

**Министерство науки и высшего образования Российской Федерации**  
федеральное государственное автономное образовательное учреждение высшего  
образования  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**


**Отчет**

Лабораторная работа No 5

Объектно-ориентированное программирование в Python

Вариант 5-5.

Автор: Киселёва Анна Николаевна

подпись 

Факультет: ФБИТ

Группа: N3147

Преподаватели:

Грозов Владимир Андреевич



Санкт-Петербург 2025

## Задание:

5 - dict (ограничение формата накладывается на значения словаря)

5 - Адрес электронной почты

## Примеры работы программы:

- 1) добавление в словарь новых значений
- 2) работа функции `d.undo()` до момента, когда в словаре остается одно значение. После программа выводит ошибку `UndoError`

```
anna@Huaweianna:~/projects/prg5ankN3147$ python3
Python 3.10.12 (main, Feb  4 2025, 14:57:36) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from prg5ankN3147 import MyDict
>>> d = MyDict([('user1', 'a@b.com'), ('user2', 'kiselevanna27@mail.ru')])
>>> d.append({'user3': 'babana@gmail.ru'})
>>> d.append({'user4': 'bbbb@lala.com'})
>>> print(d)
{'user1': 'a@b.com', 'user2': 'kiselevanna27@mail.ru', 'user3': 'babana@gmail.ru', 'user4': 'bbbb@lala.com'}
>>> d.undo()
Undo: {'user1': 'a@b.com', 'user2': 'kiselevanna27@mail.ru', 'user3': 'babana@gmail.ru'}
>>> d.undo()
Undo: {'user1': 'a@b.com', 'user2': 'kiselevanna27@mail.ru'}
>>> d.undo()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/anna/projects/prg5ankN3147/prg5ankN3147.py", line 67, in undo
    UndoError.raise_error()
  File "/home/anna/projects/prg5ankN3147/prg5ankN3147.py", line 16, in raise_error
    raise cls(f"Undo error")
prg5ankN3147.UndoError: Undo error
>>>
```

- 1) работа функции `d.undo()`
- 2) работа функции `d.redo()` до момента первого изменения. После программа выводит ошибку `RedoError`

```
prg5ankN3147.UndoError: Undo error
>>> d.redo()
Redo: {'user1': 'a@b.com', 'user2': 'kiselevanna27@mail.ru', 'user3': 'babana@gmail.ru'}
>>> d.redo()
Redo: {'user1': 'a@b.com', 'user2': 'kiselevanna27@mail.ru', 'user3': 'babana@gmail.ru', 'user4': 'bbbb@lala.com'}
>>> d.redo()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/anna/projects/prg5ankN3147/prg5ankN3147.py", line 75, in redo
    RedoError.raise_error()
  File "/home/anna/projects/prg5ankN3147/prg5ankN3147.py", line 21, in raise_error
    raise cls(f"Redo error")
prg5ankN3147.RedoError: Redo error
>>>
```

- 1) Проверка работоспособности тестов

```
__pycache__ prg5ankN3147.py test_prg5ankN3147.py
anna@Huaweianna:~/projects/prg5ankN3147$ pytest
===== test session starts =====
platform linux -- Python 3.10.12, pytest-8.3.5, pluggy-1.6.0
rootdir: /home/anna/projects/prg5ankN3147
collected 9 items

test_prg5ankN3147.py ..... [100%]

===== 9 passed in 0.02s =====
anna@Huaweianna:~/projects/prg5ankN3147$ |
```

## Исходный текст программы:

```
test_prg5ankN3147.py  prg5ankN3147.py ×
1  from email_validator import validate_email, EmailNotValidError
2  import re
3  # вызов ошибок FormatError TypeError UndoError RedoError
4  class FormatError(Exception): 3 usages new*
5      @classmethod 1 usage new*
6      def raise_error(cls, x):
7          raise cls(f"Invalid email format in values: {x}")
8  class TypeError(Exception): 1 usage new*
9      @classmethod 1 usage new*
10     def raise_error(cls, x):
11         raise cls(f"Unexpected type of {x}")
12
13     class UndoError(Exception): 3 usages new*
14         @classmethod 1 usage new*
15         def raise_error(cls):
16             raise cls(f"Undo error")
17
18     class RedoError(Exception): 3 usages new*
19         @classmethod 1 usage new*
20         def raise_error(cls):
21             raise cls(f"Redo error")
22
23     # две функции проверки корректного email адреса. Вторая с помощью библиотеки питоновской, она как то медленно работает
24     def re_is_valid_email(email): # с использованием шаблона (быстрее) но не совсем точно 3 usages new*
25         regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b'
26         if not re.fullmatch(regex, email):
27             return False
28         return True
29
30
31
32
33
34
35
36     class MyDict(dict): #объявление класса 7 usages new*
37         def __init__(self, iter=None): new*
38             super().__init__() # связь с родительским классом
39             self.history = [] # сохранение истории для undo и redo
40             self.iter = 0 # счетчик для undo и redo
41             if iter is not None:
42                 for k, v in iter: #добавление элементов
43                     self[k] = v
44                     self.current_state() #сохранение для истории
45
46         def check(self, x): #проверка корректности введенного значения в словаре 1 usage new*
47             if not isinstance(x, dict): #соответствие типу словаря
48                 TypeError.raise_error(x)
49
50             for email in x.values(): #соответствие email виду
51                 if not re_is_valid_email(str(email)):
52                     FormatError.raise_error(x)
53
54         def current_state(self): #смотрим что сейчас в словаре 2 usages new*
55             if self.iter + 1 < len(self.history): #если если сейчас элементов меньше чем было когда (бфпо вызвано undo)
56                 self.history = self.history[:self.iter+1] #обрезаем
57                 self.history.append(dict(self)) #добавляем в историю
58                 self.iter = len(self.history)-1 #обновляем указатель
59
```

```

60     def append(self, x): 5 usages new *
61         self.check(x) #проверка
62         self.update(x) #добавление
63         self.current_state() #обновление
64
65     def undo(self): 4 usages new *
66         if self.iter <= 0: #уже некуда убавлять то вызов ошибки
67             UndoError.raise_error()
68         self.iter -= 1 # переход к предыдущему элементу
69         self.clear()
70         self.update(self.history[self.iter])
71         print(f"Undo: {self}")
72
73     def redo(self): 5 usages new *
74         if self.iter >= len(self.history)-1: #если в конце словаря
75             RedoError.raise_error()
76         self.iter+=1 #переход к следующему
77         self.clear()
78         self.update(self.history[self.iter])
79         print(f"Redo: {self}")
80

```

Тестовый файл:

```

1  ✓ from prg5ankN3147 import FormatError, UndoError, RedoError, MyDict, re_is_valid_email
2      import pytest
3      #корректные email адреса
4  ✓ valid_emails = [
5      {'user1': 'a@b.com'},
6      {'user2': 'kiselevanna27@mail.ru'},
7      {'user3': 'marinovannye@ogurtsi.com'},
8      {'user4': 'babana@gmail.ru'}
9  ]
10     #некорректные email адреса
11  ✓ invalid_emails = [
12      {'user5': 'dddd@gcom'},
13      {'user6': 'dddcom'},
14      {'user7': '!@#@$@$'},
15      {'user8': 'a b c '}
16  ]
17     #проверка функции соответствия email адресу
18  ✓ @pytest.mark.parametrize("maybe_email, expected_result", [
19      ("kiselevanna27@mail.ru", True),
20      ("a@b.com", True),
21      ("aba", False),
22      ("cd.com", False),
23  ])
24  ▶ ✓ def test_re_is_valid_email(maybe_email, expected_result):
25      """Тест на проверку регулярного выражения"""
26      assert re_is_valid_email(maybe_email) is expected_result
27
28

```

```

29 ▶ def test_valid_append():
30     """Тест на проверку добавления корректного email"""
31     d = MyDict()
32     for email in valid_emails:
33         d.append(email)
34
35     for email in valid_emails:
36         for k, v in email.items():
37             assert k in d
38             assert d[k] == v
39
40 ▶ def test_invalid_append():
41     """Тест на проверку добавления некорректного email"""
42     d = MyDict()
43     for email in invalid_emails:
44         with pytest.raises(FormatError):
45             d.append(email)
46     assert len(d) == 0
47
48 ▶ def test_valid_undo_redo():
49     """Тест на проверку работающей функции undo"""
50     d = MyDict()
51     for email in valid_emails:
52         d.append(email)
53     d.undo()
54     assert 'user4' not in d
55     d.redo()
56     assert 'user4' in d
57

```

```

57
58 ▶ def test_invalid_undo():
59     """Тест на проверку неработающей функции undo"""
60     d = MyDict()
61     with pytest.raises(UndoError):
62         d.undo()
63
64 ▶ def test_invalid_redo():
65     """Тест на проверку неработающей функции redo"""
66     d = MyDict()
67     with pytest.raises(RedoError):
68         d.redo()
69

```