Python Data Model - Assignment

1. Scoreboard Class

We would like to implement a `Scoreboard` class to keep track of scores of a team.

1.1. Basic Class

Implement `Scoreboard` class as following:

- Implement its initializer method which access a string `name` and a list `scores`. Copy them to intance attributes `_name` and `_scores`.
- If `scores` is None, assign empty list to `_scores`.

Sample Output:

```
SG Team []
SG Team [1, 2, 3]
```

In [1]:

```
class Scoreboard:

    def __init__(self, name, scores = None):
        self._name = name
        if scores:
            self._scores = list(scores)
        else:
            self._scores = []

# Testing
s1 = Scoreboard('SG Team')
s2 = Scoreboard('SG Team', [1,2,3])
print(s1._name, s1._scores)
print(s2._name, s2._scores)
```

```
SG Team []
SG Team [1, 2, 3]
```

1.2 String Representations

Implement `Scoreboard2` as a derived class of `Scoreboard`.

- Implement `__str__()` method to return a string `name: scores`. E.g. `Team SG: [1, 2, 3]`
- Implement `__repr__()` method such that its returned string can be `eval()` to create an object of same value.

Sample Ouptput:

```
Team SG: [30, 40, 50]
Team SG: [30, 40, 50]
```

In [2]:

```
class Scoreboard2(Scoreboard):

    def __str__(self):
        return '{}: {}'.format(self._name, self._scores)

    def __repr__(self):
        return '{}("{}",{})'.format(self.__class__.__name__, self._name, self._scores)


# Testing
s1 = Scoreboard2('Team SG', [30, 40, 50])
print(s1)
s2 = eval(repr(s1))
print(s2)
```

```
Team SG: [30, 40, 50]
Team SG: [30, 40, 50]
```

1.3 Container Protocols

Implement subclass `Scoreboard3` from class `Scoreboard2`. Make it behave like a container type by implementing `__len__()`, `__getitem__()`, `__setitem__()` and `__delitem__()` methods.

- Score(s) can be access by indexing and slicing.
- Individual score can be updated using indexing and assignment statement.
- A score by be deleted using `del` statement.

Sample Output:

```
SG Team: [10, 20, 30]
SG Team: [50, 20, 30]
SG Team: [50, 20]
```

In [3]:

```
class Scoreboard3(Scoreboard2):

    def __len__(self):
        return len(self._scores)

    def __getitem__(self, position):
        return self._scores[position]

    def __setitem__(self, position, value):
        self._scores[position] = value

    def __delitem__(self, position):
        del self._scores[position]

# Testing
s = Scoreboard3('SG Team', [10,20,30])
print(s)

s[0] = 50
print(s)

del s[-1]
print(s)
```

```
SG Team: [10, 20, 30]
SG Team: [50, 20, 30]
SG Team: [50, 20]
```

1.4 Comparison Operators

Derive class `Scoreboard4` from class `Scoreboard3`.

- Implement `__lt__()`, `__le__()` and `__eq__()` methods so that any 2 instances of `Scoreboard4` can compare with each other.
- The scoreboard instance with larger sum of scores is considered greater.

Sample Output:

```
s1>s2: False
s1<=s2: True
s1==s2: False
```

In [4]:

```
class Scoreboard4(Scoreboard3):

    def __lt__(self, other):
        return sum(self._scores) < sum(other._scores)

    def __le__(self, other):
        return sum(self._scores) <= sum(other._scores)

    def __eq__(self, other):
        return sum(self._scores) == sum(other._scores)


# Testing
s1 = Scoreboard4('Team A', [10, 15, 20])
s2 = Scoreboard4('Team B', [15, 15, 25])
print('s1>s2:', s1>s2)
print('s1<=s2:', s1<=s2)
print('s1==s2:', s1==s2)
```

```
s1>s2: False
s1<=s2: True
s1==s2: False
```

1.5 Arithmetic Operators

Derive a subclass `Scoreboard5` from `Scoreboard4`. Implement `__iadd__()` method which will combine the scores from the other team if both teams are having same name.

Sample Output

```
Team A: [5, 10, 15, 20, 25, 30]
Cannot combine scores from different teams: Team A and Team B
```

In [5]:

```
class Scoreboard5(Scoreboard4):

    def __iadd__(self, other):
        if self._name == other._name:
            self._scores.extend(other._scores)
            return self
        else:
            print("Cannot combine scores from different teams: {} and {}".format(self._
name, other._name))


# Testing
s1 = Scoreboard5('Team A', [5, 10, 15])
s2 = Scoreboard5('Team A', [20, 25, 30])
s3 = Scoreboard5('Team B', [20, 25, 30])

s1 += s2
print(s1)
s1 += s3
```

```
Team A: [5, 10, 15, 20, 25, 30]
Cannot combine scores from different teams: Team A and Team B
```