LE - 5
Binary Search Tree

Introduction:

This exercise is a hands-on component for the concepts covered in the lectures on trees. This is intended to give you the opportunity to implement a tree data structure in the form of a binary search tree.

Expected Functionality:

The following are the expectations associated with LE - 5:

1. Implement the *insert* method to insert a new node with the passed key according to the binary search tree conditions.
2. Implement the *delete* method to delete an element based on the key and re-structure the tree as required.
3. Implement the *min_val* method to return the value of the min value (key) from the tree.
4. Implement the *max_val* method to return the value of the max value (key) from the tree.
5. Implement the *find* method to traverse through the tree and search for the element based on the key.
    - **Note**: If the key is not found the method should return **False** and if it is found then return **True.**

(This section is not part of the submission or codePost testing and is only for your own testing and exploration) Once you have implemented the five methods you may proceed to conduct the following experiments:

1. Create a 16-element BST.
2. Search element based on key (unsuccessful search.)
3. Insert and delete single element based on key (successful and unsuccessful.)
4. Repeat steps 1-2 on a 1 million element BST. Chart unsuccessful search runtime in increments of 20000 and find the time complexity of this operation.
5. Create a 1 million element left-skewed BST. Chart unsuccessful search runtime in increments of 20000 and find the time complexity of this operation.

Starter Code:

You are provided with the *LE5_BST.ipynb* file. The notebook contains a class template for **BST_Node** with a constructor that initializes a node with two pointer attributes (*left* and *right*) as well as an attribute *key* which takes the data value. There following methods are part of the class:

> **__init__(key)** - creates a new node with left and right pointers as well as a key attribute to store the data.

insert(key) - inserts a new node in the BST with the given key.
delete(key) - deletes the node which stores the passed key while maintaining the BST structure.
min_val() - finds and returns the min value from the tree.
max_val() - finds and returns the max value from the tree.
find(lkpkey) - searches for the given key in the BST and return *True* if found and *False* if not.
PrintTree() - prints the tree via inorder traversal i.e., left -> parent -> right

You need to add your code in the relevant methods to complete the BST_Node class.

Rubric:

Your code will be tested with provided test cases.

Note: Scoring will be done on the full scale (15 points) if the methods are implemented recursively and partial scale (10 points) otherwise.

Location of the code:

The code would be provided at:
https://colab.research.google.com/drive/1_SLC2XfFxVgvJeo8QudkYCIOMiBMpzck?usp=sharing

What to do when done:

Once you have completed the exercise, you should upload it to the codePost portal. Please ensure the following while submitting:

1. Once satisfied with your code, you should download the file as a python script (.py file), by going to **File > Download > Download .py**
2. The name of the file should be LE5.py
3. Upload the python script file to codePost under the LE-5 assignment.
4. You can run the test cases on your script up to a limit of 50 times.
5. Once satisfied with the test runs, complete your submission.