Laborator 6 – Testare unitara cu Python

Scrieti teste unitare pentru operatiile aritmetice de baza **(+, -, /, *)**. Dorim ca functiile care efectueaza aceste operatii sa functioneze in mai multe cazuri decat pe doi parametri de tip **number**. Astfel, la scrierea testelor unitare se vor lua in considerare mai multe scenarii cu privire la tipul celor 2 parametri. De asemenea, modificati functiile de baza astfel incat acestea sa functioneze in toate scenariile de mai jos.

Cele 30 de puncte corespund punctajului maxim pe laboratorul curent.

Scenarii:

Pentru operatia de adunare (+):

30 de puncte: 2 pentru fiecare subpunct => 1 pentru teste, 1 pentru implementare functie

a) ambii parametri sunt de tip **number** => se efectueaza suma celor 2 parametri

b) un parametru de tip **number**, celalalt parametru de tip **string** => se incearca conversia parametrului string la tipul number si se efectueaza suma. In cazul in care nu se poate, se arunca o exceptie. Parametrul string poate fi oricare din cei 2.

c) un parametru de tip **number**, celalalt parametru de tip **list** => se incearca conversia fiecarui element din lista la number si se adauga elementul number la fiecare element din lista. In cazul in care nu se poate face conversia pentru toate elementele, se arunca o exceptie. Parametrul list poate fi oricare din cei 2.

d) un parametru de tip **number**, celalalt parametru de tip **tuple** => se incearca conversia fiecarui element din tuplu la number si se efectueaza suma tuturor elementelor la care se adauga parametrul number. In cazul in care nu se poate face conversia pentru toate elementele, se arunca o exceptie. Parametrul tuple poate fi oricare din cei 2.

e) un parametru de tip **number**, celalalt parametru de tip **dictionary** => se incearca conversia fiecarei valori din dictionar la number si se adauga elementul number la fiecare valoare din dictionar. In cazul in care nu se poate face conversia pentru toate valorile, se arunca o exceptie. Parametrul dictionary poate fi oricare din cei 2.

f) ambii parametri de tip **string** => se incearca conversia celor 2 la number si se efectueaza suma. In cazul in care o conversie nu se poate face, se arunca o exceptie.

g) un parametru de tip **string**, celalalt parametru de tip **list** => la fel ca la punctul **c)**, doar ca se incearca conversia parametrului string la number.

h) un parametru de tip **string**, celalalt parametru de tip **tuple** => la fel ca la punctul **d)**, doar ca se incearca conversia parametrului string la number.

i) un parametru de tip **string**, celalalt parametru de tip **dictionary** => la fel ca la punctul **e)**, doar ca se incearca conversia parametrului string la number.

j) ambii parametri de tip **list** => se incearca conversia tuturor elementelor la number (din ambele liste), iar daca nu e posibila se arunca o exceptie. Daca cei doi parametri au acelasi numar de elemente, atunci se efectueaza suma element cu element, rezultatul fiind o lista. Daca au numar diferit de elemente, atunci se calculeaza suma tuturor elementelor, rezultatul fiind un number.

k) un parametru de tip **list**, celalalt de tip **tuple** => se incearca conversia tuturor elementelor la number (atat cele din lista cat si din tuplu), iar daca nu e posibila se arunca o exceptie. Daca cei doi parametri au acelasi numar de elemente, atunci se efectueaza suma element cu element, rezultatul fiind o lista. Daca au numar diferit de elemente, atunci se calculeaza suma tuturor elementelor, rezultatul fiind un number.

l) un parametru de tip **list**, celalalt de tip **dictionary** => se incearca conversia elementelor listei si valorilor dictionarului la number, iar daca nu e posibila se arunca o exceptie. Se calculeaza suma tuturor elementelor si valorilor, rezultatul fiind un number.

m) ambii parametri de tip **tuple** => se incearca conversia tuturor elementelor la number (din ambele tupluri), iar daca nu e posibila se arunca o exceptie. Daca cei doi parametri au acelasi numar de elemente, atunci se efectueaza suma element cu element, rezultatul fiind o lista. Daca au numar diferit de elemente, atunci se calculeaza suma tuturor elementelor, rezultatul fiind un number.

n) un parametru de tip **tuple**, celalalt de tip **dictionary** => se incearca conversia elementelor tuplului si valorilor dictionarului la number, iar daca nu e posibila se arunca o exceptie. Se calculeaza suma tuturor elementelor si valorilor, rezultatul fiind un number.

o) ambii parametri de tip **dictionary** => se incearca conversia tuturor valorilor dictionarelor la number, iar daca nu e posibila se arunca o exceptie. Daca cele doua dictionare au aceleasi chei, atunci se vor aduna cele doua valori care au aceeasi cheie. Rezultatul este un dictionar, cu aceleasi chei ca dictionarele originale. Daca au alte chei, atunci se face suma tuturor valorilor.

Exemple:

add(5, 7) => 12

add(5, “7”) => 12

add(5, [1, 2, “3”]) => [6, 7, 8]

add(5, (1, 2, “3”)) => [6, 7, 8]

add(5, (1, 2, “3”)) => [6, 7, 8]

add(5, {“a”:1, “b”:2}) => {“a”:6, “b”:7}

add(“5”, “7”) => 12

add(“5”, [1, 2, “3”]) => [6, 7, 8]

add(“5”, (1, 2, “3”)) => [6, 7, 8]

add(“5”, {“a”:1, “b”:2}) => {“a”:6, “b”:7}

add([5, “6”, 7], [“1”, 2, 3]) => [6, 8, 10]

add([5, 6, 7], [1, 2, 3, 4]) => 28

add((5, 6, 7), [1, 2, 3]) => [6, 8, 10]

add((5, 6, 7), [1, 2, 3, 4]) => 28

add([1, 2, 3], {“a”:1, “b”:”2”}) => 9

add((5, 6, 7), (1, 2, 3, 4)) => 28

add((5, 6, 7), (1, 2, 3)) => [6, 8, 10]

add((1, 2, 3), {“a”:1, “b”:”2”}) => 9

add({“a”:5, “b”:”6”}, {“a”:1, “b”:”2”}) => {“a”:6, “b”:”8”}

add({“a”:5, “b”:”6”, “c”:7}, {“a”:1, “b”:”2”}) => 21

BONUS: (20 de puncte: 4 pentru fiecare operatie)

Efectuati teste similare si implementati scenariile si pentru celelalte operatii aritmetice (scadere, inmultire, impartire, ridicare la putere, impartire intreaga)