ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИСCЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

**Дисциплина:**

«Программирование»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

**Вариант  
2-4**

**Выполнил:**

Студент гр. N3151, Мочеков Семён Сергеевич



**Проверил:**

Грозов Владимир Андреевич

Санкт-Петербург

2024

1. **Задание.**

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.

Модуль состоит из наследника класса set, и принимает в себя только строки формата: Доменное имя.

**II. Исходный код**

Модуль:

import re

class FormatError(Exception):

    pass

class UndoError(Exception):

    pass

class RedoError(Exception):

    pass

class DomainSet(set):

    def \_\_init\_\_(self, \*args):

        super().\_\_init\_\_()

        self.history = []

        self.redo\_stack = []

        if args:

            for item in args[0]:

                self.add(item)

    def \_is\_valid\_domain(self, value):

        pattern = r'^(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,}$'

        return re.match(pattern, value)

    def add(self, value):

        if not isinstance(value, str):

            raise TypeError("Only strings are allowed")

        if not self.\_is\_valid\_domain(value):

            raise FormatError("Invalid domain name format")

        self.history.append(self.copy())

        self.redo\_stack.clear()

        super().add(value)

    def remove(self, value):

        if value not in self:

            raise KeyError(f"{value} not found in set")

        self.history.append(self.copy())

        self.redo\_stack.clear()

        super().remove(value)

    def undo(self):

        if not self.history:

            raise UndoError("No actions to undo")

        self.redo\_stack.append(self.copy())

        previous\_state = self.history.pop()

        self.clear()

        super().update(previous\_state)

    def redo(self):

        if not self.redo\_stack:

            raise RedoError("No actions to redo")

        self.history.append(self.copy())

        next\_state = self.redo\_stack.pop()

        self.clear()

        super().update(next\_state)

Pytest:

import pytest

from lab6ssmN3151 import DomainSet, FormatError, UndoError, RedoError

def test\_add\_valid\_domain():

    dset = DomainSet()

    dset.add("example.com")

    assert "example.com" in dset

def test\_add\_invalid\_domain():

    dset = DomainSet()

    with pytest.raises(FormatError):

        dset.add("1312321/1123")

def test\_add\_non\_string():

    dset = DomainSet()

    with pytest.raises(TypeError):

        dset.add(123)

def test\_remove():

    dset = DomainSet(["example.com"])

    dset.remove("example.com")

    assert "example.com" not in dset

def test\_remove\_nonexistent():

    dset = DomainSet()

    with pytest.raises(KeyError):

        dset.remove("nonexistent.com")

def test\_undo():

    dset = DomainSet(["example.com"])

    dset.add("example.org")

    dset.undo()

    assert "example.org" not in dset

    assert "example.com" in dset

def test\_redo():

    dset = DomainSet(["example.com"])

    dset.add("example.org")

    dset.undo()

    dset.redo()

    assert "example.org" in dset

def test\_undo\_without\_history():

    dset = DomainSet()

    with pytest.raises(UndoError):

        dset.undo()

def test\_redo\_without\_history():

    dset = DomainSet()

    with pytest.raises(RedoError):

        dset.redo()

def test\_undo\_redo\_sequence():

    dset = DomainSet(["example.com"])

    dset.add("example.org")

    dset.undo()

    dset.redo()

    assert "example.org" in dset

def test\_undo\_after\_undo\_redo():

    dset = DomainSet(["example.com"])

    dset.add("example.org")

    dset.undo()

    dset.redo()

    dset.undo()

    assert "example.org" not in dset

    assert "example.com" in dset

def test\_add\_multiple\_valid\_domains():

    dset = DomainSet()

    domains = ["example.com", "test.org", "domain.net"]

    for domain in domains:

        dset.add(domain)

    for domain in domains:

        assert domain in dset

def test\_remove\_multiple\_domains():

    dset = DomainSet(["example.com", "test.org", "domain.net"])

    dset.remove("test.org")

    assert "test.org" not in dset

    assert "example.com" in dset

    assert "domain.net" in dset

def test\_undo\_after\_multiple\_adds():

    dset = DomainSet()

    dset.add("example.com")

    dset.add("test.org")

    dset.add("domain.net")

    dset.undo()

    assert "domain.net" not in dset

    assert "test.org" in dset

    assert "example.com" in dset

def test\_redo\_after\_multiple\_undos():

    dset = DomainSet(["example.com", "test.org", "domain.net"])

    dset.remove("test.org")

    dset.undo()

    dset.undo()

    assert "example.com" in dset

    assert "test.org" in dset

    assert "domain.net" in dset

    dset.redo()

    assert "example.com" in dset

    assert "test.org" not in dset

    assert "domain.net" in dset

def test\_mixed\_operations\_with\_undo\_redo():

    dset = DomainSet()

    dset.add("example.com")

    dset.add("test.org")

    dset.remove("example.com")

    dset.undo()

    dset.add("domain.net")

    dset.redo()

    assert "example.com" in dset

    assert "test.org" in dset

    assert "domain.net" in dset

def test\_redo\_after\_new\_addition():

    dset = DomainSet(["example.com"])

    dset.add("test.org")

    dset.undo()

    dset.add("domain.net")

    with pytest.raises(RedoError):

        dset.redo()

def test\_undo\_redo\_boundary\_conditions():

    dset = DomainSet(["example.com"])

    dset.undo()

    with pytest.raises(UndoError):

        dset.undo()

    dset.redo()

    with pytest.raises(RedoError):

        dset.redo()

def test\_add\_multiple\_valid\_domains():

    dset = DomainSet()

    domains = ["example.com", "test.org", "domain.net"]

    for domain in domains:

        dset.add(domain)

    for domain in domains:

        assert domain in dset

def test\_remove\_multiple\_domains():

    dset = DomainSet(["example.com", "test.org", "domain.net"])

    dset.remove("test.org")

    assert "test.org" not in dset

    assert "example.com" in dset

    assert "domain.net" in dset

def test\_undo\_after\_multiple\_adds():

    dset = DomainSet()

    dset.add("example.com")

    dset.add("test.org")

    dset.add("domain.net")

    dset.undo()

    assert "domain.net" not in dset

    assert "test.org" in dset

    assert "example.com" in dset

def test\_redo\_after\_multiple\_undos():

    dset = DomainSet(["example.com", "test.org", "domain.net"])

    dset.remove("test.org")

    dset.undo()

    assert "example.com" in dset

    assert "test.org" in dset

    assert "domain.net" in dset

    dset.redo()

    assert "example.com" in dset

    assert "test.org" not in dset

    assert "domain.net" in dset

def test\_mixed\_operations\_with\_undo\_redo():

    dset = DomainSet()

    dset.add("example.com")

    dset.add("test.org")

    dset.remove("example.com")

    dset.undo()

    dset.add("domain.net")

    dset.undo()

    dset.redo()

    assert "example.com" in dset

    assert "test.org" in dset

    assert "domain.net" in dset

def test\_redo\_after\_new\_addition():

    dset = DomainSet(["example.com"])

    dset.add("test.org")

    dset.undo()

    dset.add("domain.net")

    with pytest.raises(RedoError):

        dset.redo()

def test\_undo\_redo\_boundary\_conditions():

    dset = DomainSet(["example.com"])

    dset.undo()

    with pytest.raises(UndoError):

        dset.undo()

    dset.redo()

    with pytest.raises(RedoError):

        dset.redo()

def test\_domain\_format\_check():

    dset = DomainSet()

    valid\_domains = ["example.com", "sub.example.co.uk", "test.org", "my-domain.net", "another.test.edu"]

    invalid\_domains = ["example", "test@org", "my\_domain.net", "sub.example.", "example.c"]

    for domain in valid\_domains:

        dset.add(domain)

        assert domain in dset

    for domain in invalid\_domains:

        with pytest.raises(FormatError):

            dset.add(domain)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**III. Тесты**