

Lab 8A

Lab 8A - Code

```
Chapter8Lab.py (D:\github.com\anguyen738\cs131-47853-homework\Chapter 8\Wing
File Edit Source Debug Tools Window Help
New Open Save Save All Get to Definition Run Break Debug Stop Step Into Step Over Step Out
Chapter8Lab.py lab8C.py
1 # Lab 8a
2 def listUnionAB(listA, listB) :
3     listUnion = []
4     for elementA in listA:
5         if elementA not in listUnion :
6             listUnion.append(elementA)
7     for elementB in listB :
8         if elementB not in listUnion :
9             listUnion.append(elementB)
10    print("List A is %s" % listA)
11    print("List B is %s" % listB)
12    print("The union of two lists is %s" % listUnion)
13
14
15 # include below in def main() and call main():
16 # Lab 8a
17
18 def main() :
19     listA = [1, 5, 6, 8, 5]
20     listB = [3, 4, 1, 5, 1, 7]
21     listUnionAB(listA, listB)
22
23     listA = ['red', 'white', 'red']
24     listB = ['green', 'white', 'yellow']
25     listUnionAB(listA, listB)
26
27 # Lab 8b
```

Lab 8A - Output

Python Shell: Wing

Python Shell

Commands execute without debug. Use arrow keys for history.




```
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec 6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> [evaluate Chapter8Lab.py]
List A is [1, 5, 6, 8, 5]
List B is [3, 4, 1, 5, 1, 7]
The union of two lists is [1, 5, 6, 8, 3, 4, 7]
List A is ['red', 'white', 'red']
List B is ['green', 'white', 'yellow']
The union of two lists is ['red', 'white', 'green', 'yellow']
>>>
```

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)  
  
main()
```

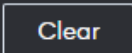
Lab 8B

Lab 8B - Code

```
main.py   
```

```
1 # Online Python compiler (interpreter) to run Python online.
2 # Write Python 3 code in this online editor and run it.
3 def intersections(dataType):
4     dataType.lower()
5     #The definition checks for data types based on whether it is a string, integer or float
6     if dataType == "float":
7         #If the float value is selected the user input is stored in two lists
8         myList = list(map(float, input("Enter elements in the list: ").strip().split()))
9         myList2 = list(map(float, input("Enter elements for the next list: ").strip().split()))
10        #A variable holds the duplicate values through the set and intersection functions being used
11        #which then is returned as a list
12        dupes = list(set(myList).intersection(myList2))
13        print("List A is: ", myList)
14        print("List B is: ", myList2)
15        print("The intersection is: ", dupes)
16    if dataType == "string":
17        myList = list(map(str, input("Enter elements in the list: ").strip().split()))
18        myList2 = list(map(str, input("Enter elements for the next list: ").strip().split()))
19        dupes = list(set(myList).intersection(myList2))
20        print("List A is: ", myList)
21        print("List B is: ", myList2)
22        print("The intersection is: ", dupes)
23    if dataType == "integer":
24        myList = list(map(int, input("Enter elements in the list: ").strip().split()))
25        myList2 = list(map(int, input("Enter elements for the next list: ").strip().split()))
26        dupes = list(set(myList).intersection(myList2))
27        print("List A is: ", myList)
28        print("List B is: ", myList2)
29        print("The intersection is: ", dupes)
30
31    #The user input is placed in a variable
32    dataType = input("Value type for list: ")
33    #The function finding the similarities between the two lists is called
34    intersections(dataType)
```

Lab 8B - Output

```
Shell 
```

```
> Value type for list: integer
> Enter elements in the list: 1 5 6 8 5
> Enter elements for the next list: 3 4 1 5 1 7
> List A is: [1, 5, 6, 8, 5]
> List B is: [3, 4, 1, 5, 1, 7]
> The intersection is: [1, 5]
> >
```

```
Shell Clear  
Value type for list: string  
Enter elements in the list: red white red  
Enter elements for the next list: green white yellow  
List A is: ['red', 'white', 'red']  
List B is: ['green', 'white', 'yellow']  
The intersection is: ['white']  
> |
```

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

lab8C.py module

```
Chapter8Lab.py lab8C.py
1 def is_uniques(string):
2     originalString = string
3     string = string.lower()
4     string = string.replace(" ", "")
5     if len(string) == len(set(string)) :
6         unique = True
7     else :
8         unique = False
9     print("The string \"%s\" has unique characters? %s"
10         % (originalString, unique))
11     return unique
12
13
14 def is_pangram(string) :
15     alphabet = set("abcdefghijklmnopqrstuvwxyz")
16     string = string.lower()
17     stringSet = set(string)
18     for letter in alphabet :
19         if letter not in stringSet :
20             print("The string %s is not a pangram!" % string)
21             return
22     print("The string %s is a pangram!" % string)
```

main().py file import lab8C

```
Chapter8Lab.py lab8C.py
62 # Lab 8c
63 def main():
64     import lab8C
65     string = "John Wick"
66     lab8C.is_uniques(string)
67     string = "Samantha Ahnmas"
68     lab8C.is_uniques(string)
69     string = "The quick brown fox jumps over the lazy dog"
70     lab8C.is_pangram(string)
71     string = "The slow brown wolf jumps over the energetic coyote"
72     lab8C.is_pangram(string)
73
74
75 main()
```

Lab 8C - Output

Python Shell: Wing

Python Shell

Commands execute without debug. Use arrow keys for history.

```
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec 6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> [evaluate Chapter8Lab.py]
The string "John Wick" has unique characters? True
The string "Samantha Ahtnamas" has unique characters? False
The string the quick brown fox jumps over the lazy dog is a pangram!
The string the slow brown wolf jumps over the energetic coyote is not a pangram!
>>>
```

Lab 8C – Written Code

from lab8C.py module

```
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)
```

main().py file import lab8C

```
def main() :  
    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)
```

main()

Lab 8D

Lab 8D - Code

```

1  # A dictionary is used and stored in a variable
2  def main(myWord) :
3      sortWord = {}
4      # Using a for loop, each value in myWord is passed into sortWord
5      for char in myWord :
6          if char in sortWord :
7              # If a similarity is found the count is incremented
8              sortWord[char] += 1
9          else :
10             # If not then instance of the character at the index is kept
11             sortWord[char] = 1
12     print("The number of times each letter occurs in the string \"%s\" "
13           % myWord)
14     # Print dictionary sortWord key, separate by space
15     for key in sortWord :
16         print("%s %3s" % (key, sortWord[key]))
17     # Function call separator
18     print("*" * 100)
19
20
21     main("mathematician")
22     # Ramanujan inspired function call
23     main("m" * 100 + "athematicia" + "n" * 100)
24

```

Lab 8D - Output

```
The number of times each letter occurs in the string "mathematician"
m 2
a 3
t 2
h 1
e 1
i 2
c 1
n 1
*****
The number of times each letter occurs in the string "mathtutoringmathematician"
m 101
a 3
t 2
h 1
e 1
i 2
c 1
n 100
*****
```

Lab 8D – Written Code

A dictionary is used and stored in a variable

```
def main(myWord) :  
    sortWord = {}  
    # 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("The number of times each letter occurs in the string \"%s\" "  
          % myWord)  
    # Print dictionary sortWord key, separate by space  
    for key in sortWord :  
        print("%s %3s" % (key, sortWord[key]))  
    # Function call separator  
    print("*" * 100)
```

```
main("mathematician")  
# Ramanujan inspired function call  
main("m" * 100 + "athematicia" + "n" * 100)
```