Python Data Structure Exercise

Exercise 1: Create a list by picking an odd-index items from the first list and even index items from the second

Given two lists, l1 and l2, write a program to create a third list l3 by picking an odd-index element from the list l1 and even index elements from the list l2.

Given:

```
l1 = [3, 6, 9, 12, 15, 18, 21]
l2 = [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]
```

Exercise 2: Remove and add item in a list

Write a program to remove the item present at index 4 and add it to the 2nd position and at the end of the list.

Given:

```
list1 = [54, 44, 27, 79, 91, 41]
```

Expected Output:

```
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]
```

Exercise 3: Slice list into 3 equal chunks and reverse each chunk

Given:

```
sample_list = [11, 45, 8, 23, 14, 12, 78, 45, 89]
```

Expected Outcome:

```
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]
```

Exercise 4: Count the occurrence of each element from a list

Write a program to iterate a given list and count the occurrence of each element and create a dictionary to show the count of each element.

Given:

```
sample_list = [11, 45, 8, 11, 23, 45, 23, 45, 89]
```

Expected Output:

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

Exercise 5: Create a Python set such that it shows the element from both lists in a pair

Given:

```
first_list = [2, 3, 4, 5, 6, 7, 8]
second_list = [4, 9, 16, 25, 36, 49, 64]
```

Expected Output:

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

Exercise 6: Find the intersection (common) of two sets and remove those elements from the first set

See: Python Set

Given:

```
first_set = {23, 42, 65, 57, 78, 83, 29}
second_set = {57, 83, 29, 67, 73, 43, 48}
```

Expected Output:

```
Intersection is  {57, 83, 29}

First Set after removing common element  {65, 42, 78, 23}
```

Exercise 7: Checks if one set is a subset or superset of another set. If found, delete all elements from that set

Given:

```
first_set = {27, 43, 34}
second_set = {34, 93, 22, 27, 43, 53, 48}
```

Expected Output:

```
First set is subset of second set - True

Second set is subset of First set -  False



First set is Super set of second set -  False

Second set is Super set of First set -  True



First Set  set()

Second Set  {67, 73, 43, 48, 83, 57, 29}
```

Exercise 8: Iterate a given list and check if a given element exists as a key's value in a dictionary. If not, delete it from the list

Given:

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

Expected Outcome:

```
After removing unwanted elements from list [47, 69, 76, 97]
```

Exercise 9: Get all values from the dictionary and add them to a list but don't add duplicates

Given:

```
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]
```

Exercise 10: Remove duplicates from a list and create a tuple and find the minimum and maximum number

Given:

```
sample_list = [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
```

Python List Exercise

Given:

```
list1 = [100, 200, 300, 400, 500]
```

Expected output:

```
[500, 400, 300, 200, 100]
```

Exercise 2: Concatenate two lists index-wise

Write a program to add two lists index-wise. Create a new list that contains the 0th index item from both the list, then the 1st index item, and so on till the last element. any leftover items will get added at the end of the new list.

Given:

```
list1 = ["M", "na", "i", "Ke"]
```

```
list2 = ["y", "me", "s", "lly"]
```

Expected output:

```
['My', 'name', 'is', 'Kelly']
```

Exercise 3: Turn every item of a list into its square

Given a list of numbers. write a program to turn every item of a list into its square.

Given:

```
numbers = [1, 2, 3, 4, 5, 6, 7]
```

Expected output:

```
[1, 4, 9, 16, 25, 36, 49]
```

Exercise 4: Concatenate two lists in the following order

```
list1 = ["Hello ", "take "]
list2 = ["Dear", "Sir"]
```

Expected output:

```
['Hello Dear', 'Hello Sir', 'take Dear', 'take Sir']
```

Exercise 5: Iterate both lists simultaneously

Given a two Python list. Write a program to iterate both lists simultaneously and display items from list1 in original order and items from list2 in reverse order.

Given

```
list1 = [10, 20, 30, 40]
list2 = [100, 200, 300, 400]
```

Expected output:

```
10 400

20 300

30 200

40 100
```

Exercise 6: Remove empty strings from the list of strings

```
list1 = ["Mike", "", "Emma", "Kelly", "", "Brad"]
```

Expected output:

```
["Mike", "Emma", "Kelly", "Brad"]
```

Exercise 7: Add new item to list after a specified item

Write a program to add item 7000 after 6000 in the following Python List

Given:

```
list1 = [10, 20, [300, 400, [5000, 6000], 500], 30, 40]
```

Expected output:

```
[10, 20, [300, 400, [5000, 6000, 7000], 500], 30, 40]
```

Exercise 8: Extend nested list by adding the sublist

You have given a nested list. Write a program to extend it by adding the sublist `["h", "i", "j"]` in such a way that it will look like the following list.

Given List:

```
list1 = ["a", "b", ["c", ["d", "e", ["f", "g"], "k"], "l"], "m", "n"]

# sub list to add
sub_list = ["h", "i", "j"]
```

Expected Output:

```
['a', 'b', ['c', ['d', 'e', ['f', 'g', 'h', 'i', 'j'], 'k'], 'l'], 'm', 'n']
```

Exercise 9: Replace list's item with new value if found

You have given a Python list. Write a program to find value 20 in the list, and if it is present, replace it with 200. Only update the first occurrence of an item.

Given:

```
list1 = [5, 10, 15, 20, 25, 50, 20]
```

Expected output:

```
[5, 10, 15, 200, 25, 50, 20]
```

Exercise 10: Remove all occurrences of a specific item from a list.

Given a Python list, write a program to remove all occurrences of item 20.

Given:

```
list1 = [5, 20, 15, 20, 25, 50, 20]
```

Expected output:

```
[5, 15, 25, 50]
```

Python String Exercise

Exercise 1A: Create a string made of the first, middle and last character

Write a program to create a new string made of an input string's first, middle, and last character.

Given:

```
str1 = "James"
```

Expected Output:

```
Jms
```

Exercise 1B: Create a string made of the middle three characters

Write a program to create a new string made of the middle three characters of an input string.

Given:

Case 1

```
str1 = "JhonDipPeta"
```

Output

```
Dip
```

Case 2

```
str2 = "JaSonAy"
```

Output

```
Son
```

Exercise 2: Append new string in the middle of a given string

Given two strings, `s1` and `s2`. Write a program to create a new string `s3` by appending `s2` in the middle of `s1`.

Given:

```
s1 = "Ault"

s2 = "Kelly"
```

Expected Output:

```
AuKellylt
```

Exercise 3: Create a new string made of the first, middle, and last characters of each input string

Given two strings, `s1` and `s2`, write a program to return a new string made of s1 and s2's first, middle, and last characters.

Given:

```
s1 = "America"

s2 = "Japan"
```

Expected Output:

```
AJrpan
```

Exercise 4: Arrange string characters such that lowercase letters should come first

Given string contains a combination of the lower and upper case letters. Write a program to arrange the characters of a string so that all lowercase letters should come first.

Given:

```
str1 = PyNaTive
```

Expected Output:

```
yaivePNT
```

Exercise 5: Count all letters, digits, and special symbols from a given string

Given:

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

Expected Outcome:

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



Chars = 8

Digits = 3

Symbol = 4
```

Exercise 6: Create a mixed String using the following rules

Given two strings, s1 and s2. Write a program to create a new string s3 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.

Given:

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

Expected Output:

```
AzbycX
```

Show Solution

Exercise 7: String characters balance Test

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. The character's position doesn't matter.

Given:

Case 1:

```
s1 = "Yn"

s2 = "PYnative"
```

Expected Output:

```
True
```

Case 2:

```
s1 = "Ynf"

s2 = "PYnative"
```

Expected Output:

```
False
```

Exercise 8: Find all occurrences of a substring in a given string by ignoring the case

Write a program to find all occurrences of “USA” in a given string ignoring the case.

Given:

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

Expected Outcome:

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

Show Hint

Show Solution

Exercise 9: Calculate the sum and average of the digits present in a string

Given a string s1, write a program to return the sum and average of the digits that appear in the string, ignoring all other characters.

Given:

```
str1 = "PYnative29@#8496"
```

Expected Outcome:

```
Sum is: 38 Average is  6.333333333333333
```

Exercise 10: Write a program to count occurrences of all characters within a string

Given:

```
str1 = "Apple"
```

Expected Outcome:

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

Exercise 11: Reverse a given string

Given:

```
str1 = "PYnative"
```

Expected Output:

```
evitanYP
```

Exercise 12: Find the last position of a given substring

Write a program to find the last position of a substring “**Emma**” in a given string.

Given:

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

Expected Output:

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

Exercise 13: Split a string on hyphens

Write a program to split a given string on hyphens and display each substring.

Given:

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

Expected Output:

```
Displaying each substring



Emma

is

a

data

scientist
```

Exercise 14: Remove empty strings from a list of strings

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']
```

Exercise 15: Remove special symbols / punctuation from a string

Given:

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

Expected Output:

```
"Jon is developer musician"
```

Exercise 16: Removal all characters from a string except integers

Given:

```
str1 = 'I am 25 years and 10 months old'
```

Expected Output:

```
2510
```

Exercise 17: Find words with both alphabets and numbers

Write a program to find words with both alphabets and numbers from an input string.

Given:

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

Expected Output:

```
Emma25
scientist50
```

Exercise 18: Replace each special symbol with # in the following string

Given:

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

Expected Output:

```
##Jon is #developer # musician##
```

Python Tuple Exercise

Exercise 1: Reverse the tuple

Given:

```
tuple1 = (10, 20, 30, 40, 50)
```

Expected output:

```
(50, 40, 30, 20, 10)
```

Exercise 2: Access value 20 from the tuple

The given tuple is a nested tuple. write a Python program to print the value 20.

Given:

```
tuple1 = ("Orange", [10, 20, 30], (5, 15, 25))
```

Expected output:

```
20
```

Exercise 3: Create a tuple with single item 50

Exercise 4: Unpack the tuple into 4 variables

Write a program to unpack the following tuple into four variables and display each variable.

Given:

```
tuple1 = (10, 20, 30, 40)
```

Expected output:

```
tuple1 = (10, 20, 30, 40)

# Your code

print(a) # should print 10

print(b) # should print 20

print(c) # should print 30
```

```
print(d) # should print 40
```

Exercise 5: Swap two tuples in Python

Given:

```
tuple1 = (11, 22)
tuple2 = (99, 88)
```

Expected output:

```
tuple1: (99, 88)

tuple2: (11, 22)
```

Exercise 6: Copy specific elements from one tuple to a new tuple

Write a program to copy elements 44 and 55 from the following tuple into a new tuple.

Given:

```
tuple1 = (11, 22, 33, 44, 55, 66)
```

Expected output:

```
tuple2: (44, 55)
```

Exercise 7: Modify the tuple

Given is a nested tuple. Write a program to modify the first item (22) of a list inside a following tuple to 222

Given:

```
tuple1 = (11, [22, 33], 44, 55)
```

Expected output:

```
tuple1: (11, [222, 33], 44, 55)
```

Exercise 8: Sort a tuple of tuples by 2nd item

Given:

```
tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))
```

Expected output:

```
(('c', 11), ('a', 23), ('d', 29), ('b', 37))
```

Exercise 9: Counts the number of occurrences of item 50 from a tuple

Given:

```
tuple1 = (50, 10, 60, 70, 50)
```

Expected output:

2

Exercise 10: Check if all items in the tuple are the same

```
tuple1 = (45, 45, 45, 45)
```

Expected output:

True

Python Set Exercise

Exercise 1: Add a list of elements to a set

Given a Python list, Write a program to add all its elements into a given set.

Given:

```
sample_set = {"Yellow", "Orange", "Black"}
sample_list = ["Blue", "Green", "Red"]
```

Expected output:

Note: Set is unordered.

```
{'Green', 'Yellow', 'Black', 'Orange', 'Red', 'Blue'}
```

Exercise 2: Return a new set of identical items from two sets

Given:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
```

Expected output:

```
{40, 50, 30}
```

Exercise 3: Get Only unique items from two sets

Write a Python program to return a new set with unique items from both sets by removing duplicates.

Given:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
```

Expected output:

```
{70, 40, 10, 50, 20, 60, 30}
```

Note: set is unordered so not necessary this will be the order of the item.

Exercise 4: Update the first set with items that don't exist in the second set

Given two Python sets, write a Python program to update the first set with items that exist only in the first set and not in the second set.

Given:

```
set1 = {10, 20, 30}
set2 = {20, 40, 50}
```

Expected output:

```
set1 {10, 30}
```

Exercise 5: Remove items from the set at once

Write a Python program to remove items 10, 20, 30 from the following set at once.

Given:

```
set1 = {10, 20, 30, 40, 50}
```

Expected output:

```
{40, 50}
```

Exercise 6: Return a set of elements present in Set A or B, but not both

Given:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
```

Expected output:

```
{20, 70, 10, 60}
```

Exercise 7: Check if two sets have any elements in common. If yes, display the common elements

Given:

```
set1 = {10, 20, 30, 40, 50}
set2 = {60, 70, 80, 90, 10}
```

Expected output:

```
Two sets have items in common

{10}
```

Exercise 8: Update set1 by adding items from set2, except common items

Given:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
```

Expected output:

```
{70, 10, 20, 60}
```

Exercise 9: Remove items from set1 that are not common to both set1 and set2

Given:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
```

Expected output:

```
{40, 50, 30}
```

Python Dictionary Exercise

Exercise 1: Convert two lists into a dictionary

Below are the two lists. Write a Python program to convert them into a dictionary in a way that item from list1 is the key and item from list2 is the value

```
keys = ['Ten', 'Twenty', 'Thirty']
values = [10, 20, 30]
```

Expected output:

```
{'Ten': 10, 'Twenty': 20, 'Thirty': 30}
```

Exercise 2: Merge two Python dictionaries into one

```
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
```

Expected output:

```
{'Ten': 10, 'Twenty': 20, 'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
```

Exercise 3: Print the value of key ‘history’ from the below `dict`

```
sampleDict = {
    "class": {
        "student": {
            "name": "Mike",
            "marks": {
                "physics": 70,
                "history": 80
            }
        }
    }
}
```

Expected output:

80

Exercise 4: Initialize dictionary with default values

In Python, we can initialize the keys with the same values.

Given:

```
employees = ['Kelly', 'Emma']
```

```
defaults = {"designation": 'Developer', "salary": 8000}
```

Expected output:

```
{'Kelly': {'designation': 'Developer', 'salary': 8000}, 'Emma': {'designation':
'Developer', 'salary': 8000}}
```

Exercise 5: Create a dictionary by extracting the keys from a given dictionary

Write a Python program to create a new dictionary by extracting the mentioned keys from the below dictionary.

Given dictionary:

```
sample_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}
```

```
# Keys to extract
keys = ["name", "salary"]
```

Expected output:

```
{'name': 'Kelly', 'salary': 8000}
```

Exercise 6: Delete a list of keys from a dictionary

Given:

```
sample_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"
}

# Keys to remove
keys = ["name", "salary"]
```

Expected output:

```
{'city': 'New york', 'age': 25}
```

Exercise 7: Check if a value exists in a dictionary

We know how to check if the key exists in a dictionary. Sometimes it is required to check if the given value is present.

Write a Python program to check if value 200 exists in the following dictionary.

Given:

```
sample_dict = {'a': 100, 'b': 200, 'c': 300}
```

Expected output:

```
200 present in a dict
```

Exercise 8: Rename key of a dictionary

Write a program to rename a key `city` to a `location` in the following dictionary.

Given:

```
sample_dict = {
  "name": "Kelly",
  "age":25,
  "salary": 8000,
  "city": "New york"
}
```

Expected output:

```
{'name': 'Kelly', 'age': 25, 'salary': 8000, 'location': 'New york'}
```

Exercise 9: Get the key of a minimum value from the following dictionary

```
sample_dict = {
  'Physics': 82,
  'Math': 65,
  'history': 75
}
```

Expected output:

`Math`

Exercise 10: Change value of a key in a nested dictionary

Write a Python program to change Brad's salary to 8500 in the following dictionary.

Given:

```
sample_dict = {
    'emp1': {'name': 'Jhon', 'salary': 7500},
    'emp2': {'name': 'Emma', 'salary': 8000},
    'emp3': {'name': 'Brad', 'salary': 500}
}
```

Expected output:

```
{

   'emp1': {'name': 'Jhon', 'salary': 7500},

   'emp2': {'name': 'Emma', 'salary': 8000},

   'emp3': {'name': 'Brad', 'salary': 8500}

}
```