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 «Радиотехнический»

Кафедра «Системы обработки информации и управления»

Рубежный контроль № 1

по дисциплине «Разработка интернет-приложений».

Вариант **E12**

Выполнил:
студент(ка) группы № РТ5-51Б
А. С. Пакало
подпись, дата

Проверил:
преподаватель
Ю. Е. Гапанюк
подпись, дата

Москва, 2021

Оглавление

Цель работы

Работа с классами в Python, организация и реализация запросов.

Задание

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.

Вариант Е.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате. Средняя зарплата должна быть округлена до 2 знака после запятой (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений; для округления необходимо использовать функцию* https://docs.python.org/3/library/functions.html#round).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.

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»,о произвольный количественный признак, например, «зарплата сотрудника».

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

Выполнение

main.py

```
from typing import Iterator, Type
from task_decorator import task_decorator
from data import programming_languages, ides, programming_languages_ides
from models import ProgrammingLanguage, Ide
from _utilities import is_substr, avg

""" Selectors-generators. """
def select_programming_languages(subtitle = '', case_sensitive = False) ->
Iterator[ProgrammingLanguage]:
    return (pl for pl in programming_languages if is_substr(pl.title,
subtitle, case_sensitive))


def select_ides(programming_language_id: int) -> Iterator[Ide]:
    return (ide for ide in ides if ide.programming_language_id ==
programming_language_id)


@task_decorator(' Task #1. ')
def task1(programming_language_subtitle) -> None:
    print(f'Searching for \'{programming_language_subtitle}\' in language
titles...')

    for pl in select_programming_languages(programming_language_subtitle,
True):
        print(pl.title)
        print('IDEs available:')
        for ide in select_ides(pl.id):
            print(f'- {ide.to_string_formatted()}')

@task_decorator(' Task #2. ')
def task2() -> None:
    print('Mean IDE price...')
    for pl in select_programming_languages():
        output = f'- for {pl.title}: '

        prices = []
        for ide in select_ides(pl.id):
            prices.append(ide.price)

        output += str(avg(prices))
        print(output)
```

```
def select_ides_many_to_many(programming_language_id: int) ->
Iterator[Ide]:
    return (ide
        for ide in ides
        for pli in programming_languages_ides
        if ide.id == pli.ide_id and programming_language_id ==
pli.programming_language_id)


@task_decorator(' Task #3. ')
def task3() -> None:
    print('IDEs starting from \'A\'')
    for pl in select_programming_languages():
        print(f'{pl.title}:')
        for ide in select_ides_many_to_many(pl.id):
            if (ide.title.startswith('A')):
                print(f'- {ide.to_string_formatted()}')


def main() -> None:
    task1('Ja')
    task2()
    task3()


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

data.py

```
from models import ProgrammingLanguage, Ide, ProgrammingLanguageIde

# Набор языков программирования.
programming_languages = [
    ProgrammingLanguage(1, 'C++'),
    ProgrammingLanguage(2, 'C#'),
    ProgrammingLanguage(11, 'Java'),
    ProgrammingLanguage(22, 'Python'),
    ProgrammingLanguage(33, 'JavaScript'),
]

# Набор средств разработки.
ides = [
    Ide(2, 'CLion', 3500, 1),

    Ide(1, 'Resharper', 2500, 2),

    Ide(4, 'IntelliJ', 3500, 11),
    Ide(9, 'SublimeText', 0, 11),

    Ide(7, 'PyCharm', 6000, 22),
    Ide(10, 'Atom', 0, 22),

    Ide(3, 'VisualStudio', 0, 33),
    Ide(6, 'WebStorm', 1000, 33),
    Ide(8, 'Eclipse', 0, 33),
]
```

```
# Реализация много-ко-многим.
programming_languages_ides = [
    ProgrammingLanguageIde(1,2),
    ProgrammingLanguageIde(1,3),
    ProgrammingLanguageIde(1,8),
    ProgrammingLanguageIde(1,9),

    ProgrammingLanguageIde(2,1),
    ProgrammingLanguageIde(2,2),
    ProgrammingLanguageIde(2,3),
    ProgrammingLanguageIde(2,8),
    ProgrammingLanguageIde(2,9),

    ProgrammingLanguageIde(11,4),
    ProgrammingLanguageIde(11,8),
    ProgrammingLanguageIde(11,3),
    ProgrammingLanguageIde(11,6),
    ProgrammingLanguageIde(11,9),
    ProgrammingLanguageIde(11,10),

    ProgrammingLanguageIde(22,7),
    ProgrammingLanguageIde(22,8),
    ProgrammingLanguageIde(22,3),
    ProgrammingLanguageIde(22,6),
    ProgrammingLanguageIde(22,9),
    ProgrammingLanguageIde(22,10),

    ProgrammingLanguageIde(33,3),
    ProgrammingLanguageIde(33,6),
    ProgrammingLanguageIde(33,8),
    ProgrammingLanguageIde(33,9),
    ProgrammingLanguageIde(33,10),
]
```

models.py

```
class ProgrammingLanguage:
    """Язык программирования. """
    def __init__(self, id: int, title: str):
        self.id = id
        self.title = title


class Ide:
    """Средство разработки. """
    def __init__(self, id: int, title: str, price: int,
programming_language_id: int):
        self.id = id
        self.title = title
        self.price = price
        self.programming_language_id = programming_language_id

    def to_string_formatted(self) -> str:
        return f'Title: {self.title}, price: {self.price if self.price !=
0 else f"{self.price} (free)"}'


class ProgrammingLanguageIde:
    """ Средства разработки для языка. """
    def __init__(self, programming_language_id: int, ide_id: int):
        self.programming_language_id = programming_language_id
        self.ide_id = ide_id
```

task_decorator.py

```
from functools import wraps

""" Decorating output of a task. """
def task_decorator(output):
    def decorator(wrapped_function):
        @wraps(wrapped_function)
        def wrapper(*args, **kwargs):
            print(f'----------------------{output}----------------------')
            wrapped_function(*args, **kwargs);
            print()

        return wrapper

    return decorator
```

_utilities.py

```
from functools import reduce
from typing import List

def is_substr(string: str, substring: str, case_sensitive: bool) -> bool:
    if case_sensitive:
        return string.find(substring) != -1

    return string.lower().find(substring.lower()) != -1


def avg(values: List[int], ndigits = 2) -> int:
    sum = reduce(lambda acc, v: acc + v, values)
    return round(sum / len(values), ndigits)
```

Результаты выполнения

```
dubuntus@DS13: .../__t3.1-Course/code/rk1_code$ python main.py
──────────────────────── Task #1. ────────────────────────
Searching for 'Ja' in language titles...
Java
IDEs available:
- Title: IntelliJ, price: 3500
- Title: SublimeText, price: 0 (free)
JavaScript
IDEs available:
- Title: VisualStudio, price: 0 (free)
- Title: WebStorm, price: 1000
- Title: Eclipse, price: 0 (free)

──────────────────────── Task #2. ────────────────────────
Mean IDE price...
- for C++: 3500.0
- for C#: 2500.0
- for Java: 1750.0
- for Python: 3000.0
- for JavaScript: 333.33

──────────────────────── Task #3. ────────────────────────
IDEs starting from 'A'
C++:
C#:
Java:
- Title: Atom, price: 0 (free)
Python:
- Title: Atom, price: 0 (free)
JavaScript:
- Title: Atom, price: 0 (free)
```

Рис. 1. Результат исполнения программы.

Вывод

Продемонстрированы навыки работы с классами в Python, организации и реализации запросов, а также работы с reduce функцией, генераторами, декораторами и аннотациями типов.