A gyakorlat kezdetén megnézzük, hogy hogyan kell futtatni Python programot terminálból. Gyakorlásra b?ven lesz lehet?ség a házi feladat megoldása során.

## Tartalomjegyzék



# Feladatok

A mai órán a rekurzív programozás és a dinamikus programozás módszereivel fogunk megoldani feladatokat. A feladat neve mellett található a feladat nehézsége (szerintem) 1-t?l 5-ig.

## deep_sum

Írjunk egy rekurzív függvényt, aminek bemenete egy olyan lista, mely listákat tartalmaz tetsz?leges mélységig, melyekben pozitív egész számok vannak.
A függvény adja vissza a listában lév? számok összegét.
Pl.:
[1, 2, 3, [4, 5], [[[6], 7]]] -> 28

## palindrom

Írjunk egy rekurzív függvény, ami eldönti egy stringr?l, hogy palindrom-e vagy sem.
Pl.:
aba -> True
abb -> False

## second_best

Írjunk egy rekurzív függvényt, aminek bemenete egy legalább kételem? lista, melyekben pozitív egész számok vannak 0-tól 100-ig.
A függvény adja vissza a második legnagyobb értéket.
Pl.:
[10,25,60,45,30] -> 45

## equal

Írjunk egy rekurzív függvényt, ami kiprinteli az összes olyan n hosszú 0/1 sorozatot, melyeknek az els? és második felében a számjegyek összege megegyezik.
Pl.:
n=4 -> 0000, 0101, 0110, 1001, 1010, 1111,
n=5 -> 00000, 00100, 01001, 01101, 01010, 01110, 10001, 10101, 10010, 10110, 11011, 11111,

## pattern

Írjunk egy rekurzív függvényt, melynek bemenete egy 5-tel osztható n egész szám, és kiprinteli a következ? számsorozatot:
n, n-5, n-10, ..., 0, 5, 10, ..., n-5, n,
A megoldáshoz ne használjunk se listát, se for ciklust, se whilet.
pl.:
15, 10, 5, 0, 5, 10, 15,

## lis

Határozzuk meg egy nem negatív számokból álló sorozat leghosszabb növekv? részsorozatát rekurzív módon.
Pl.:
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15] -> 6

## lcs

Határozzuk két karaktersorozat leghosszabb közös részsorozatának a hosszát rekurzív módon.
Pl.:
ABCBDAB, BDCABA -> 4
A megadott példában azért kapunk 4-et, mert például mindkét karaktersorozatban el?fordul BDAB részsorozat, de hosszabb nem.
Tehát nem kell, hogy a részsorozat egymásutáni elemekb?l álljon, ugorhatunk is.
Hint: Fussunk végig a két karaktersorozaton és ha adott k indexig megvan a maximum, akkor ha megegyezik s1[k] és s2[k], akkor a maximum n? 1-gyel és tovább megyünk s1[k+1]-re és s2[k+1]-re, ha pedig nem egyezik meg s1[k] és s2[k], akkor két irányba megyünk tovább, vizsgáljuk s1[k+1] és s2[k]-t, illetve s1[k] és s2[k+1]-et, a maximum pedig marad az eddigi maximum.

## interleaving

Legyen a függvényünk bemenete 3 karaktersorozat s1, s2, s3.
A függvény térjen vissza True-val, ha s1 string az s2 és s3 stringek összefésülése.
Pl.:
ADEBCF, ABC, DEF -> True
Mivel s1=s2[0]+s3[0]+s3[1]+s2[1]+s2[2]+s3[2].
s1 akkor lesz s2 és s3 összefésülése, ha s1 karakterei s2 és s3 karakterei vegyesen (mindegyik pontosan 1-szer) úgy, hogy s2, illetve s3 karaktereinek sorrendje s1-ben megegyezik s2-, illetve s3-beli sorrendjével.

## alternating_arrays

Írjunk egy rekurzív függvényt, melynek bemenete két növekv? számsorozat, és a függvény kiprinteli az

összes lehetséges olyan növekv?, legalább kételem? sorozatot, ahol az egymásután következ? tagok felváltva jönnek a két listából. (Tehát, ha az els? elem az els? listából van, akkor a második a másodikból, harmadik az els?b?l és így tovább...)
Pl.:
[10, 15, 25], [1, 5, 20, 30] bemenet esetén:
A kiprintelt sorozatok:
10 20
10 20 25 30
10 30
15 20
15 20 25 30
15 30
25 30
1 10 20
1 10 20 25 30
1 10 30
1 15 20
1 15 20 25 30
1 15 30
1 25 30
5 10 20
5 10 20 25 30
5 10 30
5 15 20
5 15 20 25 30
5 15 30
5 25 30
20 25
20 25 30

# target

Írjunk egy függvényt, aminek bemenete egy számjegyeket tartalmazó string és egy egész szám.
Írjuk fel a számjegyeknek és +, -, * m?veleteknek az összes lehetséges kombinációját, melyeknek eredménye a megadott egész szám.
Pl.:
'123',6 -> '1+2+3', '1*2*3'
'125',7 -> '1*2+5', '12-5'