T-2_Sem-3 || Python

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

String : It's Case-sensitive.

In [ ]:
```
multi = """h
    i
ee"""

print(multi)
print(len(multi))
```

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

2 type of indexing.

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

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

Slicing (String, List, Tuple) :

s[Start : End : Jump]

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

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

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

In [ ]:
```python
s = input("Enter String : ")
c = s[len(s)::-1]
if(c == s):
    print("Palindrome")
else:
    print("Not a Palindrome")
```

In [ ]:
```python
s = input("Enter String : ")
length = 0
for i in s:
    length += 1

print(length)
```

In [ ]:
```python
s = "Hello"
i = 2
new = "K"
print(s[0:i]+new+s[3:5]) # HeKlo

# we can also use [i+i:5] instead of [3:5]
```

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

In [ ]:
```python
s = input("Enter String : ") # romil
f = s[0] # r
m = s[len(s)//2] # m
l = s[-1] # l

print(f + m + l) # rml
```

String Methods :

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

In [ ]:
```python
s = "hElLo"
print(s) # hElLo
print(s.upper()) # HELLO
print(s.lower()) # hello
s = s.capitalize()
print(s) # Hello

ns = "hElLo WoRld tHis is pY"
print(ns.capitalize()) # Hello world this is py
```

In [ ]:
```python
s = 'hello'
print(s.isupper()) # False
print(s.islower()) # True

cs = 'Hello'
print(cs.isupper()) # False
print(cs.islower()) # False

astr = 'hello Py'
print(astr.isupper()) # False
print(astr.islower()) # False
```

In [ ]:
```python
s = 'abc123'
print(s.isalpha()) # False
print(s.isnumeric()) # False
print(s.isalnum()) # True
print(s.isdigit()) # False
```

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

In [ ]:
```python
s = "Hello World"
u, l = 0, 0
for i in s:
    if(i.isupper()):
        u += 1
    elif(i.islower()):
        l += 1

print("Upper :", u) # 2
print("Lower :", l) # 8
```

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

Output of Split - List

In [ ]:
```python
s = """This is python code"""
n = s.split(" ")
print(n) # ['This', 'is', 'python', 'code']
new = ""
for i in n:
    if(len(i)==1):
        i.upper
    else:
        new += i[0].upper() + i[1:-1] + i[-1].upper() + " "
print(new) # ThiS IS PythoN CodE

# Short-cutt (Don't use in Exam😂)
# for i in n:
#     print(i.capitalize()[:-1], end="")
#     print(i[-1].capitalize(),end=" ")
# Output : ThiS IS PythoN CodE

# split("char to split from", *)
# *int value that defines maximum no. of split
```

In [ ]:
```python
s1 = "Hello World"
ans = s1.find('l', 6, 10)
ans1 = s1.find('l')
a = s1.index('l')
print(ans, ans1, a) # 9 2 2

a = s1.find('x') # it will give -1
# a = s1.index('x') # it give an error.
print(a)

print(s1.count('l')) # 3
```

In [ ]:
```python
s1 = "---,,--,rgpytrghon--,--,,----"
print(s1.strip("-")) # ,,--,rgpytrghon--,--,,
print(s1.strip("-,rg")) # pytrghon
print(s1.strip("-,")) # rgpytrghon
# removes any white spaces before and after the string.

print(s1.lstrip("-,")) # rgpytrghon--,--,,----
print(s1.rstrip("-,")) # ---,,--,rgpytrghon
```

In [ ]:
```python
s = "Hello World"
ans = s.translate({108:120, 111:None})
# None - remove
print(ans) # Hexx Wrxd
```

In [ ]:
```python
s = "Hello World"
tab = s.maketrans("Hl","hL","O")
print(tab) # {72: 104, 108: 76, 79: None}
ans = s.translate(tab)
print(ans) # heLLo WorLd


tab1 = s.maketrans("Hl","aB","O")
print(tab1) # {72: 97, 108: 66, 79: None}
ans1 = s.translate(tab1)
print(ans1) # heLLo WorLd
```

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

In [ ]:
```python
s = "Hello World"
s1 = input("Enter String you wnat to find occunces : ").lower()
ans = (s.lower()).count(s1)
print(ans)
```

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

In [ ]:
```python
s = '123456789'
sum = 0
digit = 0
for i in s:
    if(i.isnumeric()):
        sum = sum + int(i)
        digit += 1

print(sum) # 45
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 [ ]:
```python
st = input("Enter : ")
s = int(input('Enter shift : '))
ans = 0
if len(st)<s:
    ans = int(st[::-1])
else:
    ans = int(st[s:]+st[:s])

print(ans)
```

326.

In [ ]:
```python
s = "Hello Pyth@n is 100% easy"
u, l, d = 0, 0, 0
for i in s:
    if(i.isupper()):
        u += 1
    elif(i.islower()):
        l += 1
    elif(i.isdigit()):
        d +=1
print("Upper :", u) # 2
print("Lower :", l) # 14
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 $]Examples : Input : Ram@_f1234Output : ValidPasswordInput : Rama_f$**oab**

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

In [ ]:
```python
RED = "\033[0;31m"
GREEN = "\033[0;32m"
BOLD = "\033[1m"
END = "\033[0m"
p = input("Enter Password : ")
u, d, s = 0, 0, 0
if(len(p) >= 8):
    for i in p:
        if(i.isupper()):
            u += 1
        elif(i.isdigit()):
            d += 1
        elif(i == '$' or i == '_' or i == '@'):
            s += 1
    if(u and d and s):
        print(GREEN + BOLD + "Valid Password" + END)
    else:
        print("Invalid Password")
else:
    print(RED + BOLD + "Invalid Password" + END)

# Output :
# Enter Password : Romil@1234
# Valid Password
```

Tuple : (Repeatation is allowed)

(Ordered || Unchangeble)

In [ ]:
```
t = (2,3,1,9,5,7)
print(t[3]) # 9
print(t[-5]) # 3
print(t[1:4]) # (3, 1, 9)
print(t[::2]) # (2, 1, 5)

t1 = (1,2,3)
t2 = (2,3,2)
print(t1 + t2) # (1, 2, 3, 2, 3, 2)
print(t2 + t1) # (2, 3, 2, 1, 2, 3)
print(t1 * 5) # (1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)

# (< && >) length ne kai leva-deva nathi😁
print(t1 > t2) # False
print(t2 > t1) # True

c = (1,2,3)
c1 = (2,2,3)
print(c > c1) # False
print(c1 > c) # True

# h1 = (10,20,30,'a') # error
h1 = (10,20,30,40)
h2 = (10,20,30,40)
print(h2 > h1) # False
print(h1 > h2) # False
print(h1 == h2) # True
```

Nested Tuple

In [ ]:
```
t1 = (10,20,30,[2,3,(100,200,300)],40,50)
print(t1[3]) # [2, 3, (100, 200, 300)]
# t[1] = 200 # -> can't change(immutable)

print(t1[3][0]) # 2
print(t1[3][2]) # (100, 200, 300)
print(t1[3][2][1]) # 200

print(t1[3]) # [2, 3, (100, 200, 300)]
t1[3][0] = 200
print(t1[3]) # [200, 3, (100, 200, 300)]
t1[3][2] = 500
print(t1[3]) # [200, 3, 500]

# t1[3][2][1] = 500 # -> error

```

if tuple - can't change

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

In [ ]:
```python
t1 = ()
print(type(t1)) # <class 'tuple'>

t2 = ('10')
print(type(t2)) # <class 'str'>

t3 = ('10',)
print(type(t3)) # <class 'tuple'>
```

In [ ]:
```python
t1 = tuple(input("Enter : ")) # 1234
print(t1) # ('1', '2', '3', '4')

t2 = eval(input("Enter : ")) # 12,13,14
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 [ ]:
```python
nt = eval(input("Enter : "))
os, es = 0,0
print(nt)

for i in nt:
    if(i%2==0):
        es += i
    elif(i%2!=0):
        os += i
print("Sum of Even :", es)
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 [ ]:
```python
str = input("Enter : ")
print("Your String :", str)
ns =""
for i in str:
    if(i.isupper()):
        ns += i.lower()
    else:
        ns += i.upper()
print("Updated String :",ns)
```

330. Write a Python program to create a Caesar encryption.Note: In cryptography, a Caesar cipher, alsoknown 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 : LJIET ENG || Shift : 3 || Cipher: OMLHW HQJ

In [ ]:
```python
cyph = input("Enter Text : ").upper()
key = int(input("Enter Key : "))
print("Inputed Text :", cyph)
nc = ""
for i in cyph:
    if i.isalpha():
        ch = ord(i)+key
        if ch>90:
            ch = (ch%91) + 65
        nc += chr(ch)
    else:
        nc += i
print("Cypher Code :",nc)
```

In [ ]:
```python
cyph = input("Enter Text : ")
key = int(input("Enter Key : "))
print("Inputed Text :", cyph)
nc = ""
for i in cyph:
    if i.islower():
        ch = ord(i)+key
        if ch>122:
            ch = (ch%123) + 97
        nc += chr(ch)
    elif i.isupper():
        ch = ord(i)+key
        if ch>90:
            ch = (ch%91) + 65
        nc += chr(ch)
    else:
        nc += i
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 [ ]:
```python
s1 = input("Enter Str-1 : ")
s2 = input("Enter Str-2 : ")
flag = False

if(len(s1)==len(s2)):
    for i in s1:
        if s1.count(i)==s2.count(i):
            flag = True
        else:
            flag = False
else:
    flag = False

if flag:
    print("Balanced")
else:
    print("Unbalanced")
```

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