LinearProbing.py

from Record import Record

class hashTable:

    # initialize hash Table

    def \_\_init\_\_(self):

        self.size = int(input("Enter the Size of the hash table : "))

        # initialize table with all elements 0

        self.table = list(None for i in range(self.size))

        self.elementCount = 0

        self.comparisons = 0

    # method that checks if the hash table is full or not

    def isFull(self):

        if self.elementCount == self.size:

            return True

        else:

            return False

    # method that returns position for a given element

    def hashFunction(self, element):

        return element % self.size

    # method that inserts element into the hash table

    def insert(self, record):

        # checking if the table is full

        if self.isFull():

            print("Hash Table Full")

            return False

        isStored = False

        position = self.hashFunction(record.get\_number())

        # checking if the position is empty

        if self.table[position] == None:

            self.table[position] = record

            print("Phone number of " + record.get\_name() + " is at position " + str(position))

            isStored = True

            self.elementCount += 1

        # collision occured hence we do linear probing

        else:

            print("Collision has occured for " + record.get\_name() + "'s phone number at position " + str(position) + " finding new Position.")

            while self.table[position] != None:

                position += 1

                if position >= self.size:

                    position = 0

            self.table[position] = record

            print("Phone number of " + record.get\_name() + " is at position " + str(position))

            isStored = True

            self.elementCount += 1

        return isStored

    # method that searches for an element in the table

    # returns position of element if found

    # else returns False

    def search(self, record):

        found = False

        position = self.hashFunction(record.get\_number())

        self.comparisons += 1

        if(self.table[position] != None):

            if(self.table[position].get\_name() == record.get\_name() and self.table[position].get\_number() == record.get\_number()):

                isFound = True

                print("Phone number found at position {} ".format(position) + " and total comparisons are " + str(1))

                return position

        # if element is not found at position returned hash function

            else:

                position += 1

                if position >= self.size-1:

                    position = 0

                while self.table[position] != None or self.comparisons <= self.size:

                    if(self.table[position].get\_name() == record.get\_name() and self.table[position].get\_number() == record.get\_number()):

                        isFound = True

                        #i=0

                        i = self.comparisons + 1

                        print("Phone number found at position {} ".format(position) + " and total comparisons are " + str(i) )

                        return position

                    position += 1

                    #print(position)

                    if position >= self.size-1:

                        position = 0

                    #print(position)

                    self.comparisons += 1

                    #print(self.comparisons)

                if isFound == False:

                    print("Record not found")

                    return false

    # method to display the hash table

    def display(self):

        print("\n")

        for i in range(self.size):

            print("Hash Value: "+str(i) + "\t\t" + str(self.table[i]))

        print("The number of phonebook records in the Table are : " + str(self.elementCount))