

- ### a) Die folgende Klassenimplementierung enthält 2 Syntax Fehler, finden und verbessern sie diese.
- ### b) Welche drei Fehler des SearchTree's treten erst zur Laufzeit auf? Korrigieren sie diese.
- ### c) Welcher Semantische Fehler steckt in der Implementierung? (Docstrings beachten)

```
class SearchTree:
```

```
    """Implements a search tree for sortable objects.
    """
```

```
    def __init__(self, items=None):
```

```
        """Initializes tree with elements from items (for
        example a list).
        """
```

```
        self.tree = None
```

```
        for entry in items:
```

```
            self.tree = self._helper_add(self.tree, entry)
```

Wenn items = None TypeError

```
    def add(self, item):
```

```
        """Insert an item in the tree. Do nothing if tree
        already contains the item.
        """
```

wird nicht geprüft

```
self.tree = self._helper_add(self.tree, item)
```

```
    def _helper_add(self, tree, item):
```

```
        if tree is None:
```

```
            return [item, None, None]
```

```
        if tree[0] > item:
```

```
            tree[2] = self._helper_add(tree[2], item)
```

```
        elif tree[0] < item:
```

```
            tree[3] = self._helper_add(tree[3], item)
```

```
        return tree
```

*contain wird beim Adden nicht aufgerufen
if tree[0] == item fehlt → Semantischer Fehler*

(LAUFZEIT FEHLER)

*[item, None, None]
(0, 1, 2)*

Index Error

```
    def __contains__(self, item):
```

```
        """Check if an item is in the tree
        """
```

```
        return self._helper_contains(self.tree, item)
```

```
    def _helper_contains(self, tree, item):
```

```
        if tree is None:
```

```
            return False
```

```
        elif tree[0] == item:
```

```
            return True
```

```
        elif tree[0] > item:
```

```
            return self._helper_contains(tree[1], item)
```

```
        else:
```

```
            return self._helper_contains(tree[2], item)
```

```
    def tolist(self):
```

```
        """Compute ordered list of all entries.
        """
```

```
        self._tmplist = []
```

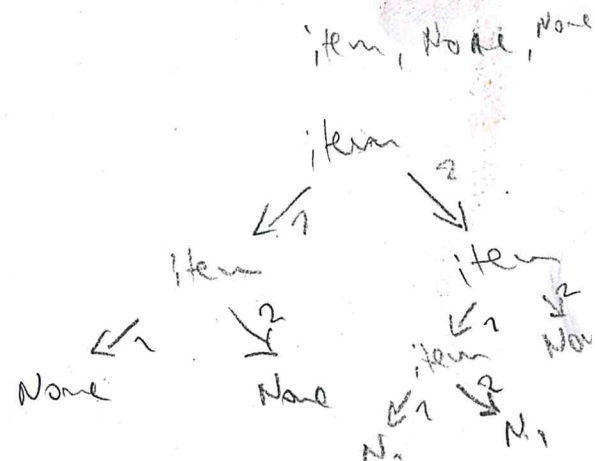
```
        self._helper_tolist(self.tree)
```

```
        return self._tmplist
```

```
    def _helper_tolist(self, node):
```

```
        if node is not None:
```

```
            self._helper_tolist(node[1])
```



```
self._tmplist.append(node[0])  
self._helper_tolist(node[2])
```