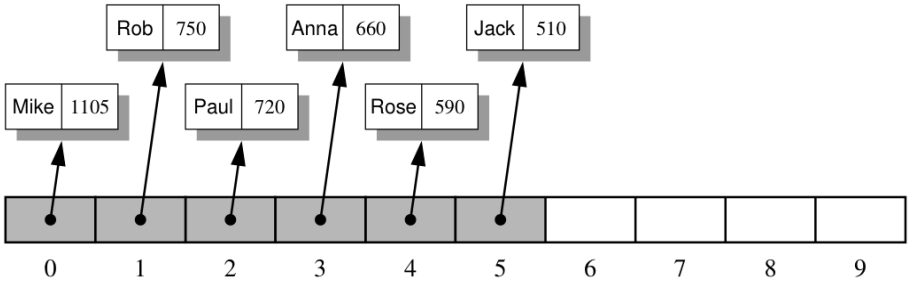
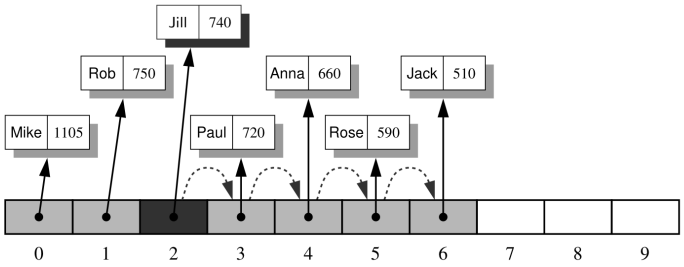
A class for high scores



Data structure ~ The array



Scoreboard Class

```python
class GameEntry:
  """Represents one entry of a list of high scores."""

  def __init__(self, name, score):
    self._name = name
    self._score = score

  def get_name(self):
    return self._name

  def get_score(self):
    return self._score

  def __str__(self):
    return '({0}, {1})'.format(self._name, self._score) # e.g., '(Bob, 98)'
```

```python
class Scoreboard:
  """Fixed-length sequence of high scores in nondecreasing order."""

  def __init__(self, capacity=10):
    """Initialize scoreboard with given maximum capacity.

    All entries are initially None.
    """
    self._board = [None] * capacity          # reserve space for future scores
    self._n = 0                              # number of actual entries

  def __getitem__(self, k):
    """Return entry at index k."""
    return self._board[k]

  def __str__(self):
    """Return string representation of the high score list."""
    return '\n'.join(str(self._board[j]) for j in range(self._n))

  def add(self, entry):
    """Consider adding entry to high scores."""
    score = entry.get_score()

    # Does new entry qualify as a high score?
    # answer is yes if board not full or score is higher than last entry
    good = self._n < len(self._board) or score > self._board[-1].get_score()

    if good:
      if self._n < len(self._board):         # no score drops from list
        self._n += 1                         # so overall number increases

      # shift lower scores rightward to make room for new entry
      j = self._n - 1
      while j > 0 and self._board[j-1].get_score( ) < score:
        self._board[j] = self._board[j-1]    # shift entry from j-1 to j
        j -= 1                               # and decrement j
      self._board[j] = entry                 # when done, add new entry
```

Calls to class to retrieve score of the entry

In case top scoreboard is already full.

Board is not full

Score is higher than the last entry

Scoreboard isn't filled yet

```python
if __name__ == '__main__':
  board = Scoreboard(5) #number of entries into scoreboard
  for e in (
    ('Rob', 750), ('Mike',1105), ('Rose', 590), ('Jill', 740),
    ('Jack', 510), ('Anna', 660), ('Paul', 720), ('Bob', 400),
    ):
    ge = GameEntry(e[0], e[1])
    board.add(ge)
    print('After considering {0}, scoreboard is:'.format(ge))
    print(board)
    print()
```

for loop

Create empty Scoreboard

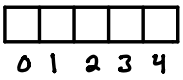
Add list into Game entry class

Add entries to board

Scoreboard

Index	0	1	2	3	4

Iteration 1 ('Rob', 750)



j ~

Start: board.add(ge) ~ add / send entry to Scoreboard class

```
score = entry.get_score( )
```
Score = 750

Self._n = 0

```
good = self._n < len(self._board) or score > self._board[−1].get_score( )
```
~5

Yes so continue

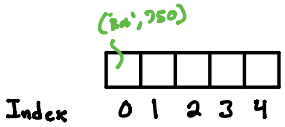
```
if good:                                  ~5
  if self._n < len(self._board):          ~ 0 < 5
    self._n += 1
```
then

Self.n += 1 = 0 + 1

Self.n = 1

Index → `j = self._n − 1` → j = 1 − 1, j = 0

j=0 so we continue →
```
while j > 0 and self._board[j−1].get_score( ) < score:
    self._board[j] = self._board[j−1]
    j −= 1
self._board[j] = entry
```
~ enter our score at index j=0




```
print('After considering {0}, scoreboard is:'.format(ge))
```
After considering (Rob, 750), scoreboard is:

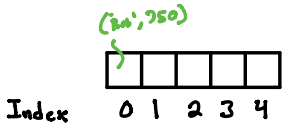
```
print(board)
```
↓ Calls this function

```
def __str__(self):
    """Return string representation of the high score list."""
    return '\n'.join(str(self._board[j]) for j in range(self._n))
```

(Rob, 750)

Iteration 2 (Mike, 1105)

Current Board




```
score = entry.get_score()
```

Score = 1050
Self._n = 1
~ len board is only 1 right now

```
good = self._n < len(self._board) or score > self._board[-1].get_score() ✓
```

```
if good:                                    ~ 5
    if self._n < len(self._board):
        self._n += 1
    j = self._n - 1
```

1 → Self.n += 1 → Self.n = 2

j = 2 - 1
j = 1

✓

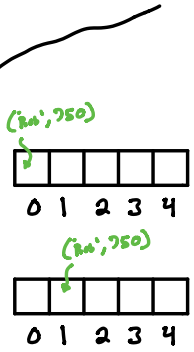
```
while j > 0 and self._board[j-1].get_score( ) < score:
    self._board[j] = self._board[j-1]
    j -= 1
```

j=0: 1-1 = 750 < 1050 then

Continue While loop until condition is false Which is the false now

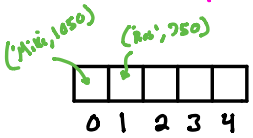
j = 1 - 1
j = 0

Swap around since smaller



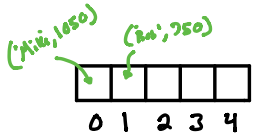
When j=0 → `self._board[j] = entry`

~ add entry @ j=0



~ print.

Iteration 3 ('Rose', 590)



Iteration 4