# ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

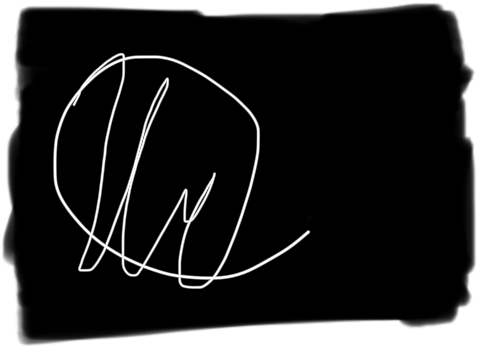
**Дисциплина:**

# «Программирование»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

**Выполнил:**

# Студент группы N3149

Ильчук Денис.



**Проверила:**

|

|

# Горлина А.В.

Задание 5-3

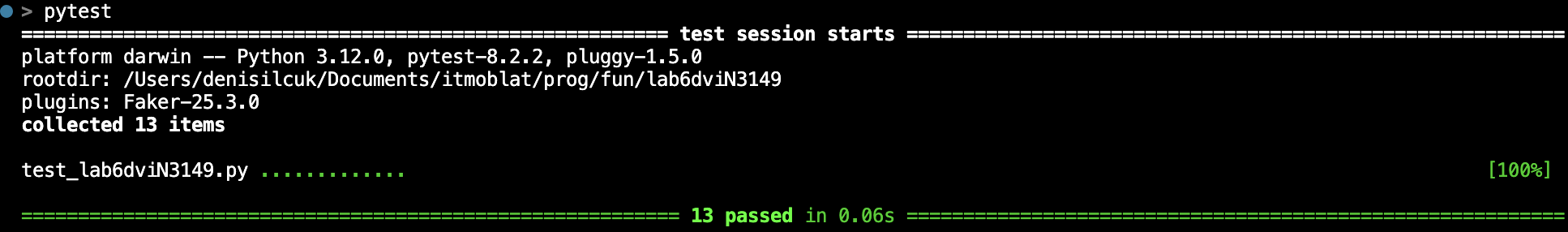








# Примеры работы pytest



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(f"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

Исходный код теста для программы

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\_mac'

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\_mac'})

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\_diict = MACAddressDict()

with pytest.raises(RedoError):

mac\_diict.redo()