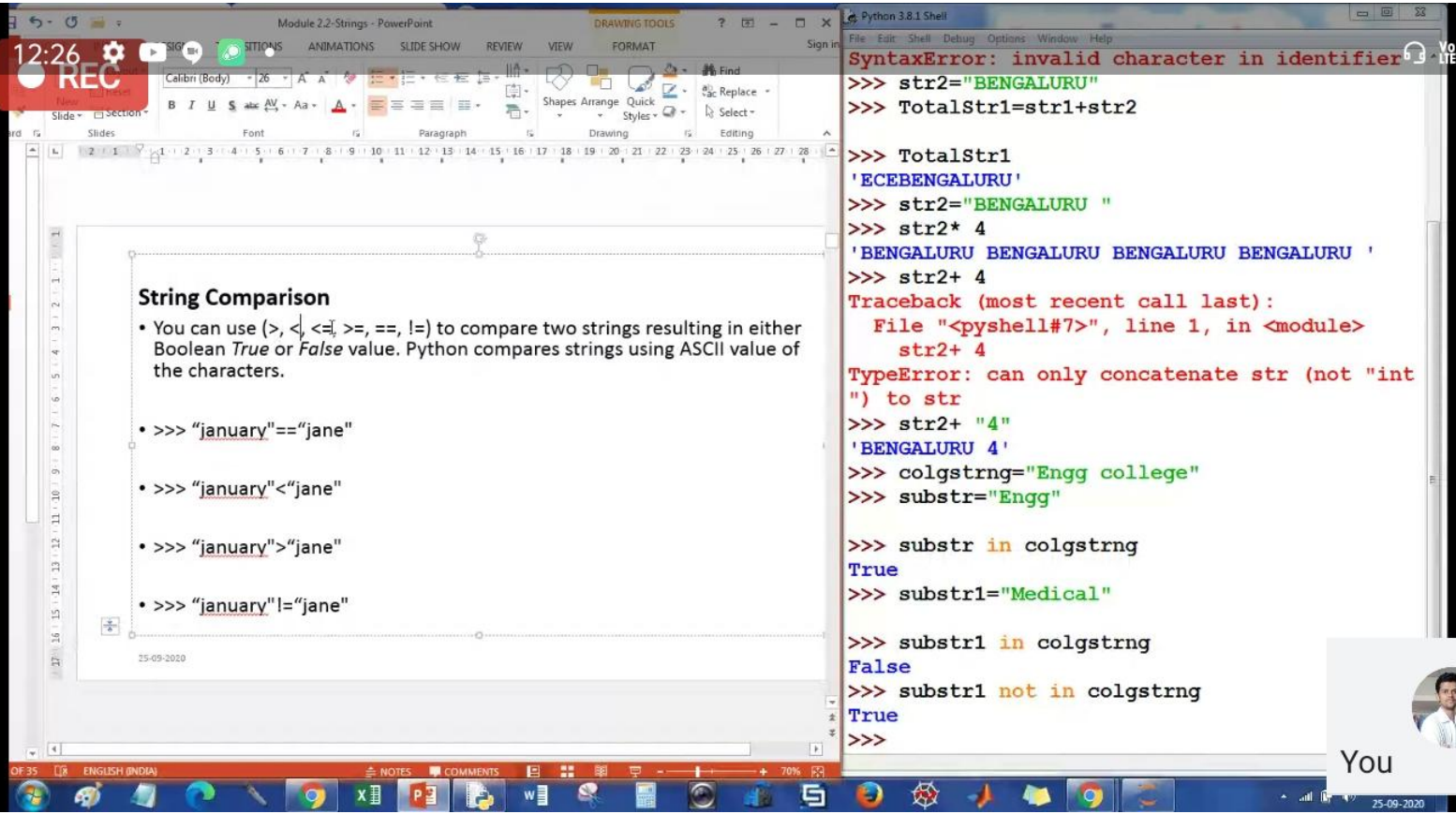
# Basic String Operations

REC

```
>>> str1="ECE"
>>> str2="BENGALURU"
```

- `>>> TotalStr1=str1+str2`
- `>>>TotalStr2="ECE "+"BENGALURU"`
- `>>> Totalstr3=560078+ "PIN"`
- `>>> Totalstr4=str(560078)+"PIN"`
- `>>> Totalstr4="560078"+"PIN"`
- `>>> Totalstr5="BENGALURU"*4`

You


12:26
REC
Module 2.2-Strings - PowerPoint
String Comparison
• You can use (>, <, <=, >=, ==, !=) to compare two strings resulting in either Boolean True or False value. Python compares strings using ASCII value of the characters.
• >>> “january"==“jane"
• >>> “january"<“jane"
• >>> “january">“jane"
• >>> “january"!=“jane"
25-09-2020
Python 3.8.1 Shell
SyntaxError: invalid character in identifier
>>> str2="BENGALURU"
>>> TotalStr1=str1+str2
>>> TotalStr1
'ECEBENGALURU'
>>> str2="BENGALURU "
>>> str2* 4
'BENGALURU BENGALURU BENGALURU BENGALURU '
>>> str2+ 4
Traceback (most recent call last):
File "<pyshell#7>", line 1, in <module>
str2+ 4
TypeError: can only concatenate str (not "int") to str
>>> str2+ "4"
'BENGALURU 4'
>>> colgstrng="Engg college"
>>> substr="Engg"
>>> substr in colgstrng
True
>>> substr1="Medical"
>>> substr1 in colgstrng
False
>>> substr1 not in colgstrng
True
>>>
You

REC

Built-In Functions Used on Strings

| Built-In Functions | Description |
|---|---|
| len() | The *len()* function calculates the number of characters in a string. The white space characters are also counted. |
| max() | The *max()* function returns a character having highest ASCII value. |
| min() | The *min()* function returns character having lowest ASCII value. |

```
>>> Count=len(“bengaluru")
>>> Count1=len(“place mysuru")
>>> max("bengaluru")
>>> min("bengaluru")
```

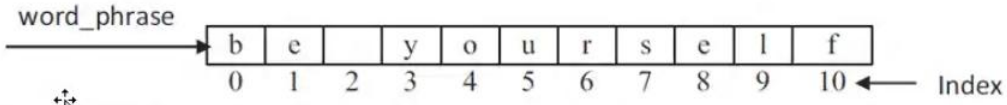
Characters with highest and lowest ASCII value are calculated

You

REC

# Accessing Characters in String by Index Number

- The index breakdown for the string *"be yourself"* assigned to *word_phrase* string variable is shown below.




The syntax for accessing an individual character in a string is as shown below.

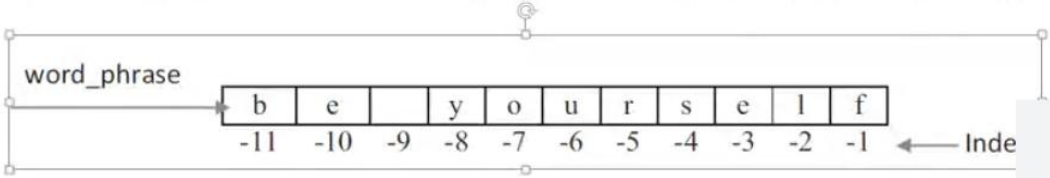
***string_name[index]***

```
>>> word_phrase[1]
'e'
```

The last character in the string is referenced by an index value which is the **(size of the string – 1)** or **(len(string) – 1)**

The negative index breakdown for the string *"be yourself"* assigned to *word_phrase* string variable is shown below.

```
>>> word_phrase[-2]
'l'
```






You

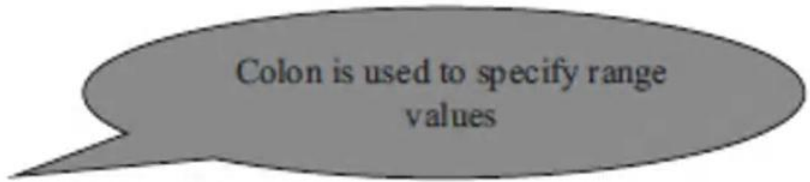
REC

## • String Slicing and Joining

The "slice" syntax is a handy way to refer to sub-parts of sequence of characters within an original string.

The syntax for string slicing is,

```
>>>ClgDept="ECE Dept Bengaluru"
>>> ClgDept[4:15:2]
>>> ClgDept[::2]
>>> ClgDept[::3]
>>>ClgDept[-9:-2]
>>>ClgDept[2:]
>>>ClgDept[:5]
>>> ClgDept[:]
>>>ClgDept[5:50]
>>>ClgDept[5:5]
```




*string_name[start:end[:step]]*



You

Apps myDSI.org Brahma Kumaris https://onlinecours... Communications Sy... Modeling the Prop... YouTube ecds_2017_21 - Go... Knimbus - Your eLi... International Journ... Sci-Hub: removing...

REC

jupyter StringsPrograms Last Checkpoint: an hour ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [15]:
```
Enterstr = input("Enter string: ")
if Enterstr == Enterstr[::-1]:
    print(f" palindrome")
else:
    print(f" Not a palindrome")
print(Enterstr, Enterstr[::-1])

```

```
Enter string: Bengaluru
 Not a palindrome
Bengaluru urulagneB
```

In [4]:
```
Example = "google"
index = 0
print(f"In the string '{Example}'")
for i in Example:
    print(f"Character '{i}' has an index value of {index}")
    index += 1
```

```
In the string 'google'
Character 'g' has an index value of 0
Character 'o' has an index value of 1
Character 'o' has an index value of 2
Character 'g' has an index value of 3
Character 'l' has an index value of 4
Character 'e' has an index value of 5
```

In [5]:
```
string_1="Jasmine"
string_2="Jasan"
for letter in string_1:
    if letter in string_2:
        print(f"Character '{letter}' is found in both the strings")

```

```
Character 'J' is found in both the strings
```

You

PM 12:35

## Joining Strings Using *join()* Method

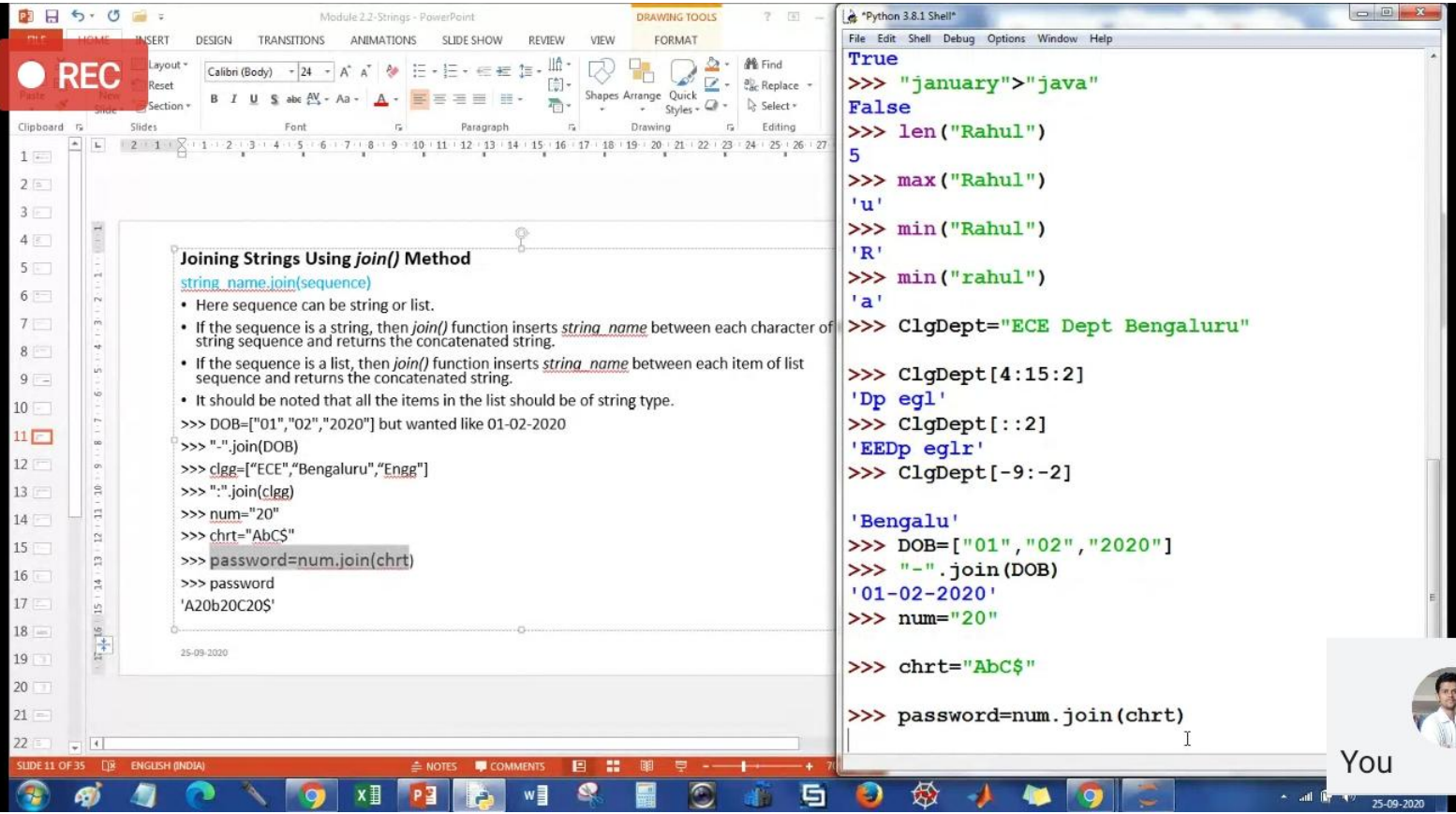
REC

string_name.join(sequence)

- Here sequence can be string or list.
- If the sequence is a string, then *join()* function inserts *string_name* between each character of the string sequence and returns the concatenated string.
- If the sequence is a list, then *join()* function inserts *string_name* between each item of list sequence and returns the concatenated string.
- It should be noted that all the items in the list should be of string type.

```
>>> DOB=["01","02","2020"] but wanted like 01-02-2020
>>> "-".join(DOB)
>>> clgg=["ECE","Bengaluru","Engg"]
>>> ":".join(clgg)
>>> num="20"
>>> chrt="AbC$"
>>> password=num.join(chrt)
>>> password
'A20b20C20$'
```

You


REC
Module 2.2-Strings - PowerPoint
DRAWING TOOLS
Joining Strings Using join() Method
string_name.join(sequence)
Here sequence can be string or list.
If the sequence is a string, then join() function inserts string_name between each character of string sequence and returns the concatenated string.
If the sequence is a list, then join() function inserts string_name between each item of list sequence and returns the concatenated string.
It should be noted that all the items in the list should be of string type.
>>> DOB=["01","02","2020"] but wanted like 01-02-2020
>>> "-".join(DOB)
>>> clgg=["ECE","Bengaluru","Engg"]
>>> ":".join(clgg)
>>> num="20"
>>> chrt="AbC$"
>>> password=num.join(chrt)
>>> password
'A20b20C20$'
25-09-2020
SLIDE 11 OF 35
ENGLISH (INDIA)
NOTES
COMMENTS
"Python 3.8.1 Shell"
File Edit Shell Debug Options Window Help
True
>>> "january">"java"
False
>>> len("Rahul")
5
>>> max("Rahul")
'u'
>>> min("Rahul")
'R'
>>> min("rahul")
'a'
>>> ClgDept="ECE Dept Bengaluru"
>>> ClgDept[4:15:2]
'Dp egl'
>>> ClgDept[::2]
'EEDp eglr'
>>> ClgDept[-9:-2]
'Bengalu'
>>> DOB=["01","02","2020"]
>>> "-".join(DOB)
'01-02-2020'
>>> num="20"
>>> chrt="AbC$"
>>> password=num.join(chrt)
You
25-09-2020

REC

## Split Strings Using *split()* Method

- The *split()* method returns a list of string items by breaking up the string using the delimiter
- The syntax of *split()* method is, ***string_name.split([separator [, maxsplit]])***

```
>>> datestr="01-02-2020"
>>> datestr.split("-")
['01', '02', '2020']
>>> datestr="01 02 2020"
>>> datestr.split(" ")
```

- ['01', '02', '2020']

```
>>> datestr.split()
```

- ['01', '02', '2020']

You

REC

## • Strings Are Immutable

- As strings are immutable, it cannot be modified. The characters in a string cannot be changed once a string value is assigned to string variable. However, you can assign different string values to the same string variable.

```
>>> clg1="bengaluru"
>>> id(clg1)
18074304
>>> clg1="engineering"
>>> id(clg1)
18080224
```

You

REC

## String Traversing

- Since the string is a sequence of characters, each of these characters can be traversed using the *for* loop.
- **Program to Demonstrate String Traversing Using the *for* Loop**

```
Example = "google"
index = 0
print(f"In the string '{Example}'")
for i in Example:
    print(f"Character '{i}' has an index value of {index}")
    index += 1
```

You

```
    print(f" Not a palindrome")
print(Enterstr, Enterstr[::-1])

```

```
Enter string: madam
 palindrome
madam madam
```

In [17]:

```
Example = "Bengaluru"
index = 0
print(f"In the string '{Example}'")
for i in Example:
    print(f"Character '{i}' has an index value of {index}")
    index += 1
```

```
In the string 'Bengaluru'
Character 'B' has an index value of 0
Character 'e' has an index value of 1
Character 'n' has an index value of 2
Character 'g' has an index value of 3
Character 'a' has an index value of 4
Character 'l' has an index value of 5
Character 'u' has an index value of 6
Character 'r' has an index value of 7
Character 'u' has an index value of 8
```

In [5]:

```
string_1="Jasmine"
string_2="Jasan"
for letter in string_1:
    if letter in string_2:
        print(f"Character '{letter}' is found in both the strings")

```

```
Character 'J' is found in both the strings
```

You

REC

- **Program to Print the Characters Which Are Common in Two Strings**

```
string_1="Jasmine"
string_2="Jasan"
for letter in string_1:
    if letter in string_2:
        print(f"Character '{letter}' is found in both the strings")
```

25-09-2020

You

REC

- **Write Python Program to Count the Total Number of Vowels, Consonants and Blanks in a String**

```
user_string = input("Enter a string: ")
vowels,consonants,blanks = 0,0,0
for each_ch in user_string:
    if(each_ch=='a' or each_ch=='e' or each_ch== 'i' or each_ch== 'o' or
      each_ch == 'u'):
        vowels += 1
    elif "a" < each_ch < "z":
        consonants += 1
    elif each_ch == " ":
        blanks += 1
print(f"Total Vowels in user entered string is {vowels}")
print(f"Total Consonants in user entered string is {consonants}")
print(f"Total Blanks in user entered string is {blanks}")
```

25-09-2020

You

```
print(f"Character '{letter}' is found in both the strings")
```

```
Character 'J' is found in both the strings
Character 'a' is found in both the strings
Character 's' is found in both the strings
Character 'n' is found in both the strings
```

In [8]:

```
#Write Python Program to Count the Total Number of Vowels,Consonants and Blanks in a String
user_string = input("Enter a string: ")
vowels,consonants,blanks = 0,0,0
for each_ch in user_string:
    if(each_ch=='a' or each_ch=='e' or each_ch== 'i' or each_ch== 'o' or
       each_ch == 'u'):
        vowels += 1
    elif "a" < each_ch < "z":
        consonants += 1
    elif each_ch == " ":
        blanks += 1
print(f"Total Vowels in user entered string is {vowels}")
print(f"Total Consonants in user entered string is {consonants}")
print(f"Total Blanks in user entered string is {blanks}")
```

```
Enter a string: This is a computer
Total Vowels in user entered string is 6
Total Consonants in user entered string is 8
Total Blanks in user entered string is 3
```

In [9]:

```
user_string = input("Enter a string: ")
count_ch = 0
for each_ch in user_string:
    count_ch += 1
print(f"The length of user entered string is {count_ch} ")
```

REC

- **Write Python Program to Calculate the Length of a String Without Using Built-In *len()* Function**

```
user_string = input("Enter a string: ")
count_ch = 0
for each_ch in user_string:
    count_ch += 1
print(f"The length of user entered string is {count_ch} ")
#using function
print("The length of user entered string using function",len(user_string))
```

25-09-2020

You

REC

- String Methods

>>> dir(str)

- ['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__',
- '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__',
- '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__
- mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
- '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize',
- 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format',
- 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower',
- 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
- 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
- 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

25-09-2020

You

- String methods like *capitalize()*, *lower()*, *upper()*, *swapcase()*, *title()* and *count()* are used for conversion purpose.
- String methods like *islower()*, *isupper()*, *isdecimal()*, *isdigit()*, *isnumeric()*, *isalpha()* and *isalnum()* are used for comparing strings.
- Some of the string methods used for padding are *rjust()*, *ljust()*, *zfill()* and *center()*.
- The string method *find()* is used to find substring in an existing string. You can use string methods like *replace()*, *join()*, *split()* and *splitlines()* to replace a string in Python.

12:52 REC

# Program using Sort()

```
# List of Integers
numbers = [1, 3, 4, 2,5,9,6,0]
numbers.sort()
print(numbers)

# List of Floating point numbers
decimalnumber = [2.01, 2.00, 3.67, 3.28, 1.68]
decimalnumber.sort()
print(decimalnumber)

# List of strings
words = ["ECE", "DSCE", "Bengaluru"]
words.sort()
print(words)
```

25-09-2020

You

```
Enter a string: This is a computer
Total Vowels in user entered string is 6
Total Consonants in user entered string is 8
Total Blanks in user entered string is 3
```

In [9]:

```python
user_string = input("Enter a string: ")
count_ch = 0
for each_ch in user_string:
    count_ch += 1
print(f"The length of user entered string is {count_ch} ")
#using function
print("The length of user entered string using function",len(user_string))


```

```
Enter a string: Computer
The length of user entered string is 8
The length of user entered string using function 8
```

In [10]:

```python
# List of Integers
numbers = [1, 3, 4, 2,5,9,6,0]
numbers.sort()
print(numbers)

# List of Floating point numbers
decimalnumber = [2.01, 2.00, 3.67, 3.28, 1.68]
decimalnumber.sort()
print(decimalnumber)

```

```
#Program 5.7: Write Python Program to Convert Uppercase Letters to
#Lowercase and Vice Versa
def case_conversion(user_string):
    convert_case = str()
    for each_char in user_string:
        if each_char.isupper():
            convert_case += each_char.lower()
        else:
            convert_case += each_char.upper()
    print(f"The modified string is {convert_case}")

input_string = input("Enter a string ")
case_conversion(input_string)
```

25-09-2020

You

```
['Bengaluru', 'DSCE', 'ECE']
```

12:53 REC

In [11]:

```python
#Program 5.7: Write Python Program to Convert Uppercase Letters to
#Lowercase and Vice Versa
def case_conversion(user_string):
    convert_case = str()
    for each_char in user_string:
        if each_char.isupper():
            convert_case += each_char.lower()
        else:
            convert_case += each_char.upper()
    print(f"The modified string is {convert_case}")

input_string = input("Enter a string ")
case_conversion(input_string)
```

```
Enter a string ComPuter
The modified string is cOMpUTER
```

In [13]:

```python
#Program 5.9: Write Python Program to Count the Occurrence of User-Entered
#Words in a Sentence
def count_word(word_occurrence, user_string):
    word_count = 0
    for each_word in user_string.split():
        if each_word == word_occurrence:
            word_count += 1
    print(f"The word '{word_occurrence}' has occurred {word_count} times")
```

You

```
Enter a word to count its occurrence is
The word 'is' has occurred 2 times
```

REC

In [14]:

```python
#Program 5.6: Write Python Program That Accepts a Sentence and Calculate
#the Number of Words, Digits, Uppercase Letters and Lowercase Letters
def string_processing(user_string):
    word_count = 0
    digit_count = 0
    upper_case_count = 0
    lower_case_count = 0
    for each_char in user_string:
        if each_char.isdigit():
            digit_count += 1
        elif each_char.isspace():
            word_count += 1
        elif each_char.isupper():
            upper_case_count += 1
        elif each_char.islower():
            lower_case_count += 1
        else:
            pass
    print(f"Number of digits in sentence is {digit_count}")
    print(f"Number of words in sentence is {word_count + 1}")
    print(f"Number of upper case letters in sentence is {upper_case_count}")
    print(f"Number of lower case letters in sentence is {lower_case_count}")

user_input = input("Enter a sentence ")
string_processing(user_input)


```

```
Enter a sentence this is laptop computer
Number of digits in sentence is 0
Number of words in sentence is 4
Number of upper case letters in sentence is 0
Number of lower case letters in sentence is 20
```

You

REC

- **Formatting Strings**
- *%-formatting* - **"f-strings"**
- *str.format()*.

- **Escape Sequences**
- Escape Sequences are a combination of a backslash (\) followed by either a letter or a combination of letters and digits. Escape sequences are also called as control sequences.
- The backslash (\) character is used to escape the meaning of characters that follow it by substituting their special meaning with an alternate interpretation. So, all escape sequences consist of two or more characters.

25-09-2020

You

REC

List of Escape Sequences

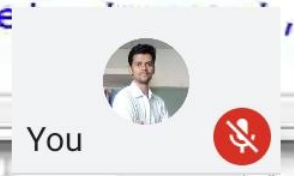
| Escape Sequence | Meaning |
|---|---|
| \ | Break a Line into Multiple lines while ensuring the continuation of the line |
| \\ | Inserts a Backslash character in the string |
| \' | Inserts a Single Quote character in the string |
| \" | Inserts a Double Quote character in the string |
| \n | Inserts a New Line in the string |
| \t | Inserts a Tab in the string |
| \r | Inserts a Carriage Return in the string |
| \b | Inserts a Backspace in the string |
| \u | Inserts a Unicode character in the string |
| \0*oo* | Inserts a character in the string based on its Octal value |
| \x*hh* | Inserts a character in the string based on its Hex value |

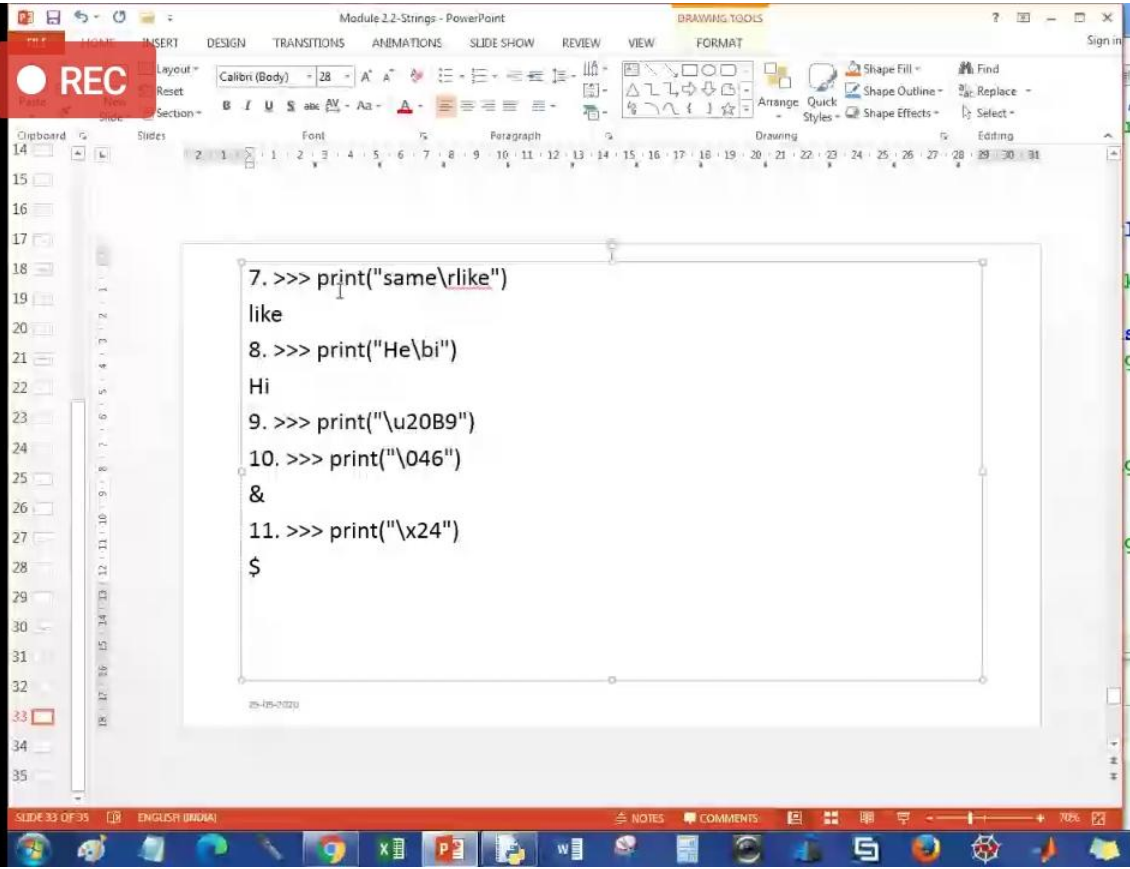
You

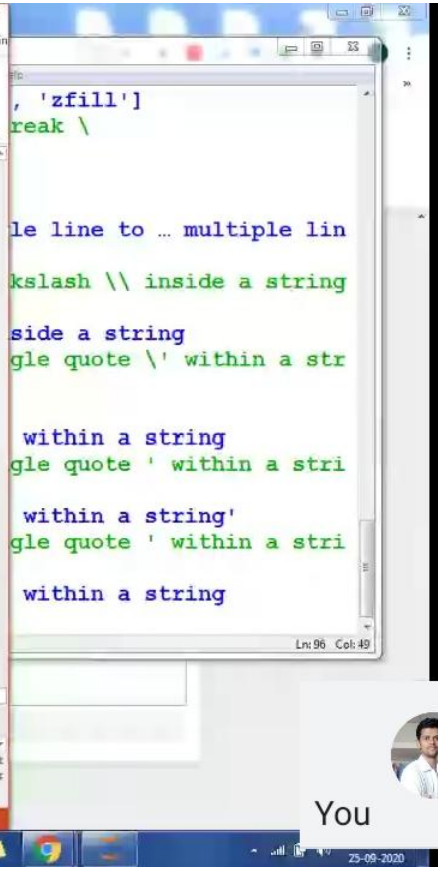
REC

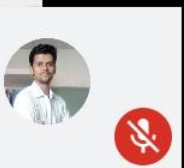
```
1. >>> print("You can break \
... single line to \
... multiple lines")
You can break single line to multiple lines
2. >>> print('print backslash \\ inside a string ')
print backslash \ inside a string
3. >>> print('print single quote \' within a string')
print single quote ' within a string
4. >>> print("print double quote \" within a string")
print double quote " within a string
5. >>> print("First line \nSecond line")
First line
Second line
6. >>> print("tab\tspacing")
tab spacing
```

```
__', '__reduce_ex__',
'__rmul__', '__setatt
tr__', '__subclasshoo
efold', 'center', 'co
h', 'expandtabs', 'fi
ap', 'index', 'isalnu
, 'isdecimal', 'isdig
lower', 'isnumeric',
, 'istitle', 'isupper
er', 'lstrip', 'maket
lace', 'rfind', 'rind
n', 'rsplit', 'rstrip
, 'startswith', 'stri
'translate
>>>
```




Module 2.2-Strings - PowerPoint
DRAWING TOOLS
7. >>> print("same\rlike")
like
8. >>> print("He\bi")
Hi
9. >>> print("\u20B9")
10. >>> print("\046")
&
11. >>> print("\x24")
$


, 'zfill']
reak \
le line to … multiple lin
kslash \\ inside a string
side a string
gle quote \' within a str
within a string
gle quote ' within a stri
within a string'
gle quote ' within a stri
within a string
Ln: 96 Col: 49


You

REC

## Raw Strings

- A raw string is created by prefixing the character ***r*** to the string. In Python, a raw string ignores all types of formatting within a string including the escape characters.
- 1. >>> print(r"Bible Says, \"Taste and see that the LORD is good; blessed is the man who takes refuge in him.\"")

Bible Says, \"Taste and see that the LORD is good; blessed is the man who takes refuge in him.\"

You

REC

**Unicodes**

```
1. >>> unicode_string = u'A unicode \u018e string \xf1'
2. >>> unicode_string
'A unicode string ñ‘
```

**Summary**

- A string is a sequence of characters.
- To access values through slicing, square brackets are used along with the index.
- Various string operations include conversion, comparing strings, padding, finding a substring in an existing string and replace a string in Python.
- Python strings are immutable which means that once created they cann changed.

You

REC

- 21. >>> "galapagos".upper()
- 'GALAPAGOS'
- 22. >>> "Centennial Light".swapcase()
- 'cENTENNIAL lIGHT'
- 23. >>> "history does repeat".replace("does", "will")
- 'history will repeat'
- 24. >>> quote = " Never Stop Dreaming "
- 25. >>> quote.rstrip()
- ' Never Stop Dreaming'
- 26. >>> quote.lstrip()
- 'Never Stop Dreaming '
- 27. >>> quote.strip()
- 'Never Stop Dreaming'
- 28. >>> 'ab c\n\nde fg\rkl\r\n'.splitlines()
- ['ab c', '', 'de fg', 'kl']
- 29. >>> "scandinavian countries are rich".center(40)
- 'scandinavian countries are rich'

You