Python (12) - alogrytmy - porównywanie szybkości

(1) Stwórz listę 8-elementową l. Przetestuj i zinterpretuj *pop*:
a=l.pop(0)
print(a)
print(l)
l.pop(2)
print(l)
l.pop(0)
print(l)

(2) Stwórz listę 10 elementową l i przetestuj:
print(l[0:3])
print(l[2:5])
Dla dowolnej listy, niech $dl = len(l)$ będzie jej długością. Znajdź odpowiednią składnię zależną od dl, aby otrzymać pierwszą połowę listy (w stylu print(l[...:...])); drugą połowę listy.
Np. dla [1,5,6,7] otrzymamy [1,5] oraz [6,7]. Dla nieparzystej liczby pierwsza połowa ma być trochę krótsza: [4,3,6,5,9] dzielimy na [4,3] oraz [6,5,9].

(3) Napisz funkcję $scal(l1, l2)$ która ma scalać dwie posortowane listy w listę l i zwracać l. Dopóki obie listy są niepuste dodajemy mniejszy z $l1[0]$ i $l2[0]$ do l i wyrzucamy go (z $l1$ lub $l2$ z pomocą pop). Gdy jedna lista jest pusta, dodajemy drugą na koniec l. Przetestuj scalając np. [3,4,6,7] oraz [1,3,5,11,12].

(4) Napisz funkcję $sortscal(l)$ sortującą przez scalanie:
jeśli lista l jest 1-elementowa zwracamy l, jeśli jest 2-elementowa ustawiamy 2 elementy rosnąco i zwracamy l. Gdy jest więcej elementów dzielimy listę na połowy $l1$ i $l2$ (jak w (2)) i stosujemy rekurencyjnie *sortscal* na $l1$ (l1=sortscal(l1)) oraz na $l2$; scalamy listy za pomocą scal z (3) w listę l i zwracamy l.
Przetestuj na dowolnej liście, czy sortscal działa.

(5) Znajdź funkcję *sortuj* z wcześniejszych zajęć (sortowanie bąbelkowe). Porównamy szybkość *sortuj* oraz *sortscal*. Stwórz listę 1000-elementową składającą się z liczb z zakresu 1..1000. Można taką listę wygenerować przez:
import random
l=[random.randint(1,1000) for _ in range(1000)]

Stosując *time.time*() sprawdź która funkcja (sortuj czy sortscal) szybciej sortuje listę. Zwiększ listę 10 razy i ponów test. Zrób to do momentu gdy któraś z funkcji stanie się zbyt wolna.

(6) W pythonie jest określona metoda *sort*() do sortowania list. Porównaj

jej szybkość z poprzednimi funkcjami.

(7) Napisz funkcję sortującą listę l opierającą się na idei: znajdujemy największy element l i wyrzucamy go na koniec listy, znajdujemy drugi element i wyrzucamy go na przedostatnią pozycję, itd. Sprawdź że funkcja działa. Porównaj jej prędkość z poprzednimi funkcjami.