Programowanie I R - zadania 15

Poziom 1: Podstawowy

Zadanie 1: **Prosty iterator** Napisz klasę `Counter`, która działa jak iterator i zwraca kolejne liczby całkowite od 1 do `n`.

```
counter = Counter(5)
for num in counter:
    print(num)
```

Zadanie 2: **Generator liczb parzystych** Napisz funkcję generatorową `even_numbers(n)`, która zwraca parzyste liczby od 0 do `n` (włącznie).

Zadanie 3: **Co się stanie?** Wyjaśnij, co się stanie po uruchomieniu poniższego kodu. Podaj wynik i uzasadnienie.

```
gen = (x**2 for x in range(5))
print(next(gen))
print(next(gen))
print(list(gen))
```

Poziom 2: Średniozaawansowany

Zadanie 4: **Generator słów** Napisz funkcję generatorową `word_by_word(text)`, która zwraca kolejne słowa z danego ciągu tekstowego.

```
for word in word_by_word("Ala ma kota"):
    print(word)
```

Zadanie 5: **Kalendarz** Napisz klasę `DaysIterator`, która iteruje po nazwach dni tygodnia w pętli (np. pon., wt., śr., ..., nd., pon., wt., ...).

```
days = DaysIterator()
for i in range(10):
    print(next(days))
```

Zadanie 6: **Generator nieskończony** Napisz generator `fibonacci()`, który zwraca nieskończoną sekwencję liczb Fibonacciego.

Poziom 3: Zaawansowany

Zadanie 7: **Plik jako iterator** Napisz klasę `FileLineIterator`, która działa jak iterator po liniach pliku. Dodaj obsługę błędów i zamykanie pliku po zakończeniu.

Zadanie 8: **Stronicowanie** Napisz funkcję generatorową `paginate(items, size)`, która zwraca kolejne strony (listy) elementów o długości `size`.

```
pages = paginate([1, 2, 3, 4, 5, 6, 7], 3)
for page in pages:
    print(page)
```

Wynik:

```
[1, 2, 3]
[4, 5, 6]
[7]
```

Zadanie 9: **Własny range** Zaimplementuj klasę `MyRange`, która działa jak `range(start, stop, step)` – ale nie używa wbudowanego `range()`.

Zadanie 10: **Generator z `send()`** Napisz generator `accumulator()`, który sumuje liczby wysyłane przez `send()`.

```
acc = accumulator()
next(acc)
acc.send(5)   # 5
acc.send(3)   # 8
acc.send(-2) # 6
```