Канаева Диана ИУ5-32Б

Текст программы

main.py

```
class HardDrive:
    def __init__(self, hd_id, capacity, name):
        self.hd_id = hd_id
        self.capacity = capacity
        self.name = name

class PC:
    def __init__(self, pc_id, ram, cores):
        self.pc_id = pc_id
        self.ram = ram
        self.cores = cores

class PC_HD:
    def __init__(self, pc_id, hd_id):
        self.hd_id = hd_id
        self.pc_id = pc_id


PCs = [
    PC(1, "DDR4-2400", 4),
    PC(2, "DDR3-1600", 8),
    PC(3, "DDR4-3200", 6),
    PC(4, "DDR3-1866", 4),
    PC(5, "DDR4-2666", 12)
]

HDs = [
    HardDrive(1, "1TB", "HDD-1TB"),
    HardDrive(2, "500GB", "SSD-500GB"),
    HardDrive(3, "1TB",  "NVMe-1TB"),
    HardDrive(4, "2TB", "SATA-2TB"),
    HardDrive(5, "2TB", "SSHD-2TB"),
    HardDrive(6, "1TB", "SSD-1TB"),
    HardDrive(7, "2TB", "HDD-2TB"),
    HardDrive(8, "500GB", "NVMe-500GB")
]

PC_HDs = [
    PC_HD(1, 1),
    PC_HD(1, 8),
    PC_HD(2, 2),
    PC_HD(2, 6),
    PC_HD(3, 4),
    PC_HD(3, 5),
    PC_HD(4, 3),
    PC_HD(5, 7)
]

def b1_solution(HDs, PCs, PC_HDs):
    sorted_HDs = sorted(HDs, key=lambda x: x.name)
    result = []
    for hd in sorted_HDs:
        for pc_hd in PC_HDs:
            if pc_hd.hd_id == hd.hd_id:
                for pc in PCs:
                    if pc_hd.pc_id == pc.pc_id:
                        result.append((hd.name, hd.capacity, pc.ram,
pc.cores))
                        break
```

```
                break
    return result

def b2_solution(PCs, PC_HDs):
    result = []
    for pc in PCs:
        count = 0
        for pc_hd in PC_HDs:
            if pc.pc_id == pc_hd.pc_id:
                count += 1
        result.append((pc.ram, pc.cores, count))
    sorted_result = sorted(result, key=lambda x: x[2])
    return sorted_result

def b3_solution(HDs, PCs, PC_HDs):
    result = []
    for hd in HDs:
        if hd.name.endswith("1TB"):
            for pc_hd in PC_HDs:
                if hd.hd_id == pc_hd.hd_id:
                    for pc in PCs:
                        if pc_hd.pc_id == pc.pc_id:
                            result.append((hd.name, pc.ram))
                            break
                    break
    return result

def main():
    print('Задание B1')
    print(b1_solution(HDs, PCs, PC_HDs))

    print('\nЗадание B2')
    print(b2_solution(PCs, PC_HDs))

    print('\nЗадание B3')
    print(b3_solution(HDs, PCs, PC_HDs))


if __name__ == '__main__':
    main()
```

test.py

```
import unittest
from main import *

class TestProgram(unittest.TestCase):
    def setUp(self):
        self.PCs = [
            PC(1, "DDR4-2400", 4),
            PC(2, "DDR3-1600", 8),
            PC(3, "DDR4-3200", 6),
            PC(4, "DDR3-1866", 4),
            PC(5, "DDR4-2666", 12)
        ]

        self.HDs = [
            HardDrive(1, "1TB", "HDD-1TB"),
            HardDrive(2, "500GB", "SSD-500GB"),
            HardDrive(3, "1TB",  "NVMe-1TB"),
            HardDrive(4, "2TB", "SATA-2TB"),
            HardDrive(5, "2TB", "SSHD-2TB"),
            HardDrive(6, "1TB", "SSD-1TB"),
            HardDrive(7, "2TB", "HDD-2TB"),
```

```
            HardDrive(8, "500GB", "NVMe-500GB")
        ]

        self.PC_HDs = [
            PC_HD(1, 1),
            PC_HD(1, 8),
            PC_HD(2, 2),
            PC_HD(2, 6),
            PC_HD(3, 4),
            PC_HD(3, 5),
            PC_HD(4, 3),
            PC_HD(5, 7)
        ]

    def test_1(self):
        expected_result = [
            ('HDD-1TB', '1TB', 'DDR4-2400', 4),
            ('HDD-2TB', '2TB', 'DDR4-2666', 12),
            ('NVMe-1TB', '1TB', 'DDR3-1866', 4),
            ('NVMe-500GB', '500GB', 'DDR4-2400', 4),
            ('SATA-2TB', '2TB', 'DDR4-3200', 6),
            ('SSD-1TB', '1TB', 'DDR3-1600', 8),
            ('SSD-500GB', '500GB', 'DDR3-1600', 8),
            ('SSHD-2TB', '2TB', 'DDR4-3200', 6)
        ]
        result = b1_solution(self.HDs, self.PCs, self.PC_HDs)
        self.assertEqual(result, expected_result)

    def test_2(self):
        expected_result = [
            ('DDR3-1866', 4, 1),
            ('DDR4-2666', 12, 1),
            ('DDR4-2400', 4, 2),
            ('DDR3-1600', 8, 2),
            ('DDR4-3200', 6, 2)
        ]
        result = b2_solution(self.PCs, self.PC_HDs)
        self.assertEqual(result, expected_result)

    def test_3(self):
        expected_result = [
            ('HDD-1TB', 'DDR4-2400'),
            ('NVMe-1TB', 'DDR3-1866'),
            ('SSD-1TB', 'DDR3-1600')
        ]
        result = b3_solution(self.HDs, self.PCs, self.PC_HDs)
        self.assertEqual(result, expected_result)

if __name__ == "__main__":
    unittest.main()
```

Результаты выполнения тестов

Пример успешного прохождения тестов:

```
C:\Users\123\Documents\PCPL_3sem\RK2>python test.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```

Пример неудачного прохождения тестов:

```
C:\Users\123\Documents\PCPL_3sem\RK2>python test.py
.F.
======================================================================
FAIL: test_2 (__main__.TestProgram)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\123\Documents\PCPL_3sem\RK2\test.py", line 59, in test_2
    self.assertEqual(result, expected_result)
AssertionError: Lists differ: [('DDR3-1866', 4, 1), ('DDR4-2666', 12, 1), ('D[54 chars], 2)] != [('DDR4-2666', 12, 1), (
'DDR3-1866', 4, 1), ('D[54 chars], 2)]

First differing element 0:
('DDR3-1866', 4, 1)
('DDR4-2666', 12, 1)

- [('DDR3-1866', 4, 1),
-  ('DDR4-2666', 12, 1),
? ^

+ [('DDR4-2666', 12, 1),
? ^

+  ('DDR3-1866', 4, 1),
   ('DDR4-2400', 4, 2),
   ('DDR3-1600', 8, 2),
   ('DDR4-3200', 6, 2)]

----------------------------------------------------------------------
Ran 3 tests in 0.002s

FAILED (failures=1)
```