Python Exercises

String:

1. Given a string of odd length greater 7, return a string made of the middle three chars of a given String.

Example:

Case 1:

```
str1 = "JhonDipPeta"
```

Expected Output:

```
Dip
```

Case 2:

```
str2 = "JaSonAy"
```

Expected Output:

```
Son
```

2. Given 2 strings, s1 and s2, create a new string by appending s2 in the middle of s1.

Example:

Given:

```
s1 = "Ault"
s2 = "Kelly"
```

Expected Output:

```
AuKellylt
```

3. Given 2 strings, s1, and s2 return a new string made of the first, middle and last char each input string.

Example:

Given:

```
s1 = "America"
s2 = "Japan"
```

Expected Output:

```
AJrpan
```

4. Arrange string characters such that lowercase letters should come first.

Given an input string with the combination of the lower and upper case arrange characters in such a way that all lowercase letters should come first.

Example:

Given:

```
str1 = PyNaTive
```

Expected Output:

```
yaivePNT
```

5. Count all lower case, upper case, digits, and special symbols from a given string.

Example:

Given:

```
str1 = "P@#yn26at^&i5ve"
```

Expected Outcome:

```
Total counts of chars, digits,and symbols

Chars = 8
Digits = 3
Symbol = 4
```

6. Given two strings, s1 and s2, create a mixed String.

Note: *create a third-string made of the first char of s1 then the last char of s2, Next, the second char of s1 and second last char of s2, and so on. Any leftover chars go at the end of the result.*

Example:

Given:

```
s1 = "Abc"
s2 = "Xyz"
```

Expected Output:

```
AzbycX
```

7. String characters balance Test.

We'll say that a String s1 and s2 is balanced if all the chars in the s1 are there in s2. characters position doesn't matter.

Example:

Case 1:

```
s1 = "Yn"
s2 = "PYnative"
```

Expected Output:

```
True
```

Case 2:

```
s1 = "Ynf"
s2 = "PYnative"
```

Expected Output:

```
False
```

8. Find all occurrences of "USA" in given string ignoring the case.

Example:

Given:

```
str1 = "Welcome to USA. usa awesome, isn't it?"
```

Expected Outcome:

```
The USA count is: 2
```

9. Given a string, return the sum and average of the digits that appear in the string, ignoring all other characters.

Example:

Given:

```
str1 = "English = 78 Science = 83 Math = 68 History = 65"
```

Expected Outcome:

```
sum is 294
average is 73.5
```

10. Given an input string, count occurrences of all characters within a string.

Example:

Given:

```
str1 = "Apple"
```

Expected Outcome:

```
{'A': 1, 'p': 2, 'l': 1, 'e': 1}
```

11. Reverse a given string.

Example:

Given:

```
str1 = "PYnative"
```

Expected Output:

```
evitanYP
```

12. Find the last position of a substring “Emma” in a given string.

Example:

Given:

```
str1 = "Emma is a data scientist who knows Python. Emma works at google."
```

Where in the string is the last occurrence of the substring “Emma”?:

Expected Output:

Last occurrence of Emma starts at index **43**

13. Split a given string on hyphens into several substrings and display each substring.

Example:

Given:

```
str1 = Emma-is-a-data-scientist
```

Expected Output:

```
Displaying each substring

Emma
is
a
data
scientist
```

14. Remove empty strings from a list of strings.

Example:

Given:

```
str_list = ["Emma", "Jon", "", "Kelly", None, "Eric", ""]
```

Expected Output:

```
Original list of sting
['Emma', 'Jon', '', 'Kelly', None, 'Eric', '']

After removing empty strings
['Emma', 'Jon', 'Kelly', 'Eric']
```

15. Remove special symbols/Punctuation from a given string.

Example:

Given:

```
str1 = "/*Jon is @developer & musician"
```

Expected Output:

"Jon is developer musician"

16. Removal all the characters other than integers from string.

Example:

Given:

```
I am 25 years and 10 months old
```

Expected Output:

2510

17. Find words with both alphabets and numbers.

Example:

Given:

```
str1 = "Emma25 is Data scientist50 and AI Expert"
```

Expected Output:

Emma25
scientist50

18. From given string replace each punctuation with #.

Example:

Given:

```
str1 = '/*Jon is @developer & musician!!'
```

Expected Output:

##Jon is #developer # musician##

Data Structure:

1. Given a two list. Create a third list by picking an odd-index element from the first list and even index elements from the second.

Example:

For Example:

```
listOne = [3, 6, 9, 12, 15, 18, 21]
listTwo = [4, 8, 12, 16, 20, 24, 28]
```

Expected Output:

```
Element at odd-index positions from list one
[6, 12, 18]
Element at even-index positions from list two
[4, 12, 20, 28]
Printing Final third list
[6, 12, 18, 4, 12, 20, 28]
```

2. Given an input list removes the element at index 4 and add it to the 2nd position and also, at the end of the list.

Example:

For example: `List = [54, 44, 27, 79, 91, 41]`

Expected Output:

```
Original list  [34, 54, 67, 89, 11, 43, 94]
List After removing element at index 4  [34, 54, 67, 89, 43, 94]
List after Adding element at index 2  [34, 54, 11, 67, 89, 43, 94]
List after Adding element at last  [34, 54, 11, 67, 89, 43, 94, 11]
```

3. Given a list slice it into a 3 equal chunks and rever each list.

Example:

For Example: `sampleList = [11, 45, 8, 23, 14, 12, 78, 45, 89]`

Expected Outcome:

```
Original list  [11, 45, 8, 23, 14, 12, 78, 45, 89]
Chunk  1 [11, 45, 8]
After reversing it  [8, 45, 11]
Chunk  2 [23, 14, 12]
After reversing it  [12, 14, 23]
Chunk  3 [78, 45, 89]
After reversing it  [89, 45, 78]
```

4. Given a list iterate it and count the occurrence of each element and create a dictionary to show the count of each element.

Example:

Expected Output:

```
Original list  [11, 45, 8, 11, 23, 45, 23, 45, 89]
Printing count of each item   {11: 2, 45: 3, 8: 1, 23: 2, 89: 1}
```

5. Given a two list of equal size create a set such that it shows the element from both lists in the pair.

Example:

Expected Output:

```
First List  [2, 3, 4, 5, 6, 7, 8]
Second List  [4, 9, 16, 25, 36, 49, 64]
Result is  {(6, 36), (8, 64), (4, 16), (5, 25), (3, 9), (7, 49), (2, 4)}
```

6. Iterate a given list and Check if a given element already exists in a dictionary as a key's value if not delete it from the list.

Example:

Given:

```
rollNumber = [47, 64, 69, 37, 76, 83, 95, 97]
sampleDict ={'Jhon':47, 'Emma':69, 'Kelly':76, 'Jason':97}
```

Expected Outcome:

```
after removing unwanted elemnts from list [47, 69, 76, 97]
```

7. Given a dictionary get all values from the dictionary and add it in a list but don't add duplicates.

Example:

```
speed ={'jan':47, 'feb':52, 'march':47, 'April':44, 'May':52, 'June':53, 'july':54, 'Aug':44, 'Sept':54}

Expected Outcome: [47, 52, 44, 53, 54]
```

8. Remove duplicate from a list and create a tuple and find the minimum and maximum number.

Example:

For Example:

```
sampleList = [87, 45, 41, 65, 94, 41, 99, 94]
```

Expected Outcome:

```
unique items [87, 45, 41, 65, 99]
tuple (87, 45, 41, 65, 99)
min: 41
max: 99
```