1. **Write a Python script to sort (ascending and descending) a dictionary by value.**

Solution:

```
import operator
d = {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}
print('Original dictionary : ',d)
sorted_d = sorted(d.items(), key=operator.itemgetter(1))
print('Dictionary in ascending order by value : ',sorted_d)
sorted_d = dict( sorted(d.items(), key=operator.itemgetter(1),reverse=True))
print('Dictionary in descending order by value : ',sorted_d)
```

2. **Write a Python script to sort (ascending and descending) a dictionary by value.**

Solution:

```
d = {0:10, 1:20}
print(d)
d.update({2:30})
print(d)
```

3. **Write a Python script to concatenate following dictionaries to create a new one.**

Sample Dictionary :

dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

Solution:

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

4. **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}
Solution :
Using a loop and update() method of a dictionary

```
keys = ['Ten', 'Twenty', 'Thirty']
values = [10, 20, 30]
# empty dictionary
res_dict = dict()
for i in range(len(keys)):
   res_dict.update({keys[i]: values[i]})
print(res_dict)
```

5. **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}

Solution

```
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
dict3 = {**dict1, **dict2}
print(dict3)
```

or

```
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
dict3 = dict1.copy()
dict3.update(dict2)
print(dict3)
```

6. **Print the value of key 'history' from the below dict**

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

Expected output: 80

Solution

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

Solution

```
print(sampleDict['class']['student']['marks']['history'])
```

7. **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': 'De

Solution

```
employees = ['Kelly', 'Emma']
defaults = {"designation": 'Developer', "salary": 8000}
res = dict.fromkeys(employees, defaults)
print(res)
# Individual data
print(res["Kelly"])
```

8. 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}
```

Solution 1:

```
Dictionary Comprehension
sampleDict = {
  "name": "Kelly",
  "age":25,
  "salary": 8000,
  "city": "New york" }
keys = ["name", "salary"]
newDict = {k: sampleDict[k] for k in keys}
print(newDict)
```

Solution 2:

Using the update() method and loop

```
sample_dict = {
   "name": "Kelly",
   "age": 25,
   "salary": 8000,
   "city": "New york"}
# keys to extract
keys = ["name", "salary"]
# new dict
res = dict()
for k in keys:
   # add current key with its va;ue from sample_dict
   res.update({k: sample_dict[k]})
print(res)
```

9. 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}
```

Solution 1:

Using the pop() method and loop

```
sample_dict = {
        "name": "Kelly",
        "age": 25,
        "salary": 8000,
        "city": "New york"
              }
    # Keys to remove
              keys = ["name", "salary"]
    for k in keys:
              sample_dict.pop(k)
    print(sample_dict)
```

Solution 2:

Dictionary Comprehension

```
    sample_dict = {
              "name": "Kelly",
                 "age": 25,
                 "salary": 8000,
                 "city": "New york"
                       }
# Keys to remove
    keys = ["name", "salary"]

    sample_dict = {k: sample_dict[k] for k in sample_dict.keys() - keys}
    print(sample_dict)
```

10. 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

Solution

```
sample_dict = {'a': 100, 'b': 200, 'c': 300}
if 200 in sample_dict.values():
   print('200 present in a dict')
```

11. 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'}
```

Solution

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

```
sample_dict['location'] = sample_dict.pop('city')
print(sample_dict)
```

12. Get the key of a minimum value from the following dictionary

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

Expected output:

```
Math
```

Solution

```
sample_dict = {
   'Physics': 82,
   'Math': 65,
   'history': 75
        }
print(min(sample_dict, key=sample_dict.get))
```

13. 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}
}
```

Solution

```
sample_dict = {
   'emp1': {'name': 'Jhon', 'salary': 7500},
   'emp2': {'name': 'Emma', 'salary': 8000},
   'emp3': {'name': 'Brad', 'salary': 6500}
        }
        sample_dict['emp3']['salary'] = 8500
        print(sample_dict)
```

RANDOM NUMBERS

1. **Generate 3 random integers between 100 and 999 which is divisible by 5**
   Solution

```
import random
print("Generating 3 random integer number between 100 and 999 divisible by 5")
for num in range(3):
    print(random.randrange(100, 999, 5), end=', ')
```

2. **Generate 3 random integers between 100 and 999 which is divisible by 5**
   Solution

```
import random
print("Generating 3 random integer number between 100 and 999 divisible by 5")
for num in range(3):
    print(random.randrange(100, 999, 5), end=', ')
```

3. **Random Lottery Pick. Generate 100 random lottery tickets and pick two lucky tickets from it as a winner.**
   Note you must adhere to the following conditions:
   - The lottery number must be 10 digits long.
   - All 100 ticket number must be unique.

   Hint: Generate a random list of 1000 numbers using randrange() and then use the sample() method to pick lucky 2 tickets.
   Solution

```
import random
lottery_tickets_list = []
print("creating 100 random lottery tickets")
# to get 100 ticket
for i in range(100):
    # ticket number must be 10 digit (1000000000, 9999999999)
    lottery_tickets_list.append(random.randrange(1000000000, 9999999999))
# pick 2 luck tickets
winners = random.sample(lottery_tickets_list, 2)
print("Lucky 2 lottery tickets are", winners)
```

4. **Generate 6 digit random secure OTP**
   Solution

```
import secrets
#Getting systemRandom class instance out of secrets module
secretsGenerator = secrets.SystemRandom()
print("Generating 6 digit random OTP")
otp = secretsGenerator.randrange(100000, 999999)
print("Secure random OTP is ", otp)
```

5. **Pick a random character from a given String**

```
import random
name = 'pynative'
char = random.choice(name)
print("random char is ", char)
```

6. **Generate random String of length 5**
   Note: String must be the combination of the UPPER case and lower case letters only. No numbers and a special symbol.

```
import random
import string
```

```
def randomString(stringLength):
    """Generate a random string of 5 charcters"""
    letters = string.ascii_letters
    return ''.join(random.choice(letters) for i in range(stringLength))
print ("Random String is ", randomString(5) )
```

7. Generate a random Password which meets the following conditions

Password length must be 10 characters long.
It must contain at least 2 upper case letters, 1 digit, and 1 special symbol.

```
import random
import string
def randomPassword():
    randomSource = string.ascii_letters + string.digits + string.punctuation
    password = random.sample(randomSource, 6)
    password += random.sample(string.ascii_uppercase, 2)
    password += random.choice(string.digits)
    password += random.choice(string.punctuation)
    passwordList = list(password)
    random.SystemRandom().shuffle(passwordList)
    password = ''.join(passwordList)
    return password
print ("Password is ", randomPassword())
```

8. Calculate multiplication of two random float numbers

Note:
First random float number must be between 0.1 and 1
Second random float number must be between 9.5 and 99.5

```
num1  = random.random()
print("First Random float is ", num1)
num2 = random.uniform(9.5, 99.5)
print("Second Random float is ", num1)
num3 = num1 * num2
print("Multiplication is ", num3)
```

9. Generate random secure token of 64 bytes and random URL

```
import secrets
print("Random secure Hexadecimal token is ", secrets.token_hex(64))
print("Random secure URL is ", secrets.token_urlsafe(64))
```

10. Roll dice in such a way that every time you get the same number

Dice has 6 numbers (from 1 to 6). Roll dice in such a way that every time you must get the same output number. do this 5 times.

```
import random
dice = [1, 2, 3, 4, 5, 6]
print("Randomly selecting same number of a dice")
for i in range(5):
    random.seed(25)
    print(random.choice(dice)).
```

FUNCTIONS

1. Write a program to create a function that takes two arguments, name and age, and print their value.

```
# demo is the function name
def demo(name, age):
    # print value
    print(name, age)
# call function
demo("Ben", 25)
```

2. Write a program to create function func1() to accept a variable length of arguments and print their value.

Note: Create a function in such a way that we can pass any number of arguments to this function and the function should process them and display each argument's value.

Read: variable length of arguments in functions

Function call:

```
# call function with 3 arguments
func1(20, 40, 60)
# call function with 2 arguments
func1(80, 100)
```

Expected Output:

```
Printing values
20
40
60

Printing values
80
100
```

```
def func1(*args):
    for i in args:
        print(i)

func1(20, 40, 60)
func1(80, 100)
```

3. Write a program to create function calculation() such that it can accept two variables and calculate addition and subtraction. Also, it must return both addition and subtraction in a single return call.

Given:

```
def calculation(a, b):
    # Your Code
res = calculation(40, 10)
print(res)
```

Expected Output

```
50, 30
```

In Python, to return multiple values from a function, use a comma to separate them.

Solution 1:

```
def calculation(a, b):
    addition = a + b
    subtraction = a - b
    # return multiple values separated by comma
    return addition, subtraction
# get result in tuple format
res = calculation(40, 10)
```

```
print(res)
```
Solution 2:
```
def calculation(a, b):
    return a + b, a - b

# get result in tuple format
# unpack tuple
add, sub = calculation(40, 10)
print(add, sub)
```

4. Write a program to create a function show_employee() using the following conditions.

It should accept the employee's name and salary and display both.
If the salary is missing in the function call then assign default value 9000 to salary

Given:
showEmployee("Ben", 12000)
showEmployee("Jessa")

Expected output:
Name: Ben salary: 12000
Name: Jessa salary: 9000

```
# function with default argument
def show_employee(name, salary=9000):
    print("Name:", name, "salary:", salary)

show_employee("Ben", 12000)
show_employee("Jessa")
```

5. Create an outer function that will accept two parameters, a and b
Create an inner function inside an outer function that will calculate the addition of a and b
At last, an outer function will add 5 into addition and return it
In Python, we can create a nested function inside a function. We can use the nested function to perform complex tasks multiple times within another function or avoid loop and code duplication.

```
# outer function
def outer_fun(a, b):
    square = a ** 2
    # inner function
    def addition(a, b):
        return a + b
    # call inner function from outer function
    add = addition(a, b)
    # add 5 to the result
    return add + 5
result = outer_fun(5, 10)
print(result)
```