Московский государственный технический университет

имени Н. Э. Баумана

# Рубежный контроль №1

по дисциплине «Парадигмы и конструкции языков программирования»

Тема: «Реализация связей один-ко-многим и многие-ко-многим в предметной области»
Вариант: 23 (предметная область «Синтаксическая конструкция — Язык программирования»)

Выполнил: студент группы ИУ3-34Б
Цугель Арсений

Преподаватель: _____

Москва, 2025 г.

## 1. Описание задания

Необходимо разработать программу на языке Python, которая создаёт два связанных класса данных в соответствии с вариантом предметной области. Для варианта 23 используются классы «Синтаксическая конструкция» (сторона «много») и «Язык программирования» (сторона «один»). Также необходимо реализовать связь многие-ко-многим через вспомогательную структуру. Программа должна выполнить три запроса варианта А:
1) вывести список всех связанных синтаксических конструкций и языков, отсортированный по языкам;
2) вывести список языков с суммарной «сложностью» конструкций, отсортированный по убыванию суммы;
3) вывести список всех языков, в названии которых есть слово «язык», и список работающих в них конструкций.

## 2. Текст программы

```
from operator import itemgetter

class Lang:
    \"\"\"Язык программирования\"\"\"
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Syntax:
    \"\"\"Синтаксическая конструкция\"\"\"
    def __init__(self, id, name, complexity, lang_id):
        # complexity – условная сложность конструкции
        self.id = id
        self.name = name
        self.complexity = complexity
        self.lang_id = lang_id

class LangSyntax:
    \"\"\"Связь язык – синтаксическая конструкция (многие-ко-многим)\"\"\"
    def __init__(self, lang_id, syntax_id):
        self.lang_id = lang_id
        self.syntax_id = syntax_id

# Языки
langs = [
    Lang(1, 'Язык Python'),
    Lang(2, 'Язык Java'),
    Lang(3, 'Функциональный язык Haskell'),
    Lang(4, 'C++'),
    Lang(5, 'Скриптовый язык JavaScript'),
    # «дополнительные» языки, чтобы показать m2m с «язык»
    Lang(11, 'Язык (альтернативный) Python'),
    Lang(22, 'Язык (альтернативный) Java'),
]

# Синтаксические конструкции
syntaxes = [
    Syntax(1, 'Список (list comprehension)', 5, 1),
    Syntax(2, 'Декоратор функции', 7, 1),
    Syntax(3, 'Лямбда-выражение', 4, 2),
    Syntax(4, 'Обобщения (templates)', 9, 4),
    Syntax(5, 'Класс', 6, 2),
    Syntax(6, 'Сопоставление с образцом', 8, 3),
]

# Связи многие-ко-многим
langs_syntaxes = [
    LangSyntax(1, 1),
    LangSyntax(1, 2),
```

```
    LangSyntax(2, 3),
    LangSyntax(2, 5),
    LangSyntax(3, 6),
    # те же конструкции в «альтернативных» языках
    LangSyntax(11, 1),
    LangSyntax(22, 3),
    LangSyntax(22, 5),
]

def main():
    # one-to-many: конструкции — языки
    one_to_many = [(s.name, s.complexity, l.name)
                   for l in langs
                   for s in syntaxes
                   if s.lang_id == l.id]

    # many-to-many
    many_to_many_temp = [(l.name, ls.lang_id, ls.syntax_id)
                         for l in langs
                         for ls in langs_syntaxes
                         if l.id == ls.lang_id]

    many_to_many = [(s.name, s.complexity, lang_name)
                    for lang_name, lang_id, syntax_id in many_to_many_temp
                    for s in syntaxes if s.id == syntax_id]

    # Задание A1
    print('Задание A1')
    res_a1 = sorted(one_to_many, key=itemgetter(2))
    for item in res_a1:
        print(item)

    # Задание A2
    print('\\nЗадание A2')
    res_a2_unsorted = []
    for l in langs:
        l_syntaxes = list(filter(lambda x: x[2] == l.name, one_to_many))
        if len(l_syntaxes) > 0:
            complexities = [c for _, c, _ in l_syntaxes]
            total_complexity = sum(complexities)
            res_a2_unsorted.append((l.name, total_complexity))
    res_a2 = sorted(res_a2_unsorted, key=itemgetter(1), reverse=True)
    for item in res_a2:
        print(item)

    # Задание A3
    print('\\nЗадание A3')
    res_a3 = {}
    for l in langs:
        if 'язык' in l.name.lower():
            l_syntaxes = list(filter(lambda x: x[2] == l.name, many_to_many))
            only_names = [x for x, _, _ in l_syntaxes]
            res_a3[l.name] = only_names
    for k, v in res_a3.items():
        print(k, '->', v)

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

## 3. Экранные формы (результаты выполнения программы)

Задание A1
('Обобщения (templates)', 9, 'C++')
('Сопоставление с образцом', 8, 'Функциональный язык Haskell')
('Лямбда-выражение', 4, 'Язык Java')
('Класс', 6, 'Язык Java')
('Список (list comprehension)', 5, 'Язык Python')
('Декоратор функции', 7, 'Язык Python')

Задание A2
('Язык Python', 12)
('Язык Java', 10)
('C++', 9)
('Функциональный язык Haskell', 8)

Задание A3
Язык Python -> ['Список (list comprehension)', 'Декоратор функции']
Язык Java -> ['Лямбда-выражение', 'Класс']
Функциональный язык Haskell -> ['Сопоставление с образцом']
Скриптовый язык JavaScript -> []
Язык (альтернативный) Python -> ['Список (list comprehension)']
Язык (альтернативный) Java -> ['Лямбда-выражение', 'Класс']