# Текст программы

## data.py

from entities import Orchestra, Musician, MusicianOrchestra  
  
orchestras = (  
 Orchestra(1, "Mariinsky Theatre Orchestra"),  
 Orchestra(2, "St. Petersburg Philharmonic Orchestra"),  
 Orchestra(3, "Bolshoi Theatre Orchestra"),  
 Orchestra(4, "Moscow City Symphony - Russian Philharmonic"),  
 Orchestra(5, "State Academic Symphony Orchestra of Russia"),  
 Orchestra(6, "Novosibirsk Symphony Orchestra"),  
)  
  
musicians = [  
 Musician(1, "Valery Gergiev", "Conductor", 1),  
 Musician(2, "Vadim Repin", "Violin", 1),  
 Musician(3, "Anna Netrebko", "Soprano", 3),  
 Musician(4, "Daniil Trifonov", "Piano", 2),  
 Musician(5, "Yuri Bashmet", "Viola", 4),  
 Musician(6, "Denis Matsuev", "Piano", 5),  
 Musician(7, "Marina Abramova", "Cello", 2),  
 Musician(8, "Maxim Vengerov", "Violin", 1),  
 Musician(9, "Evgeny Kissin", "Piano", 3),  
 Musician(10, "Natalia Gutman", "Cello", 6),  
]  
  
  
musicians\_orchestras = [  
 MusicianOrchestra(1, 2),  
 MusicianOrchestra(1, 4),  
 MusicianOrchestra(1, 5),  
 MusicianOrchestra(2, 6),  
 MusicianOrchestra(2, 3),  
 MusicianOrchestra(3, 1),  
 MusicianOrchestra(3, 2),  
 MusicianOrchestra(4, 1),  
 MusicianOrchestra(4, 6),  
 MusicianOrchestra(6, 1),  
 MusicianOrchestra(6, 2),  
 MusicianOrchestra(6, 6),  
 MusicianOrchestra(5, 3),  
 MusicianOrchestra(7, 4),  
 MusicianOrchestra(7, 5),  
 MusicianOrchestra(8, 6),  
]  
from entities import Orchestra, Musician, MusicianOrchestra  
  
orchestras = (  
 Orchestra(1, "Mariinsky Theatre Orchestra"),  
 Orchestra(2, "St. Petersburg Philharmonic Orchestra"),  
 Orchestra(3, "Bolshoi Theatre Orchestra"),  
 Orchestra(4, "Moscow City Symphony - Russian Philharmonic"),  
 Orchestra(5, "State Academic Symphony Orchestra of Russia"),  
 Orchestra(6, "Novosibirsk Symphony Orchestra"),  
)  
  
musicians = [  
 Musician(1, "Valery Gergiev", "Conductor", 1),  
 Musician(2, "Vadim Repin", "Violin", 1),  
 Musician(3, "Anna Netrebko", "Soprano", 3),  
 Musician(4, "Daniil Trifonov", "Piano", 2),  
 Musician(5, "Yuri Bashmet", "Viola", 4),  
 Musician(6, "Denis Matsuev", "Piano", 5),  
 Musician(7, "Marina Abramova", "Cello", 2),  
 Musician(8, "Maxim Vengerov", "Violin", 1),  
 Musician(9, "Evgeny Kissin", "Piano", 3),  
 Musician(10, "Natalia Gutman", "Cello", 6),  
]  
  
  
musicians\_orchestras = [  
 MusicianOrchestra(1, 2),  
 MusicianOrchestra(1, 4),  
 MusicianOrchestra(1, 5),  
 MusicianOrchestra(2, 6),  
 MusicianOrchestra(2, 3),  
 MusicianOrchestra(3, 1),  
 MusicianOrchestra(3, 2),  
 MusicianOrchestra(4, 1),  
 MusicianOrchestra(4, 6),  
 MusicianOrchestra(6, 1),  
 MusicianOrchestra(6, 2),  
 MusicianOrchestra(6, 6),  
 MusicianOrchestra(5, 3),  
 MusicianOrchestra(7, 4),  
 MusicianOrchestra(7, 5),  
 MusicianOrchestra(8, 6),  
]

## entities.py

class Musician:  
 def \_\_init\_\_(self, id\_: int, name: str, storage: str, orchestra\_id: int):  
 self.id = id\_  
 self.name = name  
 self.instrument = storage  
 self.orchestra\_id = orchestra\_id  
  
  
class Orchestra:  
 def \_\_init\_\_(self, id\_: int, name: str):  
 self.id = id\_  
 self.name = name  
  
  
class MusicianOrchestra:  
 """  
 Class implementing many-to-many binding  
 """  
  
 def \_\_init\_\_(self, musician\_id: int, orchestra\_id: int):  
 self.musician\_id = musician\_id  
 self.orchestra\_id = orchestra\_id  
class Musician:  
 def \_\_init\_\_(self, id\_: int, name: str, storage: str, orchestra\_id: int):  
 self.id = id\_  
 self.name = name  
 self.instrument = storage  
 self.orchestra\_id = orchestra\_id  
  
  
class Orchestra:  
 def \_\_init\_\_(self, id\_: int, name: str):  
 self.id = id\_  
 self.name = name  
  
  
class MusicianOrchestra:  
 """  
 Class implementing many-to-many binding  
 """  
  
 def \_\_init\_\_(self, musician\_id: int, orchestra\_id: int):  
 self.musician\_id = musician\_id  
 self.orchestra\_id = orchestra\_id

## main.py

import unittest  
import data  
from typing import List, Tuple  
from entities import Orchestra, Musician, MusicianOrchestra  
  
  
def get\_musicians\_with\_orchestras(  
 orchestras: List[Orchestra], musicians: List[Musician]  
) -> List[Tuple[str, str]]:  
 return sorted(  
 [  
 (musician.name, orchestra.name)  
 for musician in musicians  
 for orchestra in orchestras  
 if orchestra.id == musician.orchestra\_id  
 ]  
 )  
  
  
def get\_orchestras\_with\_musicians\_count(  
 orchestras: List[Orchestra], musicians: List[Musician]  
) -> List[Tuple[str, int]]:  
 return sorted(  
 [  
 (orchestra.name, musicians\_count)  
 for orchestra in orchestras  
 for musicians\_count in [  
 len(  
 [  
 musician  
 for musician in musicians  
 if musician.orchestra\_id == orchestra.id  
 ]  
 )  
 ]  
 ],  
 key=lambda item: item[1],  
 reverse=True,  
 )  
  
  
def get\_musicians\_orhestras\_by\_ending\_substring(  
 orechestras: List[Orchestra],  
 musicians: List[Musician],  
 musicians\_orchestras: List[MusicianOrchestra],  
 substring: str,  
) -> List[Tuple[str, List[str]]]:  
 return [  
 (musician.name, orchestras\_of\_musician)  
 for musician in musicians  
 for orchestras\_of\_musician in [  
 [  
 orchestra.name  
 for orchestra in orechestras  
 for musician\_orchestra in musicians\_orchestras  
 if orchestra.id == musician\_orchestra.orchestra\_id  
 and musician.id == musician\_orchestra.musician\_id  
 ]  
 ]  
 if musician.name.endswith(substring)  
 ]  
  
  
def main():  
 print("Запрос Б1")  
 print(get\_musicians\_with\_orchestras(data.orchestras, data.musicians))  
  
 print("Запрос Б2")  
 print(get\_orchestras\_with\_musicians\_count(data.orchestras, data.musicians))  
  
 print("Запрос Б3")  
 print(  
 get\_musicians\_orhestras\_by\_ending\_substring(  
 orechestras=data.orchestras,  
 musicians=data.musicians,  
 musicians\_orchestras=data.musicians\_orchestras,  
 substring="ov",  
 )  
 )  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
# %%  
import unittest  
import data  
from typing import List, Tuple  
from entities import Orchestra, Musician, MusicianOrchestra  
  
  
def get\_musicians\_with\_orchestras(  
 orchestras: List[Orchestra], musicians: List[Musician]  
) -> List[Tuple[str, str]]:  
 return sorted(  
 [  
 (musician.name, orchestra.name)  
 for musician in musicians  
 for orchestra in orchestras  
 if orchestra.id == musician.orchestra\_id  
 ]  
 )  
  
  
def get\_orchestras\_with\_musicians\_count(  
 orchestras: List[Orchestra], musicians: List[Musician]  
) -> List[Tuple[str, int]]:  
 return sorted(  
 [  
 (orchestra.name, musicians\_count)  
 for orchestra in orchestras  
 for musicians\_count in [  
 len(  
 [  
 musician  
 for musician in musicians  
 if musician.orchestra\_id == orchestra.id  
 ]  
 )  
 ]  
 ],  
 key=lambda item: item[1],  
 reverse=True,  
 )  
  
  
def get\_musicians\_orhestras\_by\_ending\_substring(  
 orechestras: List[Orchestra],  
 musicians: List[Musician],  
 musicians\_orchestras: List[MusicianOrchestra],  
 substring: str,  
) -> List[Tuple[str, List[str]]]:  
 return [  
 (musician.name, orchestras\_of\_musician)  
 for musician in musicians  
 for orchestras\_of\_musician in [  
 [  
 orchestra.name  
 for orchestra in orechestras  
 for musician\_orchestra in musicians\_orchestras  
 if orchestra.id == musician\_orchestra.orchestra\_id  
 and musician.id == musician\_orchestra.musician\_id  
 ]  
 ]  
 if musician.name.endswith(substring)  
 ]  
  
  
def main():  
 print("Запрос Б1")  
 print(get\_musicians\_with\_orchestras(data.orchestras, data.musicians))  
  
 print("Запрос Б2")  
 print(get\_orchestras\_with\_musicians\_count(data.orchestras, data.musicians))  
  
 print("Запрос Б3")  
 print(  
 get\_musicians\_orhestras\_by\_ending\_substring(  
 orechestras=data.orchestras,  
 musicians=data.musicians,  
 musicians\_orchestras=data.musicians\_orchestras,  
 substring="ov",  
 )  
 )  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
# %%

## tests.py

import unittest  
from collections import Counter  
import data  
from main import (  
 get\_musicians\_with\_orchestras,  
 get\_orchestras\_with\_musicians\_count,  
 get\_musicians\_orhestras\_by\_ending\_substring,  
)  
  
  
class TestFunctions(unittest.TestCase):  
 def setUp(self):  
 self.data = data  
  
 def test\_get\_musicians\_with\_orchestras(self):  
 want = [  
 ("Anna Netrebko", "Bolshoi Theatre Orchestra"),  
 ("Daniil Trifonov", "St. Petersburg Philharmonic Orchestra"),  
 ("Denis Matsuev", "State Academic Symphony Orchestra of Russia"),  
 ("Evgeny Kissin", "Bolshoi Theatre Orchestra"),  
 ("Marina Abramova", "St. Petersburg Philharmonic Orchestra"),  
 ("Maxim Vengerov", "Mariinsky Theatre Orchestra"),  
 ("Natalia Gutman", "Novosibirsk Symphony Orchestra"),  
 ("Vadim Repin", "Mariinsky Theatre Orchestra"),  
 ("Valery Gergiev", "Mariinsky Theatre Orchestra"),  
 ("Yuri Bashmet", "Moscow City Symphony - Russian Philharmonic"),  
 ]  
 actual = get\_musicians\_with\_orchestras(data.orchestras, data.musicians)  
 self.assertCountEqual(want, actual)  
  
 def test\_get\_orchestras\_with\_musicians\_count(self):  
 want = [  
 ("Mariinsky Theatre Orchestra", 3),  
 ("St. Petersburg Philharmonic Orchestra", 2),  
 ("Bolshoi Theatre Orchestra", 2),  
 ("Moscow City Symphony - Russian Philharmonic", 1),  
 ("State Academic Symphony Orchestra of Russia", 1),  
 ("Novosibirsk Symphony Orchestra", 1),  
 ]  
 actual = get\_orchestras\_with\_musicians\_count(data.orchestras, data.musicians)  
 self.assertCountEqual(want, actual)  
  
 def test\_get\_musicians\_orhestras\_by\_ending\_substring(self):  
 want = [  
 (  
 "Daniil Trifonov",  
 ["Mariinsky Theatre Orchestra", "Novosibirsk Symphony Orchestra"],  
 ),  
 ("Maxim Vengerov", ["Novosibirsk Symphony Orchestra"]),  
 ]  
 actual = get\_musicians\_orhestras\_by\_ending\_substring(  
 orechestras=data.orchestras,  
 musicians=data.musicians,  
 musicians\_orchestras=data.musicians\_orchestras,  
 substring="ov",  
 )  
 self.assertCountEqual(want, actual)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()  
import unittest  
from collections import Counter  
import data  
from main import (  
 get\_musicians\_with\_orchestras,  
 get\_orchestras\_with\_musicians\_count,  
 get\_musicians\_orhestras\_by\_ending\_substring,  
)  
  
  
class TestFunctions(unittest.TestCase):  
 def setUp(self):  
 self.data = data  
  
 def test\_get\_musicians\_with\_orchestras(self):  
 want = [  
 ("Anna Netrebko", "Bolshoi Theatre Orchestra"),  
 ("Daniil Trifonov", "St. Petersburg Philharmonic Orchestra"),  
 ("Denis Matsuev", "State Academic Symphony Orchestra of Russia"),  
 ("Evgeny Kissin", "Bolshoi Theatre Orchestra"),  
 ("Marina Abramova", "St. Petersburg Philharmonic Orchestra"),  
 ("Maxim Vengerov", "Mariinsky Theatre Orchestra"),  
 ("Natalia Gutman", "Novosibirsk Symphony Orchestra"),  
 ("Vadim Repin", "Mariinsky Theatre Orchestra"),  
 ("Valery Gergiev", "Mariinsky Theatre Orchestra"),  
 ("Yuri Bashmet", "Moscow City Symphony - Russian Philharmonic"),  
 ]  
 actual = get\_musicians\_with\_orchestras(data.orchestras, data.musicians)  
 self.assertCountEqual(want, actual)  
  
 def test\_get\_orchestras\_with\_musicians\_count(self):  
 want = [  
 ("Mariinsky Theatre Orchestra", 3),  
 ("St. Petersburg Philharmonic Orchestra", 2),  
 ("Bolshoi Theatre Orchestra", 2),  
 ("Moscow City Symphony - Russian Philharmonic", 1),  
 ("State Academic Symphony Orchestra of Russia", 1),  
 ("Novosibirsk Symphony Orchestra", 1),  
 ]  
 actual = get\_orchestras\_with\_musicians\_count(data.orchestras, data.musicians)  
 self.assertCountEqual(want, actual)  
  
 def test\_get\_musicians\_orhestras\_by\_ending\_substring(self):  
 want = [  
 (  
 "Daniil Trifonov",  
 ["Mariinsky Theatre Orchestra", "Novosibirsk Symphony Orchestra"],  
 ),  
 ("Maxim Vengerov", ["Novosibirsk Symphony Orchestra"]),  
 ]  
 actual = get\_musicians\_orhestras\_by\_ending\_substring(  
 orechestras=data.orchestras,  
 musicians=data.musicians,  
 musicians\_orchestras=data.musicians\_orchestras,  
 substring="ov",  
 )  
 self.assertCountEqual(want, actual)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

## Результат выполнения программы

Запрос Б1  
[('Anna Netrebko', 'Bolshoi Theatre Orchestra'), ('Daniil Trifonov', 'St. Petersburg Philharmonic Orchestra'), ('Denis Matsuev', 'State Academic Symphony Orchestra of Russia'), ('Evgeny Kissin', 'Bolshoi Theatre Orchestra'), ('Marina Abramova', 'St. Petersburg Philharmonic Orchestra'), ('Maxim Vengerov', 'Mariinsky Theatre Orchestra'), ('Natalia Gutman', 'Novosibirsk Symphony Orchestra'), ('Vadim Repin', 'Mariinsky Theatre Orchestra'), ('Valery Gergiev', 'Mariinsky Theatre Orchestra'), ('Yuri Bashmet', 'Moscow City Symphony - Russian Philharmonic')]  
Запрос Б2  
[('Mariinsky Theatre Orchestra', 3), ('St. Petersburg Philharmonic Orchestra', 2), ('Bolshoi Theatre Orchestra', 2), ('Moscow City Symphony - Russian Philharmonic', 1), ('State Academic Symphony Orchestra of Russia', 1), ('Novosibirsk Symphony Orchestra', 1)]  
Запрос Б3  
[('Daniil Trifonov', ['Mariinsky Theatre Orchestra', 'Novosibirsk Symphony Orchestra']), ('Maxim Vengerov', ['Novosibirsk Symphony Orchestra'])]

## Результат выполнения тестов

...  
----------------------------------------------------------------------  
Ran 3 tests in 0.000s  
  
OK