

T-2_Sem-3 || Python

Chap.-4 Immutable Data Structure (String && Tuple)

String : It's Case-sensitive.

```
In [1]: 1 multi = ""h
        2     i
        3 ee""
        4
        5 print(multi)
        6 print(len(multi)) # 10
```

```
h
    i
ee
10
```

```
In [ ]: 1 s = "Hello World"
        2 print(len(s)) # 11
        3 print(s[9]) # l
        4 print(s.count('o')) # 2
```

2 type of indexing.

1. Positive (Start from front with 0)
2. Negative (Start from Back with -1)

```
In [ ]: 1 s = "Hello World"
        2 print(len(s)) # 11
        3 print(s[-5]) # W
        4 print(s[-10]) # e
```

Slicing (String, List, Tuple) :

s[Start : End : Jump]

```
In [ ]: 1 s = "Slicing"
        2 print(len(s))
        3 print(s[0:7:2])
        4 # Give One+ index
        5 # default value of end = Length
        6 # default value of jump = 1
```

```
In [ ]: 1 s = "LJK University of Ahmedabad"
2 print(len(s)) # 27
3 print(s[4:14]) # University
4 print(s[-23:-13]) # University
5 print(s[18:25:2]) # Amdb
6 print(s[-9:-2:2]) # Amdb
7 print(s[2: :-1]) # KJL
8 print(s[-25::-1]) # KJL
```

P.b. 306 - Check a Entered String is Palindrome or Not.

```
In [ ]: 1 s = input("Enter String : ")
2 c = s[len(s)::-1]
3 if(c == s):
4     print("Palindrome")
5 else:
6     print("Not a Palindrome")
```

```
In [ ]: 1 s = input("Enter String : ")
2 length = 0
3 for i in s:
4     length += 1
5
6 print(length)
```

```
In [ ]: 1 s = "Hello"
2 i = 2
3 new = "K"
4 print(s[0:i]+new+s[3:5]) # HeKLo
5
6 # we can also use [i+i:5] instead of [3:5]
```

P.b. : 312 (first, middle, last)

```
In [ ]: 1 s = input("Enter String : ") # romil
2 f = s[0] # r
3 m = s[len(s)//2] # m
4 l = s[-1] # l
5
6 print(f + m + l) # rml
```

String Methods :

1. Lower()
2. upper()
3. capitalize()

```
In [ ]: 1 s = "hELLo"
2 print(s) # hELLo
3 print(s.upper()) # HELLO
4 print(s.lower()) # hello
5 s = s.capitalize()
6 print(s) # Hello
7
8 ns = "hELLo World tHis is pY"
9 print(ns.capitalize()) # Hello world this is py
```

```
In [ ]: 1 s = 'hello'
2 print(s.isupper()) # False
3 print(s.islower()) # True
4
5 cs = 'Hello'
6 print(cs.isupper()) # False
7 print(cs.islower()) # False
8
9 astr = 'hello Py'
10 print(astr.isupper()) # False
11 print(astr.islower()) # False
```

```
In [ ]: 1 s = 'abc123'
2 print(s.isalpha()) # False
3 print(s.isnumeric()) # False
4 print(s.isalnum()) # True
5 print(s.isdigit()) # False
```

P.b. : 309 (find how many Upper && Lower words in Str)

```
In [ ]: 1 s = "Hello World"
2 u, l = 0, 0
3 for i in s:
4     if(i.isupper()):
5         u += 1
6     elif(i.islower()):
7         l += 1
8
9 print("Upper :", u) # 2
10 print("Lower :", l) # 8
```

P.b. : 318 (First & Last char. Capitalize)

Output of Split - List

```
In [ ]: 1 s = """This is python code"""
2 n = s.split(" ")
3 print(n) # ['This', 'is', 'python', 'code']
4 new = ""
5 for i in n:
6     if(len(i)==1):
7         i.upper
8     else:
9         new += i[0].upper() + i[1:-1] + i[-1].upper() + " "
10 print(new) # ThiS IS PythoN CodE
11
12 # Short-cuttt (Don't use in Exam 😊)
13 # for i in n:
14 #     print(i.capitalize()[:-1], end="")
15 #     print(i[-1].capitalize(),end=" ")
16 # Output : ThiS IS PythoN CodE
17
18 # split("char to split from", *)
19 # *int value that defines maximum no. of split
```

```
In [ ]: 1 s1 = "Hello World"
2 ans = s1.find('l', 6, 10)
3 ans1 = s1.find('l')
4 a = s1.index('l')
5 print(ans, ans1, a) # 9 2 2
6
7 a = s1.find('x') # it will give -1
8 # a = s1.index('x') # it give an error.
9 print(a)
10
11 print(s1.count('l')) # 3
```

```
In [ ]: 1 s1 = "---,--,rgpytrghon---,--,----"
2 print(s1.strip("-")) # ,--,rgpytrghon---,--,
3 print(s1.strip("-,rg")) # pytrghon
4 print(s1.strip("-",)) # rgpytrghon
5 # removes any white spaces before and after the string.
6
7 print(s1.lstrip("-",)) # rgpytrghon---,--,----
8 print(s1.rstrip("-",)) # ---,--,rgpytrghon
```

```
In [ ]: 1 s = "Hello World"
2 ans = s.translate({108:120, 111:None})
3 # None - remove
4 print(ans) # Hexx Wrxld
```

```
In [ ]: 1 s = "Hello World"
2 tab = s.maketrans("Hl", "hL", "O")
3 print(tab) # {72: 104, 108: 76, 79: None}
4 ans = s.translate(tab)
5 print(ans) # heLLo WorLd
6
7
8 tab1 = s.maketrans("Hl", "aB", "O")
9 print(tab1) # {72: 97, 108: 66, 79: None}
10 ans1 = s.translate(tab1)
11 print(ans1) # heLLo WorLd
```

313. Write a program to find all occurrences of a sub string in a given string by ignoring the case.

```
In [ ]: 1 s = "Hello World"
2 s1 = input("Enter String you want to find occurrences : ").lower()
3 ans = (s.lower()).count(s1)
4 print(ans)
```

314. Write a program to calculate the sum and average of the digits present in a string.

```
In [ ]: 1 s = '123456789'
2 sum = 0
3 digit = 0
4 for i in s:
5     if(i.isnumeric()):
6         sum = sum + int(i)
7         digit += 1
8
9 print(sum) # 45
10 print(sum/digit) # 5.0
```

325. Write a Python program using function to shift the decimal digits n places to the left, wrapping the extra digits around. If shift > the number of digits of n, then reverse the string.

Note: Function will take two parameters:

1. The number
2. How much shift user want Example: Input: n=12345 shift=1 Output: Result=23451 Input: n=12345 shift=3 Output: Result=45123 Input: n=12345 shift=5 Output: Result=12345 Input: n=12345 shift=6 Output: Result=54321

```
In [ ]: 1 st = input("Enter : ")
        2 s = int(input('Enter shift : '))
        3 ans = 0
        4 if len(st)<s:
        5     ans = int(st[::-1])
        6 else:
        7     ans = int(st[s:]+st[:s])
        8
        9 print(ans)
```

326.

```
In [ ]: 1 s = "Hello Pyth@n is 100% easy"
        2 u, l, d = 0, 0, 0
        3 for i in s:
        4     if(i.isupper()):
        5         u += 1
        6     elif(i.islower()):
        7         l += 1
        8     elif(i.isdigit()):
        9         d +=1
        10 print("Upper :", u) # 2
        11 print("Lower :", l) # 14
        12 print("Digit :", d) # 3
```

327. Write a python program to check the validity of a Password.

Primary conditions for password validation:

- 1. Minimum 8 characters.**
- 2. The alphabet must be between [a-z]**
- 3. At least one alphabet should be of Upper Case [A-Z]**
- 4. At least 1 number or digit between [0-9]**

5. At least 1 character from [_ or @ or

]Examples : Input : Ram@_1234Output : ValidPasswordInput : Ramafoab

Output: Invalid Password Explanation: Number is missing Input: Rama#fo9c Output: Invalid Password Explanation: Must consist from _ or @ or \$



In []:

```
1 RED = "\033[0;31m"
2 GREEN = "\033[0;32m"
3 BOLD = "\033[1m"
4 END = "\033[0m"
5 p = input("Enter Password : ")
6 u, d, s = 0, 0, 0
7 if(len(p) >= 8):
8     for i in p:
9         if(i.isupper()):
10             u += 1
11         elif(i.isdigit()):
12             d += 1
13         elif(i == '$' or i == '_' or i == '@'):
14             s += 1
15     if(u and d and s):
16         print(GREEN + BOLD + "Valid Password" + END)
17     else:
18         print("Invalid Password")
19 else:
20     print(RED + BOLD + "Invalid Password" + END)
21
22 # Output :
23 # Enter Password : Romil@1234
24 # Valid Password
```

Tuple : (Repeataion is allowed)

(Ordered || Unchangeble)

```
In [ ]: 1 t = (2,3,1,9,5,7)
2 print(t[3]) # 9
3 print(t[-5]) # 3
4 print(t[1:4]) # (3, 1, 9)
5 print(t[:2]) # (2, 1, 5)
6
7 t1 = (1,2,3)
8 t2 = (2,3,2)
9 print(t1 + t2) # (1, 2, 3, 2, 3, 2)
10 print(t2 + t1) # (2, 3, 2, 1, 2, 3)
11 print(t1 * 5) # (1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)
12
13 # (< && >) length ne kai Leva-deva nathi 😊
14 print(t1 > t2) # False
15 print(t2 > t1) # True
16
17 c = (1,2,3)
18 c1 = (2,2,3)
19 print(c > c1) # False
20 print(c1 > c) # True
21
22 # h1 = (10,20,30,'a') # error
23 h1 = (10,20,30,40)
24 h2 = (10,20,30,40)
25 print(h2 > h1) # False
26 print(h1 > h2) # False
27 print(h1 == h2) # True
```

Nested Tuple

```
In [ ]: 1 t1 = (10,20,30,[2,3,(100,200,300)],40,50)
2 print(t1[3]) # [2, 3, (100, 200, 300)]
3 # t[1] = 200 # -> can't change(immutable)
4
5 print(t1[3][0]) # 2
6 print(t1[3][2]) # (100, 200, 300)
7 print(t1[3][2][1]) # 200
8
9 print(t1[3]) # [2, 3, (100, 200, 300)]
10 t1[3][0] = 200
11 print(t1[3]) # [200, 3, (100, 200, 300)]
12 t1[3][2] = 500
13 print(t1[3]) # [200, 3, 500]
14
15 # t1[3][2][1] = 500 # -> error
16
```

if tuple - can't change


```
In [ ]: 1 t1 = (10,3,7,5,2,15,9)
2 print(min(t1)) # 2
3 print(max(t1)) # 15
4 print(len(t1)) # 7
5 print(sorted(t1)) # [2, 3, 5, 7, 9, 10, 15]
6 print(sorted(t1, reverse=True)) # [15, 10, 9, 7, 5, 3, 2]
7 print(tuple(reversed(t1))) # (9, 15, 2, 5, 7, 3, 10)
8 print((reversed(t1))) # <reversed object at 0x0000023C0D2C81F0>
9 print(t1.count(2)) # 1
10 print(t1.index(5)) # 3
```

```
In [ ]: 1 t1 = ()
2 print(type(t1)) # <class 'tuple'>
3
4 t2 = ('10')
5 print(type(t2)) # <class 'str'>
6
7 t3 = ('10',)
8 print(type(t3)) # <class 'tuple'>
```

```
In [ ]: 1 t1 = tuple(input("Enter : ")) # 1234
2 print(t1) # ('1', '2', '3', '4')
3
4 t2 = eval(input("Enter : ")) # 12,13,14
5 print(t2) # (12, 13, 14)
```

324. Write a program to print sum of even numbers and sum of odd numbers from elements given in tuple.

```
In [ ]: 1 nt = eval(input("Enter : "))
2 os, es = 0,0
3 print(nt)
4
5 for i in nt:
6     if(i%2==0):
7         es += i
8     elif(i%2!=0):
9         os += i
10 print("Sum of Even :", es)
11 print("Sum of Odd :", os)
```

328. Write a Python program to return another string similar to the input string, but with its case inverted. For example, input of “Mr. Ed” will result in “mR. eD” as the output string. Note: Use of built in swapcase function is prohibited.

```
In [ ]: 1 str = input("Enter : ")
        2 print("Your String :", str)
        3 ns = ""
        4 for i in str:
        5     if(i.isupper()):
        6         ns += i.lower()
        7     else:
        8         ns += i.upper()
        9 print("Updated String :",ns)
```

330. Write a Python program to create a Caesar encryption. Note: In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a right shift of 3, A would be replaced by D, E would become H, and so on. The method is named after Julius Caesar, who used it in his private correspondence. For Example: Input Text : LJ IET ENG || Shift : 3 || Cipher: OMLHW HQJ

```
In [ ]: 1 cyph = input("Enter Text : ").upper()
        2 key = int(input("Enter Key : "))
        3 print("Inputed Text :", cyph)
        4 nc = ""
        5 for i in cyph:
        6     if i.isalpha():
        7         ch = ord(i)+key
        8         if ch>90:
        9             ch = (ch%91) + 65
        10        nc += chr(ch)
        11    else:
        12        nc += i
        13 print("Cypher Code :",nc)
```

```
In [ ]: 1 cyph = input("Enter Text : ")
2 key = int(input("Enter Key : "))
3 print("Inputed Text :", cyph)
4 nc = ""
5 for i in cyph:
6     if i.islower():
7         ch = ord(i)+key
8         if ch>122:
9             ch = (ch%123) + 97
10        nc += chr(ch)
11    elif i.isupper():
12        ch = ord(i)+key
13        if ch>90:
14            ch = (ch%91) + 65
15        nc += chr(ch)
16    else:
17        nc += i
18 print("Cypher Code :",nc)
```

331. Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2 and length of s1 & s2 should be same. The character's position doesn't matter. Example : s1 = hello s2 = olleh | Balanced

```
In [ ]: 1 s1 = input("Enter Str-1 : ")
2 s2 = input("Enter Str-2 : ")
3 flag = False
4
5 if(len(s1)==len(s2)):
6     for i in s1:
7         if s1.count(i)==s2.count(i):
8             flag = True
9         else:
10            flag = False
11 else:
12     flag = False
13
14 if flag:
15     print("Balanced")
16 else:
17     print("Unbalanced")
```

```
In [ ]: 1 s1 = sorted(tuple(input("Enter Str-1 : ")))
2 s2 = sorted(tuple(input("Enter Str-2 : ")))
3 if s1==s2:
4     print("Balanced")
5 else:
6     print("Unbalanced")
```