Practical 1

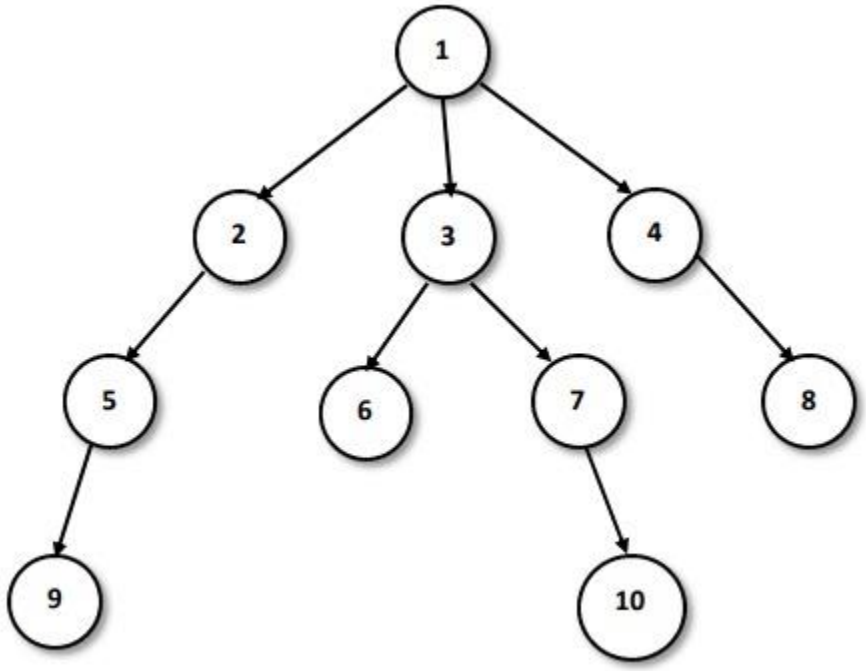
1a) BFS

Aim :Implement BFS treverse on given tree:

Theory :

BFS : Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search when a dead end occurs in any iteration.

Tree :

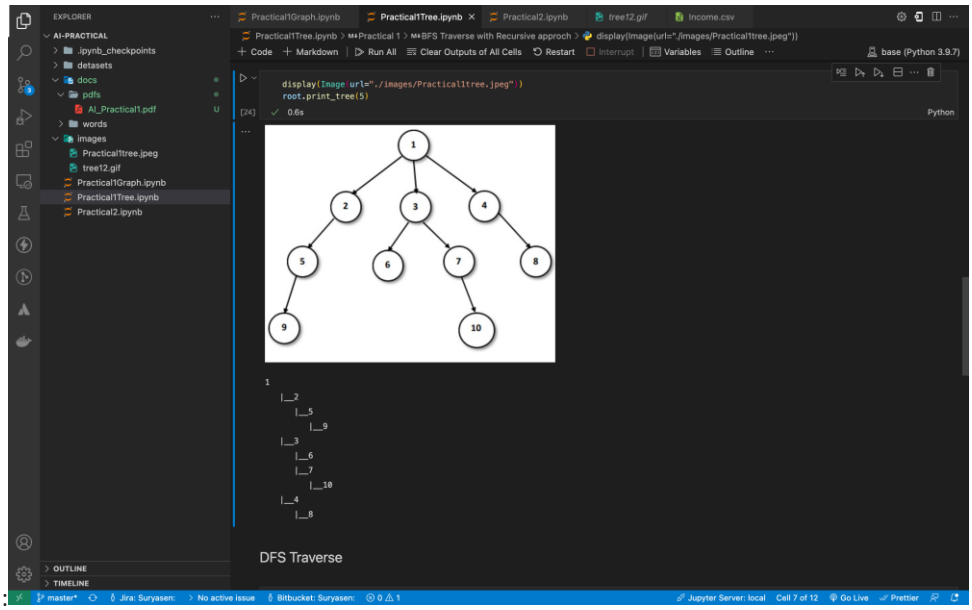


Code :

```
from IPython.core.display import Image,display
display(Image(url="./images/Practical1tree.jpeg"))
```

```
class TreeNode:
  def __init__(self,data):
    self.data=data

    self.children=[]
    self.parent=None
  def add_child(self,child):
```

```
        child.parent=self
        self.children.append(child)
    def get_levels(self):
        level=0
        p=self.parent
        while p:
            level +=1
            p=p.parent
        return level
    def print_tree(self, level):
        if self.get_levels() > level:
            return
        spaces = ' ' * self.get_levels() * 3
        prefix = spaces + "|__" if self.parent else ""
        print(prefix + self.data)
        if self.children:
            for child in self.children:
                child.print_tree(level)


def create_product_tree():
    root=TreeNode('1')
    node2=TreeNode('2')
    node3=TreeNode('3')
    node4=TreeNode('4')
    node5=TreeNode('5')
    node6=TreeNode('6')
    node7=TreeNode('7')
    node8=TreeNode('8')
    node9=TreeNode('9')
    node10=TreeNode('10')
    node5.add_child(node9)
    node2.add_child(node5)
    root.add_child(node2)
    node7.add_child(node10)
    node3.add_child(node6)
    node3.add_child(node7)
    root.add_child(node3)
    node4.add_child(node8)
    root.add_child(node4)

    return root


root=create_product_tree()

display(Image(url="./images/Practical1tree.jpeg"))
root.print_tree(5)
```

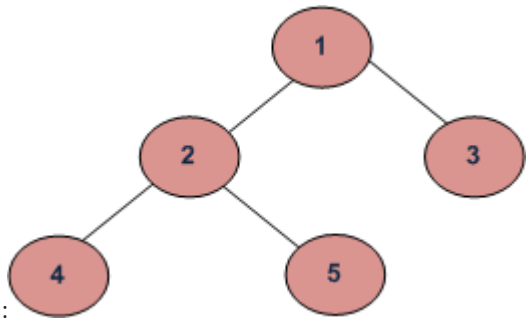
Output : 

Code Link : https://github.com/tonudon86/AI-practicals/blob/master/Practical1Tree.ipynb

1b)DFS

Aim :Implement DFS on Binary Tree :

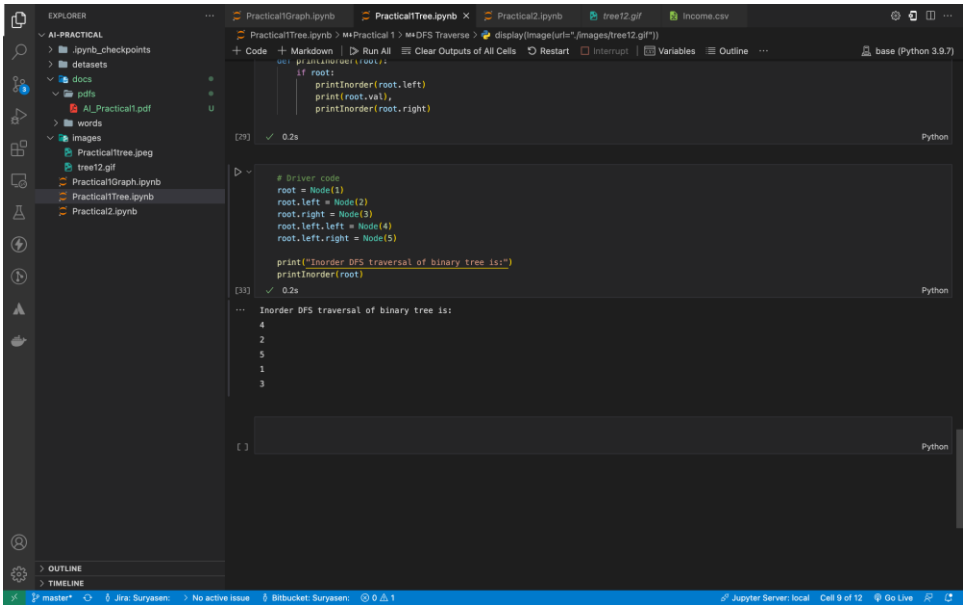
Thoery : DFS : Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search when a dead end occurs in any iteration.

Tree : 

Code :

```
display(Image(url="./images/tree12.gif"))

class Node:
    def __init__(self, key):
        self.left = None
        self.right = None
        self.val = key




def printInorder(root):
    if root:
        printInorder(root.left)
        print(root.val),
        printInorder(root.right)


# Driver code
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)

print("Inorder DFS traversal of binary tree is:")
printInorder(root)
```

Output :



Code link : https://github.com/tonudon86/AI-practicals/blob/master/Practical1Tree.ipynb