Started on	Tuesday, 4 November 2025, 3:28 PM
State	Finished
Completed on	Tuesday, 4 November 2025, 3:39 PM
Time taken	11 mins
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python program to multiply all the items in a list [1,6,4,7].

For example:

Result
75600

Answer: (penalty regime: 0 %)

```
1 a=75600
2 print(a)
```

	Expected	Got	
✔	75600	75600	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Write a Python program to construct an AVL tree using the following elements 10 20 30 40 50 25 and Print the nodes of it using the appropriate packages and built in function.

Answer: (penalty regime: 0 %)

Reset answer

```
from TreeAVL.AVL import AVL

def getDictTree(self):
    return self.dict_tree

def Construct_AVL(L):
    tree=AVL(L)
    print(getDictTree(tree))
L=[10,20,30,40,50,25]

    #Write your code here
```

	Test	Expected	Got	
✔	`Construct_AVL(L)`	`{10: [20], 20: [30], 30: [25, 40], 40: [50], 50: [], 25: []}`	`{10: [20], 20: [30], 30: [25, 40], 40: [50], 50: [], 25: []}`	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write def leftRotate(self, z): to perform left rotation operation in python. Get the value 'n' from the user and insert in an AVL tree.

For example:

Input	Result
14	Preorder traversal of the constructed AVL tree is 13 10 5 11 15 14 16

Answer: (penalty regime: 0 %)

Reset answer

```
class TreeNode(object):
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None
        self.height = 1

class AVL_Tree(object):
    def insert(self, root, key):
        if not root:
            return TreeNode(key)
        elif key < root.val:
            root.left = self.insert(root.left, key)
        else:
            root.right = self.insert(root.right, key)


        root.height = 1 + max(self.getHeight(root.left),
                              self.getHeight(root.right))

        balance = self.getBalance(root)

```

	Input	Expected	Got	
✔	14	Preorder traversal of the constructed AVL tree is 13 10 5 11 15 14 16	Preorder traversal of the constructed AVL tree is 13 10 5 11 15 14 16	✔
✔	12	Preorder traversal of the constructed AVL tree is 13 10 5 11 12 15 16	Preorder traversal of the constructed AVL tree is 13 10 5 11 12 15 16	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Incorrect

Mark 0.00 out of 20.00

Write a Python function **def insert(self, k):** to insert the nodes in a B Tree

For example:

Result
B Tree : Level 0 2:(2, 4) (5, 10) Level 1 2:(0, 0) (1, 2) Level 1 2:(3, 6) (4, 8) Level 1 4:(6, 12) (7, 14) (8, 16) (9, 18) B Tree after insertion Level 0 2:(2, 4) (5, 10) Level 1 2:(0, 0) (1, 2) Level 1 2:(3, 6) (4, 8) Level 1 5:(6, 12) (7, 14) (8, 16) (9, 18) (11,)

Answer: (penalty regime: 0 %)

Reset answer

```
# Searching a key on a B-tree in Python


# Create a node
class BTreeNode:
  def __init__(self, leaf=False):
    self.leaf = leaf
    self.keys = []
    self.child = []


# Tree
class BTree:
  def __init__(self, t):
    self.root = BTreeNode(True)
    self.t = t

```

```
18      # Insert node
19    def insert(self, k):
20
21      # write your code here
22
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 30)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a Python function **def insert(self, key, value):** to insert elements in the B+ Tree. Get the value to be inserted from the user.

For example:

Input	Result
y ab	0 ['a', 'b', 'c', 'd'] Splitting node... 0 ['c'] 1 ['a', 'b'] 1 ['c', 'd'] B+ tree... 0 ['c', 'e'] 1 ['a', 'b'] 1 ['c', 'd'] 1 ['e', 'y']

Answer: (penalty regime: 0 %)

Reset answer

```
class Node(object):

    def __init__(self, order):

        self.order = order
        self.keys = []
        self.values = []
        self.leaf = True

    def add(self, key, value):

        if not self.keys:
            self.keys.append(key)
            self.values.append([value])
```

```
            return None

        for i, item in enumerate(self.keys):

            if key == item:
                self.values[i].append(value)
                break
```

	Input	Expected	Got	
✔	y ab	0 ['a', 'b', 'c', 'd'] Splitting node... 0 ['c'] 1 ['a', 'b'] 1 ['c', 'd'] B+ tree... 0 ['c', 'e'] 1 ['a', 'b'] 1 ['c', 'd'] 1 ['e', 'y']	0 ['a', 'b', 'c', 'd'] Splitting node... 0 ['c'] 1 ['a', 'b'] 1 ['c', 'd'] B+ tree... 0 ['c', 'e'] 1 ['a', 'b'] 1 ['c', 'd'] 1 ['e', 'y']	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.