

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**
Факультет безопасности информационных технологий

Дисциплина:

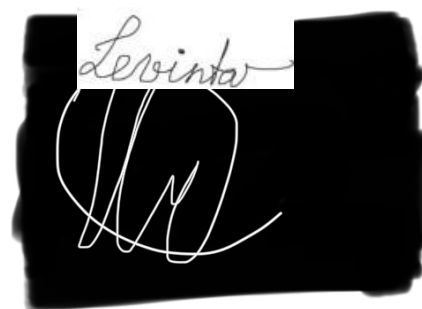
«Программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

Выполнил:

Студент группы N3149

Ильчук Денис.

A handwritten signature in white ink on a black rectangular background. The signature appears to be 'Levintov' or similar, written in a cursive style.

Проверила:

Горлина А.В.

Задание 5-3

Таблица 1. Базовый класс

№ варианта	Базовый класс
5	dict (ограничение формата накладывается на значения словаря)

Таблица 2. Формат данных в строке

№ варианта	Формат данных в строке
3	MAC-адрес в формате XX:XX:XX:XX:XX:XX

Примеры работы pytest

```
> pytest
===== test session starts =====
platform darwin -- Python 3.12.0, pytest-8.2.2, pluggy-1.5.0
rootdir: /Users/denisilcuk/Documents/itmoblat/prog/fun/lab6dviN3149
plugins: Faker-25.3.0
collected 13 items

test_lab6dviN3149.py ..... [100%]

===== 13 passed in 0.06s =====
```

Исходный код программы

```
import collections
import re

class FormatError(Exception):
    pass

class UndoError(Exception):
    pass

class RedoError(Exception):
    pass

class MACAddressDict(dict):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.history = []
        self.redo_stack = []

    def __setitem__(self, key, value):
        if not self._validate_mac(value):
            raise FormatError(f"Неправильный формат MAC адреса: {value}")
        super().__setitem__(key, value)
        self.history.append(('setitem', key, value))
        self.redo_stack.clear() # Очистить redo stack после действия

    def __delitem__(self, key):
        value = self[key]
```

```

super().__delitem__(key)
self.history.append(('delitem', key, value))
self.redo_stack.clear() # как и раньше

def update(self, *args, **kwargs):
    for key, value in dict(*args, **kwargs).items():
        if not self._validate_mac(value):
            raise FormatError("Неправильный формат MAC адреса: {value}")
    super().update(*args, **kwargs)
    self.history.append(('update', dict(*args, **kwargs)))
    self.redo_stack.clear() # на западном фронте без перемен

def undo(self):
    if not self.history:
        raise UndoError("Нет действий для отмены")
    action, *data = self.history.pop()
    if action == 'setitem':
        key, value = data
        if key in self and self[key] == value:
            super().__delitem__(key)
    elif action == 'delitem':
        key, value = data
        super().__setitem__(key, value)
    elif action == 'update':
        for key in data[0]:
            super().__delitem__(key)
    self.redo_stack.append((action, *data))

def redo(self):
    if not self.redo_stack:
        raise RedoError("Нет действий для отмены отмены")
    action, *data = self.redo_stack.pop()
    if action == 'setitem':
        key, value = data
        self.__setitem__(key, value)
        self.history.pop() # Удалить повторяющееся действие из истории
    elif action == 'delitem':
        key, value = data
        self.__delitem__(key)
        self.history.pop() # Удалить повторяющееся действие из истории
    elif action == 'update':
        self.update(data[0])
        self.history.pop() # Удалить повторяющееся действие из истории

def _validate_mac(self, mac): #проверка MAC адреса
    pattern = r'^([0-9A-Fa-f]{2}:){5}([0-9A-Fa-f]{2})$'
    return re.match(pattern, mac) is not None

```

```

import pytest
from lab6dviN3149 import FormatError, UndoError, RedoError, MACAddressDict

def test_setitem_valid_mac():
    mac_dict = MACAddressDict()
    mac_dict['key1'] = '01:23:45:67:89:AB'
    assert mac_dict['key1'] == '01:23:45:67:89:AB'

def test_setitem_invalid_mac():
    mac_dict = MACAddressDict()
    with pytest.raises(FormatError):
        mac_dict['key1'] = 'инвалид_маc'

def test_delitem():
    mac_dict = MACAddressDict({'key1': '01:23:45:67:89:AB'})
    del mac_dict['key1']
    assert 'key1' not in mac_dict

def test_update_valid_macs():
    mac_dict = MACAddressDict()
    mac_dict.update({'key1': '01:23:45:67:89:AB', 'key2': 'CD:EF:01:23:45:67'})
    assert mac_dict['key1'] == '01:23:45:67:89:AB'
    assert mac_dict['key2'] == 'CD:EF:01:23:45:67'

def test_update_invalid_mac():
    mac_dict = MACAddressDict()
    with pytest.raises(FormatError):
        mac_dict.update({'key1': '01:23:45:67:89:AB', 'key2': 'инвалид_маc'})

def test_undo_setitem():
    mac_dict = MACAddressDict()
    mac_dict['key1'] = '01:23:45:67:89:AB'
    mac_dict.undo()
    assert 'key1' not in mac_dict

def test_undo_delitem():
    mac_dict = MACAddressDict({'key1': '01:23:45:67:89:AB'})
    del mac_dict['key1']
    mac_dict.undo()
    assert mac_dict['key1'] == '01:23:45:67:89:AB'

def test_undo_update():
    mac_dict = MACAddressDict()
    mac_dict.update({'key1': '01:23:45:67:89:AB', 'key2': 'CD:EF:01:23:45:67'})
    mac_dict.undo()
    assert 'key1' not in mac_dict

```

```
assert 'key2' not in mac_dict
```

```
def test_redo_setitem():
```

```
    mac_dict = MACAddressDict()
    mac_dict['key1'] = '01:23:45:67:89:AB'
    mac_dict.undo()
    mac_dict.redo()
    assert mac_dict['key1'] == '01:23:45:67:89:AB'
```

```
def test_redo_delitem():
```

```
    mac_dict = MACAddressDict({'key1': '01:23:45:67:89:AB'})
    del mac_dict['key1']
    mac_dict.undo()
    mac_dict.redo()
    assert 'key1' not in mac_dict
```

```
def test_redo_update():
```

```
    mac_dict = MACAddressDict()
    mac_dict.update({'key1': '01:23:45:67:89:AB', 'key2': 'CD:EF:01:23:45:67'})
    mac_dict.undo()
    mac_dict.redo()
    assert mac_dict['key1'] == '01:23:45:67:89:AB'
    assert mac_dict['key2'] == 'CD:EF:01:23:45:67'
```

```
def test_undo_error_empty_history():
```

```
    mac_dict = MACAddressDict()
    with pytest.raises(UndoError):
        mac_dict.undo()
```

```
def test_redo_error_empty_future():
```

```
    mac_dict = MACAddressDict()
    with pytest.raises(RedoError):
        mac_dict.redo()
```