

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4.2
«Перегрузка операторов в языке Python»

Выполнил:
Горшков Виталий Игоревич
3 курс, группа ИВТ-б-о-21-1,
09.03.01 «Информатика
и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем»,
очная форма обучения

(подпись)

Ставрополь, 2023 г.

Цель работы: приобретение навыков по перегрузке операторов при написании программ с помощью языка программирования Python.

Выполнение работы:

1. Создали общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

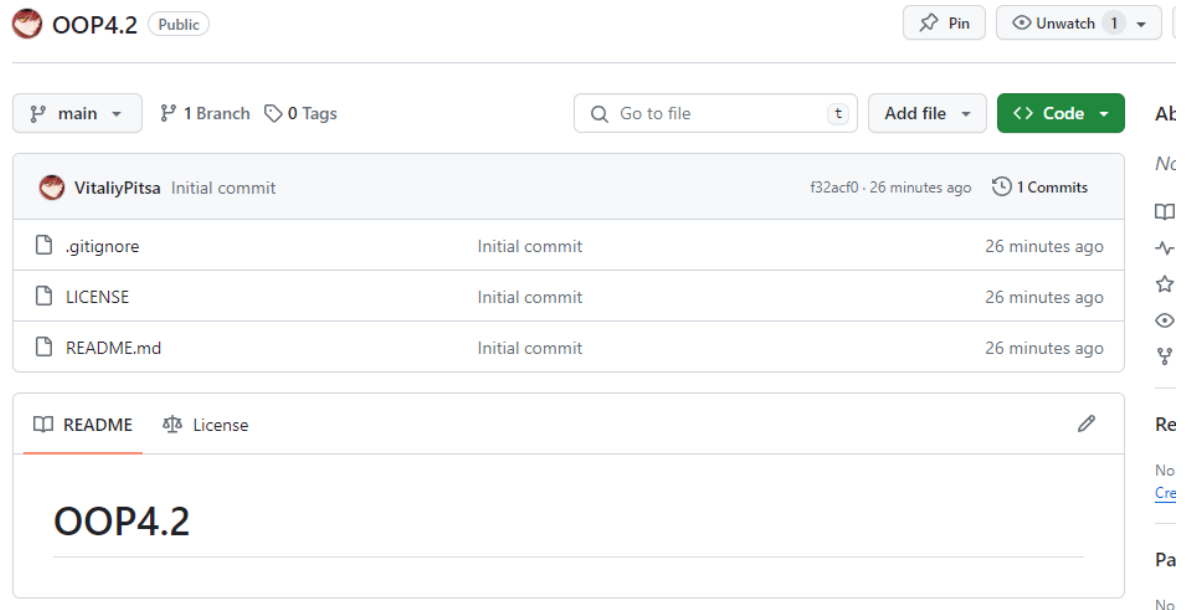


Рисунок 1 – Создание репозитория

2. Выполнили клонирование созданного репозитория.

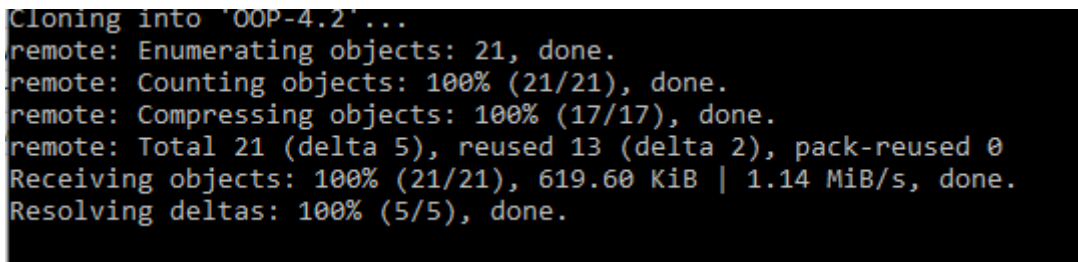


Рисунок 2 – Клонирование репозитория

3. Организовали свой репозиторий в соответствии с моделью git-flow.

4. Выполнение индивидуального задания №1.

Поле first — целое положительное число, номинал купюры; номинал может принимать значения 1, 2, 5, 10, 50, 100, 500, 1000, 5000 Поле second — целое положительное число, количество купюр данного достоинства. Реализовать метод summa() — вычисление денежной суммы. Максимально задействовав имеющиеся в Python средства перегрузки операторов.

```
Номинал купюры: 100
Количество купюр: 5
Сумма: 500
Номинал купюры: 100
Количество купюр: 3
Сумма: 300
Номинал купюры: 100
Количество купюр: 8
Сумма: 800
```

Рисунок 4 – Результат выполнения

5. Выполнение индивидуального задания №2

Карточка иностранного слова представляет собой словарь, содержащую иностранное слово и его перевод. Для моделирования электронного словаря иностранных слов реализовать класс Dictionary. Данный класс имеет поле-название словаря и содержит список словарей WordCard, представляющих собой карточки иностранного слова. Название словаря задается при создании нового словаря, но должна быть предоставлена возможность его изменения во время работы. Карточки добавляются в словарь и удаляются из него. Реализовать поиск определенного слова как отдельный метод. Аргументом операции индексирования должно быть иностранное слово. В словаре не должно быть карточек-дублей. Реализовать операции объединения, пересечения и вычитания словарей. При реализации должен создаваться новый словарь, а исходные словари не должны изменяться. При объединении новый словарь должен содержать без повторений все слова, содержащиеся в обоих словарях-операндах. При пересечении новый словарь должен состоять только из тех слов, которые имеются в обоих словарях-операндах. При вычитании новый словарь должен содержать слова первого словаря-операнда, отсутствующие во втором.

Рисунок 5 – Результат выполнения

Контрольные вопросы:

1 Какие средства существуют в Python для перегрузки операций?

В Python для перегрузки операций используются магические методы (или "dunder" методы), начинающиеся и заканчивающиеся двумя подчеркиваниями (например, `__add__`, `__sub__`, `__eq` и т.д.). Эти методы позволяют объектам классов вести себя определенным образом при использовании операторов.

2 Какие существуют методы для перегрузки арифметических операций и операций отношения в языке Python?

- Арифметические операции: `__add__` (сложение), `__sub__` (вычитание), `__mul__` (умножение), `__truediv__` (деление), и т.д.
- Операции отношения: `__eq__` (равенство), `__ne__` (неравенство), `__lt__` (меньше), `__le__` (меньше или равно), `__gt__` (больше), `__ge__` (больше или равно).

3 В каких случаях будут вызваны следующие методы: `__add__`, `__iadd__` и `__radd__`?

Приведите примеры.

- `__add__`: Вызывается при использовании оператора `+`. Например, `a + b`.
- `__iadd__` (in-place addition): Вызывается при использовании оператора `+=`. Например, `a += b`.
- `__radd__` (right-side addition): Вызывается при использовании оператора `+`, когда левый операнд не поддерживает операцию, но правый операнд поддерживает. Например, `b + a`, если для `b` не определен метод `__add__`.

class Example:

```
def __add__(self, other):
    return Example(self.value + other.value)
```

```
def __iadd__(self, other):
    self.value += other.value
    return self
```

```
def __radd__(self, other):
    return Example(self.value + other)
```

4 Для каких целей предназначен метод `__new__`? Чем он отличается от метода `__init__`?

`__new__` отвечает за создание нового экземпляра объекта перед его инициализацией (`__init__`). Метод `__new__` чаще всего используется в неизменяемых типах данных, таких как строки и кортежи. Отличие от `__init__`: `__new__` вызывается перед `__init__` и имеет возможность изменить создаваемый объект или даже вернуть другой объект вместо создаваемого.

5 Чем отличаются методы `__str__` и `__repr__`?

- `__str__`: Вызывается функцией `str()`. Предназначен для представления объекта в человекочитаемой форме. Если `__str__` отсутствует, вызывается `__repr__`.

- `__repr__`: Вызывается функцией `repr()` и используется для представления объекта в форме, пригодной для воспроизведения (по возможности). Если `__repr__` отсутствует, вызывается `_str_`.