

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Факультет безопасности информационных технологий

Дисциплина:
«Программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
«Объектно-ориентированное программирование в Python»
Вариант 3-10

Выполнил:

Суханкулиев Мухаммет,
студент группы N3146



(подпись)

Проверила:

Сыдыкова Эмилия,
преподаватель программирования

(отметка о выполнении)

(подпись)

Санкт-Петербург
2024 г.

СОДЕРЖАНИЕ

Введение	3
1 Объектно-ориентированное программирование в Python	4
1.1 Задание.....	4
Базовый класс – deque (из модуля collections).	4
1.2 Примеры работы программы.....	4
1.3 Исходный текст программы	4
1.3.1 lab6msN3146.py.....	4
1.3.2 test_lab6msN3146.py	6
Заключение.....	8
Список использованных источников.....	9

ВВЕДЕНИЕ

Разработать на языке Python для ОС Linux модуль, содержащий определение заданного типа, и тестовый модуль для pytest, содержащий набор тестов для проверки корректности определения типа.

Для достижения поставленной цели необходимо решить следующие задачи:

- Выполнить задание;
- Протестировать программу;
- Заархивировать папку проекта.

1 ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В PYTHON

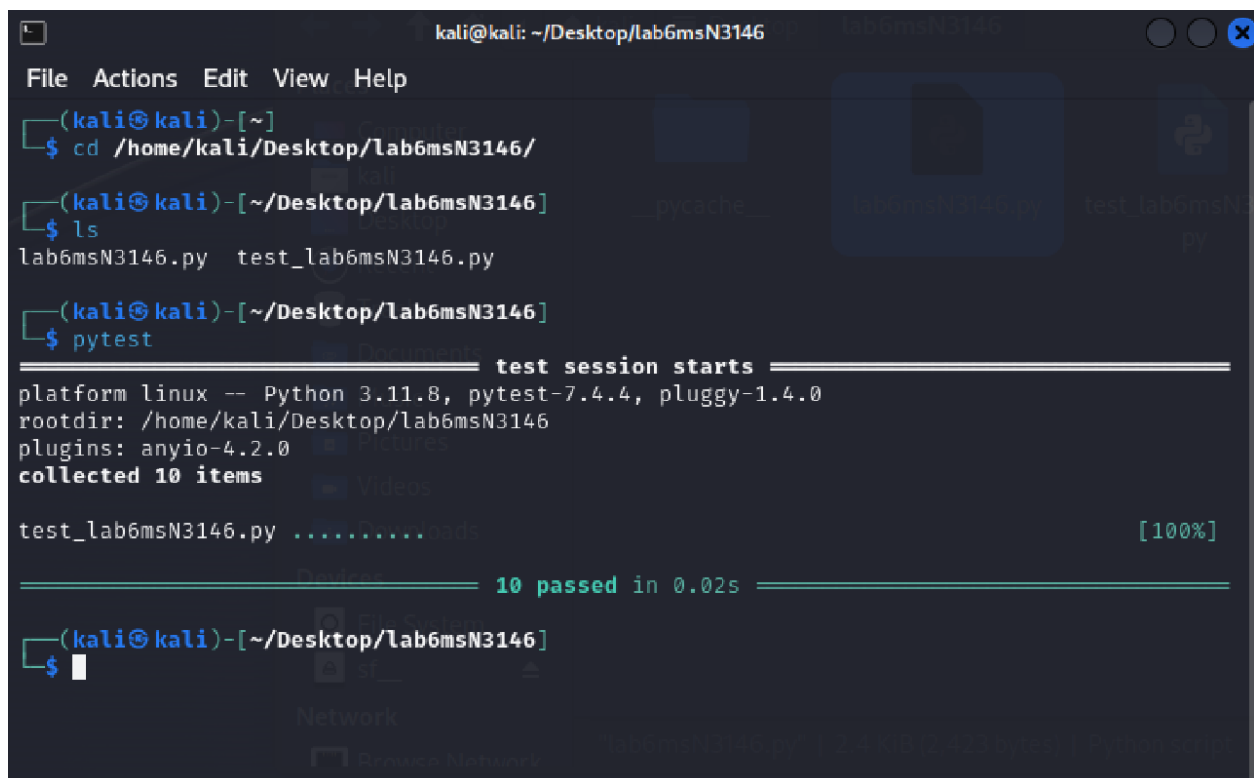
Создание модуля, содержащего определение заданного типа и тестовый модуль, который проверит корректность определения типа в соответствии с вариантом 3-10.

1.1 Задание

Базовый класс – deque (из модуля collections).

Формат данных в строке – Автомобильный номер РФ

1.2 Примеры работы программы



```
kali@kali: ~/Desktop/lab6msN3146
File Actions Edit View Help
(kali@kali)-[~]
$ cd /home/kali/Desktop/lab6msN3146/
(kali@kali)-[~/Desktop/lab6msN3146]
$ ls
lab6msN3146.py  test_lab6msN3146.py
(kali@kali)-[~/Desktop/lab6msN3146]
$ pytest
===== test session starts =====
platform linux -- Python 3.11.8, pytest-7.4.4, pluggy-1.4.0
rootdir: /home/kali/Desktop/lab6msN3146
plugins: anyio-4.2.0
collected 10 items

test_lab6msN3146.py ..... [100%]

===== 10 passed in 0.02s =====
(kali@kali)-[~/Desktop/lab6msN3146]
$
```

1.3 Исходный текст программы

1.3.1 lab6msN3146.py

```
import re
from collections import deque

class FormatError(Exception): pass
class UndoError(Exception): pass
class RedoError(Exception): pass

class MyDeque(deque):
```

```

def __init__(self, iterable=()):
    super().__init__()
    self.history = []
    self.future = []
    for item in iterable:
        self._check_format(item)
        super().append(item)
    self._save_state(initial=True)

def _check_format(self, item):
    if not isinstance(item, str):
        raise TypeError("Только строки дозволены") # Кек
    pattern = r'^[АВЕКМНОРСТУХ]\d{3}[АВЕКМНОРСТУХ]{2,3}\d{2,3}$'
    if not re.match(pattern, item):
        raise FormatError("Неверный формат Автомобильного номера")

def append(self, item):
    self._check_format(item)
    super().append(item)
    self._save_state()

def appendleft(self, item):
    self._check_format(item)
    super().appendleft(item)
    self._save_state()

def pop(self):
    item = super().pop()
    self._save_state()
    return item

def popleft(self):
    item = super().popleft()
    self._save_state()
    return item

def insert(self, index, item):
    self._check_format(item)
    super().insert(index, item)
    self._save_state()

def remove(self, value):
    super().remove(value)
    self._save_state()

def clear(self):
    super().clear()
    self._save_state()

def _save_state(self, initial=False):
    if not initial:
        self.history.append(list(self))
    else:
        self.history.append(list(self))
    self.future.clear()

def undo(self):
    if len(self.history) <= 1:
        raise UndoError("Нет истории для отмены")
    self.future.append(self.history.pop())
    last_state = self.history[-1]
    self._restore_state(last_state)

```

```

def redo(self):
    if not self.future:
        raise RedoError("Нет действий для повтора")
    next_state = self.future.pop()
    self.history.append(next_state)
    self._restore_state(next_state)

def _restore_state(self, state):
    super().clear()
    super().extend(state)

```

1.3.2 test_lab6msN3146.py

```

import pytest
from lab6msN3146 import MyDeque, FormatError, UndoError, RedoError

def test_append():
    dq = MyDeque()
    dq.append('A123AA123')
    assert dq == MyDeque(['A123AA123'])
    dq.append('B333BB333')
    assert dq == MyDeque(['A123AA123', 'B333BB333'])

def test_appendleft():
    dq = MyDeque()
    dq.appendleft('A123AA123')
    assert dq == MyDeque(['A123AA123'])
    dq.appendleft('B333BB333')
    assert dq == MyDeque(['B333BB333', 'A123AA123'])

def test_pop():
    dq = MyDeque(['A123AA123', 'B333BB333'])
    assert dq.pop() == 'B333BB333'
    assert dq == MyDeque(['A123AA123'])

def test_popleft():
    dq = MyDeque(['A123AA123', 'B333BB333'])
    assert dq.popleft() == 'A123AA123'
    assert dq == MyDeque(['B333BB333'])

def test_insert():
    dq = MyDeque(['A123AA123', 'B333BB333'])
    dq.insert(1, 'A456AA456')
    assert dq == MyDeque(['A123AA123', 'A456AA456', 'B333BB333'])

def test_remove():
    dq = MyDeque(['A123AA123', 'B333BB333'])
    dq.remove('A123AA123')
    assert dq == MyDeque(['B333BB333'])

def test_clear():
    dq = MyDeque(['A123AA123', 'B333BB333'])
    dq.clear()
    assert dq == MyDeque()

def test_undo_redo():
    dq = MyDeque(['A123AA12'])
    dq.append('B333BB333')
    assert dq == MyDeque(['A123AA12', 'B333BB333'])
    dq.undo()
    assert dq == MyDeque(['A123AA12'])
    dq.redo()

```

```

    assert dq == MyDeque(['A123AA12', 'B333BB333'])

def test_exceptions():
    dq = MyDeque()
    with pytest.raises(FormatError):
        dq.append('hren\\')
    with pytest.raises(UndoError):
        dq.undo()
    dq.append('A123AA123')
    with pytest.raises(RedoError):
        dq.redo()

def test_type_error():
    dq = MyDeque()
    with pytest.raises(TypeError):
        dq.append(123)
    with pytest.raises(TypeError):
        dq.appendleft({'A123AA123'})
    with pytest.raises(TypeError):
        dq.insert(0, ['B333BB333'])

```

ЗАКЛЮЧЕНИЕ

Был создан модуль, содержащий определение deque (из модуля collections) и тестовый модуль, который проверяет корректность определения типа.

Это позволило закрепить навыки ООП в Python.

Я все сделал, я молодец. =)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лекции Гирика Алексея Валерьевича по программированию – 2024. – URL :
<https://drive.google.com/drive/folders/1eAiMW4hD9TLhZH2vtpKPWzZkzKp10BnL>
(дата обращения: 14.05.2024).