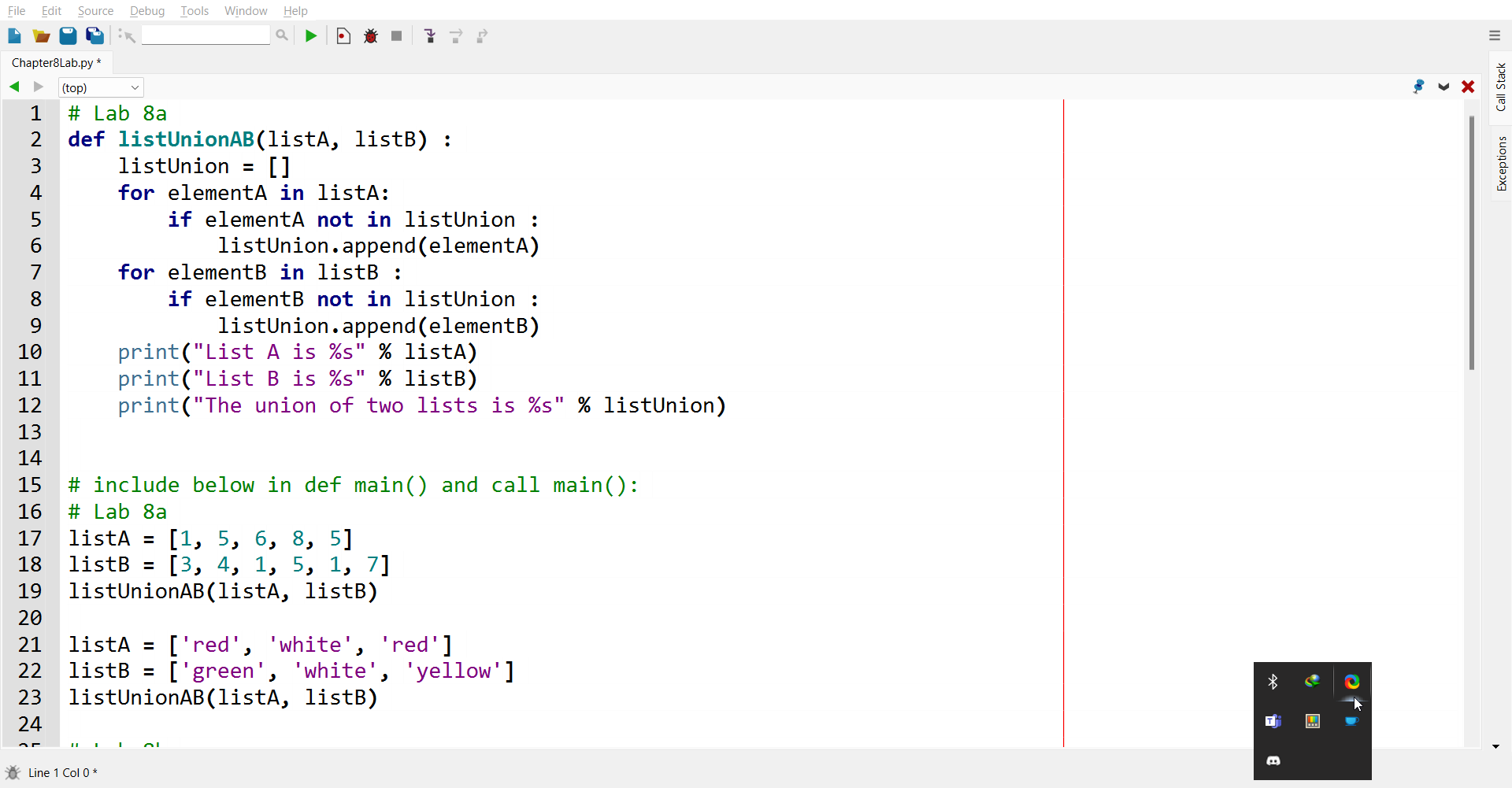
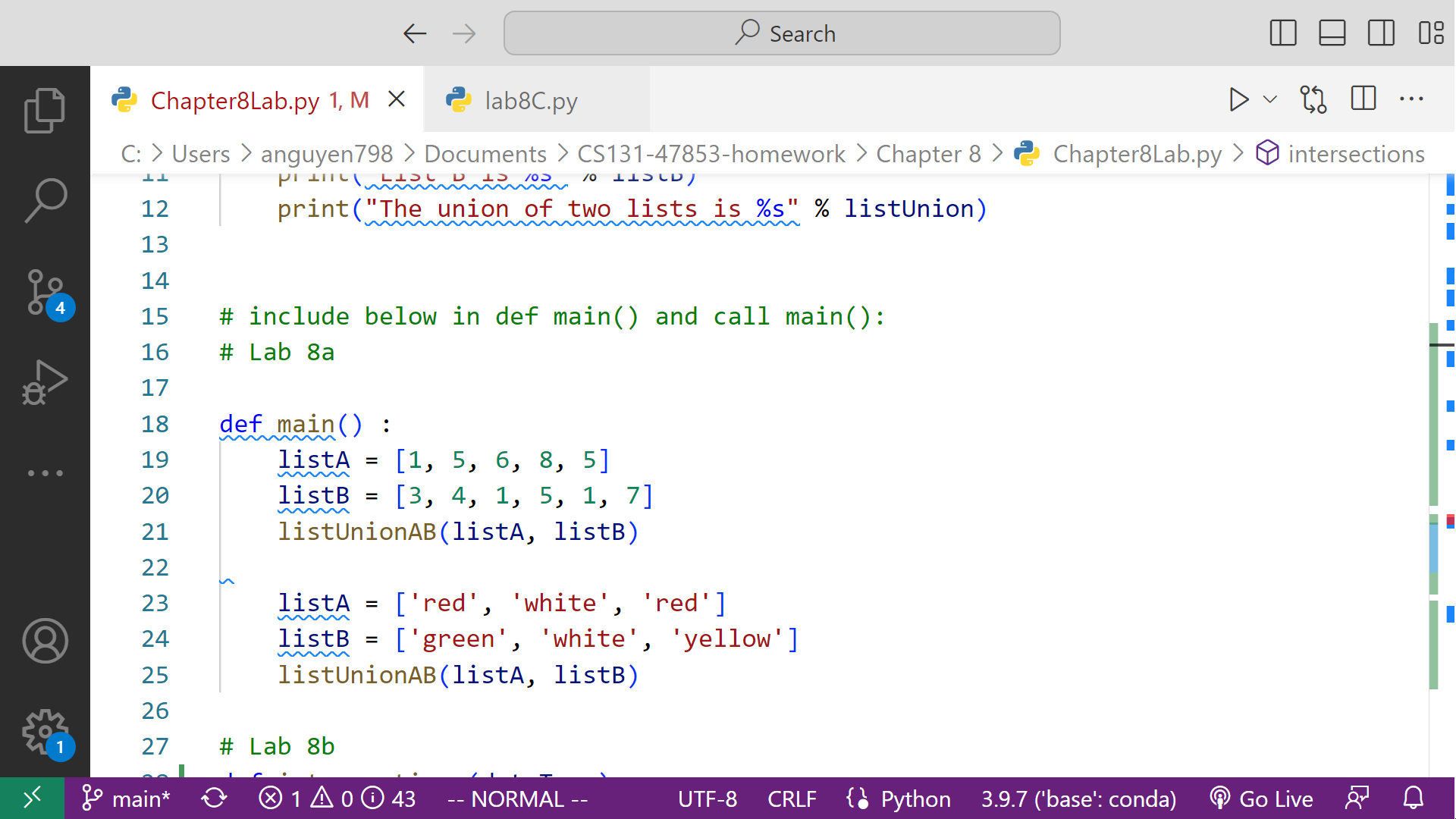
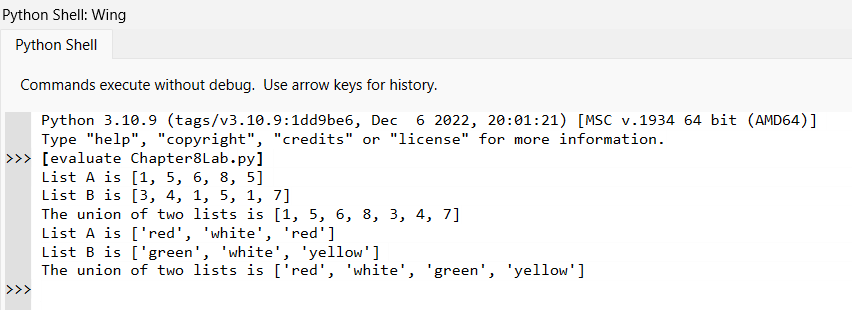
**Lab 8A**

**Lab 8A - Code**



****

**Lab 8A - Console**



**Lab 8A – Written Code**

def listUnionAB(listA, listB) :

    listUnion = []

    for elementA in listA:

        if elementA not in listUnion :

            listUnion.append(elementA)

    for elementB in listB :

        if elementB not in listUnion :

            listUnion.append(elementB)

    print("List A is %s" % listA)

    print("List B is %s" % listB)

    print("The union of two lists is %s" % listUnion)

def main() :

    listA = [1, 5, 6, 8, 5]

    listB = [3, 4, 1, 5, 1, 7]

    listUnionAB(listA, listB)

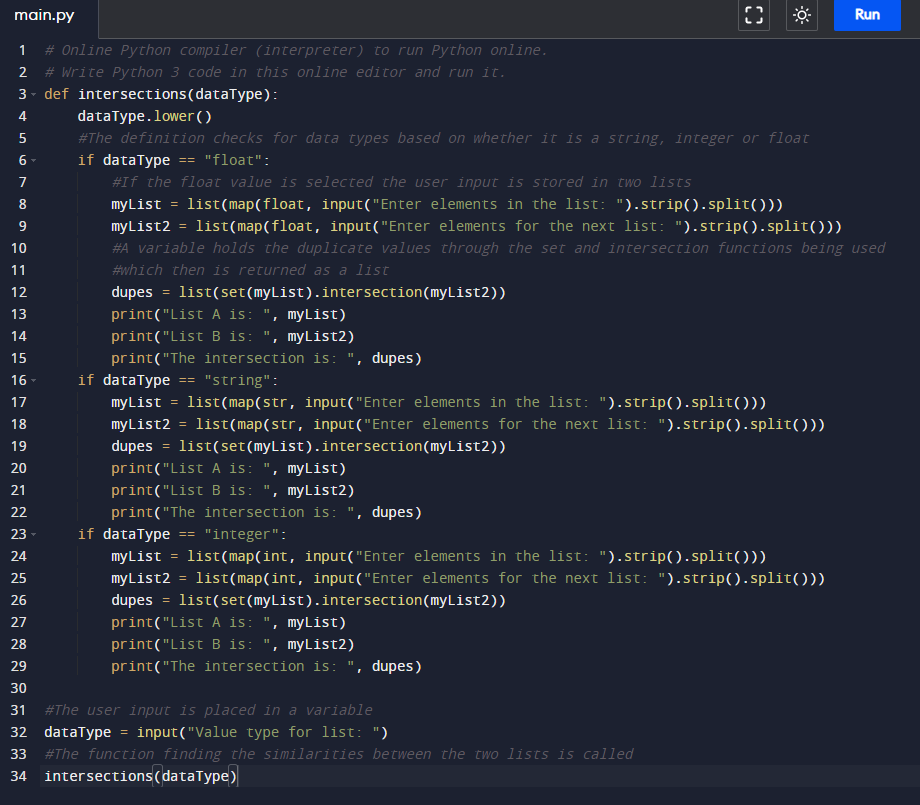
    listA = ['red', 'white', 'red']

    listB = ['green', 'white', 'yellow']

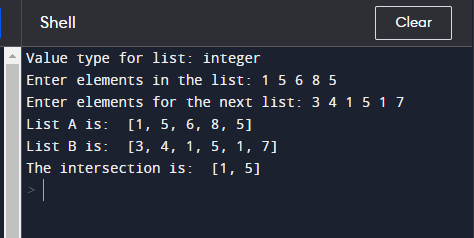
    listUnionAB(listA, listB)

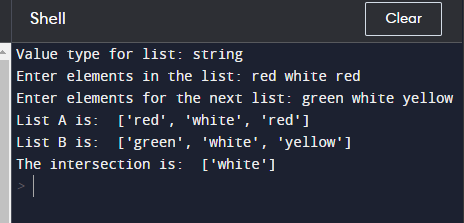
**Lab 8B**

**Lab 8B - Code**



**Lab 8B – Output**





**Lab 8B – Written Code**

def intersections(dataType) :

    dataType.lower()

    #The definition checks for data types based on whether it is a string, integer or float

    if dataType == "float" :

        # If the float value is selected the user input is stored in two lists

        myList = list(map(float, input("Enter elements in the list: ").strip().split()))

        myList2 = list(map(float, input("Enter elements for the next list: ").strip().split()))

        # A variable holds the duplicate values through the set and intersection functions being used

        # Which then is returned as a list

        dupes = list(set(myList).intersection(myList2))

        print("List A is: ", myList)

        print("List B is: ", myList2)

        print("The intersection is ", dupes)

    if dataType == "string" :

        myList = list(map(str, input("Enter elements in the list: ").strip().split()))

        myList2 = list(map(str, input("Enter elements for the next list: ").strip().split()))

        dupes = list(set(myList).intersection(myList2))

        print("List A is: ", myList)

        print("List B is: ", myList2)

        print("The intersection is ", dupes)

    if dataType == "integer" :

        myList = list(map(int, input("Enter elements in the list: ").strip().split()))

        myList2 = list(map(int, input("Enter elements for the next list: ").strip().split()))

        dupes = list(set(myList).intersection(myList2))

        print("List A is: ", myList)

        print("List B is: ", myList2)

        print("The intersection is ", dupes)

# The user input is placed in a variable

dataType = input("Value type for list: ")

# The function finding the similarities between the two lists is called

intersections(dataType)

**Lab 8C**

**Lab 8C - Code**

**Lab 8C - Console**

**Lab 8C - Output File**

**Lab 8C – Written Code**

From lab8C.py

def is\_uniques(string):

    originalString = string

    string = string.lower()

    string = string.replace(" ", "")

    if len(string) == len(set(string)) :

        unique = True

    else :

        unique = False

    print("The string \"%s\" has unique characters? %s" % (originalString, unique))

    return unique

def is\_pangram(string) :

    alphabet = set("abcdefghijklmnopqrstuvwxyz")

    string = string.lower()

    stringSet = set(string)

    for letter in alphabet :

        if letter not in stringSet :

            print("The string %s is not a pangram!" % string)

            return

    print("The string %s is a pangram!" % string)

import lab8C

string = "John Wick"

lab8C.is\_uniques(string)

string = "Samantha Ahtnamas"

lab8C.is\_uniques(string)

string = "The quick brown fox jumps over the lazy dog"

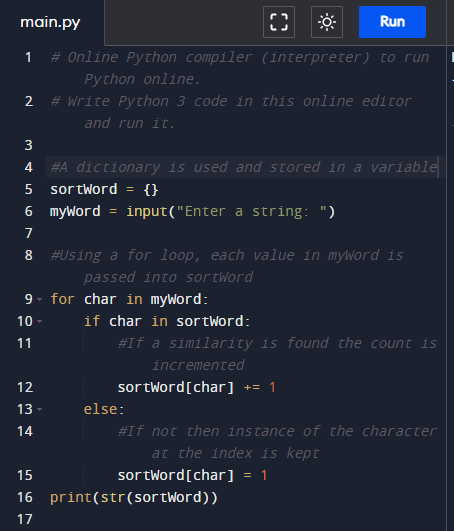
lab8C.is\_pangram(string)

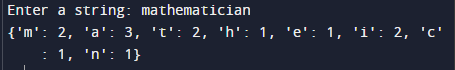
string = "The slow brown wolf jumps over the energetic coyote"

lab8C.is\_pangram(string)

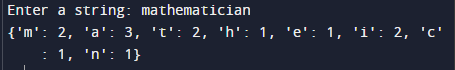
**Lab 8D**

**Lab 8D - Code**





**Lab 8D - Output**



**Lab 8D – Written Code**

# A dictionary is used and stored in a variable

sortWord = {}

myWord = input("Enter a string: ")

# Using a for loop, each value in myWord is passed into sortWord

for char in myWord :

    if char in sortWord :

        # If a similarity is found the count is incremented

        sortWord[char] += 1

    else :

        # If not then instance of the character at the index is kept

        sortWord[char] = 1

print(str(sortWord))