CS6.302 - Software System Development Monsoon 2025

Lab - 12: Python Session 3

BST vs AVL

dict-based vs class-based

```
node = {'value': 10,
'left': None,
'right': None,
'height': 1}

node['value']

node['left']

get_height(node)

root = insert_node(root, 10)
```

```
@dataclass
class Node:
key: int
left: Optional['Node'] = None
right: Optional['Node'] = None
height: int = 1

node.key

node.left

my_tree.get_height()

my_tree.insert(10)
```

dict-based vs class-based

dict-based

```python
def insert_node(root, value):
    if root is None:
        return {'value': value, ...} # create_node

    if value < root['value']:
        root['left'] = insert_node(root['left'], value)
    else:
        root['right'] = insert_node(root['right'], value)

    # ... (update height, balance) ...
    return root # Return the new root
```

class-based

```python
# Inside your BST or AVLTree class
def _insert(self, root_node, key):
    if root_node is None:
        return Node(key=key) # from @dataclass

    if key < root_node.key:
        root_node.left = self._insert(root_node.left, key)
    else:
        root_node.right = self._insert(root_node.right, key)

    # ... (update height, balance) ...
    return root_node # Return the new root
```

Right Rotate: Left-Left case

```python
# In your AVLTree class
def _right_rotate(self, y: Node) -> Node:

    # 1. Identify the new root (x) and the node that gets moved (T2)
    x = y.left
    T2 = x.right

    # 2. Perform the rotation (swap the pointers)
    x.right = y   # x becomes the new parent of y
    y.left = T2   # y's old left child is gone, so it adopts T2

    # 3. Update heights. IMPORTANT: Update the child (y) FIRST!
    y.height = 1 + max(self._get_node_height(y.left),
                       self._get_node_height(y.right))

    x.height = 1 + max(self._get_node_height(x.left),
                       self._get_node_height(x.right))

    # 4. Return the new root of this subtree
    return x
```