Dictionary

```python
# Python
# collection of key value pair
# it is mutable . each key is unique .


my_dictionary = {
    "name": "Anna",
    "age": 21,
    "dept" : "dev"

}

x = my_dictionary["name"]
print(x)

y = my_dictionary.get("name")
print(y)


my_keys = my_dictionary.keys()
print(my_keys)

my_values = my_dictionary.values()
print(my_values)

my_dictionary["name"] = "John"

print(my_dictionary)


# POP

my_dictionary.popitem()
print(my_dictionary)

my_dictionary.update({"dept":"dev"})
print(my_dictionary)

#clear

for i in my_dictionary:
    print(my_dictionary[i])
```

```python
for j in my_dictionary.keys():
    print(j)

for x,y in my_dictionary.items():
    print(x,y)


## Print the dictionary 1 to 5
# n = int(input("Enter a number"))
# d = dict()
# for x in range(1, n+1):
#     d[x] = x * x
# print(d)


# # if key is present

# def is_key_present(x):
#     if x in d:
#         print("key is present")
#     else:
#         print("Not present")

# is_key_present(5)


d = {
    "a" : 200,
    "b": 300,
    "c" : 100,
    "d": 700
}


# .copy

d2 = d.copy()
print(d2)

# sort the dictionary by values

sorted_items = sorted(d.items() , key = lambda item: item[1])
```

```
print(sorted_items)

key_min = sorted_items[0][0]
MIn_value = sorted_items[0][1]
print(key_min)
print(MIn_value)

str = "helloooo"
char_count = {}

for char in str:
    if char in char_count:
        char_count[char] += 1
    else:
        char_count[char] = 1

print(char_count)

space_dict = {'first     name': "john" , 'last name': "Doe" , "age":
21}

new_dict ={}


for key, value in space_dict.items():
    new_dict[key.replace(" ","")] = value

print(new_dict)
```

Zip.py

```
# //zip()

list1 =[1,2,3,4]
list2=['a','b','c']

zipped = list(zip(list1,list2))
print(zipped)
```

```
unzip = list(zip(*zipped))
print(unzip)

names = ["anna","Bob","John"]
score1 = [80, 70 , 90]

# for name , score in zip(names, score):
#     print(f"{name} scored {score}")

scores = dict(zip(names,score1))
print(scores)
```

1. Create a dictionary with keys as country names and values as their capital cities. Write a function that takes a country name as input and returns its capital city. If the country is not in the dictionary, return "Country not found".
   Hint : use → dictionary.get(key, default_value)
   Default_value is when key is not found

2. Write a function that takes a list of tuples and converts it into a dictionary. Each tuple contains a key and a value. For example, [(1, "one"), (2, "two")] should be converted into {1: "one", 2: "two"}

3. Given the following dictionary of student scores, calculate the average score and output a dictionary with the students' names and their respective average score.

   ```
   sum(marks) / len(marks)
   ```

4. Write a function that merges two dictionaries. If there are duplicate keys, the value from the second dictionary should be taken.
   Use `del d[key]`

5. Write a function that takes a dictionary and a key, and deletes the key-value pair from the dictionary. If the key does not exist, print "Key not found"
   Hint:
   Calculate length : len(dict)
6. Write a function to check if a dictionary is empty.
7. Write a function to find the key of the maximum value in a dictionary.
   Hint:
   ```
       return max(d, key=d.get)
   ```

8. Write a function that counts the occurrence of each character in a string and returns the result in a dictionary.

9. Write a function that swaps the keys and values in a dictionary.
10. Write a function that finds the common keys between two dictionaries.

Hint: `return d1.keys() & d2.keys()`

11. Write a function that adds a new key-value pair to a dictionary only if the key does not already exist.
12. Write a function to sort a dictionary by its values in ascending order.
13. Write a function that removes all the keys with None values in a dictionary.
14. Write a function to find the intersection of two dictionaries (keys and values must be the same).
15. Write a function that returns the dictionary after updating all values by adding 1.
16. Write a function that returns a dictionary where the keys are unique values of a list, and the values are the count of those elements.
17. Write a function that checks if a given key exists in a dictionary.
18. Write a function that takes a dictionary and returns a new dictionary that has all the keys with string values.
    Hint:

```python
def filter_string_values(d):
    filtered_dict = {}
    for k, v in d.items():
        if isinstance(v, str):
            filtered_dict[k] = v
    return filtered_dict
```

    For each pair, we check if the value is an instance of `str` using `isinstance(v, str)`.

19. Write a function to concatenate all values of a dictionary (assuming all values are strings) into a single string.
20. Write a function that takes a dictionary and returns a list of keys sorted by their corresponding values

```python
# 1. Create a dictionary with keys as country names and values as their capital cities.
# Write a function that takes a country name as input and returns its capital city.
# If the country is not in the dictionary, return "Country not found".

countries_capitals = {
    "USA": "Washington, D.C.",
    "India": "New Delhi",
    "Germany": "Berlin",
    "Canada": "Ottawa",
    "Australia": "Canberra"
}

def get_capital(country):
    return countries_capitals.get(country, "Country not found")

print(get_capital("India"))  # Output: New Delhi
print(get_capital("France"))  # Output: Country not found

# 2. Write a function that takes a list of tuples and converts it into a dictionary.
# Each tuple contains a key and a value. For example, [(1, "one"), (2, "two")]
# should be converted into {1: "one", 2: "two"}.

def list_to_dict(lst):
    return dict(lst)

lst = [(1, "one"), (2, "two"), (3, "three")]
print(list_to_dict(lst))  # Output: {1: 'one', 2: 'two', 3: 'three'}

# 3. Given the following dictionary of student scores, calculate the average score
# and output a dictionary with the students' names and their respective average score.

students_scores = {
    "Alice": [85, 90, 78],
    "Bob": [92, 88, 84],
    "Charlie": [89, 91, 85]
}
```

```python
def average_scores(scores):
    avg_scores = {}
    for student, marks in scores.items():
        avg_scores[student] = sum(marks) / len(marks)
    return avg_scores

print(average_scores(students_scores))  # Output: {'Alice': 84.33,
'Bob': 88.0, 'Charlie': 88.33}

# 4. Write a function that merges two dictionaries. If there are
duplicate keys,
# the value from the second dictionary should be taken.

dict1 = {"a": 1, "b": 2, "c": 3}
dict2 = {"b": 4, "d": 5}

def merge_dicts(d1, d2):
    d1.update(d2)
    return d1

print(merge_dicts(dict1, dict2))  # Output: {'a': 1, 'b': 4, 'c': 3,
'd': 5}

# 5. Write a function that takes a dictionary and a key, and deletes
the key-value pair
# from the dictionary. If the key does not exist, print "Key not
found".

def delete_key(d, key):
    if key in d:
        del d[key]
        return d
    else:
        return "Key not found"

print(delete_key({"a": 1, "b": 2}, "a"))  # Output: {'b': 2}
print(delete_key({"a": 1, "b": 2}, "c"))  # Output: Key not found

# 6. Write a function to check if a dictionary is empty.

def is_empty(d):
    return len(d) == 0
```

```python
print(is_empty({}))  # Output: True
print(is_empty({"a": 1}))  # Output: False

# 7. Write a function to find the key of the maximum value in a
dictionary.

def max_value_key(d):
    return max(d, key=d.get)

print(max_value_key({"a": 10, "b": 30, "c": 20}))  # Output: 'b'

# 8. Write a function that counts the occurrence of each character in a
string
# and returns the result in a dictionary.

def char_count(string):
    count = {}
    for char in string:
        count[char] = count.get(char, 0) + 1
    return count

print(char_count("hello"))  # Output: {'h': 1, 'e': 1, 'l': 2, 'o': 1}

# 9. Write a function that swaps the keys and values in a dictionary.

def swap_keys_values(d):
    swapped_dict = {}
    for key, value in d.items():
        swapped_dict[value] = key
    return swapped_dict

# Example usage
my_dict = {"a": 1, "b": 2, "c": 3}
swapped_dict = swap_keys_values(my_dict)

print(swapped_dict)  # Output: {1: 'a', 2: 'b', 3: 'c'}

# 10. Write a function that finds the common keys between two
dictionaries.

def common_keys(d1, d2):
    return d1.keys() & d2.keys()
```

```python
print(common_keys({"a": 1, "b": 2}, {"b": 3, "c": 4}))  # Output: {'b'}

# 11. Write a function that adds a new key-value pair to a dictionary
only
# if the key does not already exist.

def add_if_not_exists(d, key, value):
    if key not in d:
        d[key] = value
    return d

print(add_if_not_exists({"a": 1}, "b", 2))  # Output: {'a': 1, 'b': 2}
print(add_if_not_exists({"a": 1}, "a", 2))  # Output: {'a': 1}

# 12. Write a function to sort a dictionary by its values in ascending
order.

def sort_dict_by_value(d):
    return dict(sorted(d.items(), key=lambda item: item[1]))

print(sort_dict_by_value({"a": 3, "b": 1, "c": 2}))  # Output: {'b': 1,
'c': 2, 'a': 3}

# 13. Write a function that removes all the keys with None values in a
dictionary.


def remove_none_values(d):
    return {k: v for k, v in d.items() if v is not None}

# Example usage
my_dict = {"a": 1, "b": None, "c": 2, "d": None}
cleaned_dict = remove_none_values(my_dict)

print(cleaned_dict)  # Output: {'a': 1, 'c': 2}


# 14. Write a function to find the intersection of two dictionaries
# (keys and values must be the same).

def dict_intersection(d1, d2):
    return {k: v for k, v in d1.items() if d2.get(k) == v}
```

```
print(dict_intersection({"a": 1, "b": 2}, {"b": 2, "a": 1}))  # Output:
{'a': 1, 'b': 2}

# 15. Write a function that returns the dictionary after updating all
values by adding 1.

def increment_values(d):
    return {k: v + 1 for k, v in d.items()}

print(increment_values({"a": 1, "b": 2}))  # Output: {'a': 2, 'b': 3}

# 16. Write a function that returns a dictionary where the keys are
unique
# values of a list, and the values are the count of those elements.

def count_elements(lst):
    return {x: lst.count(x) for x in set(lst)}

print(count_elements([1, 2, 2, 3, 1]))  # Output: {1: 2, 2: 2, 3: 1}

# 17. Write a function that checks if a given key exists in a
dictionary.

def key_exists(d, key):
    return key in d

print(key_exists({"a": 1, "b": 2}, "b"))  # Output: True
print(key_exists({"a": 1, "b": 2}, "c"))  # Output: False

# 18. Write a function that takes a dictionary and returns a new
dictionary
# that has all the keys with string values.

def filter_string_values(d):
    filtered_dict = {}
    for k, v in d.items():
        if isinstance(v, str):
            filtered_dict[k] = v
    return filtered_dict

# Example usage
my_dict = {"a": "apple", "b": 2, "c": "cat", "d": 3.5}
filtered_dict = filter_string_values(my_dict)
```

```
print(filtered_dict)  # Output: {'a': 'apple', 'c': 'cat'}
# Output: {'a': 'apple', 'c': 'cat'}

# 19. Write a function to concatenate all values of a dictionary
# (assuming all values are strings) into a single string.

def concatenate_values(d):
    return ''.join(d.values())

print(concatenate_values({"a": "hello", "b": "world"}))  # Output:
'helloworld'

# 20. Write a function that takes a dictionary and returns a list of
keys
# sorted by their corresponding values.

def sort_keys_by_value(d):
    return [k for k, v in sorted(d.items(), key=lambda item: item[1])]

print(sort_keys_by_value({"a": 3, "b": 1, "c": 2}))  # Output: ['b',
'c', 'a']
```

```
####################PRINT UNIQUE VALUES
L = [{"V": "S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"},
{"VII": "S005"}, {"V": "S009"}, {"VIII": "S007"}]

unique_values = set(val for dictionary in L for val in
dictionary.values())
print(unique_values)


student = [{'id': 1, 'success': True, 'name': 'Lary'},
 {'id': 2, 'success': False, 'name': 'Rabi'},
 {'id': 3, 'success': True, 'name': 'Alex'}]

print(sum((d['id'] for d in student)))

my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

########## Key exists or not ############
```

```
my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

keys_to_check =['name', 'age']

key_exists= all(key in my_dict for key in keys_to_check )
print(key_exists)
```

NESTED Dictionary

```
students = {
    '101' : {"name" :"Alice"  , 'age': 22 , 'courses': ["Maths",
"Physics"]} ,
    '102' : {"name" :"Alice"  , 'age': 22 , 'courses': ["Maths",
"Physics"]},
     '103' : {"name" :"Alice"  , 'age': 22 , 'courses': ["Maths",
"Physics"]}
}


##Access the value for nested dictionary
print(students["101"]["name"])
```

ENUMS

```
from enum import Enum

class Color(Enum):
    RED = 1
    GREEN = 2
    BLUE = 3

print(Color.RED)
print(Color.GREEN)
print(Color.BLUE)

print(Color.RED.value)
```