# Testy jednostkowe

Wskazówki:

- logikę piszemy w głównym katalogu projektu, lub jakimś podkatalogu (jak Wam wygodnie),
- testy jednostkowe piszemy w podkatalogu `tests` (jest to dobra praktyka)
- do uruchamiania testów zalecam używać bibliotekę `pytest`, dokumentacja: https://docs.pytest.org/en/stable/

**Przed implementacją należy stworzyć zadanie na Clickup, tak aby w commitach umieszczać ID zadania.**

Funkcje do zaimplementowania:

1. `is_palindrome(text: str) -> bool` - sprawdza, czy dany ciąg znaków jest palindromem (ignorując wielkość liter i spacje).
2. `fibonacci(n: int) -> int` - zwraca n-ty element ciągu Fibonacciego (Załóż, że `fibonacci(0) == 0`, `fibonacci(1) == 1`.).
3. `count_vowels(text: str) -> int` - zlicza liczbę samogłosek w podanym ciągu (a, e, i, o, u, y – wielkość liter bez znaczenia).
4. `calculate_discount(price: float, discount: float) -> float` - zwraca cenę po uwzględnieniu zniżki (np. calculate_discount(100, 0.2) → 80).
   Jeśli discount jest spoza zakresu 0–1, ma zostać zgłoszony wyjątek ValueError.
5. `flatten_list(nested_list: list) -> list` - przyjmuje listę (mogącą zawierać zagnieżdżone listy) i zwraca ją „spłaszczoną”.
   Przykład: [1, [2, 3], [4, [5]]] → [1, 2, 3, 4, 5]
6. `word_frequencies(text: str) -> dict` - zwraca słownik z częstością występowania słów w tekście (ignorując wielkość liter i interpunkcję).

7. `is_prime(n: int) → bool` - sprawdza, czy liczba jest pierwsza.
   Jeśli n < 2, zwraca False.

Testy do zaimplementowania:

1. `is_palindrome(text)`
   - `"kajak"` → `True`
   - `"Kobyła ma mały bok"` → `True`
   - `"python"` → `False`
   - `""` (pusty ciąg) → `True`
   - `"A"` (pojedynczy znak) → `True`
2. `fibonacci(n)`
   - `fibonacci(0)` → `0`
   - `fibonacci(1)` → `1`
   - `fibonacci(5)` → `5`
   - `fibonacci(10)` → `55`
   - `fibonacci(-1)` → oczekiwany wyjątek `ValueError` lub `None` (zależnie od implementacji)
3. `count_vowels(text)`
   - `"Python"` → `1`
   - `"AEIOUY"` → `6`
   - `"bcd"` → `0`
   - `""` → `0`
   - `"Próba żółwia"` → `4` (uwzględnić polskie znaki, jeśli obsługiwane)
4. `calculate_discount(price, discount)`
   - `calculate_discount(100, 0.2)` → `80.0`
   - `calculate_discount(50, 0)` → `50.0`

- `calculate_discount(200, 1)` → `0.0`
- `calculate_discount(100, -0.1)` → wyjątek `ValueError`
- `calculate_discount(100, 1.5)` → wyjątek `ValueError`

5. `flatten_list(nested_list)`
   - `[1, 2, 3]` → `[1, 2, 3]`
   - `[1, [2, 3], [4, [5]]]` → `[1, 2, 3, 4, 5]`
   - `[]` → `[]`
   - `[[[1]]]` → `[1]`
   - `[1, [2, [3, [4]]]]` → `[1, 2, 3, 4]`
6. `word_frequencies(text)`
   - `"To be or not to be"` → `{"to": 2, "be": 2, "or": 1, "not": 1}`
   - `"Hello, hello!"` → `{"hello": 2}`
   - `""` → `{}`
   - `"Python Python python"` → `{"python": 3}`
   - `"Ala ma kota, a kot ma Ale."` → poprawność ignorowania interpunkcji\
7. `is_prime(n)`
   - `2` → `True`
   - `3` → `True`
   - `4` → `False`
   - `0` → `False`
   - `1` → `False`
   - `5` → `False`
   - `97` → `True`

**Zadania należy umieścić na branchu** `unit_tests`