Unit II: Computational Thinking and Programming - I
● Strings: introduction, string operations (concatenation, repetition, membership and slicing), traversing a string using loops,
built-in functions/methods–len(), capitalize(), title(), lower(), upper(), count(), find(), index(), endswith(), startswith(), isalnum(), isalpha(), isdigit(), islower(), isupper(), isspace(), lstrip(), rstrip(), strip(), replace(), join(), partition(), split()

Main point

- String is **sequence** created by **enclosing** zero or more UNICODE characters **in** single, double or triple **quote**.
- Each character is string has **unique index**. The index must be an **integer** (positive, zero or negative). The **forward** index (**from left**) starts from **0** and end at **n-1**. The **backward** index (**from right**) starts from **-1** and end at **–n.**
- Syntax to **access** character from string
    - **Stringname[index]**
- If we give **index value out of** this **range** then we get an **IndexError**
- If we give **float value** as **index** we get a **TypeError**
- A string is an **immutable:-** string **cannot be changed**. Produce **error** if **try to do so.**

String Operation: - Concatenation

- **join** two **strings** using **concatenation** operator plus(**+**)
- syntax: str1/strname1 + str2/strname2 , Example:- **'peace'+ 'ful'** gives **peaceful**
- **string** can be **concatenated with** another **string only.**
- **no changes** made by Concatenation operation **in original string**

String Operation: - Repetition

- **repeat** the **string** using repetition operator **(*)**
- syntax: str1/strname1 * n , Example:- **'Peace' * 2** gives **PeacePeace**
- **no changes** made by repletion operation **in original string**

String Operation: - Membership

- membership operators '**in**' and '**not in'** returns **True or False**
- syntax: substr in str or substr not in str, Example:- **'vers' in 'Traversing'** gives **True**

String Operation: - Slicing

- **access some part** of a string by method called **slicing**
- syntax:- str1[**start: end: step**] Example **name[2:6:2]**
- **start , end , step** must be **integer (positive, zero or negative)**
- **default** value of **start is 0** and **end is n** where **n is length** of string and **step is 1** if not provided
- **default** value of **start is -1** and **end is –(n+1)** where **n is length** of string if **step =-1**

Traversing a String

- **access each character** of a string by using **for or while loop**
- Syntax:- for ch in string_name:
- starts from the first character of the string and automatically ends when the last character is accessed

Built-in Function
len():- Returns the length of the given string, syntax:- len(string_name)
Example len(“my name”) :- return 7 len("hi \n hello"):- return 10 as \n is a escape sequence that take only one space so \n take only one index.

```
len("hi \n hello")
    012 3456789
```

String Methods

Method	Description	Example

endswith(**suffix**[, start[, end]])	Return **True** if test **string ends** with the specified **suffix**, False otherwise.
	suffix is **mandatory** and **start** and **end** is **optional**
	Test string beginning at start index and end before given value of end. (character at **end index** is **not included** for comparison)
	Suffix can be **string** or **tuple of string** to try
		s="We, the people of INDIA"
		>>> s.endswith("INDIA")
		True
		>>> s.endswith("ple", 8, 14)
		True
		>>> s.endswith(("NEPAL", "BHUTAN", "INDIA"))
		True

startswith (prefix[, start[, end]])	Return **True** if test **string starts** with the specified **prefix**, False otherwise.
	prefix is **mandatory** and **start** and **end** is **optional**
	prefix can be **string** or **tuple of string** to try
		>>> s.startswith("We")
		True
		>>> s.startswith("peo",8,15)
		True
		>>> s.startswith(("I", "We", "You"))
		True

isalnum()	Returns **True** if the string is **non-empty** and **all characters** in the string are **alphabet or numeric**, **otherwise** Return **False**	>>> s.isalnum()
		False
		>>> k=""
		>>> k.isalnum()
		False

isalpha()	Returns **True** if the string is **non-empty** and **all characters** in the string are **alphabet**, **otherwise** Return **False**	>>> s.isalpha()
		False
		>>> k.isalpha()
		False

isdigit()	Returns **True** if the string is **non-empty** and **all characters** in the string are **digits** ,**otherwise** Return **False**	>>> s.isdigit()
		False
		>>> k.isdigit()
		False

islower()	Returns **True** if the string is **non-empty** and if **all alphabet characters** in the string are **lowercase alphabet**.
	there is at **least one lowercase alphabet** character in the string , **rest** can be **non-alphabet** characters
		>>> s.islower()
		False
		>>> k.islower()
		False
		>>>m="j76"
		>>> m.islower()

		True

isupper()	Returns **True** if the string is **non-empty** and if **all alphabet characters** in the string are **uppercase alphabet**.
	there is at least **one uppercase alphabet** character in the string **, rest** can be **non-alphabet** characters
		>>> s.isupper()
		False
		>>> k.islower()
		False
		>>> m.islower()
		False

isspace()	Returns **True** if the string is **non-empty** and **all characters** are **white spaces** (blank, tab, newline, carriage return)
	>>> l="\n"
	>>>p= “ ”
		>>> s.isspace()
		False
		>>> k.isspace()
		False
		>>> l.isspace()
		True
		>>> p.isspace()
		True

istitle()	Returns **True** if the string is **non-empty** and title case, i.e., **upper- and title-case** characters (first letter) of **each word** may only **follow non alphabet characters and lowercase** characters
	>>> k="Riya Has Locker **7a#**"
	>>> k.istitle()
	False
		>>> n="Riya Has Locker 578"
		>>> n.istitle()
		True
		>>> m="Amit Has Locker **7A#**"
		>>> m.istitle()
		True

title()	**Returns** the **string** with **first letter** of **every word** in the string in **uppercase** and **rest in lowercase**	>>> k="ana has Locker 7ab#"
		>>> k.title()
		'Ana Has Locker 7Ab#'

lower()	**Returns** the **string** with **all uppercase** letters **converted** to **lowercase**	>>> k="Ana has Locker 7ab#"
		>>> k.lower()
		'ana has locker 7ab#'

upper()	**Returns** the **string** with **all lowercase** letters **converted** to **uppercase**	>>> k.upper() 'ANA HAS LOCKER 7AB#'

count(sub [, start [, end]])	Returns **number of times** substring sub occurs in the given string[start:end]
	sub is **mandatory** and **start** and **end** is **optional**
		>>> str1 = "Hello World! Hello"
		>>> str1.count("Hello")
		2
		>>> str1.count("Hello",10,20)
		1

find(sub[, start[, end]])	**Returns** the **index** of **first occurrence** of substring **sub** in the given string. Return **-1** if sub is **not present** in given string	>>> str1.find("World")
		6
		>>> str1.find("World",10,15)
		-1

index(str, start, end)	**Returns** the **index** of **first occurrence** of substring **sub** in the given string**. raises an exception** (**ValueError**: substring not found) if the **substring is not present** in the given string	>>> str1.index("World")
		6
		>>> str1.index("World",10,15)
		ValueError: substring not found

lstrip(chars=None)	Return a **copy of the string** after removing **leading whitespace.**
	If **chars are given**, start **removing** character from **left** side of given string **if** that **character** present **in chars**. **If** get a **character** that is **not** present **in char** then **stop**.
		>>> x="*.*.*..*.hi*.*.*."
		>>> x.lstrip("*.")
		'hi*.*.*.'
		>>> k=" hello "
		>>> k.lstrip()
		'hello '

rstrip(chars=None)	Return a **copy of the string** after removing **trailing whitespace.**
	If **chars are given**, start **removing** character from **right** side of given string **if** that **character** present **in chars**. **If** get a **character** that is **not** present **in chars** then **stop.**
		>>> s.rstrip(".*")
		'*.*.*..*.hi'
		>>> k.rstrip()
		' hello'

strip(chars=None)	Return a **copy of the string** after removing **leading and trailing whitespace.**
	If **chars are given**, start **removing** character from **right** side of given string **if** that **character** present **in chars**. **If** get a **character** that is **not** present **in chars** then **stop.**
		>>> x.strip("*.")
		'hi'
		>>> k.strip()
		'hello'

replace(old, new, count=-1)	Return a **copy of the string** with all occurrences of **old substring replaced** by new substring.
	count is **Maximum number** of **occurrences to replace,** if **not given replace all** occurrence
		>>>m='ho ho hore haiya nikali khewaiya ho ho haiya'
		>>> m.replace("ho","yo")
		'**yo yo yo**re haiya nikali khewaiya **yo yo** haiya'
		>>> m.replace("ho","yo",2)
		'**yo yo** hore haiya nikali khewaiya ho ho haiya'

join(iterable)	Return a **string** in which the **string** whose method is called is **inserted in between** each element of given **string / list of string /tuple of string.**	>>> "*".join("HAPPY")
		'H*A*P*P*Y'
		>>> "#".join(["ANU","RINA"])
		'ANU#RINA'

partition(sep)	Returns a **3-tuple** containing the **part before** the separator, the **separator itself**, and the **part after** it, if **sep** is **present** in string **otherwise** Return a **3-tuple** containing the **original string** and **two empty strings**	>>> s.partition("people")
		('**we, the** ', 'people', ' **of INDIA**')
		>>> s.partition("in")
		('**we, the people of INDIA**', '', '')

split(sep=None, maxsplit=-1)	Returns a **list of words** delimited by the specified substring. None (the default value) means **split** according to **whitespace**, and **discard empty strings** from the result.
	maxsplit :- Maximum number of splits to do
		>>> s.split()
		['**we,**', 'the', '**people**', 'of', '**INDIA**']
		>>> s.split(" ",2)
		['**we,**', 'the', '**people of INDIA**']

Question based on string

1. Select the correct output of the code :

```
S="Amrit Mahotsav @ 75"
A=S.split(" ",2)
print(A)
```

(a) ('Amrit', 'Mahotsav', '@', '75')
(b) ['Amrit', 'Mahotsav', '@ 75']
(c) ('Amrit', 'Mahotsav', '@ 75')
(d) ['Amrit', 'Mahotsav', '@', '75'

2. Which of the following statement(s) would give an error after executing the following code ?

```
S="Happy" # Statement 1
print(S*2) # Statement 2
S+="Independence" # Statement 3
S.append("Day") # Statement 4
print(S) # Statement 5
```

(a) Statement 2 (b) Statement 3 (c) Statement 4 (d) Statement 3 and 4

3 (a) Given is a Python string declaration : NAME = "Learning Python is Fun"
Write the output of : print(NAME[-5:-10:-1])

4. (i) Predict the output of the code given below :

```
text="LearningCS"
L=len(text)
ntext=""
for i in range (0,L):
  if text[i].islower():
    ntext=ntext+text[i].upper()
  elif text [i].isalnum():
    ntext=ntext+text[i-1]
  else:
    ntext=ntext+'&&'
print(ntext)
```

2023

1. Select the correct output of the code :

```
S= "Amrit Mahotsav @ 75"
A=S.partition (" ")
print (a)
```

(a) ('Amrit Mahotsav','@','75')
(b) ['Amrit','Mahotsav','@','75']
(c) ('Amrit', 'Mahotsav @ 75')
(d) ('Amrit','' , 'Mahotsav @ 75')

2. Predict the output of the code given below :

```
def makenew(mystr) :
     newstr=""
     count=0
     for i in mystr :
          if count%2!=0:
               newstr=newstr+str(count)
          else :
               if  i.lower():
                    newstr=newstr+i.upper()
               else:
                    newstr=newstr+i
          count+=1
     print(newstr)
makenew("No@1")
```

2019

1. Find and write the output of the following python code

```
Msg1="WeLcOME"
Msg2="GUeSTs"
Msg3=""
for I in range(0,len(Msg2)+1):
    if Msg1[I]>="A" and Msg1[I]<="M":
        Msg3=Msg3+Msg1[I]
    elif Msg1[I]>="N" and Msg1[I]<="Z":
        Msg3=Msg3+Msg2[I]
    else:
        Msg3=Msg3+"*"
print Msg3
```

2019

1. Find and write the output of the following Python code :

```
Text1="AISSCE 2018"
Text2=""
I=0
while I<len(Text1):
  if Text1[I]>="0" and Text1[I]<="9":
    Val = int(Text1[I])
    Val = Val + 1
    Text2=Text2 + str(Val)
  elif Text1[I]>="A" and Text1[I] <="Z":
    Text2=Text2 + (Text1[I+1])
  else:
   Text2=Text2 + "*"
  I=I+1
print Text2
```

SQP-2024

1. Select the correct output of the code:

```
s = "Python is fun"
l = s.split()
s_new = "-".join([l[0].upper(), l[1], l[2].capitalize()])
print(s_new)
```

a. PYTHON-IS-Fun b. PYTHON-is-Fun c. Python-is-fun d. PYTHON-Is -Fun

2. Consider the statements given below and then choose the correct output from the given options:

```
pride="#G20 Presidency"
print(pride[-2:2:-2])
```

a. ndsr b. ceieP0 c. ceieP d. yndsr

3. Write a function, lenWords(STRING), that takes a string as an argument and returns a tuple containing length of each word of a string. For example, if the string is "Come let us have some fun", the tuple will have (4, 3, 2, 4, 4, 3)

4. Write the Python statement for each of the following tasks using BUILT-IN functions/methods only:

(ii) To check whether a string named, message ends with a full stop / period or not.

5. Predict the output of the Python code given below:

```
Text1="IND-23"
Text2=""
I=0
while I<len(Text1):
    if Text1[I]>="0" and Text1[I]<="9":
        Val = int(Text1[I])
        Val = Val + 1
        Text2=Text2 + str(Val)
    elif Text1[I]>="A" and Text1[I]<="Z":
        Text2=Text2 + (Text1[I+1])
    else:
        Text2=Text2 + "*"
    I+=1
print(Text2)
```

SQP-2023

1. Select the correct output of the code:

```
a = "Year 2022 at All the best"
a = a.split('2')
b = a[0] + ". " + a[1] + ". " + a[3]
print (b)
```

(a) Year . 0. at All the best (b) Year 0. at All the best
(c) Year . 022. at All the best (d) Year . 0. at all the best

2. Which of the following statement(s) would give an error after executing the following code?

```
S="Welcome to class XII" # Statement 1
print(S) # Statement 2
S="Thank you" # Statement 3
S[0]= '@' # Statement 4
S=S+"Thank you" # Statement 5
```

(a) Statement 3 (b) Statement 4 (c) Statement 5 (d) Statement 4 and 5

3. (a) Given is a Python string declaration: myexam="@@CBSE Examination 2022@@"
Write the output of: print(myexam[::-2])

(a) Predict the output of the code given below:

```
s="welcome2cs"
n = len(s)
m=""
for i in range(0, n):
    if (s[i] >= 'a' and s[i] <= 'm'):
        m =   m +s[i].upper()
    elif (s[i] >= 'n' and s[i] <= 'z'):
        m = m +s[i-1]
    elif (s[i].isupper()):
        m = m + s[i].lower()
    else:
        m = m +'&'
print(m)
```

SQP-2022
1. If the following code is executed, what will be the output of the following code?
name="ComputerSciencewithPython"
print(name[3:10])

2. Find the output of the following code:

```
Name="PythoN3.1"
R=""
for x in range(len(Name)):
    if Name[x].isupper():
       R=R+Name[x].lower()
    elif Name[x].islower():
        R=R+Name[x].upper()
    elif Name[x].isdigit():
        R=R+Name[x-1]
    else:
        R=R+"#"
print(R)
```

a. pYTHOn##@ b. pYTHOnN#@ c. pYTHOn#@ d. pYTHOnN@#

SQP-2021
1. Find and write the output of the following Python code:

```
def Display(str):
    m=""
    for i in range(0,len(str)):
        if(str[i].isupper()):
            m=m+str[i].lower()
        elif str[i].islower():
            m=m+str[i].upper()
        else:
            if i%2==0:
                m=m+str[i-1]
            else:
                m=m+"#"
    print(m)
Display('Fun@Python3.0')
```

SQP-2020

1. Find and write the output of the following python code:

```
def fun(s):
    k=len(s)
    m=" "
    for i in range(0,k):
        if(s[i].isupper()):
            m=m+s[i].lower()
        elif s[i].isalpha():
            m=m+s[i].upper()
        else:
            m=m+'bb'
    print(m)

fun('school2@com')
```

2. Find and write the output of the following python code:

```
x = "abcdef"
i = "a"
while i in x:
    print(i, end = " ")
```