Project 4 – Calcudoku

Section: CPE101 – 03

Name: Claire Minahan

Instructor: S. Einakian

funcs.py template

#creates the cages

#int int → list

create_cage(num, tot)

- Take input value representing number of boxes in cage and what the boxes should add to
- If the count is less than the num add a '0' item to the list
- Return list(cage)

self.assertEqualTo(create_cage(3, 5), [[0, 0, 0], [5])

#gets the sum of a cage

#list → int

get_cage_sum(cages)

- Read the cage list
- Take the current index and add it to the total
- Move onto the next index and add it to the total
- Return the final total

cage = [3, 2, 0]

self.assertEqualTo(get_cage_sum(cage), 5)

#return true if all 3 validation functions below return True and False otherwise

#list list → boolean

validate_all(grid, cages)

- Call validate_rows(grid)
- Call validate_cols(grid)
- Call validate_cages(grid, cages)
- If all the functions above return True, return True
- Else return false

grid1 = [1, 2, 3, 4, 1]

cage1 = [[2, 3], [4]]

self.assertFalse(validate_all(grid1, cage1))

#return true if all rows contain no duplicate positive numbers and False otherwise

#list → boolean

validate_rows(grid)

- Read the list
- Take the first index in the row and compare it to the rest
- if it has a duplicate in the row return False
- go onto next index and compare it to the rest of the row
- if none are duplicates return True

rows = [2, 3, 1, 5, 4]

self.assertTrue(validate_rows(rows))

#return true if all columns contain no duplicate positive number and False otherwise

#list → boolean

validate_cols(grid)

- read the list
- take the first index in the column and compare it the rest
- if it has a duplicate in the column return False
- go onto the next index and compare it to the rest of the column
- if none are duplicates return True

cols = [2, 1, 4, 3, 2]

self.assertFalse(validate_cols(cols))

#return true if the sum of values in a fully populated cage equals the required sum or the sum values in a
#a partially populated cage is less than the required sum and False otherwise

#list list → boolean

validate_cages(grid, cages)

- take the grid list
- take the cage list
- add the first index in the cage to the total (starting at 0)
- move onto the next index and add it to the total
- if the cage is fully populated and the sum is equal to the required sum, return True
- else if the cage is not fully populated and the sum is less than the required sum, return True
- else return False

grid1 = [2, 3, 4, 5, 0]

cage1 = [[2, 3, 4], [13]]

self.assertFalse(validate_cages(grid1, cage1))

#returns False is a spot contains a 0 and True otherwise

#list → boolean

check_for_zero(grid)

- read the list
- take the first index
- check if it is equal to 0
- return False if a spot contains a 0
- else return True

grid1 = [2, 1, 5, 4, 3, 0]

self.assertFalse(check_for_zero(grid1))