* Dureaza mai mult sa adaugi elemente cu append intr-o lista sau sa folosesti list comprehension?

Creeaza 2 functii care returneaza timpul necesar celor 2 operatiuni.

Poti folosi time.time() pentru a obtine timpul curent.

* Se da un string s. Scrie o functie care returneaza suma caracterelor de tip cifra din s.

Modifica functia astfel incat sa returneze suma numerelor din s. Un numar este reprezentat de caractere numerice alaturate.

ex:

s = 'eu am 33 de ani. in 2 luni va fi ziua mea de nume'

result: 35

* Scrieti o functie care primeste un numar variabil de argumente de tip lista si returneaza o lista cu toate elementele argumentelor.
* Scrieti o functie care primeste numele unui fisier ca parametreu si returneaza tipul acelui fisier.

ex:

def file\_type(file\_name):

pass

file\_type(‘20221107.jpeg’) -> imagine

file\_type(‘test.txt’’) -> text

file\_type(‘music\_20221107.mp3’) -> muzica

* Scrieti o functie care determina daca un cuvant contine litere care nu se repeta (este format din caractere unice)

Functia poate returna True sau False

* \* Se da un numar. Scrie o functie care returneaza numarul ca un string in urmatorul format:

Input: 37 → output: ‘30 + 7’

Input: 114 → output: ‘100 + 10 + 4’

Input: 30165 → output: ‘30000 + 100 + 60 + 5’

* Scrie o functie care returneaza cuvantul cu scorul cel mai mare.

Scorul unui cuvant este dat de suma pozitiilor literelor cuvantului, din alfabet.

Functia primeste ca argument o lista de cuvinte.

ex:

a = 1, b = 2, c = 3, d = 4, etc

Word: mama, scor: 13 + 1 + 13 + 1 = 30

* Se dau 3 tupluri ca input. Fiecare dintre aceste tupluri contine urmatoarele informatii (nume, varsta, inaltime) Afiseaza cele 3 tupluri, dupa sortarea (a-z) si in ordine crescatoare, tinand cont, pe rand, de nume, varsta si inaltime.   
  Ex.

input: (Dan, 33, 170), (Mihai, 20, 180), (Daniel, 40, 173)

dupa nume: (Dan, 33, 170), (Daniel, 40, 173), (Mihai, 20, 180)

dupa varsta: (Mihai, 20, 180), (Dan, 33, 170), (Daniel, 40, 173)

dupa inaltime: (Dan, 33, 170), (Daniel, 40, 173), (Mihai, 20, 180)

* Se da o lista de intregi. Pentru fiecare element din lista sa se determine cate numere din cadrul acelei liste sunt mai mici decat acel element. Rezultatul se returneaza tot intr-o lista

Ex. nbs = [3, 7, 8, 5]

output: [0, 2, 3, 1]

pentru nbs[0] = 3 nu exista niciun numar mai mic in nbs pentru nbs[1] = 7 exista 2 numere mai mici, etc.

* \* Fie un intreg pozitiv n, dat ca input. Scrie o aplicatie care sa printeze o forma triunghiulara realizata din caracterul '\*'.

n reprezinta numarul de randuri ale acestui pattern. Numarul de pe fiecare rand va creste cu 2.

Ex. input n = 3

output:

\*

\*\*\*

\*\*\*\*\*

input: n = 5

output:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

* Fie un string s care contine cuvinte separate de unul sau mai multe spatii. Sa se returneaza un string cu toate cuvintele inversate si separate de un singur spatiu.

Ex. input s = 'python java javascript'

output: 'javascript java python'

EX 1

import time  
  
  
def list\_append(numar):  
 start = time.time()  
 lista\_char = []  
 for i in range(numar):  
 lista\_char.append(i)  
 end = time.time()  
 return end - start  
  
def list\_compre(numar):  
 start = time.time()  
 lista\_char = [i for i in range(numar)]  
 end = time.time()  
 return end - start  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 numar = 5\_000\_000  
 print("Timp cu append", list\_append(numar))  
 print("Timp cu comprehension", list\_compre(numar))

EX 2

def suma\_cifre(s):  
  
 lista\_char = [char for char in s]  
 for i in range(len(lista\_char)):  
 if not lista\_char[i].isdigit():  
 lista\_char[i] = " "  
 numere = "".join(lista\_char).split()  
  
 suma = 0  
  
 for numar in numere:  
 suma += int(numar)  
  
 return suma  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 s = "eu am 33 de ani. in 2 luni va fi ziua mea de nume "  
 print(suma\_cifre(s))

EX 3

def numar\_variabil(\*args):  
  
 lista\_numere = []  
 for i in args:  
 lista\_numere += i  
  
 return lista\_numere  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(numar\_variabil([1,2,3,4,5],[6,7,8,9],[10, 11, 12, 13] ))

EX 4

def file\_type(file\_name):  
 extensions = {  
 'jpg': 'imagine',  
 'jpeg': 'imagine',  
 'png': 'imagine',  
 'gif': 'imagine',  
 'bmp': 'imagine',  
 'txt': 'text',  
 'pdf': 'text',  
 'doc': 'text',  
 'docx': 'text',  
 'mp3': 'muzica'  
 }  
 if "." not in file\_name:  
 return "Fara extensie"  
 ext = file\_name.rsplit(".", 1)[-1].lower()  
 return extensions.get(ext, "Fara extensie")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(file\_type("imagine.png"))  
 print(file\_type("text.txt"))  
 print(file\_type("fisier"))  
 print(file\_type("music\_20221107.mp3"))

EX 5

def char\_unic(cuvant):  
 return len(set(cuvant)) != len(cuvant)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(char\_unic("extraterestru"))  
 print(char\_unic("mar"))  
 print(char\_unic("albastru"))  
 print(char\_unic("alb"))

EX 6

def f\_numar(numar):  
 n\_str = str(numar)  
 lungine = len (n\_str)  
 termeni = []  
  
 for i, cifra in enumerate(n\_str):  
 if cifra != "0":  
 putere = lungine - i - 1 # putere = 3 - 0 - 1 = 2  
 print(putere)  
 termen = cifra + "0" \* putere # termen = 1 + "0" \* 2  
 termeni.append(termen)  
  
  
 print(termeni)  
 return "+".join(termeni)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(f\_numar(114))

EX 7

from traceback import print\_tb  
  
  
def scor\_max(nume):  
 abc = {"a":1, "b":2, "c":3, "d":4, "e":5, "f":6, "g":7, "h":8, "i":9, "j":10, "k":11, "l":12, "m":13, "n":14, "o":15, "p":16, "q":17,  
 "r":18, "s":19, "t":20, "u":21, "v":22, "w":23, "x":24, "y":25, "z":26}  
 def nume\_scor\_max(num):  
 return sum(abc.get(char.lower(), 0) for char in num)  
 for num in nume:  
 print(f"{num} -> {nume\_scor\_max(num)}")  
  
  
  
 return max(nume, key= nume\_scor\_max)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(scor\_max(["mama", "tata", "Andrei", "Vlad"]))

EX 8

def sortare(pers1, pers2, pers3):  
 sort\_nume = [pers1, pers2, pers3]  
 sort\_nume.sort()  
  
 inversate = [(v[1], v[0],v[2]) for v in sort\_nume]  
 inversate.sort()  
 sort\_varsta = [(v[1], v[0], v[2]) for v in inversate]  
  
 inaltime = [(i[2], i[1], i[0]) for i in sort\_nume]  
 inaltime.sort()  
 sort\_inaltime = [(i[2], i[1], i[0]) for i in inaltime]  
  
  
 print(sort\_nume)  
 print(sort\_varsta)  
 print(sort\_inaltime)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(sortare(("Dan", 33, 170), ("Mihai", 20, 180), ("Daniel", 40, 173)))

EX 9

exemplu = [3, 7, 8, 5]  
rezultat = []  
for i in exemplu:  
 calcul = sum(1 for x in exemplu if i > x)  
 rezultat.append((i, calcul))  
print("Ficare element este mai mare decat alte elemente de cate ori:")  
for element, nr in rezultat:  
 print(f"{element} -> {nr}")

EX 10

def stele(n):  
 for i in range(n):  
  
 numar\_stele = 2 \* i + 1  
 print("\*" \* numar\_stele)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 print(stele(10))

EX 11

def sting\_nou(text):  
 text1 = text.split()  
 lista\_noua = []  
 for char in range(len(text1)):  
 if char == 0:  
 lista\_noua.append(text1[-1])  
 elif char == len(text1) - 1:  
 lista\_noua.append(text1[0])  
 else:  
 lista\_noua.append(text1[char])  
 return " ".join(lista\_noua)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(sting\_nou("python java javascript"))  
 print(sting\_nou("mere pere mure"))  
 print(sting\_nou("Mihai Andrei Ion"))