Topics covered in the lecture:

1. Slicing strings
2. len(s), in, not in, + (concat), * (replicate),
3. Splitting strings
4. String joining
5. Strings are immutable
6. s.lower(), s.upper(), s.strip(), s.replace(), s.count("str"), s.find("str"), s.isalpha(), s.isalnum(), s.isdigit(), s.capitalize(), s.title(), s.isupper(), s.islower(), s.startswith(str), s.endswith(str), ord(), chr()
7. Escape characters
8. String comparison
9. Taking multiple inputs
10. Traversing strings

Practice/Tutorial Questions:

Some output/error finding/mcq type questions for practice:

1. What would be the output of the given code:

```
def replace_substring(test_str, s1, s2):
  # Initialize an empty string to store the result
  result = ""
  # Initialize a variable to keep track of our position in the string
  i = 0
  # Loop through the string one character at a time
  while i < len(test_str):
    # Check if the current substring matches the substring we want to replace
    if test_str[i:i+len(s1)] == s1:
      # If it does, add the replacement substring to the result and move the pointer forward
      result += s2
      i += len(s1)
    else:
      # If it doesn't, add the current character to the result and move the pointer forward
      result += test_str[i]
      i += 1
  # Return the final result
```

```
    return result

# Test
test_str = "eatclasscodeeatcodesleep"
s1 = "code"
s2 = "learn"

print(replace_substring(test_str, s1, s2))
```

2. What is the output:

```
test_str = 'ankankaaabcgu'
print("The original string is : " + str(test_str))

# Initializing K
K = 'a'
cnt = 0
res = 0

for idx in range(len(test_str)):
    # Increment counter on checking
    if test_str[idx] == K:
        cnt += 1
    else:
        cnt = 0
    # Retaining max
    res = max(res, cnt)

# Printing result
print("The Longest Substring Length : " + str(res))
```

3. What is the output

```
def string_balance_test(s1, s2):
    flag = True
    for char in s1:
        if char in s2:
            continue
        else:
            flag = False
    return flag

s1 = "turn"
s2 = "Return"
flag = string_balance_test(s1, s2)
print("s1 and s2 are balanced:", flag)
```

```
s1 = "Ynf"
s2 = "PYnative"
flag = string_balance_test(s1, s2)
print("s1 and s2 are balanced:", flag)
```

4. Report errors in the given code and write the correct version

   Problem stmt: last position of a given substring

```
def find_last_position(s, sub):
  # Attempt to find the last occurrence of the substring
  position = s.index(sub)
  return position

# Example usage:
input_string = "hello world, welcome to the world"
substring = "world"
output = find_last_position(input_string, substring)
print("Last position of the substring:", output)
```

5. Report errors in the given code and write the correct version

   Problem stmt: Python program in to check string is anagrams or no

```
def are_anagrams(str1, str2):
  if str1 == str2:
    return True
  else:
    return False

# Example usage:
string1 = "listen"
string2 = "silent"
result = are_anagrams(string1, string2)
if result:
  print(f"{string1} and {string2} are anagrams.")
else:
  print(f"{string1} and {string2} are not anagrams.")
```

Write code for the given problems:

1. Write a program that checks whether a given word is a palindrome. A palindrome is a word that reads the same forwards and backwards. Examples: “madam”, malayalam etc.
2. Write a program that checks whether a given sentence is a palindrome. A palindrome is a sentence that reads the same forwards and backwards when

spaces, punctuation, and capitalization are ignored. Examples: “"No lemon, no melon.", "Was it a car or a cat I saw?", "Madam, in Eden, I’m Adam.", “"A man, a plan, a canal, Panama!"

3. Given a sentence as an input, reverse each word in the sentence
   Example:
   Input: I like python programming
   Output: I ekil nohtyp gnimmargorp

4. You are given a string. Split the string on a **“ “** (space) delimiter and join using a **“$”** (dollar character). Give 2 versions of code, one using list and one not using a list
   Example:
   - Input: this is a string
   - Output: this$is$a$string

5. Write a Python Program Print First Half of the String in Uppercase and the Next Half in Lowercase

6. Given two strings s and t, return the minimum window substring of s such that every character in t (including duplicates) is included in the window. If there is no such substring, return the empty string.
   - s = "ADOBECODEBANC", t = "ABC"
     - Output: "BANC"
   - s = "!@#AB$%^&*()ABC" , t = "AAB"
     - Output: "AB$%^&*()A"

Some scenario-based questions:

1. **Password Validator:**
   Write a program that validates user-entered passwords based on the following criteria:

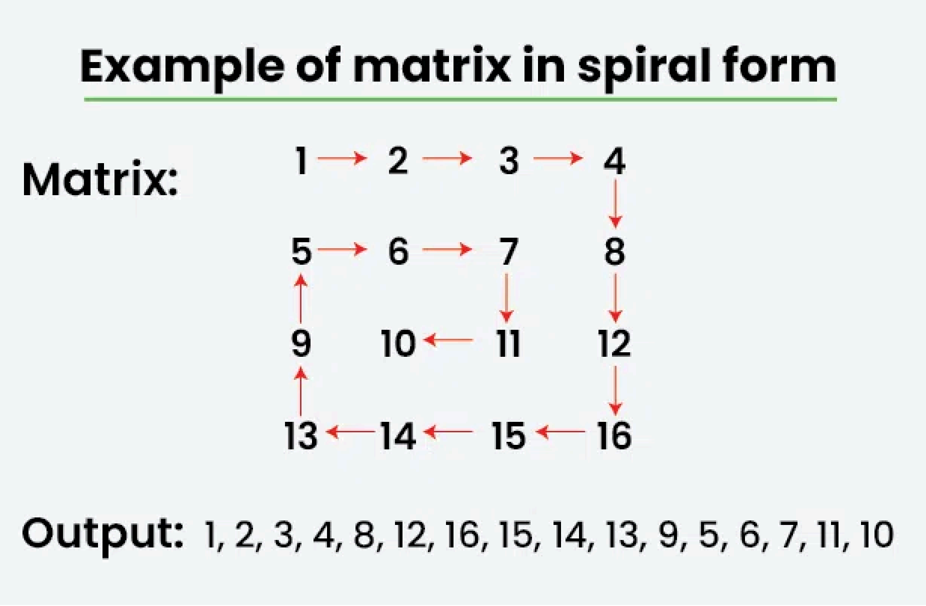
- Must be at least 8 characters long.
- Must contain at least one uppercase letter, one lowercase letter, and one digit
- Must not contain any spaces.
- Should not start or end with a space.

2. **Sentence Analyzer**
   Write a program that takes a sentence as input and provides the following analysis:
   - Total number of characters (excluding spaces).
   - Total number of words.
   - Number of vowels and consonants.
   - Whether the sentence is in uppercase, lowercase, or title case.
   - Checks if the sentence starts with a capital letter and ends with a proper punctuation mark (., !, ?).
3. **Movie Review Analyzer:** After the release of the movie trailer, the marketing team decides to analyze their social media engagement.
   Tasks:
   - Count how many comments mention either "Alex" or "Henry."
   - Filter out the comments that contain the word "love" and store the usernames and their comments in a new list.
   - Create a summary of the filtered comments that includes the total number of comments containing "love."

   Input:

```
comments = [
  ["user1", "I love Alex and Henry!"],
  ["user2", "What a great event."],
  ["user3", "Henry is amazing."],
  ["user4", "I love the movie!"],
  ["user5", "Alex and Henry are perfect!"],
  ["user6", "Love conquers all!"]
  [“user7”, “Very Demure, Very Mindful”]
]
```

4. Given a 2-D nested list (matrix), write a python code to return the spiral representation of the matrix.


Example of matrix in spiral form
Matrix:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Output: 1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10

[Image Source]

    - Input1: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
        - Output1: [1, 2, 3, 6, 9, 8, 7, 4, 5]
    - Input2: [[1, 2, 3,4,5,6], [7,8,9,10,11,12], [13,14,15,16,17, 18]]
        - Output2: [1, 2, 3, 4, 5, 6, 12, 18, 17, 16, 15, 14, 13, 7, 8, 9, 10, 11]

5. Take some text (e.g. from some site) and assign it to a string. (i) Write a program that takes as input a word and returns how many times the word is mentioned in the text. (ii) Find the frequency of different words in this text, and how many different words are used. (Pls do not use dictionary - you can use two lists)
6. **Caesar Cipher Encryption:** You need to implement a simple text encryption program using the Caesar cipher. Your program should encrypt messages by shifting each letter by a specified number of positions in the alphabet.

    Requirements:

    I. Create a function encrypt_message(text, shift) that:

    A. Encrypts the text using a Caesar cipher with a specified shift.

    B. Preserves the case of each letter.

C. Leaves non-alphabetic characters unchanged.

II. User Input:

A. A message to encrypt.

B. A non-negative integer for the shift value.

- If the user enters a negative shift, ask them to enter a valid value.

III. Output: Display the encrypted message.

Some examples:
Input format: text shift
Input1: Hello, World! 3
Output1: Khoor, Zruog!

Input2: Introduction to Programming 5
Output2: Lqwurgxfwlrq wr Surjudpplqj
Input3: zoom 3
Output3: crrp

Hint: Use strings+multiple inputs

```
str1 = input()  #Introduction to Programming!!! 3
words = str1.split()
text = ' '.join(words[:len(words)-1])
shift = int(words[len(words)-1])
print(text) #Introduction to Programming!!!
print(shift) #3
```