Lab 4
Lists & Tuples

Tasks:

1. (List Slicing) Create a list called numbers containing "1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20," then perform slicing operations to obtain the following:
a) The third number,
b) The first five numbers,
c) The first half of the list,
d) The last five numbers,
e) Every other number,
f) The numbers in reverse order, and
g) The third last number.

2. (Fibonacci) The Fibonacci numbers are a sequence in which each number is the sum of the two preceding ones. Define a function fib that receives three consecutive numbers of the Fibonacci series and returns the three subsequent values. Then, call the function three times starting with the numbers 0, 1,and 1 and restarting the function each time with the resulting numbers of the previous iteration.

3. (Duplicate Elimination) In organizations, a list of email addresses is often compiled for marketing purposes. However, duplicate email addresses need to be removed from this list. Write a function that receives a list and returns a list containing only unique values. Test your function with a list of numbers alist = [23,34,67,87,22,43,34,87,23,33,22,34].

4. (Insertion Sort Algorithm)** In computer science, there are several algorithms available to sort the elements of a list such as quicksort, merge, heapsort, and insertion sort. Although insertion sort is not the most efficient sorting algorithm when dealing with large lists, it is very efficient for small data sets and easy to implement. The algorithm splits a list into a sorted and unsorted part. Each value from the unsorted part is sequentially compared to each element in the sorted part and placed in the correct position. The process to sorting the list is:

a) Get a list of unsorted numbers.
b) Compare the first two elements.
c) If they are in descending order, switch them. This is now your sorted sub-list.
d) Take the third element of the list. Compare, and if necessary, swap it with the second element. Now compare the new second element and swap, if necessary, with the first element.

e) Take the fourth element of the list. Compare and swap with the third element, if necessary.
f) Keep repeating Steps (a) to (e) until every element of the list is in its correct place.

When this process completes, the list of numbers is sorted in ascending order. Write a function **`insertion_sort`** implementing the described algorithm. Use a list of 10 unique and randomly picked numbers between 0 and 100 to test this function, try the list **`alist = [23, 34, 67, 87, 22, 43, 33, 99, 23, 77]`**. Display the unsorted and the sorted list to evaluate the validity of your function.

5. (Palindrome Tester) A string that's spelled identically backward and forward, like 'radar', is a palindrome. Write a function is_palindrome that takes a string and returns True if it's a palindrome and False otherwise. Use a **stack** to help determine whether a string is a palindrome. Your function should ignore case sensitivity (that is, 'a' and 'A' are the same), spaces and punctuation. Test with **`test_str1 = "rA;da. .R"`** and **`test_str2 = "rader"`**. **Hint:** (1) convert the string into a list of lowercase characters; (2) remove all the spaces and punctions; (3) the stack is used to create a reverse list of characters.

6. (Encryption Key) During the second world war, messages were often encrypted based on an extract of a book. The unique letters of such an extract were summarized into an encryption key. This key was then used to encrypt a message. Write a function that accepts a string as input and isolates the different unique letters ignoring punctuation, spaces, and case sensitivity. The function should check if the encryption key contains all letters of the alphabet. If not, the missing letters should be added to the key. Test with the string "Give me my key please! Don't hesitate to generate one now."

7. (Encrypt) As seen in Task 6, an extract of a book was often used to encrypt messages during the second world war. They were encrypted by calculating the position of each of its letters in the regular alphabet. This position was used to select the **corresponding** letter in the encryption key. For example, the letter A has position 1 in the alphabet. The letter A in the message is thus replaced by the first letter in the encryption key. Write a script that uses the encryption key composed in Task 6 to encrypt "**Help me.**"

8. (List Sorting) Write a function **`is_sorted`** that compares the results from your **`insertion_sort`** function to the result from the built-in sorted function. If both functions return the same sorted list, the **`is_sorted`** function returns **True**. Test your function with several sorted and unsorted lists.

9. (Sorting User IDs) For each new user, a system generates a user ID consisting of a randomly chosen letter and two random numbers. Write a script that randomly generates a list containing 8000 userIDs using random letters from the range "a" through "f" and numbers from the range 1 to 4. Subsequently, remove all duplicate entries from the list and sort this list in descending order.

10. (Summing the Triples of the Even Integers from 2 through 10) Starting with a list containing 1 through 10, use **`filter`**, **`map`** and sum to calculate the total of the triples of the even integers from 2 through 10. Reimplement your code with list comprehensions rather than filter and map.

11. (Finding the People with a Specified Last Name) Create a list of tuples containing first and last names. Use **`filter`** to locate the tuples containing the last name "Nguyen". Ensure that several tuples in your list have that last name.