Zadania z programowania w języku Python
listy i zbiory

dr Agnieszka Zbrzezny

Listy

1. Zakładając, iż dana jest lista `lista = []`, napisać jedną lub dwie instrukcje iteracyjne **`for`** wypełniające listę `lista` następującymi liczbami:
   (a) `1 2 3 4 5 6 7 8 9 10`
   (b) `0 2 4 6 8 10 12 14 16 18 20`
   (c) `1 4 9 16 25 36 49 64 81 100`
   (d) `0 0 0 0 0 0 0 0 0 0`
   (e) `0 1 0 1 0 1 0 1 0 1`
   (f) `0 1 2 3 4 0 1 2 3 4`
2. Zrób poprzednie zadanie używając instrukcji iteracyjnej **`while`**.
3. Napisz funkcję `ile_ujemnych`, która zwraca jako swój wynik liczbę ujemnych elementów listy podanej jako jedyny argument tej funkcji. Przetestuj tę funkcję w funkcji `main`.
4. Napisz funkcję `iloczyn`, która zwraca jako swój wynik iloczyn liczb będących elementami listy podanej jako jedyny argument tej funkcji. Przetestuj tę funkcję w funkcji `main`.
5. Napisz funkcję `minmax`, która zwraca jako swój wynik krotkę dwóch liczbi, z których pierwsza to minimum a druga yo maksimum z listy podanej jako jedyny argument tej funkcji. Przykładowo, dla listy `a = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]` funkcja `minmax` powinna zwrócić krotkę `(28, 31)`. Przetestuj tę funkcję w funkcji `main`.
6. Napisz funkcję obliczającą sumę przemienną wszystkich elementów na liście. Przykładowo, dla listy `1 4 16 9 9 7 4 9 11` funkcja powinna obliczyć wynik funkcji jako `1 - 4 + 16 - 9 + 9 - 7 + 4 - 9 + 11 = -2` Przetestuj tę funkcję w funkcji `main`.
7. Napisz program, który wczytuje kolejną liczbę i dodaje ją do listy, o ile nie występuje ona jeszcze na liscie. Gdy lista zawiera dziesięć liczb, program wyświetla jej zawartość i kończy pracę.

8. Napisz program, który dodaje wszystkie liczby od 2 do 10000 do początkowo pustej listy, po czym usuwa wielokrotności 2 (ale nie 2), wielokrotności 3 (ale nie 3), i tak dalej, aż do wielokrotności 100. Następnie program wypisuje liczby pozostałe na liście.

9. Dla każdego z poniższych punktów napisz funkcję, która wykonuje odpowiednie zadanie dla listy liczb całkowitych będącej jej argumentem. Przetestuj każdą funkcję w funkcji `main`.

   (a) Zamienia miejscami pierwszy i ostatni element w liście.

   (b) Przesuwa wszystkie elementu o jedną pozycję w prawo a ostatni element przenosi na poczatek listy. Przykładowo, lista `[1 4 9 16 25]` ma być przekształcona w listę `[25 1 4 9 16]`.

   (c) Zastępuje wszystkie parzyste elementy listy liczbą `0`.

   (d) Zastępuje każdy element listy, z wyjątkiem pierwszego i ostatniego, przez większą z dwóch sąsiednich elementów.

   (e) Usuwa element środkowy, jeśli długość lista jest nieparzysta, lub dwa środkowe elementy, jeśli długość jest parzysta.

   (f) Przenosi wszystkie parzyste elementy listy na jej poczatek, z zachowaniem kolejności elementów.

   (g) Zwraca drugi co do wielkości element na liście.

   (h) Zwraca **`True`**, jeśli lista jest posortowana w porządku rosnącym. W przeciwnym przypadku zwraca **`False`**.

   (i) Zwraca **`True`**, jeśli lista zawiera jakiekolwiek dwie sąsiadujące równe elementy. W przeciwnym przypadku zwraca **`False`**.

   (j) Zwraca **`True`**, jeśli lista zawiera jakiekolwiek dwa równe elementy (które nie muszą znajdować się obok siebie). W przeciwnym przypadku zwraca **`False`**.

10. Napisz funkcję `equals(a, b)`, która sprawdza, czy dwie listy są identyczne, to znaczy mają tyle samo takich samych elementów (i w tej samej kolejności). Przetestuj tę funkcję w funkcji `main`.