# Project 4 – Calcudoku

# Section: CPE101 – 03

# Name: Claire Minahan

# Instructor: S. Einakian

funcs.py template

#creates the cages

#input 🡪 list

create\_cages()

* 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 cage item to the list
* Input the numbers into the individual cage
* Return list(cages)

#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\_columns(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\_columns(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))