Assignment - 04 (Dict, Set)

1. What is a dictionary in Python? List out all properties of Dictionary.

In [ ]:
```
Dictionary in Data type which in collection of key and value pairs.
eXample:- {key1 : Value1, Key2 : Value2, . . . . . , KeyN : ValueN}

Properties :
1. It is collection of key and value pair
2. It is mutable
3. It is ordered / unordered also
4. Indexing and slicing is not allowed
5. Duplicate key are not allowed
6. Duplicate Values are allowed
7. Enclosed by curly brackets {}
```

2. Explain the concept of dictionary comprehension in Python.

In [ ]:
```
Dictionary comprehension is a technique for creating python dictionaries. The method create dictionaries from iterable objects, like list, tuple.
Dictionary comprehension also allowes to filtering and modifying key-value pairs.
```

3. Explain the difference between the dict.get(key) method and accessing dict[key] directly.

In [ ]:
```
dict.get() will always return the value, if specified key does not exist then dict.get(key) returns None

dict[key] will raise keyerror if the given key is not present.
```

4. How can you merge two dictionaries in Python?

In [4]:
```
x = {"A":10,"B":60,"C":90}
y = {"D":120,"E":150,"F":180}
x.update(y)
print(x)

{'A': 10, 'B': 60, 'C': 90, 'D': 120, 'E': 150, 'F': 180}
```

5. How can you update dictionary values in Python? Provide examples.

In [5]:
```
x = {"A":10,"B":60,"C":90}
x.update({"B":120})
print(x)

{'A': 10, 'B': 60, 'C': 90, 'B': 120}
```

6. How can you iterate over a dictionary to access keys, values, and key-value pairs?

In [6]:
```
x = {"A":10,"B":60,"C":90}
print(x.keys())
print(x.values())
print(x.items())

dict_keys(['A', 'B', 'C'])
dict_values([10, 60, 90])
dict_items([('A', 10), ('B', 60), ('C', 90)])
```

7. What are the different ways to delete items from dict

In [9]:
```
1. using del method :

x = {"A":10,"B":60,"C":90}
del x["B"]
x

{'A': 10, 'C': 90}

2. using pop method :

x = {"A":10,"B":60,"C":90}
x.pop("B")
x

{'A': 10, 'C': 90}
```

In [11]:
```
x = {"A":10,"B":60,"C":90}
x.popitem()
x

{'A': 10, 'B': 60}

3. using clear method :

x = {"A":10,"B":60,"C":90}
x.clear()
x

{}
```

8. Count the frequency of each word in a given sentence using a dictionary.

In [13]:
```
x = "Dictionary Set Functions Tutorial".lower().split()
count = {}
for i in x:
    count.update({i:x.count(i)})
print(count)

{'dictionary': 1, 'set': 1, 'functions': 1, 'tutorial': 1}
```

9. Count the frequency of each character in a given word using a dictionary.

In [14]:
```
x = "Dictionary Set Functions Tutorial".lower()
count = {}
for i in x:
    count.update({i:x.count(i)})
print(count)

{'d': 1, 'i': 4, 'c': 2, 't': 5, 'o': 3, 'n': 3, 'a': 2, 'r': 2, 'y': 1, ' ': 3, 's': 2, 'e': 1, 'f': 1, 'u': 2, 'l': 1}
```

10. Sort a dictionary by keys and values.

In [16]:
```
# Sort by Keys

x = {"Mumbai":5, "Pune":6, "Delhi":9, "Chennai":12}

sort_by_keys = dict(sorted(x.items()))
print("Sort_by_keys :", sort_by_keys)

Sort_by_keys : {'Chennai': 12, 'Delhi': 9, 'Mumbai': 5, 'Pune': 6}
```

In [18]:
```
# Sort by values

x = {"Mumbai":30, "Pune":14, "Delhi":9, "Chennai":6}

sort_by_values = dict(sorted(x.items(),key=lambda item: item[1]))
print("Sort_by_values :", sort_by_values)

Sort_by_values : {'Chennai': 6, 'Delhi': 9, 'Pune': 14, 'Mumbai': 30}
```

11. Find the sum of all values in a dictionary.

In [17]:
```
x = {"A": 20, "B": 60, "C": 90, "D": 120, "E": 180, "F": 210}
sum_values = sum(x.values())
sum_values

630
```

In [19]:
```
x = {"A": 10, "B": 60, "C": 90, "D": 120, "E": 180, "F": 210}
sum_values = 0
for i in x.values():
    sum_values += i
print("sum_values :", sum_values)

sum_of_values : 670
```

12. Create a dictionary using dictionary comprehension that contains even numbers as keys and their squares as values from 1 to 10.

In [20]:
```
x = {i:i*2 for i in range(1,11) if i%2==0}
print(x)

{2: 4, 4: 8, 6: 12, 8: 16, 10: 20}
```

13. Group a list of words into lists of anagrams a dictionary. Print the list of lists where each inner list contains words that are anagrams of each other.

Input: ["listen", "silent", "enlist", "hello", "lemon", "melon", "debitcard", "badcredit"]

Anagram groups: ["listen", "silent", "enlist"], ["hello"], ["lemon", "melon"], ["debitcard", "badcredit"]

In [21]:
```
words = ["listen", "silent", "enlist", "hello", "lemon", "melon", "debitcard", "badcredit"]

anagram_dict = {}

for word in words:
    sorted_word = tuple(sorted(word))
    if sorted_word in anagram_dict:
        anagram_dict[sorted_word].append(word)

    else:
        anagram_dict[sorted_word]=[word]
list_anagram=list(anagram_dict.values())
print(list_anagram)

[['listen', 'silent', 'enlist'], ['hello'], ['lemon', 'melon'], ['debitcard', 'badcredit']]
```

14. Merge two dictionaries. If there are common keys, sum their values. Print the merged dictionary.

In [22]:
```
x = {"A":10,"B":60,"C":1200}
y = {"A":180,"F":360,"C":60}
z = {}
for i in x.keys():
    for j in y.keys():
        if i==j:
            z.update({i:x.get(i)+y.get(j)})
print(z)

{'A': 90, 'B': 160, 'R': 1600}
```

15. Merge two lists into a dictionary using dictionary comprehension.

In [24]:
```
x = ["Mumbai", "Delhi", "Chennai"]
y = [1, 2, 3]
new_dict = {key : value for key, value in zip(x, y)}
print(new_dict)

{1: 'Mumbai', 2: 'Delhi', 3: 'Chennai'}
```

16. Find mirror characters in a string using a dictionary.

For example, 'A' mirrors to 'Z', 'B' mirrors to 'Y',

Input string: "Hello World"

In [4]:
```
x = "Hello World"
s1 = {chr(i):chr(155-i) for i in range(65,91)}
s1.update({chr(i):chr(219-i) for i in range(97,123)})

for k,v in s1.items():
    print(f"{k}: {v}")
mirror_string = ""
for char in x :
    if char in s1:
        mirror_string += s1[char]

    else:
        mirror_string += char
print(f"Original String : {x}")
print(f"Mirror_string : {mirror_string}")

A: Z
B: Y
C: X
D: W
E: V
F: U
G: T
H: S
I: R
J: Q
K: P
L: O
M: N
N: M
O: L
P: K
Q: J
R: I
S: H
T: G
U: F
V: E
W: D
X: C
Y: B
Z: A
a: z
b: y
c: x
d: w
e: v
f: u
g: t
h: s
i: r
j: q
k: p
l: o
m: n
n: m
o: l
p: k
q: j
r: i
s: h
t: g
u: f
v: e
w: d
x: c
y: b
z: a
Original String : Hello World
Mirror_string : Svool Dliow
```

17. Convert a list of tuples into a dictionary.

In [5]:
```
list = [("A",10),("B",20),("C",30)]
Dict1 = dict(list)
print(Dict1)

{'A': 10, 'B': 20, 'C': 30}
```

18. Convert a dictionary to a list of tuples.

In [12]:
```
Dict1 = {'A': 10, 'B': 20, 'C': 30}
list = [(k,v) for k, v in Dict1.items()]
list
```

Out[12]:
```
[('A', 10), ('B', 20), ('C', 30)]
```

19. Create a dictionary that maps each lowercase and uppercase alphabet to its corresponding ASCII code.

In [13]:
```
Dict1 = {chr(i):i for i in range(65,91) }
Dict1.update({chr(i):i for i in range(97,123) })
print(Dict1)

{'A': 65, 'B': 66, 'C': 67, 'D': 68, 'E': 69, 'F': 70, 'G': 71, 'H': 72, 'I': 73, 'J': 74, 'K': 75, 'L': 76, 'M': 77, 'N': 78, 'O': 79, 'P': 80, 'Q': 81, 'R': 82, 'S': 83, 'T': 84, 'U': 85, 'V': 86, 'W': 87, 'X': 88, 'Y': 89, 'Z': 90, 'a': 97, 'b': 98, 'c': 99, 'd': 100, 'e': 101, 'f': 102, 'g': 103, 'h': 104, 'i': 105, 'j': 106, 'k': 107, 'l': 108, 'm': 109, 'n': 110, 'o': 111, 'p': 112, 'q': 113, 'r': 114, 's': 115, 't': 116, 'u': 117, 'v': 118, 'w': 119, 'x': 120, 'y': 121, 'z': 122}
```

20. Count the number of vowels (both uppercase and lowercase) in a given string and store the counts in a dictionary.

In [18]:
```
x = "Python and Data Science"
vowels = "AEIOUaeiou"
dict1 = {i:x.count(i) for i in vowels if i in x}
dict1

{'a': 3, 'e': 2, 'i': 1, 'o': 1}
```

21. Create a nested dictionary where the keys are letters from 'A' to 'E' and the values are dictionaries mapping numbers from 1 to 5 to their squares.

In [20]:
```
x = {chr(letter):{number : number**2 for number in range(1,6)}for letter in range(65,71)}
print(x)

{'A': {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}, 'B': {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}, 'C': {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}, 'D': {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}, 'E': {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}, 'F': {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}}
```

22. What is a set? List out all the properties of set.

In [ ]:
```
Set is an unordered collection of unique elements. Set are Defined by using the set keyword or by using curly braces {}

Properties of Set :
1. Set is unordered , They do not follow any order or sequence
2. No duplicate items are allowed
3. Set are Mutable, we can add or remove elements from the set
4. Indexing and Slicing is not possible in sets
5. Set is a collection of immutable Datatype
```

23. Explain the difference between a set and a frozenset.

In [ ]:
```
Set :
    -- set is mutable , we can change or remove elements from the sets
    -- Sets are used when you need a collection of unique elements

Frozenset :
    -- Frozenset are immutable, we can't change it after creation.
    -- Frozensets are typically used as keys in dictionaries or elements in other sets because they are hashable and immutable
```

24. Can you have a set of sets in Python? Explain with reasons.

In [ ]:
```
NO
Sets are mutable and therefore not hashable, which prevents them from being elements of other sets.
```

25. How are sets different from lists and tuples in Python?

In [ ]:
```
Set :
    -- set is an unordered
    -- duplicate items are not allowed
    -- does not support indexing

List :
    -- list is orderd
    -- duplicate items are allowed
    -- support indexing

Tuple :
    -- tuple is ordered
    -- duplicate items are allowed
    -- support indexing
```

26. Find the minimum and maximum elements in a set without using the built-in min() and max() functions.

In [22]:
```
x = {10,50,30,1,25,66,13,99,44}
max_num = 0
for num in x:
    if max_num < num:
        max_num = num

print(f"Maximum Number is {max_num}")

Maximum Number is 99
```

In [23]:
```
x = {10,50,30,1,25,66,13,99,44}
min_num = x.pop()
for num in x:
    if num < min_num:
        min_num = num

print(f"Minimum Number is {min_num}")

Minimum Number is 1
```

27. Iterate through a set and calculate the sum of its elements without using the built-in sum() function.

In [25]:
```
x = {10,50,30,1,25,66,13,99,44}
sum = 0
for num in x:
    sum = sum + num
print(f"Sum is {sum}")

Sum is 329
```

28. Find the intersection of two sets with and without using built-in methods.

In [28]:
```
x = {10,50,30,1,25,66,13,99,44}
y = {11,22,33,99,2,50,0,1,30}
intersected_elements = []
for num in x:
    if num in x:
        intersected_elements.append(num)

print(f"Intersected elements : {intersected_elements}",end=" ")

Intersected elements : [1, 50, 99, 30]
```

29. Find the symmetric difference of two sets with and without using built-in methods.

In [31]:
```
x = {10,50,30,1,13,66,44}
y = {11,22,33,99,2,50,0,1,30}
z = {i for i in x if i not in y if i in y if i not in x}
z = {z for z in x if z not in y}
print(f"Symmetric difference of two sets is{z}",end=" ")

Symmetric difference of two sets is{5, 66, 13, 44, 11}
```

30. Perform set operations (union, intersection, difference) on sets containing both integers and strings without using built-in methods.

Set 1: {1, 2, 'apple', 'orange'} Set 2: {2, 3, 'orange', 'banana'}

In [32]:
```
set1 = {1, 2, 'apple', 'orange'}
set2 = {2, 3, 'orange', 'banana'}
set1.union(set2)
```

Out[32]:
```
{1, 2, 3, 'apple', 'banana', 'orange'}
```

In [33]:
```
set1 = {1, 2, 'apple', 'orange'}
set2 = {2, 3, 'orange', 'banana'}
set2.union(set1)
```

Out[33]:
```
{1, 2, 3, 'apple', 'banana', 'orange'}
```

In [34]:
```
set1 = {1, 2, 'apple', 'orange'}
set2 = {2, 3, 'orange', 'banana'}
set1.intersection(set2)
```

Out[34]:
```
{2, 'orange'}
```

In [35]:
```
set1 = {1, 2, 'apple', 'orange'}
set2 = {2, 3, 'orange', 'banana'}
set2.intersection(set1)
```

Out[35]:
```
{2, 'orange'}
```

In [36]:
```
set1 = {1, 2, 'apple', 'orange'}
set2 = {2, 3, 'orange', 'banana'}
set1.difference(set2)
```

Out[36]:
```
{1, 'apple'}
```

In [37]:
```
set1 = {1, 2, 'apple', 'orange'}
set2 = {2, 3, 'orange', 'banana'}
set2.difference(set1)
```

Out[37]:
```
{3, 'banana'}
```

In [ ]: