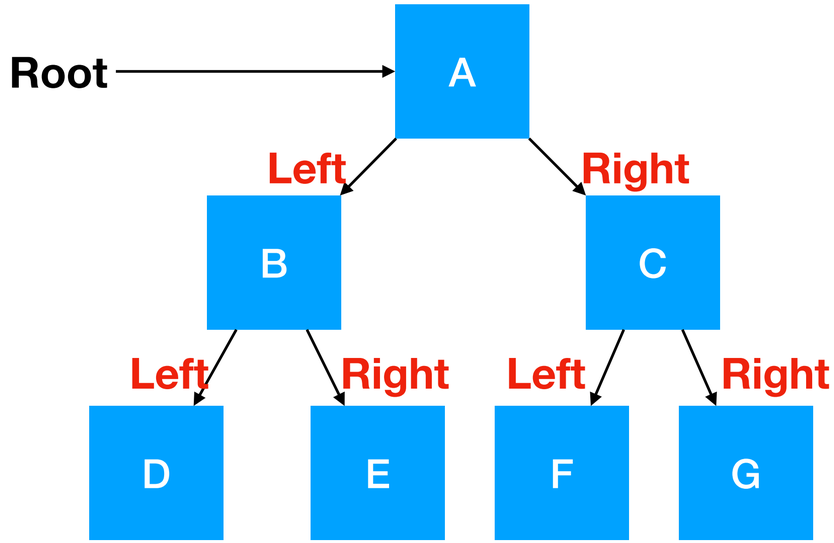
- A tree has a root node, this is where you access information on the tree



Concept

Definition

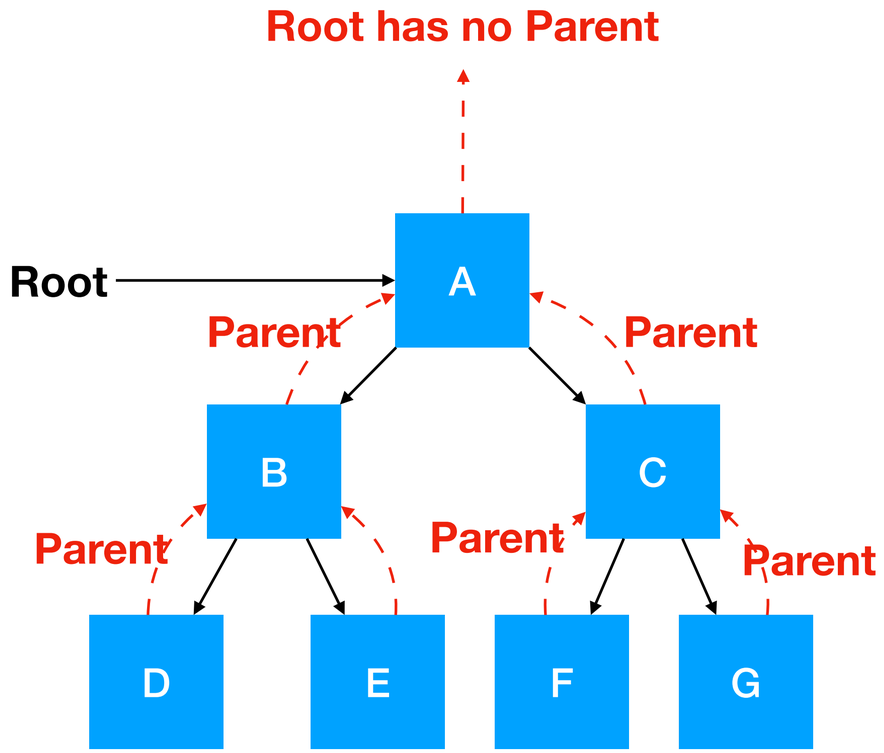
- In a binary tree, each node can have two "**children**": **left** and **right**
  - Node A: **Left** B, **Right** C
    A is B and C's **parent**
  - Node B: **Left** D, **Right** E
    B is D and E's **parent**
  - Node C: **Left** F, **Right** G
    C is F and G's **parent**
- Any node except for the root can **only have 1 parent**



Concept

Definition

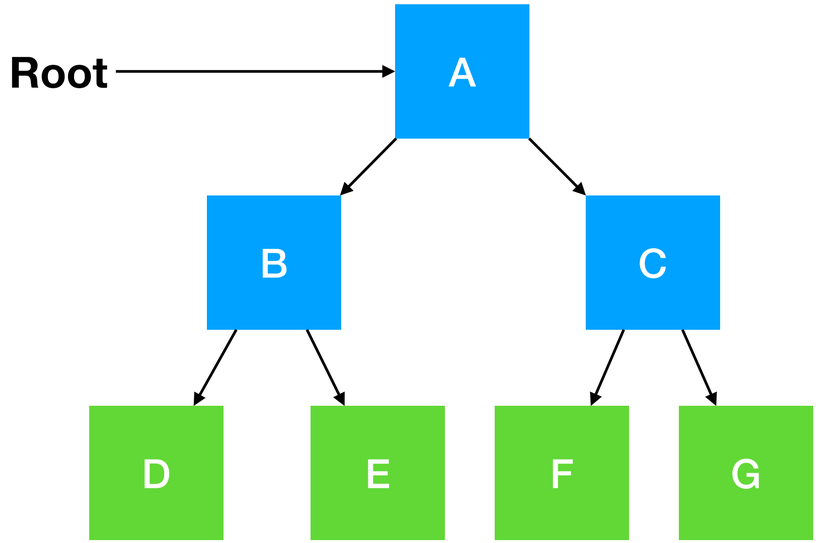
- In a binary tree, each node can have two "**children**": **left** and **right**
  - Node A: **Left** B, **Right** C
    A is B and C's **parent**
  - Node B: **Left** D, **Right** E
    B is D and E's **parent**
  - Node C: **Left** F, **Right** G
    C is F and G's **parent**
- Any node except for the root can **only have 1 parent**



Concept

Definition

- There are also two types of nodes
  - **Leaf node**: a node without children
  - **Internal node**: a node with children
- **Complete Binary Tree**
  - All internal nodes have 2 children

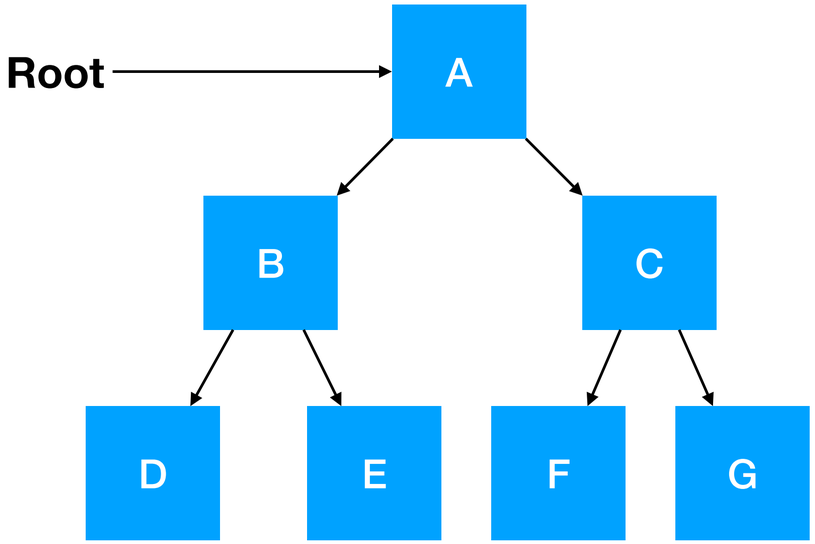


Concept

P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the Root, then Left, then Right



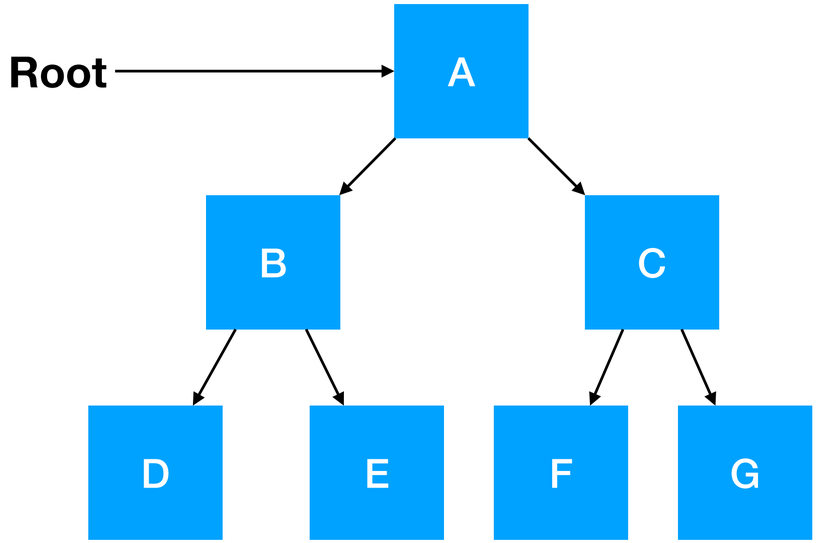
Concept

P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the **Root**, then **Left** (subtree), then **Right** (subtree)

Visiting Order:

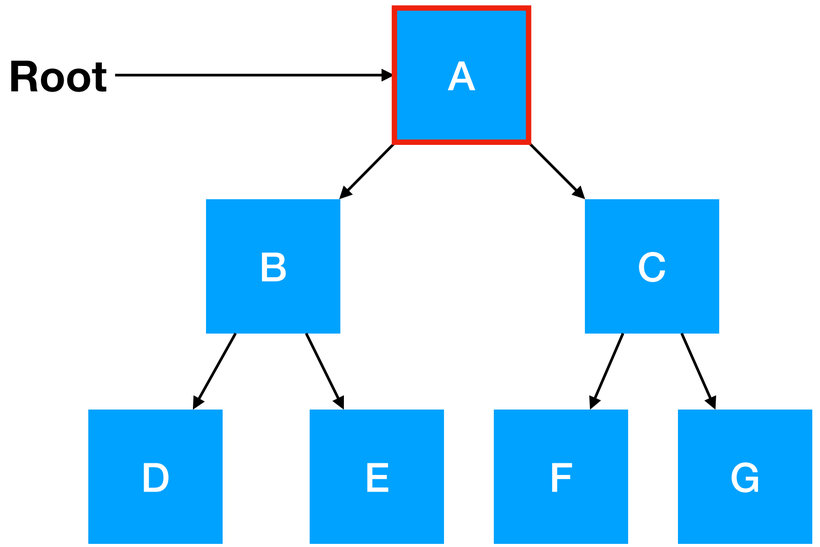


Concept

P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the **Root**, then **Left** (subtree), then **Right** (subtree)



Visiting Order:

A

Concept

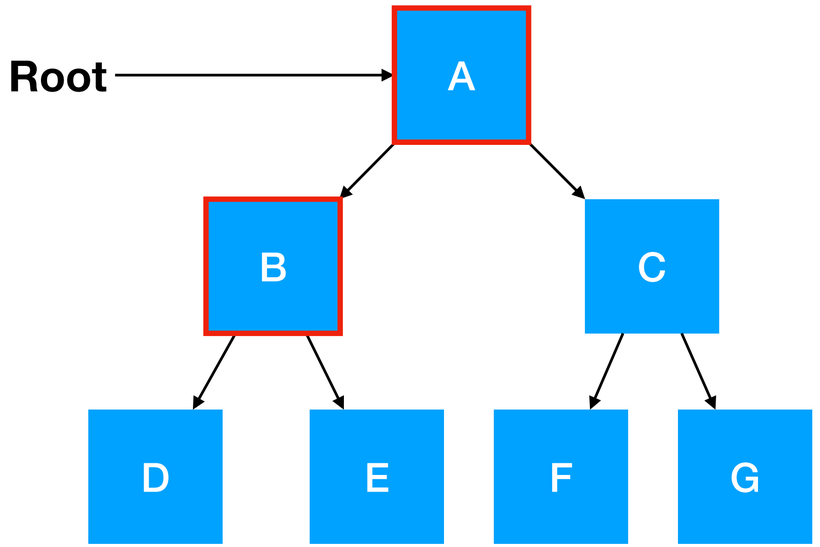
P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the **Root**, then **Left** (subtree), then **Right** (subtree)

Visiting Order:

A → **B**



Concept

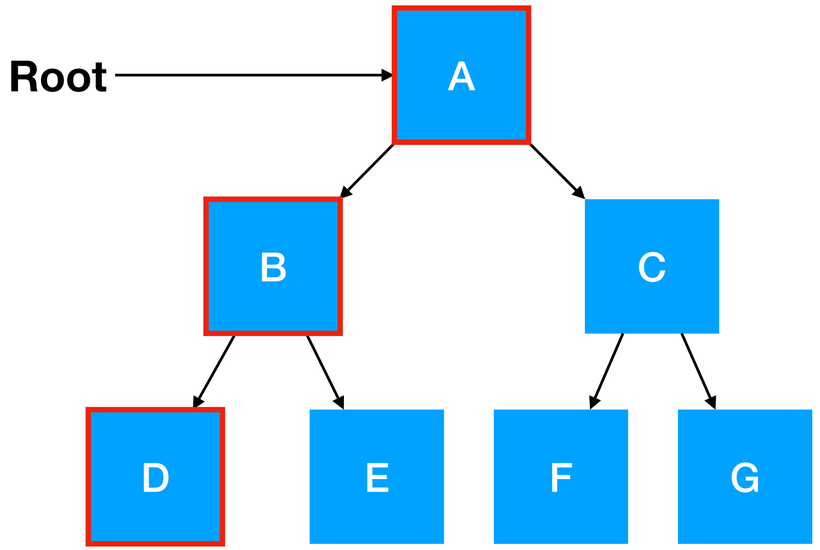
P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the **Root**, then **Left** (subtree), then **Right** (subtree)

Visiting Order:

A → B → D



Concept

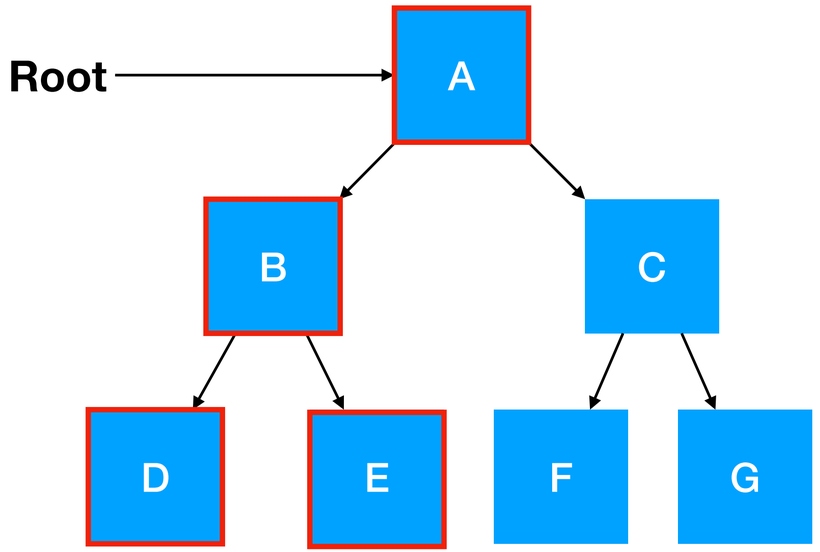
P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the **Root**, then **Left** (subtree), then **Right** (subtree)

Visiting Order:

A → **B** → **D** → **E**



Concept

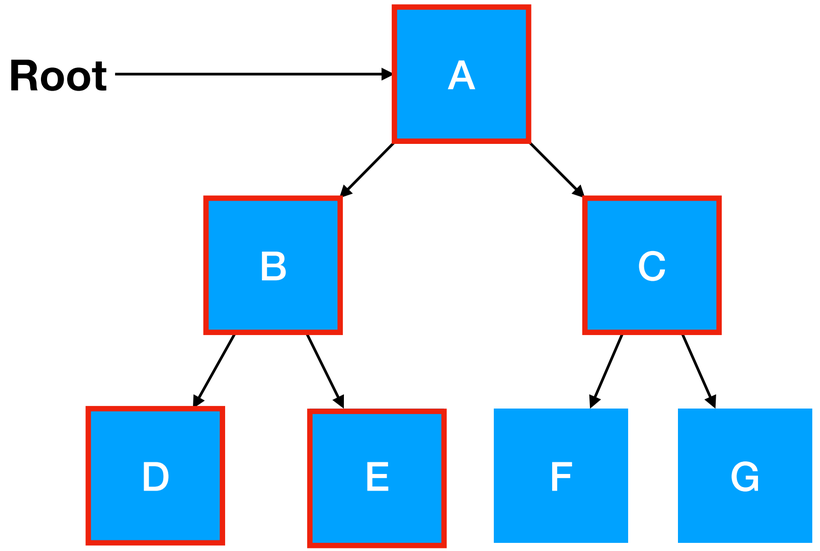
P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the **Root**, then **Left** (subtree), then **Right** (subtree)

Visiting Order:

A → B → D → E → C



Concept

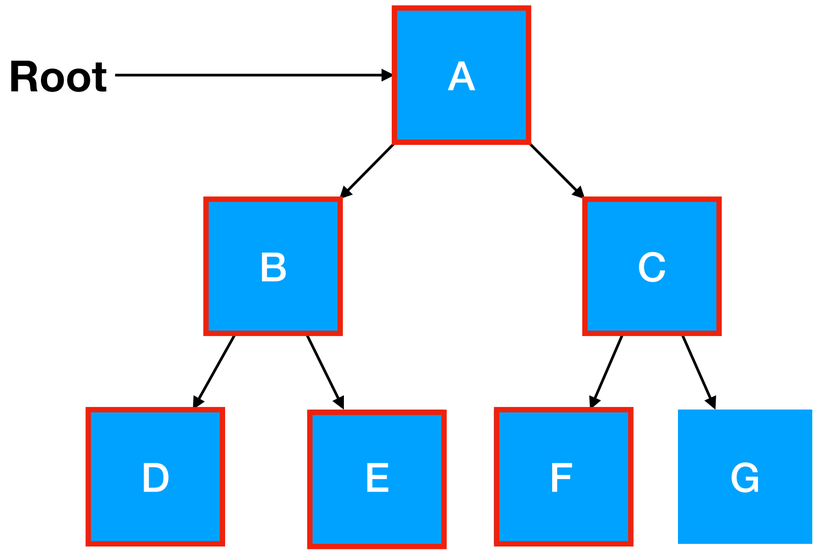
P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the **Root**, then **Left** (subtree), then **Right** (subtree)

Visiting Order:

A → B → D → E → C → F



Concept

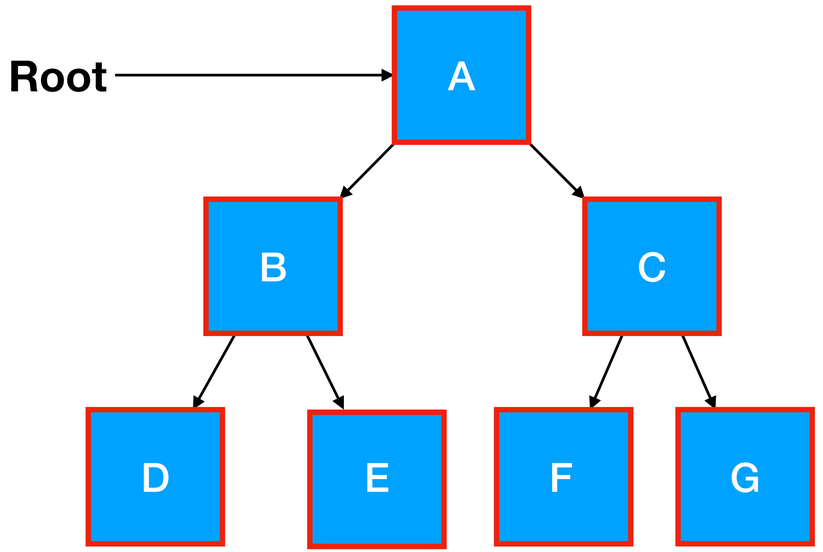
P1
Tree

Tree Traversal

- There's a few types of binary tree traversals, first let's do **Preorder Traversal**
- First visit the **Root**, then **Left** (subtree), then **Right** (subtree)

Visiting Order:

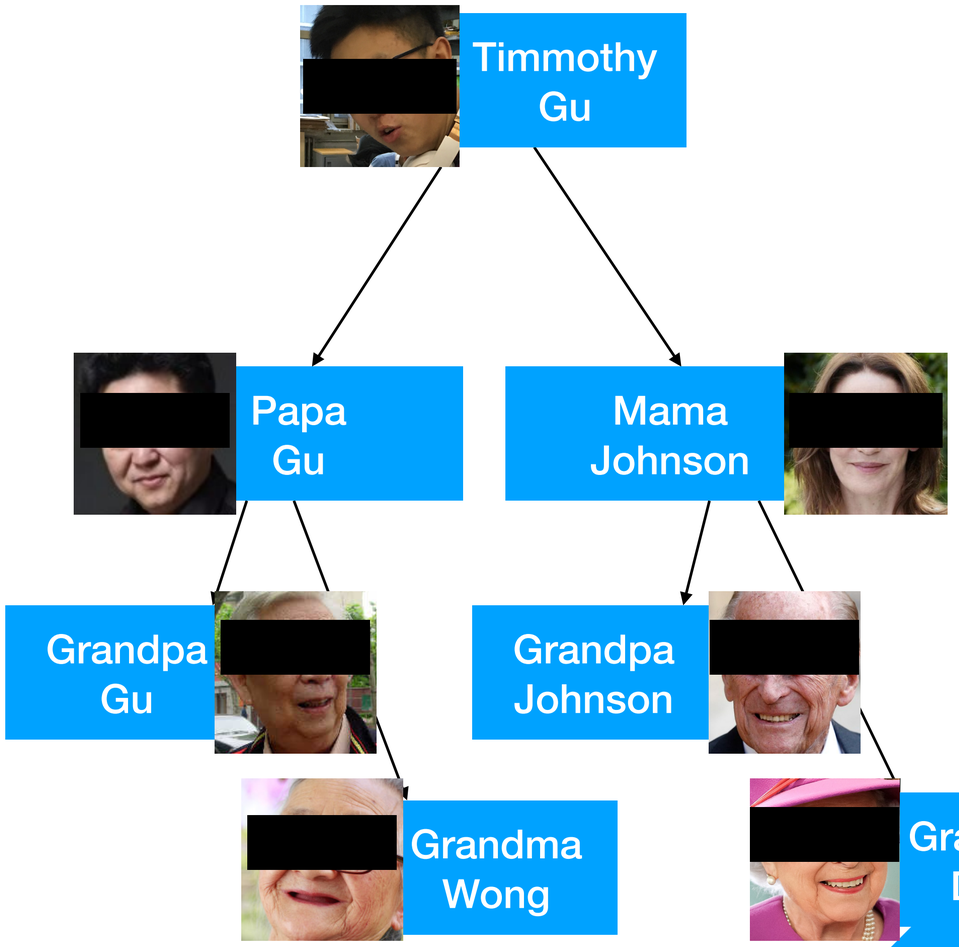
A → B → D → E → C → F → G



Concept

A Family Tree Problem

- Visit all family members in **preorder**
-
```
class Family:
    def __init__(...):
        ...
    def traverse(self):
        print(self.name)
        if self.papa is not None:
            self.papa.traverse()
        if self.mama is not None:
            self.mama.traverse()
```



P1
Tree

A Family Tree Problem

- What is the output?
- Can you make it prettier?