from operator import itemgetter

class HardDrive:

"""Hard Drive"""

def \_\_init\_\_(self, id, model, capacity\_gb, computer\_id):

self.id = id

self.model = model

self.capacity\_gb = capacity\_gb

self.computer\_id = computer\_id

class Computer:

"""Computer"""

def \_\_init\_\_(self, id, name, type, hard\_drive\_id, cost):

self.id = id

self.name = name

self.type = type

self.hard\_drive\_id = hard\_drive\_id

self.cost = cost

class ComputerHardDrive:

"""'Computers with Hard Drives' for implementing many-to-many relationship"""

def \_\_init\_\_(self, computer\_id, hard\_drive\_id):

self.computer\_id = computer\_id

self.hard\_drive\_id = hard\_drive\_id

# Hard Drives

hard\_drives = [

HardDrive(1, 'Seagate 1TB', 1000, 1),

HardDrive(2, 'Western Digital 2TB', 2000, 2),

HardDrive(3, 'Samsung 500GB', 500, 2),

]

# Computers

computers = [

Computer(1, 'Computer 1', 'Desktop', 1, 800),

Computer(2, 'Laptop 1', 'Laptop', 2, 1200),

Computer(3, 'Computer 2', 'Desktop', 3, 700),

]

computer\_hard\_drives = [

ComputerHardDrive(1, 1),

ComputerHardDrive(2, 2),

ComputerHardDrive(3, 3),

ComputerHardDrive(3, 2),

]

def main():

"""Main function"""

# One-to-many relationship

one\_to\_many = [(h.model, h.capacity\_gb, c.name)

for c in computers

for h in hard\_drives

if h.computer\_id == c.id]

# Many-to-many relationship

many\_to\_many\_temp = [(c.name, ch.computer\_id, ch.hard\_drive\_id)

for c in computers

for ch in computer\_hard\_drives

if c.id == ch.computer\_id]

many\_to\_many = [(h.model, h.capacity\_gb, comp\_name)

for comp\_name, comp\_id, hd\_id in many\_to\_many\_temp

for h in hard\_drives if h.id == hd\_id]

print('Task A1')

res\_a1 = sorted(one\_to\_many, key=itemgetter(2))

print(res\_a1)

print('\nTask A2')

res\_a2\_unsorted = []

for c in computers:

c\_hard\_drives = list(filter(lambda i: i[2] == c.name, one\_to\_many))

if len(c\_hard\_drives) > 0:

c\_capacities = [capacity for \_, capacity, \_ in c\_hard\_drives]

c\_capacity\_sum = sum(c\_capacities)

res\_a2\_unsorted.append((c.name, c\_capacity\_sum))

res\_a2 = sorted(res\_a2\_unsorted, key=itemgetter(1), reverse=True)

print(res\_a2)

print('\nTask A3')

res\_a3 = {}

for c in computers:

if 'computer' in c.name.lower():

c\_hard\_drives = list(filter(lambda i: i[2] == c.name, many\_to\_many))

c\_hard\_drive\_models = [x for x, \_, \_ in c\_hard\_drives]

res\_a3[c.name] = c\_hard\_drive\_models

print(res\_a3)

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

main()  
  
  
  
  
  
**Результаты выполнения:**

Task A1

[('Seagate 1TB', 1000, 'Computer 1'), ('Western Digital 2TB', 2000, 'Laptop 1'), ('Samsung 500GB', 500, 'Laptop 1')]

Task A2

[('Laptop 1', 2500), ('Computer 1', 1000)]

Task A3

{'Computer 1': ['Seagate 1TB'], 'Computer 2': ['Samsung 500GB', 'Western Digital 2TB']}