***Python zadaci***

1. Zadati string poredjati alfabetno. Takodje voditi racuna da veliak slova imaju prednost :

Primer :

Input1 : hello

Input2 : abAbccdaB

Output1: ehllo

Output2 : ABaabbccd

tekst = input("Unesite neki tekstualni podatak: ").strip().lower()

tekst2 = input("Unesite neki tekstualni podatak: ").strip()

# sortira elemente stringa, ali ih pretvori u LISTU!

sortiraniString = sorted(tekst)

tekst = "".join(sortiraniString)

print(tekst)

# sortira elemente stringa, ali ih pretvori u LISTU!

sortiraniString2 = sorted(tekst2)

tekst2 = "".join(sortiraniString2)

print(tekst2)

Metoda sorted(promenljiva) prosledjenu promenljivu SORTIRA I PRETVORI JE U LISTU.

Tako da sledeci korak je konvertovati SORTIRANU LISTU u spojeni sortirani string.

To se radi sa : “”.join(sortiranaLista)

1. Proveriti da li se SVI elementi jedne manje liste, nalaze medju elementima druge vece liste :

brojevi = [1, 2, 3, 4, 5, 6]

print("brojevi je : " + str(brojevi))

brojeviAparata = [1, 22, 3, 4, 5, 6, 2, 8, 9,

                  10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

print("brojeviAparata je : " + str(brojeviAparata))

if all(x in brojeviAparata for x in brojevi):  # dzekpot

    print("Svi elementi liste brojevi su u listi brojeviAparata")

Ugradjena metoda **all(x in VecaLista for x in ManjaLista)** proverava da li se Svi elementi manje liste nalaze medju elementima vece liste.

Pored all(…) , postoje I

1. Dat je niz od nekoliko brojeva. Pronaci koji od elemenata iz niza, daju sabiranjem odredjeni broj.

Primer :

Input : numbers = [1, 2, 6, 8, 11, 15], target = 13

Output : [2,11]

# Prvo resenje zadatka na jednostavan nacin :

# Vreme izvrsavanja : O(n^2)

def twoSumMojeResenje(*ar*, *target*):

    for i in range(len(*ar*)-1):

        for j in range(i+1,len(*ar*)):

            if i != j:

                if *ar*[i] + *ar*[j] == *target*:

                    print(*ar*[i],*ar*[j])

# Drugi nacin koji je malo precizniji :

# Time complexity : O(n^2)

# Space complexity : O(1)

def target\_sum(A, target):

    for i in range(len(A)-1):

        for j in range(i+1, len(A)):

            if A[i] + A[j] == target:

                print(A[i], A[j])

print(target\_sum(ar, 13))

# Treci nacin koriscenjem HASH TABELE :

# Time complexity : O(n)

# Space complexity : O(n)

def target\_sum\_hash(A, target):

    ht = dict()

    for i in range(len(A)):

        if A[i] in ht:

            print(ht[A[i]], A[i])

            return True

        else:

            ht[target - A[i]] = A[i]

    return False

print(target\_sum\_hash(ar, 13))

1. Dat je niz sa brojevima. Potrebno je napraviti novi niz koji sadrzi zbir svih brojeva osim onog na kome se nalazi pokazivac.

Primer :

Input : brojevi = [3, 5, 6, 8]

Output : novaLista = [240, 144, 120, 90]

Racna se po sistemu :

240 = 5\*6\*8

144 = 3\*6\*8

120 = 3\*5\*8

90 = 3\*5\*6

1. Fibonacijeva sekvenca. Fibonaci brojevi su : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144…

A formula po kojoj se racuna sekvenca je : Fn = Fn-1 + Fn-2

# Fibonaci za odredjeni broj

def Fibonacci(n):

    # Provera da li je broj manji od 0

    if n < 0:

        print("Netacan unos")

    # Provera da li je uneti broj 0

    elif n == 0:

        return 0

    # Provera da li je uneti broj 1 ili 2

    elif n == 1 or n == 2:

        return 1

    else:

        return Fibonacci(n-1) + Fibonacci(n-2)

print(Fibonacci(10))

1. Funkcija za racunanje Faktorijela. Faktorijel se racuna po formuli :

N! = N \* (N-1) \* (N-2) \* ……. \* 1

Primer :

4! = 4 \* 3 \* 2 \* 1 = 24

6! = 6 \* 5 \* 4 \* 3 \* 2 \* 1 = 720

Faktorial brojeva 1 i 0 = 0

def factorial(n):

    return 1 if (n == 1 or n == 0) else n \* factorial(n - 1)

num = 5

print("Factorial broja", num, "je",

      factorial(num))