**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Институт Цифры**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №3**

по дисциплине «Разработка современных приложений на Python»

студента 3 курса

**Лободы Максима Алексеевича**

Направление 02.03.02– «Фундаментальная информатика и информационные технологии»

Преподаватель:

ст. пр. Зимин А. И.

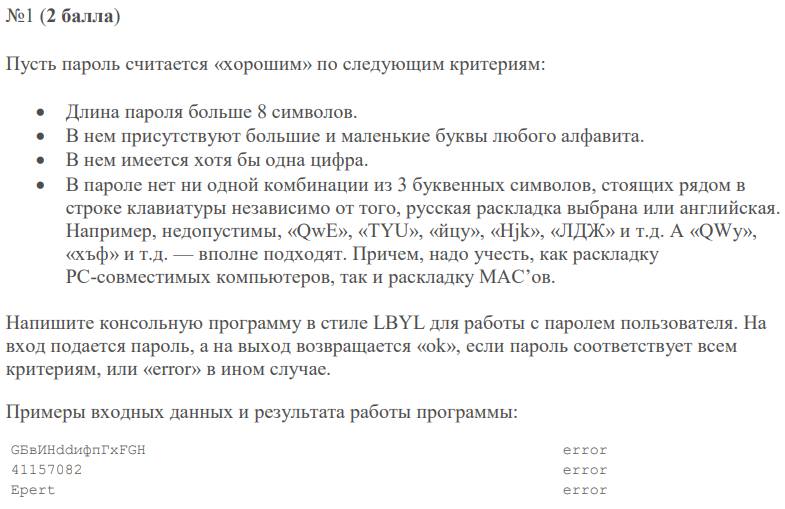
Работа защищена

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Кемерово 2023

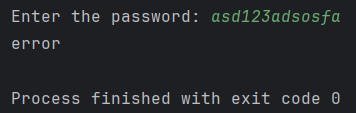
**Основная часть**

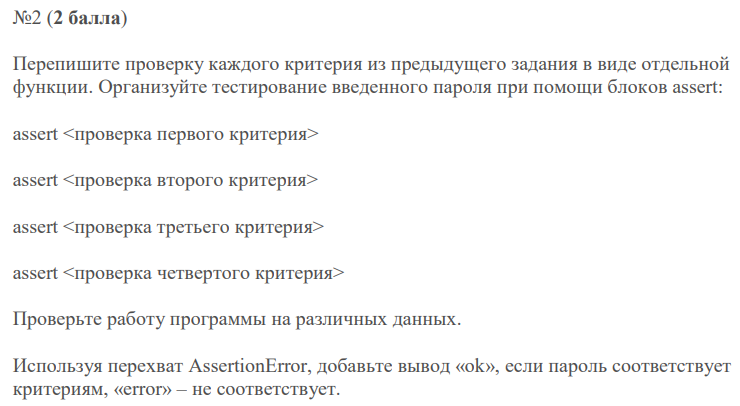


Код приложения

def check\_password(password):  
 if len(password) <= 8:  
 return "error"  
  
 has\_upper = False  
 has\_lower = False  
 for char in password:  
 if char.islower():  
 has\_lower = True  
 if char.isupper():  
 has\_upper = True  
 if not (has\_lower and has\_upper):  
 return "error"  
  
 has\_digit = any(char.isdigit() for char in password)  
 if not has\_digit:  
 return "error"  
  
 keyboard\_layouts = ["qwertyuiop", "asdfghjkl", "zxcvbnm", "йцукенгшщзхъ", "фывапролджэ", "ячсмитьбю"]  
  
 for layout in keyboard\_layouts:  
 for i in range(len(layout) - 2):  
 substring = layout[i:i + 3]  
 if substring in password.lower():  
 return "error"  
  
 return "ok"  
  
  
password = input("Enter the password: ")  
result = check\_password(password)  
print(result)

Результат выполнения

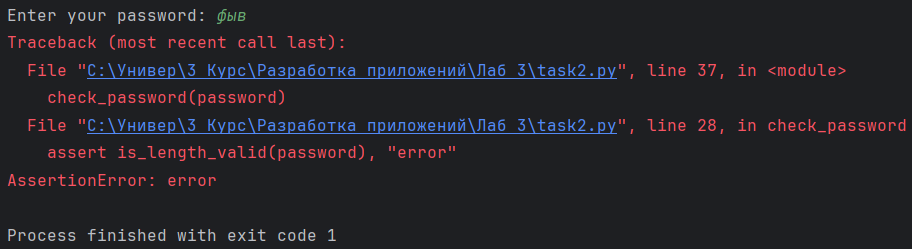


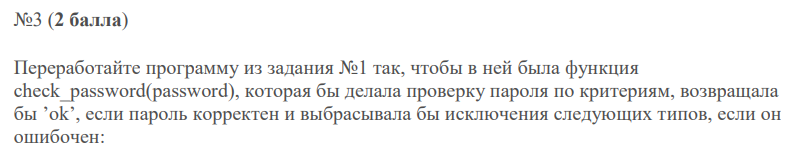


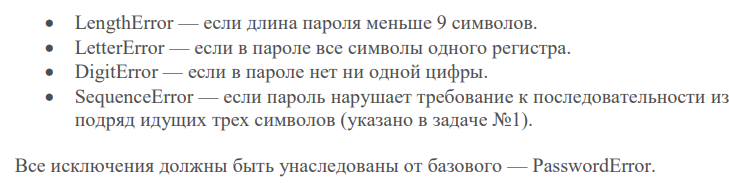
Код приложения

def is\_length\_valid(password):  
 return len(password) > 8  
  
  
def has\_upper\_and\_lower(password):  
 has\_upper = any(char.isupper() for char in password)  
 has\_lower = any(char.islower() for char in password)  
 return has\_upper and has\_lower  
  
  
def has\_digit(password):  
 return any(char.isdigit() for char in password)  
  
  
def no\_keyboard\_patterns(password):  
 keyboard\_layouts = ["qwertyuiop", "asdfghjkl", "zxcvbnm", "йцукенгшщзхъ", "фывапролджэ", "ячсмитьбю"]  
  
 for layout in keyboard\_layouts:  
 for i in range(len(layout) - 2):  
 substring = layout[i:i + 3]  
 if substring in password.lower():  
 return False  
  
 return True  
  
  
def check\_password(password):  
 assert is\_length\_valid(password), "error"  
 assert has\_upper\_and\_lower(password), "error"  
 assert has\_digit(password), "error"  
 assert no\_keyboard\_patterns(password), "error"  
  
 return "ok"  
  
  
password = input("Enter your password: ")  
check\_password(password)

Результат выполнения



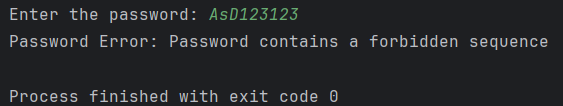


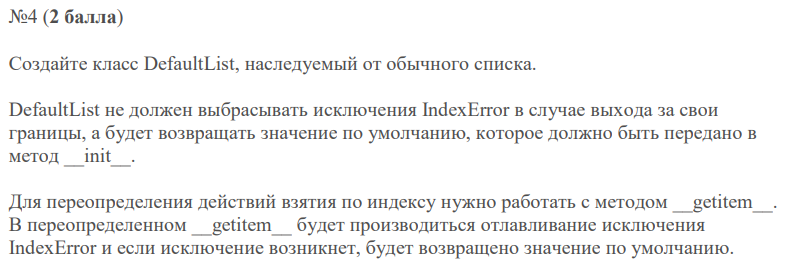


Код приложения

class PasswordError(Exception):  
 pass  
  
  
class LengthError(PasswordError):  
 pass  
  
  
class LetterError(PasswordError):  
 pass  
  
  
class DigitError(PasswordError):  
 pass  
  
  
class SequenceError(PasswordError):  
 pass  
  
  
def check\_password(password):  
 if len(password) <= 8:  
 raise LengthError("Password is too short")  
  
 has\_upper = any(char.isupper() for char in password)  
 has\_lower = any(char.islower() for char in password)  
  
 if not (has\_lower and has\_upper):  
 raise LetterError("Password must contain both upper and lower case letters")  
  
 has\_digit = any(char.isdigit() for char in password)  
  
 if not has\_digit:  
 raise DigitError("Password must contain at least one digit")  
  
 keyboard\_layouts = ["qwertyuiop", "asdfghjkl", "zxcvbnm", "йцукенгшщзхъ", "фывапролджэ", "ячсмитьбю"]  
  
 for layout in keyboard\_layouts:  
 for i in range(len(layout) - 2):  
 substring = layout[i:i + 3]  
 if substring in password.lower():  
 raise SequenceError("Password contains a forbidden sequence")  
  
 return "ok"  
  
  
def main():  
 try:  
 password = input("Enter the password: ")  
 result = check\_password(password)  
 print(result)  
 except PasswordError as e:  
 print(f"Password Error: {e}")  
  
  
main()

Результат выполнения

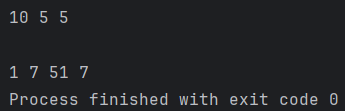


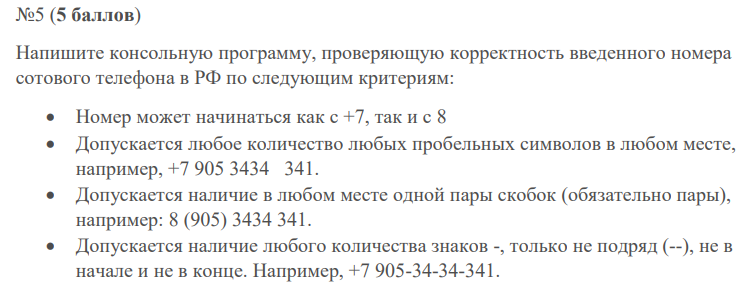


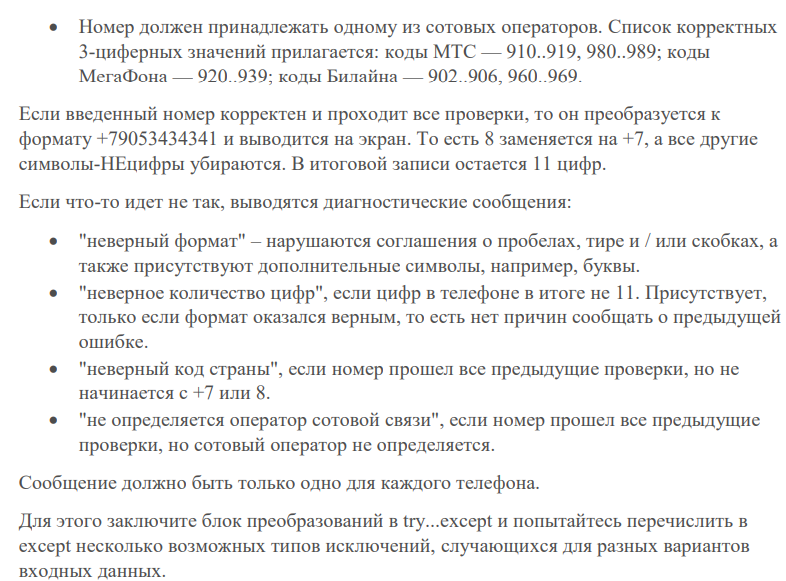
Код приложения

class DefaultList(list):  
 def \_\_init\_\_(self, default\_value):  
 self.default\_value = default\_value  
 super().\_\_init\_\_()  
  
 def \_\_getitem\_\_(self, index):  
 try:  
 return super().\_\_getitem\_\_(index)  
 except IndexError:  
 return self.default\_value  
  
  
# Пример 1  
s1 = DefaultList(5)  
s1.extend([4, 10])  
indexes1 = [1, 124, 1882]  
for i in indexes1:  
 print(s1[i], end=" ")  
  
print('\n')  
  
# Пример 2  
s2 = DefaultList(51)  
s2.extend([1, 5, 7])  
indexes2 = [0, 2, 1000, -1]  
for i in indexes2:  
 print(s2[i], end=" ")

Результат выполнения



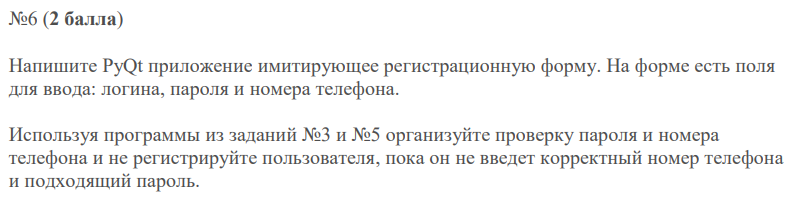
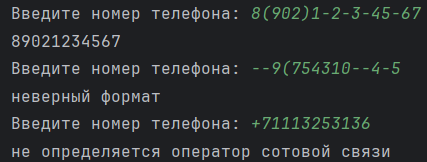




Код приложения

class FormatError(Exception):  
 pass  
  
  
class DigitCountError(Exception):  
 pass  
  
  
class CountryCodeError(Exception):  
 pass  
  
  
class OperatorError(Exception):  
 pass  
  
  
def remove\_spaces(s):  
 return "".join(s.split())  
  
  
def has\_single\_closed\_bracket\_pair(number):  
 stack = []  
 for char in number:  
 if char == '(':  
 stack.append(char)  
 elif char == ')':  
 if not stack:  
 return False  
 stack.pop()  
 return not stack  
  
  
def has\_valid\_hyphens(number):  
 for i in range(1, len(number) - 1):  
 if number[i] == '-':  
 if number[i - 1] == '-' or number[i + 1] == '-':  
 return False  
 return True  
  
  
def has\_letters(number):  
 return any(char.isalpha() for char in number)  
  
  
def format\_phone\_number(number):  
 if (not has\_single\_closed\_bracket\_pair(number) or not has\_valid\_hyphens(number) or has\_letters(number)  
 or len(number) == 0):  
 raise FormatError("неверный формат")  
  
 cleaned\_number = ''.join(filter(str.isdigit, number))  
  
 operator\_codes = {  
 'МТС': ['910', '911', '912', '913', '914', '915', '916', '917', '918', '919', '980', '981', '982', '983', '984', '985', '986', '987', '988', '989'],  
 'МегаФон': ['920', '921', '922', '923', '924', '925', '926', '927', '928', '929', '930', '931', '932', '933', '934', '935', '936', '937', '938', '939'],  
 'Билайн': ['902', '903', '904', '905', '906', '960', '961', '962', '963', '964', '965', '966', '967', '968', '969']  
 }  
  
 operator\_code = cleaned\_number[1:4]  
  
 operator\_name = None  
 for operator, codes in operator\_codes.items():  
 if operator\_code in codes:  
 operator\_name = operator  
 break  
  
 if len(cleaned\_number) != 11:  
 raise DigitCountError("неверное количество цифр")  
  
 if number.startswith('+7'):  
 number = '+7' + number[2:]  
 elif number.startswith('8'):  
 number = '+7' + number[1:]  
 else:  
 raise CountryCodeError("неверный код страны")  
  
 if operator\_name is None:  
 raise OperatorError("не определяется оператор сотовой связи")  
  
 return cleaned\_number  
  
  
def main():  
 while True:  
 try:  
 phone\_number = input("Введите номер телефона: ")  
 formatted\_number = format\_phone\_number(phone\_number)  
 print(formatted\_number)  
 except (CountryCodeError, DigitCountError, OperatorError, FormatError) as e:  
 print(e)  
  
  
main()

Результат выполнения

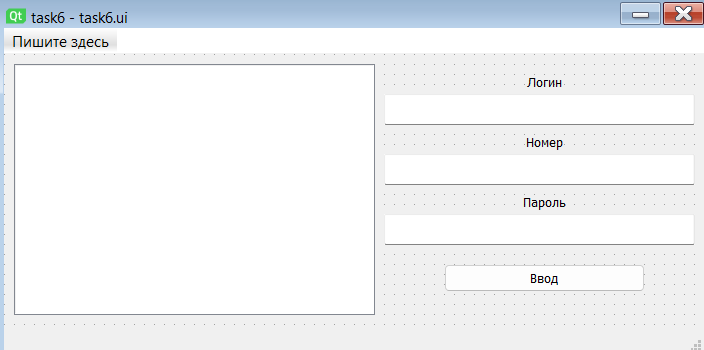


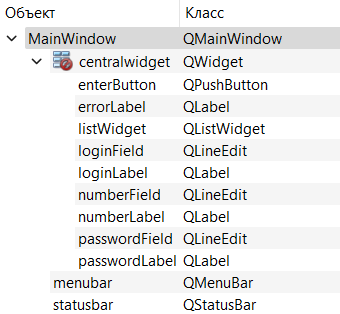
Код приложения

import sys  
  
from task5 import \*  
from task3 import \*  
  
from PyQt5 import uic  
from PyQt5.QtWidgets import QApplication, QMainWindow  
  
  
class MyWidget(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 uic.loadUi('task6.ui', self)  
 self.setFixedSize(700, 325)  
 self.enterButton.clicked.connect(self.addRecord)  
  
 def addRecord(self):  
 login = self.loginField.text()  
 number = self.numberField.text()  
 password = self.passwordField.text()  
  
 try:  
 check\_password(password)  
 formatted\_number = format\_phone\_number(number)  
 self.listWidget.addItem(f"{login}: {formatted\_number}")  
 self.errorLabel.setText("")  
 except PasswordError as e:  
 self.errorLabel.setText(f"Password Error: {e}")  
 except (CountryCodeError, DigitCountError, OperatorError, FormatError) as e:  
 self.errorLabel.setText(f"Number Error: {e}")  
  
  
def except\_hook(cls, exception, traceback):  
 sys.\_\_excepthook\_\_(cls, exception, traceback)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = MyWidget()  
 ex.show()  
 sys.excepthook = except\_hook  
 sys.exit(app.exec\_())

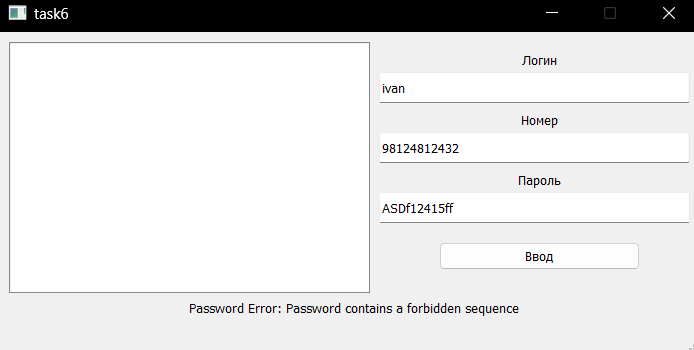
Как можно видеть, методы check\_password() и format\_phone\_number() здесь импортируются из 3 и 5 заданий.

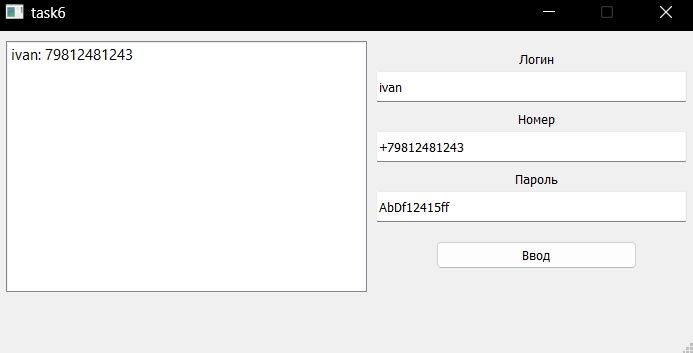
Интерфейс

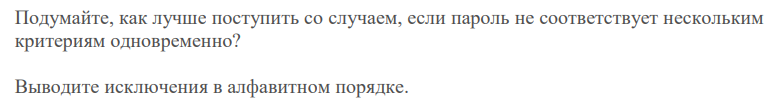
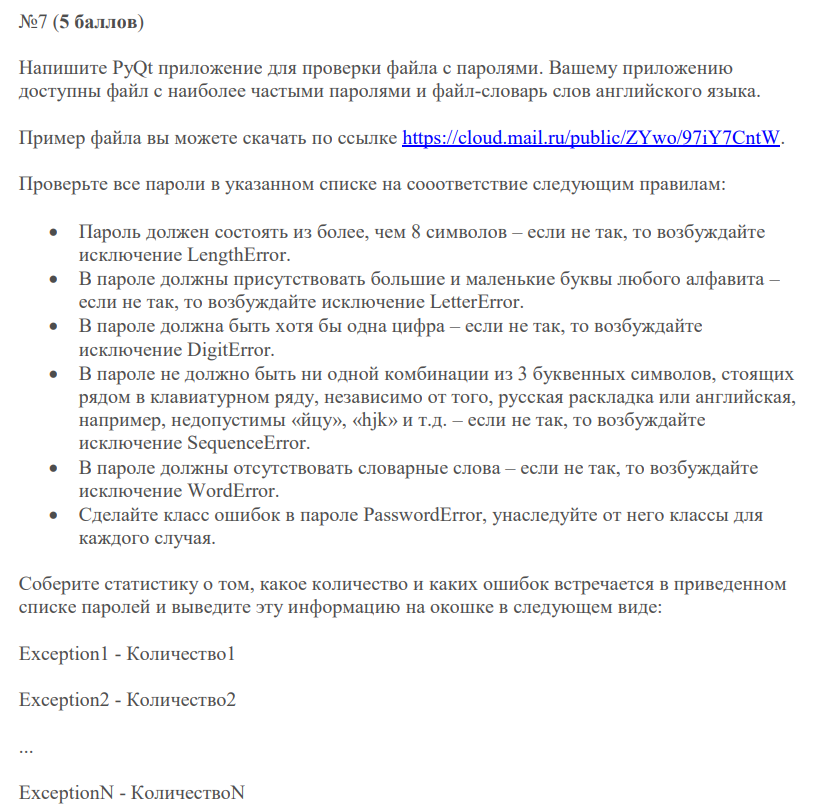




Демонстрация работы







Код приложения

Для выполнения этого задания был слегка изменен код метода проверки пароля из 3 задания.

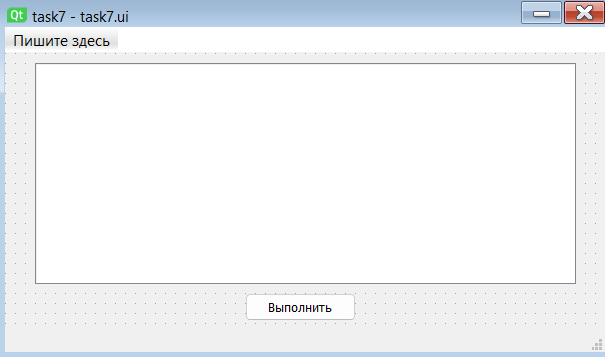
dict\_password\_check.py:

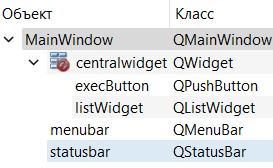
class PasswordError(Exception):  
 pass  
  
  
class LengthError(PasswordError):  
 pass  
  
  
class LetterError(PasswordError):  
 pass  
  
  
class DigitError(PasswordError):  
 pass  
  
  
class SequenceError(PasswordError):  
 pass  
  
  
class WordError(PasswordError):  
 pass  
  
  
def dict\_password\_check(password, word\_dict):  
 errors = []  
  
 try:  
 if len(password) <= 8:  
 errors.append(LengthError("Password is too short"))  
  
 has\_upper = any(char.isupper() for char in password)  
 has\_lower = any(char.islower() for char in password)  
  
 if not (has\_lower and has\_upper):  
 errors.append(LetterError("Password must contain both upper and lower case letters"))  
  
 has\_digit = any(char.isdigit() for char in password)  
  
 if not has\_digit:  
 errors.append(DigitError("Password must contain at least one digit"))  
  
 keyboard\_layouts = ["qwertyuiop", "asdfghjkl", "zxcvbnm", "йцукенгшщзхъ", "фывапролджэ", "ячсмитьбю"]  
  
 for layout in keyboard\_layouts:  
 for i in range(len(layout) - 2):  
 substring = layout[i:i + 3]  
 if substring in password.lower():  
 errors.append(SequenceError("Password contains a forbidden sequence"))  
  
 if any(word in password.lower() for word in word\_dict):  
 errors.append(WordError("Password contains a dictionary word"))  
  
 except PasswordError as e:  
 errors.append(e)  
  
 return errors

task7.py:

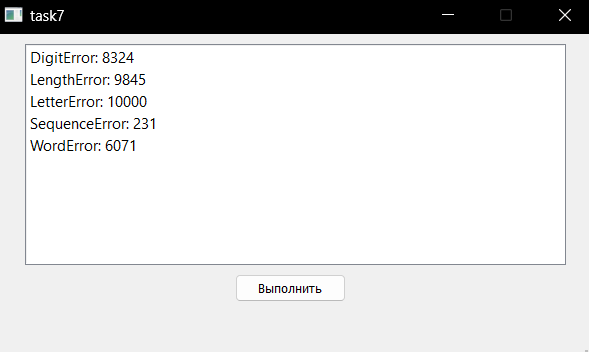
import sys  
  
from dict\_password\_check import \*  
  
from PyQt5 import uic  
from PyQt5.QtWidgets import QApplication, QMainWindow  
  
  
class MyWidget(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 uic.loadUi('task7.ui', self)  
 self.setFixedSize(600, 325)  
  
 self.passwords = []  
 self.english\_words = []  
  
 self.init()  
 self.execButton.clicked.connect(self.on\_exec\_button\_click)  
  
 def init(self):  
 with open('data/top 10000 passwd.txt', 'r') as file:  
 self.passwords = file.read().splitlines()  
  
 with open('data/top-9999-words.txt', 'r') as file:  
 self.english\_words = set(file.read().splitlines())  
  
 def on\_exec\_button\_click(self):  
 error\_stats = {}  
  
 for password in self.passwords:  
 errors = dict\_password\_check(password, self.english\_words)  
 for error in errors:  
 error\_name = error.\_\_class\_\_.\_\_name\_\_  
 if error\_name in error\_stats:  
 error\_stats[error\_name] += 1  
 else:  
 error\_stats[error\_name] = 1  
  
 for error, count in sorted(error\_stats.items()):  
 self.listWidget.addItem(f"{error}: {count}")  
  
  
def except\_hook(cls, exception, traceback):  
 sys.\_\_excepthook\_\_(cls, exception, traceback)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = MyWidget()  
 ex.show()  
 sys.excepthook = except\_hook  
 sys.exit(app.exec\_())

Интерфейс





Демонстрация работы



Ввиду того, что ни один из паролей не содержал заглавных букв и не был длиннее 8-ми символов, программа изначально выводила только два типа ошибок: LengthError и LetterError, потому было решено считать несколько типов ошибок для каждого из паролей.