Practise programmes for Python :-

① WAP to reverse a given String ? → `s=input("Enter the name:")` `print(s[::-1])`

② WAP to reverse order of words ?

Ans :-

```
s = input("Enter some string")
l = s.split()
l1 = l[::-1]
output = ' '.join(l1)
print(output)
```

→ suhas is best → best is suhas.

② WAP to reverse Internal content of each word :-

```
s=input("Enter some string")
l = s.split()
l1 = []
for words in l:
    l1.append(words[::-1])
output = ' '.join(l1)
print(output)
```

→ suhas is best → sahus si tseb

③ WAP to reverse internal content of every second word present in the given string ?

```
s=input("Enter the string")
l = s.split()
i = 0
l1 = []
while i < len(l):
    if i%2 == 0:
        l1.append(l[i])
    else:
        l1.append(l[i][::-1])
    i=i+1
output = ' '.join(l1)
print(output)
```

→ One two three four

(or)

→ One owt three ruof

④ WAP to print characters present at even index & odd index seperately for the given String :-

```
s=input("Enter the String")
print("The characters present at even index is:", s[::2])
print("The characters present at odd index is:", s[1::2])
```

⑥ WAP to merge characters of 2 strings into a single string by taking characters alternatively?

```
S1 = input("Enter the Frist name")
S2 = input("Enter the second name")
i, j = 0, 0
output = ''
while i<len(S1) or j<len(S2):
    if i<len(S1):
        output = output + S1[i]
        i = i+1
    if j<len(S2)
        output = output + S2[j]
        j = j+1
print(output)
```

i/p: suhas
rocky

o/p: sruohcaksy

⑦ Assume input string contains only alphabet symbols & digits. WAP to sort characters of the string, first alphabet symbols followed by digits:-

```
S=input("Enter some alphanumeric string to sort:")
alphabets = []
digits = []
for ch in S:
    if ch.isalpha():
        alphabets.append(ch)
    else:
        digits.append(ch)
output = ''.join(sorted(alphabets)+sorted(digits))
print(output)
```

→ if : A1B4C6E5D9

→ ABCDE14659

Eg: S=B1A2C3D4
print(sorted(S))

o/p:
['1', '2', '3', '4', 'A', 'B', 'C', 'D']

⑧ Write a program for the following requirement :-

```
S=input("Enter some string followed by digits")
output = ''
for ch in S:
    if ch.isalpha():
        x=ch
    else:
        d=int(ch)
        output = output + x*d
print(output)
```

i/p: aH b3C2

o/p: aaabbbCC

(14) WAP for the following requirement :- i/p : a3z2b4
o/p : aaabbbbzz (sorted string)

```
s=input("Enter the string:")
target=''
for ch is s:
    if ch.isalpha():
        x=ch
    else:
        d=int(ch)
        target = target + x*d
output = ''.join(sorted(target))
print(output)
```

(15) WAP for the following requirement :- i/p = aaaabbCCzz
o/p : 4a2b2C2z (Same as problem 15)

```
s=input("Enter the string:")
previous = s[0]
(count) → C = 1
i = 1
output = ''
while i<len(s):
    if s[i]==previous:
        C=C+1
    else:
        output = output + str(C) + previous
        previous = s[i]
        C = 1
    if i==len(s)-1:
        output = output + str(C) + previous
    i=i+1
print(output)
```

05/11/21
(17) WAP for the following requirement ?
i/p : a4k3b2
o/p : aeknbd

```
s=input("Enter the string with requirement:")
output = ''
for ch in s:
    if ch.isalpha():
        output = output+ch
        x=ch
    else:
        d=int(ch)
        newc = chr(ord(x)+d)
        output = output+newc
Print(output)
```

Chr to unicode
ord('a') = 97
Chr(97) = a
Unicode to char

(12) WAP to remove duplicate characters from the given i/p string:

```
s=input("Enter the string to remove duplicates:")
output=''
for ch in s:
    if ch not in output:
        output=output+ch

print(output)
```

i/p: AABBCDEGHH
o/p: ABCDEGH

(13) WAP to find the no of occurances of each character present in the given string: Using Count Method.

```
s=input("Enter the string")
l=[]
for ch in s:
    if ch not in l:
        l.append(ch)

for ch in sorted(l):
    print('{} occurs {} times'.format(ch, s.count(ch)))
```

i/p: AABBCCCDDDD
o/p: A occurs 2 times.
B occurs 2 times
C occurs 3 times
D occurs 4 times.

(14) No of occurance of each character present in given string without Count():-

```
s=input("Enter the string:")
d={}
for ch in s:
    d[ch]=d.get(ch,0)+1
for k,v in sorted(d.items()):
    print("{} occurs {} times".format(k,v))
```

(15) WAP for the following Requirement-

i/p: ABAABCCADBB
o/p: 4A4B2C1D

```
s=input("Enter the string:")
d={}
for ch in s:
    d[ch]=d.get(ch,0)+1
output=''
for k,v in sorted(d.items()):
    output=output+str(v)+k

print(output)
```

Dict Concept :- It will always be in key value pairs.

d = { k1 : v1 , k2 : v2 }

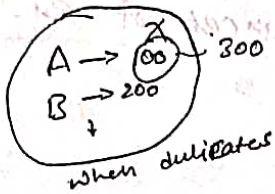
d = { 'A' : 2 , 'B' : 3 }

To create empty dictory :- d = { }

To add one key value pair d[k] = v

d = { }
d['A'] = 100
d['B'] = 200
print(d) → o/p { 'A' : 100 , 'B' : 200 }

Ex: d = { }
d['A'] = 100
d['B'] = 200
d['A'] = 300
print(d) → { 'A' : 300 , 'B' : 200 }



when duplicates inserted

∴ d.get('A') → 300
d.get('z') → None

∴ d.get(k, defaultvalue) ↑

∴ d = { 'A' : 100 , 'z' : 200 , 'B' : 200 }
① items ② item ③ item

for k, v in d.items() :
print(k, v)

o/p : A 100
z 200
B 300

no error old values will be inserted into new value.

Ex:- d = { }
d['A'] = 1
d['B'] = 2
d['A'] = d.get('A', 0) + 1
print(d) → o/p : { 'A' : 2 , 'B' : 2 }

(16) WAP to find the no of occurances of each vowel present in the given string.

→ ip durgasoft solutions

o/p : a occurs 1 times
u occurs 2 times
o occurs 1 times
i occurs 3 times

s = input("Enter the string or name:")
v = { 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' }
d = { }
for ch in s :
if ch in v :
d[ch] = d.get(ch, 0) + 1
for k, v in sorted(d.items()) :
print("{} occurs {} times".format(k, v))

(17) WAP to check whether the given two strings are anagrams (or) not?

∴ Two strings are said to be anagrams iff both are having same content irrespective of characters position.

eg: lazy & zaly
Silent & Listen
Triangle & Integral.

sorted() :- sorts any sequence (list, tuple) & always returns a list with the elements in sorted manner
Ex: sorted(s) = BADGC
print(s) = ['A', 'B', 'C', 'D', 'G']

```
S1 = input ("Enter the First string: ")
S2 = input ("Enter the Second string: ")
if sorted(S1) == sorted(S2):
        print ("Both strings are anagrams")
else:
    print ("strings are not anagrams")
```

(15) WAP to check whether the given string is Palindrome (or) not :-

∴ A string is said to be Palindrome if original string & its reversed strings are equal.

eg :- level, malayalam, eye

```
S = input ("Enter some string")
if S == S[::-1]:
        print ("It is palindrome")
else:
    print ("Not palindrome")
```

(16) WAP for the following requirement :-

```
S1 = input ("Enter 1st string:")
S2 = input ("Enter 2nd string:")
S3 = input ("Enter 3rd string:")
i = j = k = 0
output = ''
while i<len(S1) or j<len(S2) or k<len(S3):
        if i<len(S1):
            output = output + S1[i]
            i = i+1
        if j<len(S2):
            output = output + S2[j]
            j = j+1
        if k<len(S3):
            output = output + S3[k]
            k = k+1
print (output)
```

I/P :- S1 = 'abcdefg'
S2 = 'xyz'
S3 = '12345'

O/P :- ax1, by2, cz3, d4, e5, f, g