# используется для сортировки

from operator import itemgetter

class IDE:

"""Средство разработки"""

def \_\_init\_\_(self, id, name, platform ,license ):

self.id = id

self.name = name

self.platform = platform

self.license = license

class PL:

"""Язык программирования"""

def \_\_init\_\_(self, id, memory, name, type, progParadigm, IDE\_id):

self.id = id

self.memory = memory # кол-во занимаемой памяти языком программирования(мб)

self.name = name

self.type = type

self.progParadigm = progParadigm

self.IDE\_id = IDE\_id

class IDE\_PL:

"""

'Языки программмирования средств разработки' для реализации

связи многие-ко-многим

"""

def \_\_init\_\_(self, PL\_id, IDE\_id):

self.IDE\_id = IDE\_id

self.PL\_id= PL\_id

# Средства разработки

IDEs = [

IDE(1, 'Visual Studio Code', 'Windows, macOS, Linux', 'Бесплатная, открытый код'),

IDE(2, 'IntelliJ ideA', 'Windows, macOS, Linux', 'Коммерческая'),

IDE(3, 'Eclipse', 'Windows, macOS, Linux', 'Бесплатная, открытый код'),

IDE(4, 'PyCharm', 'Windows, macOS, Linux', 'Коммерческая'),

IDE(5, 'Xcode', 'macOS', 'Бесплатная'),

]

# Языки программирования

PLs = [

PL(1, 1,'Python', 'Скриптовый', 'Объектно-ориентированный', 1),

PL(2, 3,'Java', 'Компилируемый', 'Объектно-ориентированный', 3),

PL(3, 2,'JavaScript', 'Скриптовый', 'Объектно-ориентированный', 2),

PL(4, 5,'C++', 'Компилируемый', 'Объектно-ориентированный', 4),

PL(5, 4,'Go', 'Компилируемый', 'Процедурный', 5),

]

IDEs\_PLs = [

IDE\_PL(1, 1),

IDE\_PL(2, 2),

IDE\_PL(3, 1),

IDE\_PL(4, 3),

IDE\_PL(3, 5),

IDE\_PL(2, 1),

IDE\_PL(1, 4),

IDE\_PL(5, 4),

IDE\_PL(1, 3),

IDE\_PL(2, 5),

]

def main():

"""Основная функция"""

# Соединение данных один-ко-многим

one\_to\_many = [(p.name, p.memory, I.name)

for p in PLs

for I in IDEs

if p.IDE\_id == I.id]

# Соединение данных многие-ко-многим

many\_to\_many\_temp = [(I.name, ip.IDE\_id, ip.PL\_id)

for I in IDEs

for ip in IDEs\_PLs

if I.id == ip.IDE\_id]

many\_to\_many = [(p.name, p.memory, IDE\_name)

for IDE\_name, IDE\_id, PL\_id in many\_to\_many\_temp

for p in PLs if p.id == PL\_id]

print('Задание А1')

res\_11 = sorted(one\_to\_many, key=itemgetter(2))

print(res\_11)

print('\nЗадание А2')

res\_12\_unsorted = []

# Перебираем все средства разработки

for I in IDEs:

# Список языков программирования средства разработки

I\_PLs = list(filter(lambda i: i[2] == I.name, one\_to\_many))

if len(I\_PLs) > 0:

# Памяти, занимаемые языками программирвания

I\_memories = [memory for \_, memory, \_ in I\_PLs]

# Суммарная память, занимаемая всеми языками программирвания

I\_memories\_sum = sum(I\_memories)

res\_12\_unsorted.append((I.name, I\_memories\_sum))

# Сортировка по суммарной памяти

res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)

print(res\_12)

print('\nЗадание А3')

res\_13 = {}

# Перебираем все средства разработки

for I in IDEs:

# Список языков программирвания средства разработки

i\_PLs = list(filter(lambda i: i[2] == I.name, many\_to\_many))

# Только название языков программирвания

i\_PLs\_names = [x for x, \_, \_ in i\_PLs]

# Добавляем результат в словарь

# ключ - средство разработки, значение - список языков программирвания

res\_13[I.name] = i\_PLs\_names

print(res\_13)

if \_\_name\_\_ == '\_\_main\_\_':

main()

Результаты работы программы:

Задание А1

[('Java', 3, 'Eclipse'), ('JavaScript', 2, 'IntelliJ ideA'), ('C++', 5, 'PyCharm'), ('Python', 1, 'Visual Studio Code'), ('Go', 4, 'Xcode')]

Задание А2

[('PyCharm', 5), ('Xcode', 4), ('Eclipse', 3), ('IntelliJ ideA', 2), ('Visual Studio Code', 1)]

Задание А3

{'Visual Studio Code': ['Python', 'JavaScript', 'Java'], 'IntelliJ ideA': ['Java'], 'Eclipse': ['C++', 'Python'], 'PyCharm': ['Python', 'Go'], 'Xcode': ['JavaScript', 'Java']}