*Государственное образовательное учреждение высшего профессионального образования*



ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Системы обработки информации и управления» (ИУ-5)

**РУБЕЖНЫЙ КОНТРОЛЬ**

**№ 2**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Группа ИУ5-35Б

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **16.12.2024** /**А. А. Торопыгин/** 

(Подпись, дата) (И.О.Фамилия)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /**Ю. Е. Гапанюк/**

(Подпись, дата) (И.О.Фамилия)

2024

**Текст программы:**

from operator import itemgetter

def unit\_test(\*\*kwargs):

def inner\_decorator(func):

def wrapped(\*args):

assert 'output' in kwargs

result = func()

assert result == kwargs['output'],\

f'incorrecr return from function {func.\_\_name\_\_}()\n\

recieved: {result}\n\

expected: {kwargs['output']}'

print(f'\nfunction \'{func.\_\_name\_\_}\' test completed successfully\n')

return func(\*args)

return wrapped

return inner\_decorator

class Driver:

"""Водитеь"""

def \_\_init\_\_(self, id, fio, exp, ap\_id):

self.id = id

self.fio = fio

self.exp = exp #стаж в годах

self.ap\_id = ap\_id

class AP:

"""Автопарк"""

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

self.id = id

self.name = name

class DriverAP:

"""

'Водители автопарка' для реализации

связи многие-ко-многим

"""

def \_\_init\_\_(self, ap\_id, driver\_id):

self.ap\_id = ap\_id

self.driver\_id = driver\_id

# Автопарки

aps = [

AP(1, 'Транспортный Альянс'),

AP(2, 'ЭкоДрайв'),

AP(3, 'АвтоМир')

]

# Водители

drivers = [

Driver(1, 'Алексеев', 10, 1),

Driver(2, 'Борисов', 11, 1),

Driver(3, 'Васильев', 15, 2),

Driver(4, 'Герасимов', 30, 2),

Driver(5, 'Данилов', 13, 3)

]

drivers\_aps = [

DriverAP(1, 1),

DriverAP(1, 2),

DriverAP(1, 3),

DriverAP(2, 3),

DriverAP(2, 4),

DriverAP(2, 5),

DriverAP(3, 1),

DriverAP(3, 2),

DriverAP(3, 4),

DriverAP(3, 5)

]

# Соединение данных один-ко-многим

one\_to\_many = [(d.fio, d.exp, ap.name)

for ap in aps

for d in drivers

if d.ap\_id==ap.id]

@unit\_test(

output = [

('Борисов', 'Транспортный Альянс'),

('Герасимов', 'ЭкоДрайв'),

('Данилов', 'АвтоМир')

]

)

def drivers\_end\_with\_ov():

res\_1 = [(i[0], i[2]) for i in one\_to\_many if i[0].endswith("ов")]

return res\_1

@unit\_test(

output = [

('ЭкоДрайв', 22.5),

('АвтоМир', 13.0),

('Транспортный Альянс', 10.5)

]

)

def sort\_by\_mid\_exp():

res\_2\_unsorted = []

for ap in aps:

ap\_drivers = list(filter(lambda i: i[2]==ap.name, one\_to\_many)) # получаем список водителей

if len(ap\_drivers) > 0:

ap\_exps = [exp for \_,exp,\_ in ap\_drivers] # получаем список зарплат

ap\_exps\_mid = sum(ap\_exps)/len(ap\_exps) # вычисляем среднюю з/п

res\_2\_unsorted.append((ap.name, ap\_exps\_mid))

res\_2 = sorted(res\_2\_unsorted, key=itemgetter(1), reverse=True) # сортируе по убыванию среней з/п

return res\_2

# Соединение данных многие-ко-многим

many\_to\_many\_temp = [(ap.name, dap.ap\_id, dap.driver\_id)

for ap in aps

for dap in drivers\_aps

if ap.id==dap.ap\_id]

many\_to\_many = [(d.fio, d.exp, dep\_name)

for dep\_name, ap\_id, driver\_id in many\_to\_many\_temp

for d in drivers if d.id==driver\_id]

@unit\_test(

output = {

'АвтоМир': ['Алексеев', 'Борисов', 'Герасимов', 'Данилов']

}

)

def ap\_starts\_with\_a():

res\_3 = {}

for ap in aps:

if ap.name.startswith("А"):

# получаем все записи о водителях автопарка

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

# получаем список имен водителей

ap\_drivers\_names = [x for x,\_,\_ in ap\_drivers]

res\_3[ap.name] = ap\_drivers\_names

return res\_3

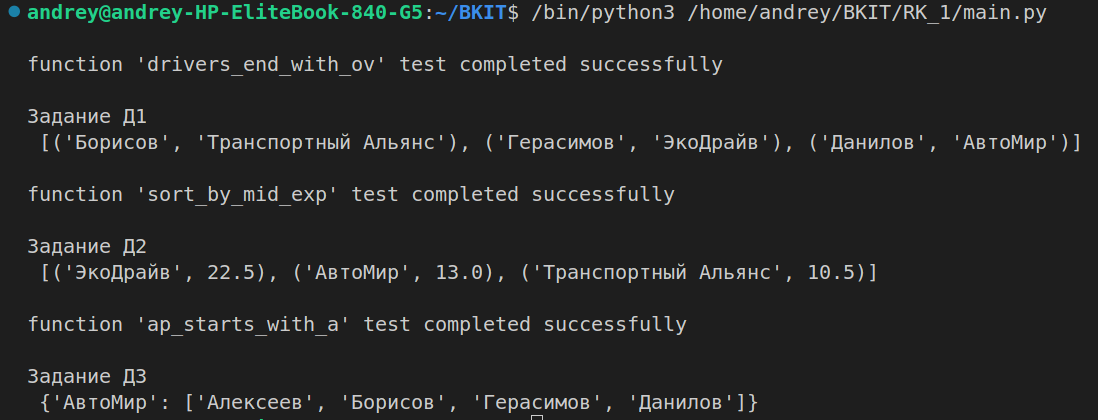
if \_\_name\_\_ == '\_\_main\_\_':

print('Задание Д1\n', drivers\_end\_with\_ov())

print('Задание Д2\n', sort\_by\_mid\_exp())

print('Задание Д3\n', ap\_starts\_with\_a())

**Вывод:**

****