



**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»  
Кафедра ИУ5 «Системы обработки информации и управления»**

**Курс «Базовые компоненты интернет-технологий»  
Отчет по РК1  
«Основные конструкции языка Python»**

**Выполнил:  
студент группы ИУ5-33Б  
Климов Н. С.**

**Проверил:**

**2021 г.**

Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс «Сотрудник», содержащий поля:
    - ID записи о сотруднике;
    - Фамилия сотрудника;
    - Зарплата (количественный признак);
    - ID записи об отделе. (для реализации связи один-ко-многим)
  2. Класс «Отдел», содержащий поля:
    - ID записи об отделе;
    - Наименование отдела.
  3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
    - ID записи о сотруднике;
    - ID записи об отделе.
- 2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.
- 3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

Результатом рубежного контроля является документ в формате PDF, который содержит текст программы и результаты ее выполнения.

## Текст программы:

```
from operator import itemgetter
from typing import List, Tuple, Any
```

```
class microprocessor:
    """Микропроцессор"""
```

```

def __init__(self, id, title, sal, comp_id):
    self.id = id
    self.title = title
    self.sal = sal
    self.comp_id = comp_id

class Computer:
    """Компьютер"""

    def __init__(self, id, name):
        self.id = id
        self.name = name

class ComputerMicroprocessor:
    """
    " для реализации
    связи многие-ко-многим
    """

    def __init__(self, comp_id, mic_id):
        self.comp_id = comp_id
        self.mic_id = mic_id

# Компьютеры
Computers = [
    Computer(1, 'MSI'),
    Computer(2, 'ASUS'),
    Computer(3, 'Lenovo'),
    Computer(4, 'Imac'),
    Computer(5, 'Xiaomi'),
]

# Микропроцессоры
Microprocessors = [
    microprocessor(1, 'AMD A8', 253, 3),
    microprocessor(2, 'AMD Ryzen 5', 222, 2),
    microprocessor(3, 'Intel Core i5', 434, 5),
    microprocessor(4, 'AMD Athlon', 370, 2),
    microprocessor(5, 'Intel Core i7', 556, 3),
]

Microprocessor_Computer = [
    ComputerMicroprocessor(1, 3),
    ComputerMicroprocessor(2, 2),
    ComputerMicroprocessor(3, 5),
    ComputerMicroprocessor(4, 2),
    ComputerMicroprocessor(5, 3),
]

```

```

    ComputerMicroprocessor(1, 2),
    ComputerMicroprocessor(2, 4),
    ComputerMicroprocessor(3, 1),
    ComputerMicroprocessor(4, 4),
    ComputerMicroprocessor(5, 1),
]
def main():
    """Основная функция"""
    # Соединение данных один-ко-многим
    one_to_many = [(m.title, m.sal, c.name)
                    for c in Computers
                    for m in Microprocessors
                    if m.comp_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, cm.comp_id, cm.mic_id)
                          for c in Computers
                          for cm in Microprocessor_Computer
                          if c.id == cm.comp_id]

    many_to_many = [(c.title, c.sal, Computer_name)
                    for Computer_name, Computer_id, microprocessor_id in many_to_many_temp
                    for c in Microprocessors if c.id == microprocessor_id]

    print('Задание B1:')
    res11 = []
    for title, _, Computer in one_to_many:
        if title[0] == "A":
            res11.append((title, Computer))
    print(res11, "\n")

    print('Задание B2')
    res12 = [[one_to_many[0][2], one_to_many[0][1]]]
    for title, sal, name in one_to_many:
        if name == res12[len(res12)-1][0]:
            if sal < res12[len(res12)-1][1]:

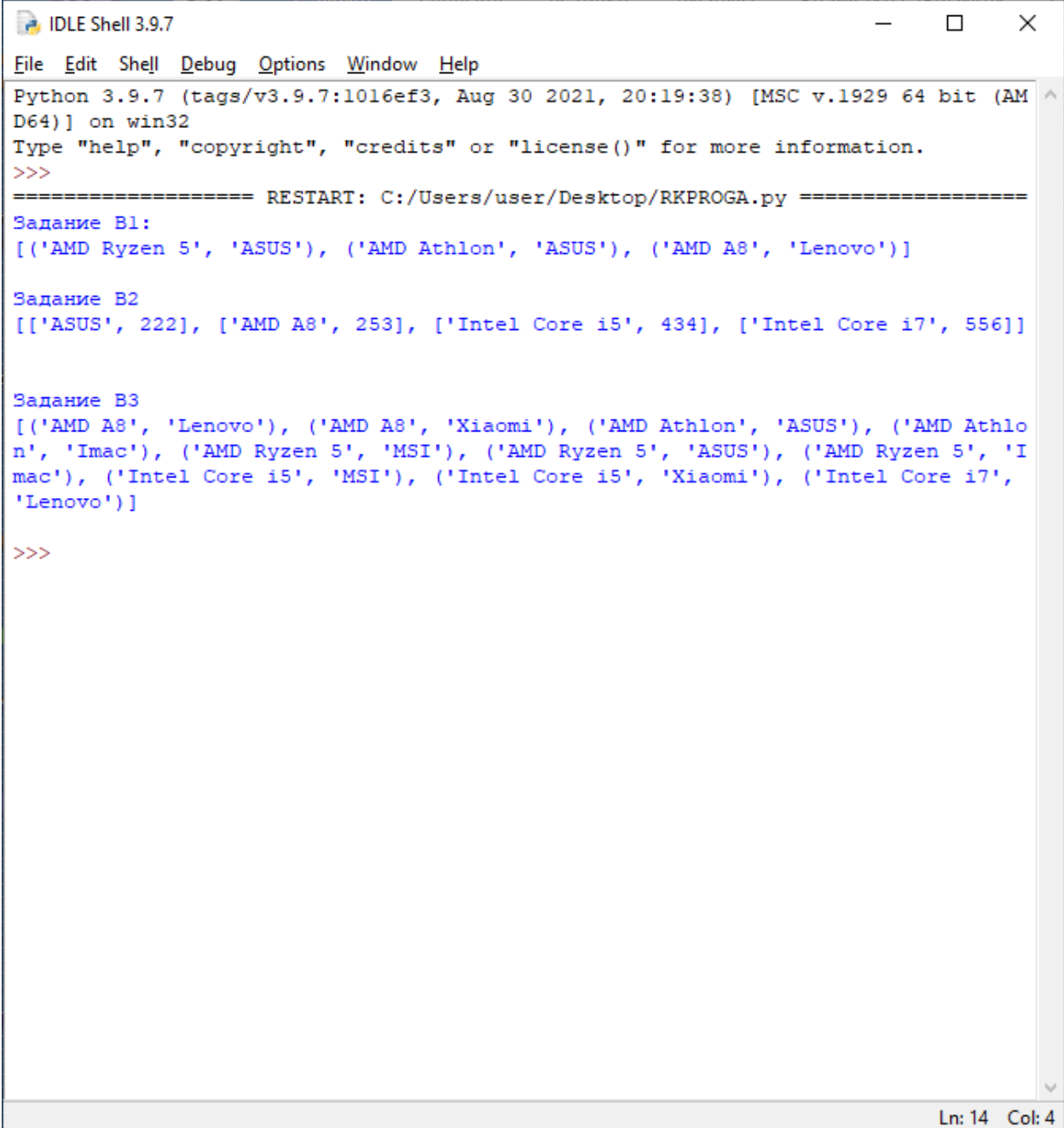
                res12[len(res12)-1][1] = sal
        else:
            res12.append([title, sal])
    print(sorted(res12, key=itemgetter(1)), "\n")

    print('Задание B3')
    res13 = []
    for title, sal, Computer in many_to_many:
        res13.append((title, Computer))
    # print(res_13)
    print(sorted(res13, key=itemgetter(0)), "\n")

```

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

## Экранные формы с примерами выполнения задания:



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/user/Desktop/RKPROGA.py =====  
Задание B1:  
[('AMD Ryzen 5', 'ASUS'), ('AMD Athlon', 'ASUS'), ('AMD A8', 'Lenovo')]  
  
Задание B2  
[['ASUS', 222], ['AMD A8', 253], ['Intel Core i5', 434], ['Intel Core i7', 556]]  
  
Задание B3  
[('AMD A8', 'Lenovo'), ('AMD A8', 'Xiaomi'), ('AMD Athlon', 'ASUS'), ('AMD Athlon', 'Imac'), ('AMD Ryzen 5', 'MSI'), ('AMD Ryzen 5', 'ASUS'), ('AMD Ryzen 5', 'Imac'), ('Intel Core i5', 'MSI'), ('Intel Core i5', 'Xiaomi'), ('Intel Core i7', 'Lenovo')]  
>>>  
  
Ln: 14 Col: 4
```