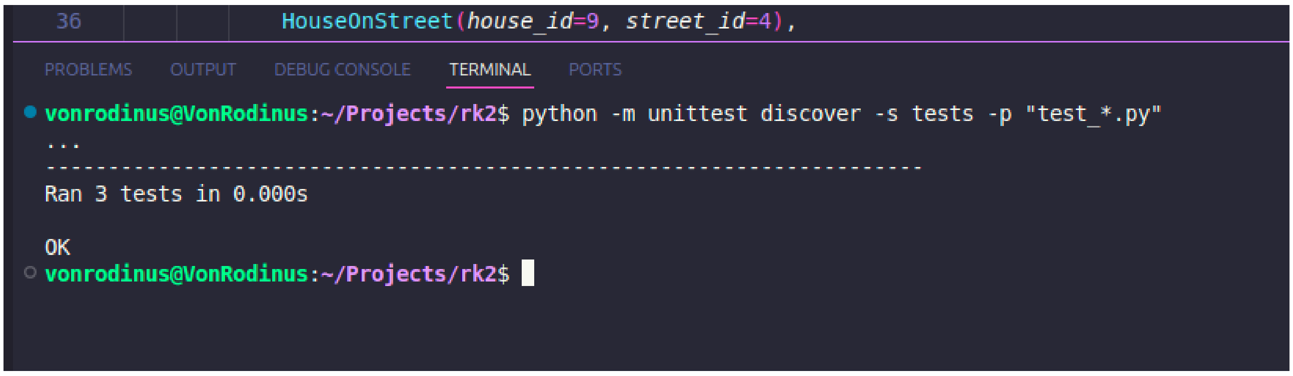
```
from dataclasses import dataclass
from typing import List, Dict


@dataclass
class House:
house_id: int
address: str
apartment_count: int
street_id: int

@dataclass
class Street:
street_id: int
name: str

@dataclass
class HouseOnStreet:
house_id: int
street_id: int


def list_houses_by_street(houses: List[House], streets: List[Street]) -> List[tuple]:
result = []

for street in sorted(streets, key=lambda s: s.name):

sorted_houses = sorted(filter(lambda h: h.street_id == street.street_id, houses), key=lambda h:
h.address)
for house in sorted_houses:
result.append((house.address, house.apartment_count, street.name))
return result


def list_streets_with_total_apartments(houses: List[House], streets: List[Street]) -> List[tuple]:
street_apartments = {street.street_id: 0 for street in streets}

for house in houses:
street_apartments[house.street_id] += house.apartment_count

sorted_streets = sorted(streets, key=lambda s: street_apartments[s.street_id], reverse=True)
return [(street.name, street_apartments[street.street_id]) for street in sorted_streets]

def list_streets_with_houses_with_keyword(streets: List[Street], houses_on_streets:
List[HouseOnStreet], houses: List[House], keyword: str = "улиц") -> Dict[str, List[str]]:
result = {}
for street in filter(lambda s: keyword in s.name.lower(), streets):
result[street.name] = [
next(h for h in houses if h.house_id == link.house_id).address
for link in filter(lambda hs: hs.street_id == street.street_id, houses_on_streets)
]
return result
```

```
import unittest
from src.main import House, Street, HouseOnStreet, list_houses_by_street,
list_streets_with_total_apartments, list_streets_with_houses_with_keyword

class TestHouseFunctions(unittest.TestCase):
def setUp(self):
self.streets = [
Street(street_id=1, name="Улица Мира"),
Street(street_id=2, name="Пролетарская"),
Street(street_id=3, name="Октябрьская"),
Street(street_id=4, name="Улица Победы")
]
self.houses = [
House(house_id=1, address="Дом 1", apartment_count=10, street_id=1),
House(house_id=2, address="Дом 2", apartment_count=15, street_id=1),
House(house_id=3, address="Дом 3", apartment_count=20, street_id=2),
House(house_id=4, address="Дом 4", apartment_count=25, street_id=2),
House(house_id=5, address="Дом 5", apartment_count=23, street_id=3),
House(house_id=6, address="Дом 6", apartment_count=13, street_id=3),
House(house_id=7, address="Дом 7", apartment_count=15, street_id=3),
House(house_id=8, address="Дом 8", apartment_count=10, street_id=4),
House(house_id=9, address="Дом 9", apartment_count=30, street_id=4),
House(house_id=10, address="Дом 10", apartment_count=20, street_id=4),
House(house_id=11, address="Дом 11", apartment_count=10, street_id=4)
]
self.houses_on_streets = [
HouseOnStreet(house_id=1, street_id=1),
HouseOnStreet(house_id=2, street_id=1),
HouseOnStreet(house_id=3, street_id=2),
HouseOnStreet(house_id=4, street_id=2),
HouseOnStreet(house_id=5, street_id=3),
HouseOnStreet(house_id=6, street_id=3),
HouseOnStreet(house_id=7, street_id=3),
HouseOnStreet(house_id=8, street_id=4),
HouseOnStreet(house_id=9, street_id=4),
HouseOnStreet(house_id=10, street_id=4),
HouseOnStreet(house_id=11, street_id=4)
]

def test_list_houses_by_street(self):
result = list_houses_by_street(self.houses, self.streets)
self.assertEqual(result[0], ('Дом 5', 23, 'Октябрьская'))

def test_list_streets_with_total_apartments(self):
result = list_streets_with_total_apartments(self.houses, self.streets)
self.assertEqual(result[0], ('Улица Победы', 70))

def test_list_streets_with_houses_with_keyword(self):
result = list_streets_with_houses_with_keyword(self.streets, self.houses_on_streets, self.houses)
self.assertTrue('Улица Мира' in result)
```

Результаты тестов представлены ниже:



```
HouseOnStreet(house_id=9, street_id=4),
```

```
vonrodinus@VonRodinus:~/Projects/rk2$ python -m unittest discover -s tests -p "test_*.py"
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
vonrodinus@VonRodinus:~/Projects/rk2$ 
```