Python - Sortowanie bąbelkowe

Uwagi ogólne

Uwaga! Wcięcia w kodzie, mają dla Pythona znaczenie (są konieczne w odpowiednich miejscach)! Python do pobrania (dla Windows): *https://www.python.org/downloads/windows/*

Uruchamianie skryptu:

C:\> **python naszSkrypt.py**

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii i tzw. twarde odstępy. Kod należy przepisać ze zrozumieniem.

Nasz skrypt

Jeśli chcemy posortować od najmniejszej do największej liczby, porównujemy najpierw dwie sąsiadujące liczby (3 i 1) i ewentualnie zamieniamy je miejscami. Na przykład:

8
12
5
1
3

W ten sposób, po kilku przebiegach, najmniejsza liczba przesunięta zostanie na samą górę (uniesie się jak bąbelek w szampanie).

Przedstawienie procedury w ujęciu poziomym (pierwszy przebieg pętli FOR):

8, 12, 5, 1, 3
8 ? 12, 5, 1, 3
8, **12 ? 5**, 1, 3
8, 5, **12 ? 1**, 3
8, 5, 1, **12 ? 3**
8, 5, 1, 3, 12

Drugi przebieg pętli FOR:

8, 5, 1, 3, 12
8 ? 5, 1, 3, 12
5, **8 ? 1**, 3, 12
5, 1, **8 ? 3**, 12
5, 1, 3, **8 ? 12**
5, 1, 3, 8, 12

Trzeci przebieg pętli FOR:

5, 1, 3, 8, 12
5 ? 1, 3, 8, 12
1, **5 ? 3**, 8, 12
1, 3, **5 ? 8**, 12
1, 3, 5, **8 ? 12**
1, 3, 5, 8, 12

Kod skryptu - wersja podstawowa

```
def Sortowanie_Babelkowe(liczby):
    czyKolejnyPrzebieg = 'TAK'
    while czyKolejnyPrzebieg == 'TAK':
        czyKolejnyPrzebieg = 'NIE'
        for x in range(len(liczby)-1):
            if liczby[x] > liczby[x+1]:
                liczby[x], liczby[x+1] = liczby[x+1], liczby[x]
                czyKolejnyPrzebieg = 'TAK'
        print(liczby)

Sortowanie_Babelkowe([18,12,5,9,1,-3])
input('Naciśnij ENTER, aby zakończyć...')
```

Kod skryptu - wersja rozbudowana

```
#!/usr/bin/env python
# Powyższa linia tylko dla osób korzystających z systemu Linux.

# ========= SORTOWANIE BĄBELKOWE ===============
# Dzielimy ciąg liczb na pary i sprawdzamy, która z nich jest mniejsza.
# Liczbę mniejszą umieszczamy jako pierwszą, a następnie drugą porównujemy
# z trzecią, itd. Cały proces powtarzamy do momentu, aż podczas
# kolejnego przejścia pętli nie zajdą żadne zmiany.

from os import system # Wymagane do kolorowana składni w systemie Windows 10/11:
system("")
# Uwaga: Składnia nie będzie kolorowana, gdy uruchomimy kod w IDLE Shell (to nie jest
# prawdziwy terminal) oraz w Windows 7/8.

def Sortowanie_Babelkowe(pobraneLiczbyJakoIntegerLista):
    # Wstępnie zakładamy, że ciąg liczb wymaga posortowania:
    czyKolejnyPrzebieg = 'TAK'
    # Wykonuj kolejne serie pętli FOR dopoki zmienna ustawiona jest na 'TAK':
    while czyKolejnyPrzebieg == 'TAK':
    # Każda z podanych liczb zostanie porównana z następną w jednym pełnym przebiegu pętli FOR. Ale pojedynczy przebieg pętli FOR nie gwarantuje jeszcze posortowania wszystkich liczb. Musi być powtarzany do momentu, aż żadna z kolejnych liczb nie będzie większa od następnej. Poniżej, domyślnie przyjmujemy, że niepotrzebny będzie kolejny przebieg pętli FOR, chyba że instrukcja IF stwierdzi inaczej:
        czyKolejnyPrzebieg = 'NIE'
        for kolejnaLiczba in range(len(pobraneLiczbyJakoIntegerLista)-1):
            if pobraneLiczbyJakoIntegerLista[kolejnaLiczba] >
pobraneLiczbyJakoIntegerLista[kolejnaLiczba+1]:
                # Jeśli liczba pierwsza jest większa niż liczba druga, liczby zamieniają się
                # miejscami:
                pobraneLiczbyJakoIntegerLista[kolejnaLiczba],
pobraneLiczbyJakoIntegerLista[kolejnaLiczba+1] =
pobraneLiczbyJakoIntegerLista[kolejnaLiczba+1], pobraneLiczbyJakoIntegerLista[kolejnaLiczba]
```

```
                # Jeśli podczas kolejnej pętli FOR nastąpiła chociaż jedna zmiana miejsc, to jest powód, aby przypuszczać, że całość nie jest jeszcze odpowiednio posortowana. Inicjujemy więc kolejny przebieg pętli WHILE, aby to jeszcze raz sprawdzić.
                    czyKolejnyPrzebieg = 'TAK'
                # Wyświetlanie kolejnych etapów sortowania:
                print("\n\033[0;31;40m", pobraneLiczbyJakoIntegerLista, "\033[0m \n")
        print("\nOstateczny efekt sortowania liczb:\n")

        return pobraneLiczbyJakoIntegerLista

try:
        print("\n\n\n\033[0;37;45m ===== SORTOWANIE BĄBELKOWE ===== \033[0m")
        # Pobranie liczb od użytkownika jako pojedynczy string, podzielenie go na autonomiczne stringi (kryterium podziału to spacja) i przekształcenie w listę stringów:
        pobraneLiczbyJakoStringLista = input("\nPodaj kilka liczb całkowitych, oddzielonych spacją: \033[1;36;40m").split(" ")
        # Mapuje na integer każdą liczbę w drugim argumencie, po czym zamienia na listę:
        pobraneLiczbyJakoIntegerLista = list(map(int, pobraneLiczbyJakoStringLista))
        print("\n\n\033[1;30;40mKolejne etaty sortowania:\033[0m\n")

        print("\033[1;33;40m", Sortowanie_Babelkowe(pobraneLiczbyJakoIntegerLista), "\033[0m")
except ValueError:
        print("\n\n\033[1;37;41m BŁĄD: \033[0m")
        print("\033[1;31;40m1. Wprowadzone wartości albo nie są liczbami...\n2. Albo zawierają spację na końcu...\n3. Albo w ogóle nie wprowadzono żadnych wartości.\n\nSpróbuj jeszcze raz.\033[0m\n")
finally:
        input('\n\n\033[1;30;40mNaciśnij ENTER, aby zakończyć...\033[0m\n')
```

Ostatnia aktualizacja: 6 listopada 2023.