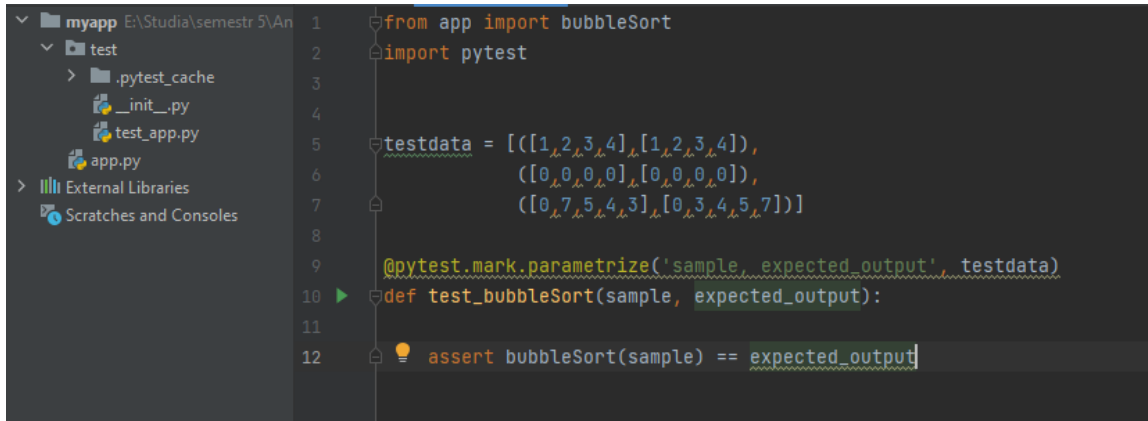


## SPRAWOZDANIE LAB13


Agata Słonka AiBD gr1  
407121

1. **Faza Red:** napisanie testu do algorytmu sortowania bąbelkowego (samo sortowanie bąbelkowe nie jest zaimplementowane)



```
1 from app import bubbleSort
2 import pytest
3
4
5 testdata = [([1,2,3,4],[1,2,3,4]),
6             ([0,0,0,0],[0,0,0,0]),
7             ([0,7,5,4,3],[0,3,4,5,7])]
8
9 @pytest.mark.parametrize('sample, expected_output', testdata)
10 def test_bubbleSort(sample, expected_output):
11
12     assert bubbleSort(sample) == expected_output
```

2. **Faza Green:** implementacja algorytmu w pliku app, tak, aby przechodziły testy

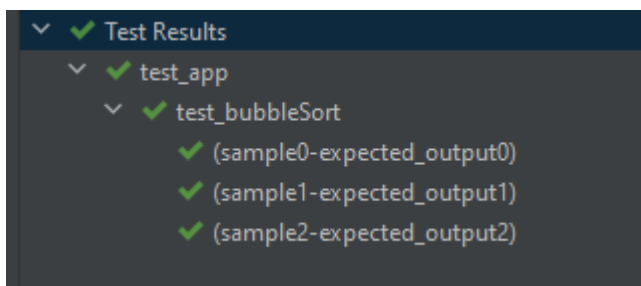


```
def bubbleSort(list1):
    has_swapped = True

    total_iteration = 0

    while has_swapped:
        has_swapped = False
        for i in range(len(list1) - total_iteration - 1):
            if list1[i] > list1[i + 1]:
                # Swap
                list1[i], list1[i + 1] = list1[i + 1], list1[i]
                has_swapped = True
            total_iteration += 1

    return list1
```



```
✓ Test Results
  ✓ test_app
    ✓ test_bubbleSort
      ✓ (sample0-expected_output0)
      ✓ (sample1-expected_output1)
      ✓ (sample2-expected_output2)
```

3. **Faza Refactor:** refaktoryzacja, czyli ulepszenie istniejącego kodu. Jako że funkcja, którą sprawdzaliśmy była bardzo prosta, w tej fazie pojawiła się tylko jedna zmiana: sprawdzanie argumentów wejściowych.

```
def bubbleSort(list1):  
  
    if not all(isinstance(num, int) for num in list1):  
        return False  
  
    has_swapped = True  
    total_iteration = 0  
  
    while has_swapped:  
        has_swapped = False  
        for i in range(len(list1) - total_iteration - 1):  
            if list1[i] > list1[i + 1]:  
                # Swap  
                list1[i], list1[i + 1] = list1[i + 1], list1[i]  
                has_swapped = True  
            total_iteration += 1  
  
    return list1
```

**WNIOSKI:** Pytest pozwala na szybkie sprawdzanie poprawności naszej funkcji. Piszac test nie musimy posiadać jeszcze gotowego algorytmu (byłoby to głupie, gdybyśmy musieli - testy mają nam pomóc właśnie w napisaniu go) jedynie musimy wiedzieć, jak chcemy, aby nasz algorytm działał. Pozwala nam na szybkie sprawdzenie poprawności działania funkcji dla wielu argumentów, co znacząco ułatwia znajdowanie teoretycznych błędów w implementacji