1 Laborator 2

Saptamana 1-5 martie 2021

Continut:

1. Exercitii cu functii
2. Exercitii cu NumPy

1.1 Precizari

1. Tema va fi implementata intr-un singur fisier Jupyter Notebook, avand numele: Tema2_IDS_NumeCoechipier1_NumeCoechipier2.ipynb. Exemplu: Tema2_IDS_IoanaPopescu_DanIonescu.ipynb
2. Repository-ul va contine in radacina un fisier "README.md" in care vor fi mentionati autorii temelor, cu nume complet si grupa din care fac parte.
3. Tema va fi pusa in lucrarea "Tema 2" de pe site-ul de elearning si pe repository-ul github al echipei pana la data de 8 martie ora 22.
4. Se vor folosi type annotations pentru variabile, parametrii functiilor, tipuri de retur. Se vor folosi docstrings pentru functii. Neindeplinirea acestei cerinte duce la injumatatirea punctajului pe exercitiul in cauza.
5. Inainte de a trimite tema, verificati in cadrul echipei acoperirea cerintelor.

Pentru docstring exista 3 stiluri de lucru. Exemple scurte sunt la https://stackoverflow.com/questions/3898572/what-is-the-standard-python-docstring-format (https://stackoverflow.com/questions/3898572/what-is-the-standard-python-docstring-format). Se va folosi stilul reST, recomandat de PEP287 (https://www.python.org/dev/peps/pep-0287/).

1.2 Exercitii cu functii

Precizare. Fiecare din problemele din aceasta sectiune este notata cu un punct.

1. Scrieti o functie care determina daca un numar este egal cu suma divizorilor sai, mai putin numarul insusi.

Exemplu: divizorii lui 6 sunt 1, 2, 3, 6; suma celor mai mici decat 6 este 1+2+3=6=numarul de plecare. Numarul 8 nu este in aceasta situatie: $8 \neq 1 + 2 + 4 = 7$. Folositi aceasta functie pentru a afisa toate numere cu proprietatea ceruta pana la un n dat.

2. Scrieti o functie care determina daca un sir de caractere - presupus a fi scris cu litere mici - este sau nu palindrom. Un palindrom este un sir de caractere care citit de la dreapta la stanga are acelasi continut ca si citirea de la stanga la dreapta. In functie de valoarea unui indicator boolean `ignore_spaces`, avand valoarea implicita `False`, se vor ignora (elimina) sau nu spatiile. Daca sirul dat ca parametru nu are toate literele mici, se va arunca exceptie; documentatie exceptii in Python (https://www.w3schools.com/python/gloss_python_raise.asp).

Exemple:

- `is_palindrome('ele fac cafele', ignore_spaces=True)` returneaza `True`; `is_palindrome('ele fac cafele')` returneaza `False`. `is_palindrome('Ele fac cafele')` arunca o exceptie, indiferent

de valoarea lui `ignore_spaces`.
- `is_palindrome('abaac')` returneaza `False`

3. Sa se scrie o functie care primeste caile catre doua fisiere de tip text. Despre fiecare fiecare se stie ca are cate un numar pe linie (numarul de linii din fisiere este necunoscut apriori). Functia va returna un tuplu cu: lista de numere care apar in ambele fisiere si diferenta maxima care exista intre numerele din al doilea fisier si numerele din primul fisier. Numerele pot fi cu semn, intregi sau in virgula mobila (functia trebuie sa mearga in orice situatie). Incercati sa folositi functii predefinite din Python (sau NumPy) si collection comprehension.

Exemplu: a.txt contine numerele 1, 2, 3, 4 (cate unul pe linie). b.txt contine 2, 3, 4, 10 (cate un numar pe linie). Functia va returna tuplul `([2, 3, 4], 9)`. Numarul 9 este realizat de 10-1.

4. Sa se scrie o functie `sum_divisors_digits` care preia un numar natural strict pozitiv n si returneaza suma cifrelor divizorilor sai. Intr-o alta functie `black_hole` se apeleaza in mod repetat `sum_divisors_digits` pe numerele rezultate, pana cand se atinge un numar maxim de iteratii (implicit 1000) sau se ajunge la numarul 15. Puteti crea alte functii auxiliare.

Exemplul 1: se pleaca de la $n = 15$; divizorii sunt $1, 3, 5, 15$; suma cifrelor divizorilor este $1 + 3 + 5 + 1 + 5 = 15$, acesta fiind rezultatul apelului `sum_divisors_digits(15)`. Functia `black_hole` se opreste deci dupa un singur apel al functiei `sum_divisors_digits`.

Exemplul 2: se pleaca de la $n = 21$; divizorii sunt $1, 3, 7, 21$ iar `sum_divisors_digits(21)` este $1 + 3 + 7 + 2 + 1 = 14$; se reia in functia `black_hole` apelul functiei `sum_divisors_digits` pana la epuizarea numarului de apeluri sau potentiala stabilizare in 15.

5. Sa se scrie o functie care primeste doua dictionare si returneaza `True` daca primul dictionar este continut in al doilea si `False` altfel. Spunem ca un dictionar `a` este continut in dictionarul `b` daca toate cheile din `a` se gasesc printre cheile din `b` si toate pentru orice cheie `c` din `a`, `a[c] == b[c]`. Incercati sa folositi functii predefinite din Python (`any, all`), tipuri de date incluse in Python si collection comprehension.

1.3 Exercitii cu NumPy

Precizări:

- in rezolvarea exercitiilor se va folosi cod *vectorizat* si collection comprehension (preferabil insa doar cod vectorizat). Puteti folosi alte functii de NumPy.
- fiecare din probleme este notata cu un punct

1. Scrieti o functie care pentru un vector dat a dat, returneaza diferenta de ordinul intai intre elementele sale: $b[i] = a[i + 1] - a[i]$. Scrieti apoi o alta functie care face acelasi lucru pentru o matrice, pe linii sau pe coloane.

Exemple:

- `a = np.array([1, 2, 10, 3])`; `diff1_vector(a)` va returna vectorul NumPy cu continutul `(1, 8, -7)`.

- `a = np.array([[1, 2, 3], [40, 50, 60]])` ; `diff1_mat(a, axis=0)` returneaza o matrice cu o singura linie, `[[39, 48, 57]]` . `diff1_mat(a, axis=1)` returneaza o matrice cu 2x2, `[[1, 1], [10, 10]]` .

2. Sa se scrie o functie care gaseste pozitiile maximelor locale dintr-un vector numpy. Un maxim local este o valoare care are in vecinii imediati (indicele curent ± 1, fara a iesi din vector) valori strict mai mici decat ea.

Exemplu: `[-1, 3, -7, 1, 2, 6, 0, 1] -> [1, 5, 7]` .

3. Pentru un vector NumPy dat, sa se calculeze toate ferestrele de o anumita lungime, cu o anumita dilatare. Lungimea unei ferestre inseamna numarul de elemente din vectorii rezultati, dilatare reprezinta peste cate elemente vecine se sare in construirea unei "ferestre".

Exemple:

- v = np.arange(20), win_len=4, dilation=1 -> [0, 1, 2, 3], [1, 2, 3, 4], [2, 3, 4, 5], ...[16, 17, 18, 19]
- v = np.arange(20), win_len=3, dilation=2 -> [0, 2, 4], [1, 3, 5], ..., [15, 17, 19]

Valorile implicite pentru `win_len` si `dilation` sunt 3, respectiv 1. Veti verifica in cadrul functiilor daca: `0 < win_len <= numarul de elemente din vector` , `dilation > 0` si daca pentru vectorul de intrare, `win_len` si `dilation` date rezulta macar o fereastra; daca vreuna din aceste conditii nu se indeplineste, nu se executa restul functiei si se arunca exceptie (se pot folosi assertiuni sau aruncari de exceptii). Rezultatul se va da ca un tablou cu `win_len` coloane.

4. Descarcati setul de date Wine descris la 'http://archive.ics.uci.edu/ml/datasets/Wine' (http://archive.ics.uci.edu/ml/datasets/Wine'), fisierul `wine.data` , si incarcati-l intr-o matrice NumPy, folosind np.genfromtxt. Alegeti aleator 35 de pozitii din cadrul matricei de valori, setati pe aceste pozitii NaN. Construiti o functie care, primind la intrare o matrice, returneaza o colectie cu indicii de linii respectiv de coloane in care se gasesc valori Na, precum si matricea 'corectata': valorile NaN se umplu cu o valoare implicita data ca parametru al functiei dvs.

5. Se da un vector `v` de numere floating point, un numar floating point `x` si un numar intreg `k>0` , `k<=len(v)` . Sa se scrie o functie care determina pozitiile celor mai apropiate `k` valori din `v` fata de `x` (k nearest neighbors, 1d).