Zaawansowane techniki programowania w języku Python

Lista zadań – Generatory

Zadanie 1 - Even

Zaimplementuj własny generator o nazwie `even`, zwracający kolejne liczby parzyste. Generator powinien zwracać wartości w nieskończoność.

Przykład:

```
even() → 0 2 4...
```

Wersja rozszerzona: Uzupełnij generator o możliwość zwracania kolejnych liczb, rozpoczynając od wartości ustalonej przez użytkownika generatora.

Przykład:

```
even(10) → 10 12 14...
```

Co powinno się stać, gdy użytkownik generatora jako wartość startową poda liczbę nieparzystą?

Zadanie 2 – Repeat

Zaimplementuj własny generator o nazwie `repeat`, zwracający obiekt podany przez użytkownika dokładnie `times` razy. Jeśli wartość parametru `times` nie została określona, generator powinien zwracać wartości w nieskończoność.

Przykład:

```
repeat(10, 3) → 10 10 10
repeat(10, 5) → 10 10 10 10 10
repeat(5) → 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5…
repeat(5, None) → 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5…
```

Notatki:

.....

.....

Zadanie 3 – Accumulate

Zaimplementuj własny generator o nazwie `accumulate`, zwracający kolejne sumy elementów z podanego obiektu iterowalnego.

Przykład:

```
accumulate([1,2,3,4,5]) →  1 3 6 10 15
accumulate(["ala ", "ma", " kota"]) → 'ala ', 'ala ma', 'ala ma kota'
```

Zadanie 4 - xrange

Wykorzystując pętlę `while`, zaimplementuj własny generator o nazwie xrange[1], zwracający kolejne liczby całkowite. Liczba, od której należy zacząć odliczanie, a także liczba, do której należy odliczać, powinna być podana jako argument.

Przykład:

```
xrange(1, 6) → 1 2 3 4 5
xrange(5, 10) → 5 6 7 8 9
xrange(5) → 0 1 2 3 4
```

Napisz program, w którym wykorzystasz swój generator do wypisania na ekranie kolejnych 10 liczb całkowitych.

---

repeat, **accumulate** są częścią modułu `itertools`[2].

1 W Pythonie 2 były dostępne dwie funkcje: range oraz xrange. Ta pierwsza zwracała listę, druga obiekt iterowalny, którego iterator zwracał kolejne liczby bez potrzeby przechowywania wszystkich w pamięci.

2 zobacz: itertools — Functions creating iterators for efficient looping, https://docs.python.org/3/library/itertools.html

Notatki:

.....

.....