Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №3**

По дисциплине: «Новые технологии в РПС»

Тема: «GUI»

|  |
| --- |
| Выполнил:  студент группы  М.ПИН.РИС - 23.08  Тимофеев Александр Сергеевич |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь 2023

**Вариант №1-2**

**Постановка задачи:**

Создать приложение с графическим интерфейсом, повторяющее функционал лабораторной работы 2. Необходимо создать файл main\_window.py в котором будет реализован графический интерфейс вашего приложения. Здесь не должно быть реализации алгоритмов, все алгоритмы должны быть импортированы как функции или объекты из других файлов.

* Приложение должно запрашивать у пользователя путь к папке исходного датасета
* Приложение должно иметь кнопку для создания файла аннотации исходного датасета. Для этого потребуется запросить у пользователя путь к файлу назначения.
* Приложение должно иметь кнопку для создания датасета с другой организацией файлов (в соответствии с пунктами лабораторной работы 2) и файла аннотации создаваемого датасета. Для этого потребуется запросить у пользователя путь к папке назначения.

*Вариант 1-2*

1. Приложение должно иметь поле ввода даты и кнопку "Получить данные". Пользователь вводит дату и видит в интерфейсе данные для этой даты.

*Вариант 3-10*

1. Приложение должно иметь кнопки для получения следующего экземпляра класса из датасета. Например, если у вас датасет из кошек и собак, то должна быть кнопка "Следующая кошка" и кнопка "Следующая собака". После нажатия на них должен быть получен следующий путь при помощи итератора, а затем отображена картинка новой кошки или собаки из вашего датасета в интерфейсе вашей программы. В случае варианта с текстом отображаем текст.

**Код программы (Python):**

Библиотеки главной формы:

import pandas as pd

from datetime import datetime

from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton, QVBoxLayout, QWidget, QFileDialog, QLineEdit, QTableWidget, QTableWidgetItem, QHBoxLayout, QLabel, QMessageBox, QComboBox

from PyQt6.QtCore import QDate

from dataset\_handler import DatasetHandler

from operations\_dataset import DatasetOperations

from directory\_handler import DirectoryHandler as dir\_h

from form\_images import ImageWindow

from form\_currency import CurrencyWindow

Константы:

CURRENCY\_FIELDS = ['date', 'nominal', 'value', 'vunitRate']

IMAGES\_FIELDS = ['date', 'file\_name', 'url', 'path']

CURR\_DIR = dir\_h.set\_current\_dir()

Инициализация главной формы:

class MainWindow(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.folder\_path = ""

        self.df = pd.DataFrame()

        self.dataset\_operations = DatasetOperations(CURR\_DIR)

        self.init\_ui()

        self.choose\_dataset\_from\_file()

Инициализация компонентов формы:

def init\_ui(self):

        self.setWindowTitle('Операции с датасетом')

        v\_layout = QVBoxLayout()

        self.button\_choose\_dataset = QPushButton('Выбор датасета')

        self.button\_choose\_dataset.clicked.connect(self.choose\_dataset\_from\_file)

        self.combo\_box\_fields = QComboBox()

        self.combo\_box\_fields.addItem("'date', 'nominal', 'value', 'vunitRate'")

        self.combo\_box\_fields.addItem("'date', 'file\_name', 'url', 'path'")

        self.textbox\_date = QLineEdit()

        self.textbox\_date.setPlaceholderText('Введите дату')

        self.button\_search = QPushButton('Получить данные')

        self.button\_search.clicked.connect(self.get\_data\_with\_date)

        self.label\_separation\_operations = QLabel('Разделение по ...')

        self.button\_separation\_date\_by\_data = QPushButton('... даты от данных')

        self.button\_separation\_date\_by\_data.clicked.connect(self.separation\_date\_by\_data)

        self.button\_separation\_by\_years = QPushButton('... годам')

        self.button\_separation\_by\_years.clicked.connect(self.separation\_by\_years)

        self.button\_separation\_by\_weeks = QPushButton('... неделям')

        self.button\_separation\_by\_weeks.clicked.connect(self.separation\_by\_weeks)

        self.button\_show\_images = QPushButton('Просмотр данных')

        self.button\_show\_images.clicked.connect(self.show\_form\_data)

        v\_layout.addWidget(self.button\_choose\_dataset)

        v\_layout.addWidget(self.combo\_box\_fields)

        v\_layout.addWidget(self.textbox\_date)

        v\_layout.addWidget(self.button\_search)

        v\_layout.addWidget(self.label\_separation\_operations)

        v\_layout.addWidget(self.button\_separation\_date\_by\_data)

        v\_layout.addWidget(self.button\_separation\_by\_years)

        v\_layout.addWidget(self.button\_separation\_by\_weeks)

        v\_layout.addWidget(self.button\_show\_images)

        v\_layout.addStretch(1) #чтобы кнопки не расплылись по форме от горизонтального слоя

        self.table = QTableWidget()

        h\_layout = QHBoxLayout()

        h\_layout.addLayout(v\_layout)

        h\_layout.addWidget(self.table)

        container = QWidget()

        container.setLayout(h\_layout)

        self.setCentralWidget(container)

Разделение полей датасета при его загрузки

    def split\_fields(self, text: str) -> str:

        text = text.replace("'", "")

        res = text.split(", ")

        return res

Выбор датасета

def choose\_dataset\_from\_file(self):

        try:

            dat\_h = DatasetHandler()

            file\_dialog = QFileDialog()

            self.folder\_path = file\_dialog.getOpenFileName()[0]

            if(self.df is None or self.df.empty):

                self.df = dat\_h.create\_dataset([self.folder\_path])

                self.current\_fields = self.df.columns.tolist()

            else:

                self.df = dat\_h.create\_dataset\_from\_files([self.folder\_path], self.split\_fields(self.combo\_box\_fields.currentText()))#self.combo\_box\_fields.currentText())

                self.current\_fields = self.df.columns.tolist()

            self.update\_table()

        except Exception as e:

            self.show\_message\_box("Ошибка", str(e))

Метод отображения окна сообщения результата

    def show\_message\_box(self, title: str, text: str):

        message\_box = QMessageBox()

        message\_box.setIcon(QMessageBox.Icon.Information)

        message\_box.setText(text)

        message\_box.setWindowTitle(title)

        message\_box.exec()

Получение данных по дате

    def get\_data\_with\_date(self):

        date\_str = self.textbox\_date.text()

        date = QDate.fromString(date\_str, 'dd.MM.yyyy')

        date\_datetime = datetime(date.year(), date.month(), date.day())

        data = self.dataset\_operations.get\_data\_from\_date(self.df, date\_datetime)

        self.show\_message\_box("Полученные данные по дате", str(data))

Разделение даты от данных, данных по годам/неделям

    def separation\_date\_by\_data(self):

        self.dataset\_operations.separation\_date\_by\_data(self.df)

        self.show\_message\_box("Успех", 'Данные отделены от даты')

    def separation\_by\_years(self):

        self.dataset\_operations.separation\_by\_years(self.df)

        self.show\_message\_box("Успех", 'Данные разделены по годам')

    def separation\_by\_weeks(self):

        self.dataset\_operations.separation\_by\_weeks(self.df)

        self.show\_message\_box("Успех", 'Данные разделены по неделям')

Обновление таблицы датасета

    def update\_table(self):

        if not self.df.empty:

            self.table.setRowCount(len(self.df))

            self.table.setColumnCount(len(self.df.columns))

            self.table.setHorizontalHeaderLabels(self.df.columns)

            for i in range(len(self.df)):

                for j in range(len(self.df.columns)):

                    item = self.df.iat[i, j]

                    if j == 0: #'date'

                        date\_str = item.strftime('%Y-%m-%d')

                        date = QDate.fromString(date\_str, 'yyyy-MM-dd')

                        item\_str = date.toString('yyyy-MM-dd')

                    else:

                        item\_str = str(item)

                    self.table.setItem(i, j, QTableWidgetItem(item\_str))

                    #self.table.setItem(i, j, QTableWidgetItem(str(self.df.iat[i, j])))

            self.table.resizeColumnsToContents()

Вызов формы отображения данных (картинки/числа)

    def show\_form\_data(self):

        if(self.current\_fields == self.split\_fields("'date', 'nominal', 'value', 'vunitRate'")):

            self.currency\_window = CurrencyWindow(self.df)

            self.currency\_window.show()

        elif(self.current\_fields == self.split\_fields("'date', 'file\_name', 'url', 'path'")):

            self.image\_window = ImageWindow(self.df)

            self.image\_window.show()

Форма отображения данныых Валюты

Импорты

import pandas as pd

from PyQt6.QtWidgets import QLabel, QPushButton, QVBoxLayout, QWidget

Код формы CurrencyWindow

class CurrencyWindow(QWidget):

    def \_\_init\_\_(self, df):

        super().\_\_init\_\_()

        self.setWindowTitle('Просмотр данных')

        self.label\_name = QLabel()

        self.iterator = iter(DataIterator(df))

        self.button\_next = QPushButton(">")

        self.button\_next.clicked.connect(self.next\_currency)

        v\_layout = QVBoxLayout()

        v\_layout.addWidget(self.label\_name)

        v\_layout.addWidget(self.button\_next)

        self.setLayout(v\_layout)

        self.next\_currency()

    def next\_currency(self):

        try:

            data = next(self.iterator)

            self.label\_name.setText(str(data))

        except StopIteration:

            pass

Итератор по данным

class DataIterator:

    def \_\_init\_\_(self, df: pd.DataFrame):

        self.df = df

        self.counter = 0

    def \_\_iter\_\_(self):

        return self

    def \_\_next\_\_(self):

        if self.counter < len(self.df):

            result = tuple(self.df.iloc[self.counter])

            self.counter += 1

            return result

        else:

            raise StopIteration

Форма отображение данных Картинки

Импорты

import pandas as pd

from PyQt6.QtWidgets import QLabel, QPushButton, QVBoxLayout, QWidget

from PyQt6.QtGui import QPixmap

Код формы ImageWindow

class ImageWindow(QWidget):

    def \_\_init\_\_(self, df):

        super().\_\_init\_\_()

        self.setWindowTitle('Просмотр данных')

        self.label\_image = QLabel()

        self.label\_name = QLabel()

        self.iterator = iter(ImageIterator(df))

        self.button\_next = QPushButton(">")

        self.button\_next.clicked.connect(self.next\_image)

        v\_layout = QVBoxLayout()

        v\_layout.addWidget(self.label\_image)

        v\_layout.addWidget(self.label\_name)

        v\_layout.addWidget(self.button\_next)

        self.setLayout(v\_layout)

        self.next\_image()

    def next\_image(self):

        try:

            image\_path = next(self.iterator)

            pixmap = QPixmap(image\_path)

            self.label\_image.setPixmap(pixmap.scaled(500, 500))

            self.label\_name.setText(image\_path)

        except StopIteration:

            pass

Итератор по данным

class ImageIterator:

    def \_\_init\_\_(self, df: pd.DataFrame):

        self.df = df

        self.counter = 0

    def \_\_iter\_\_(self):

        return self

    def \_\_next\_\_(self):

        if self.counter < len(self.df):

            result = self.df.iloc[self.counter, 3]

            self.counter += 1

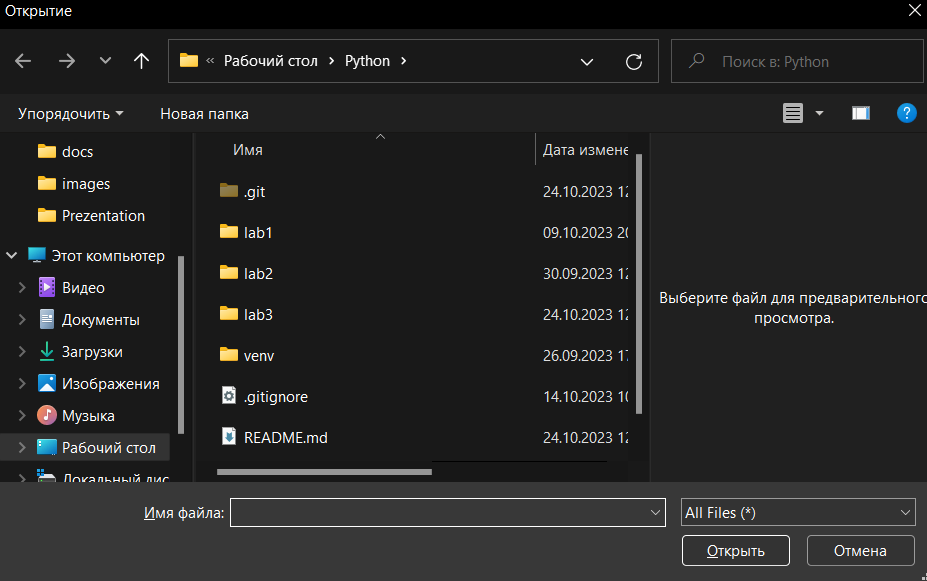
            return result

        else:

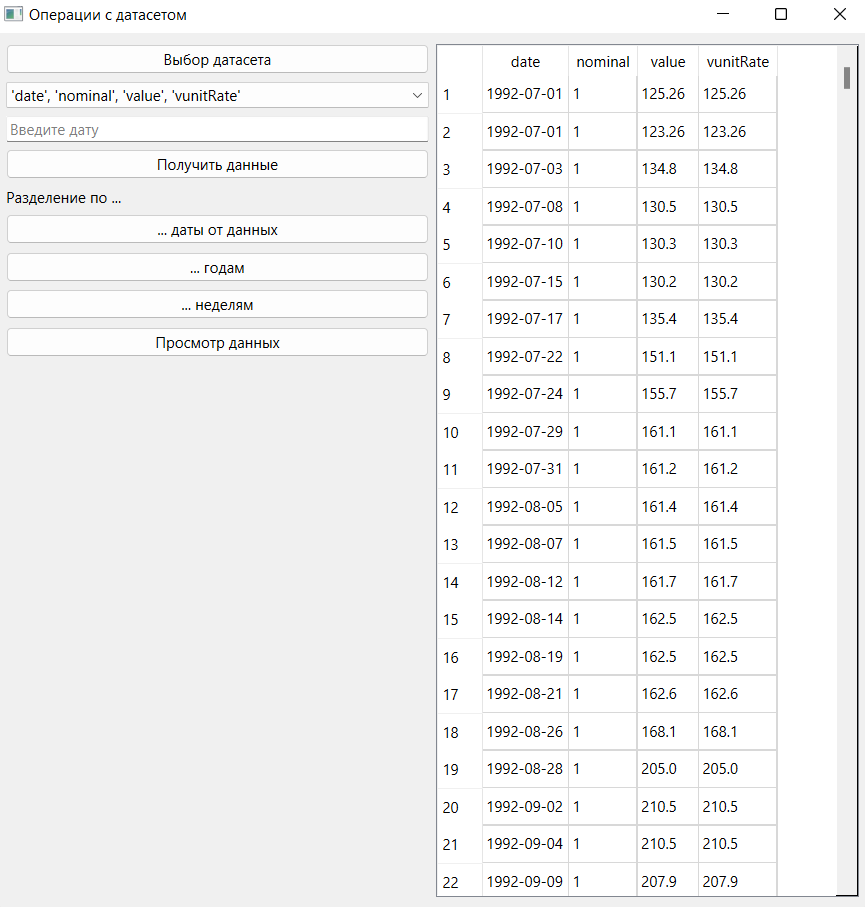
            raise StopIteration

Тестирование:

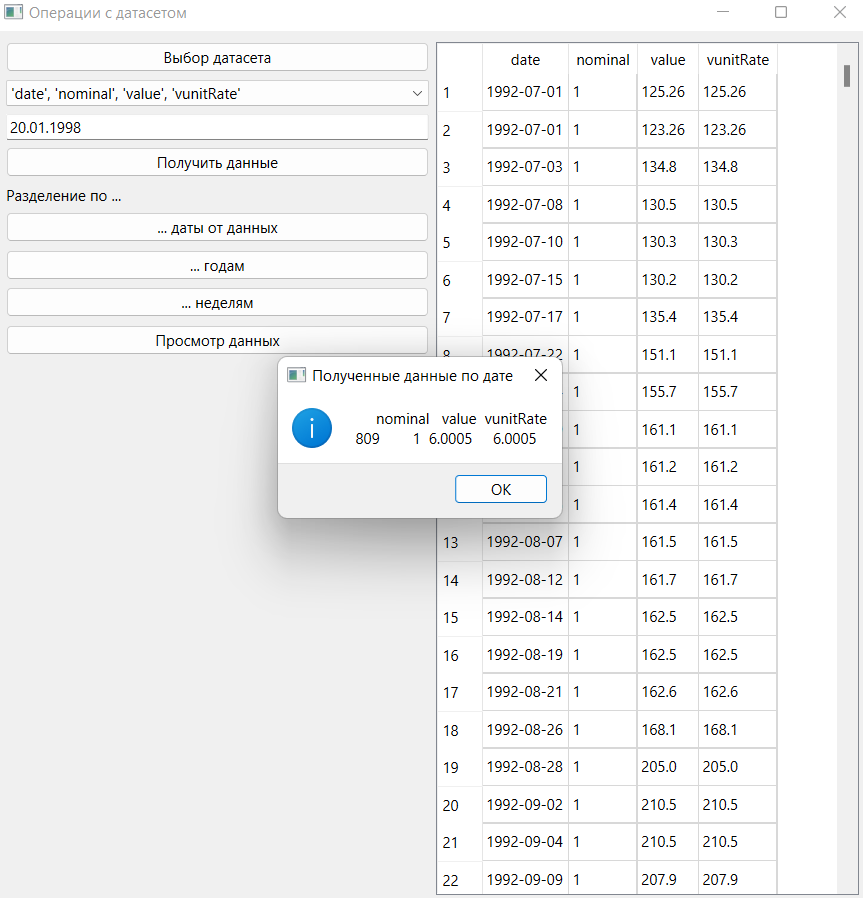
Выбор датасета



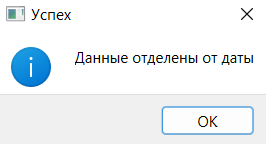
Главная форма

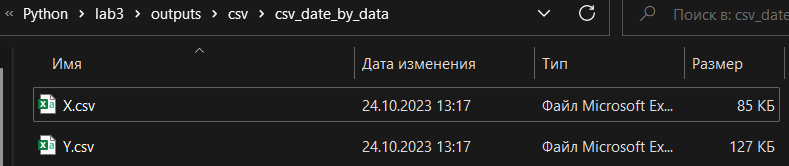


Получение данных по дате

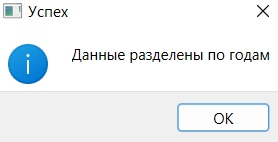


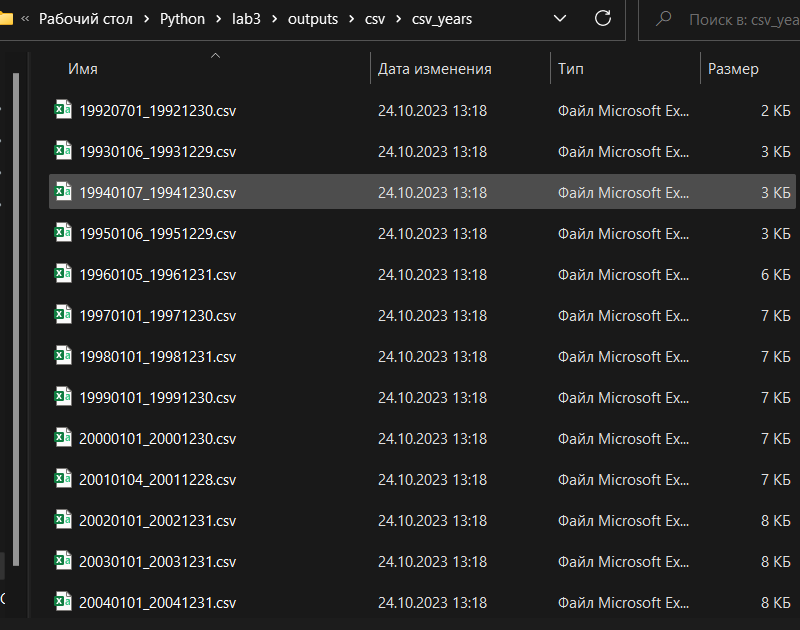
Разделение даты от данных



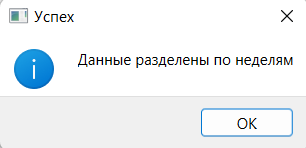


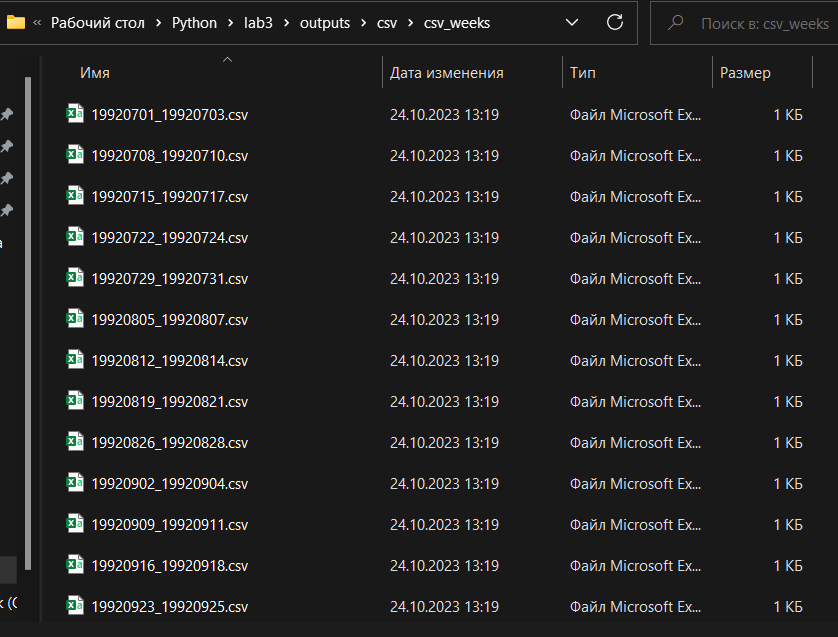
Разделение по годам



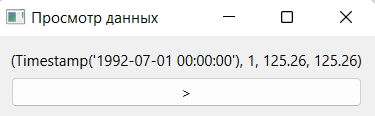


Разделение по неделям

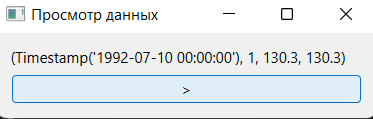




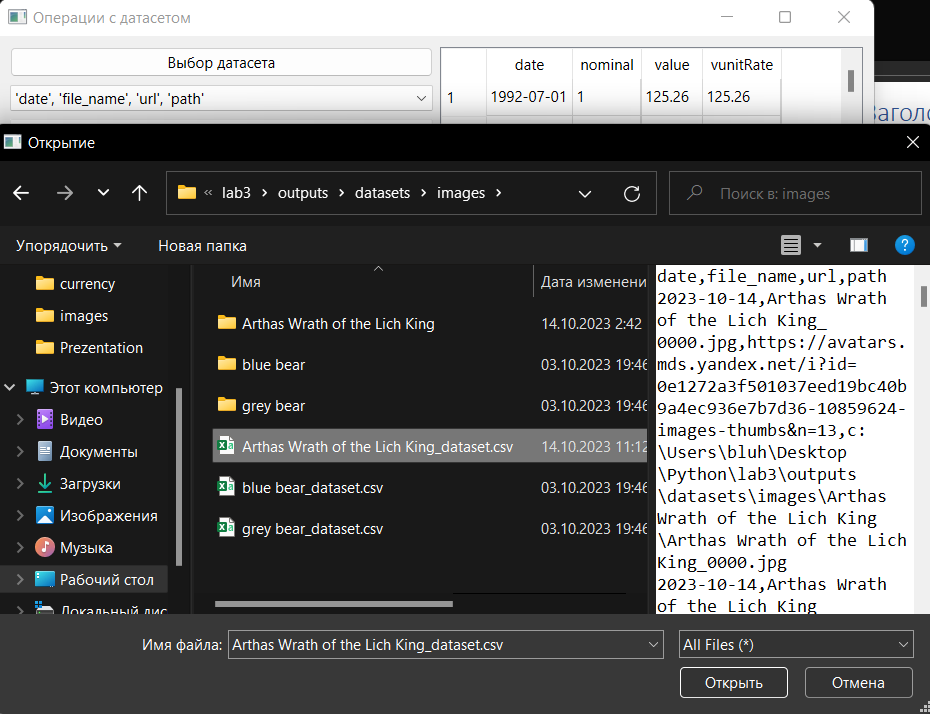
Просмотр данных валюта

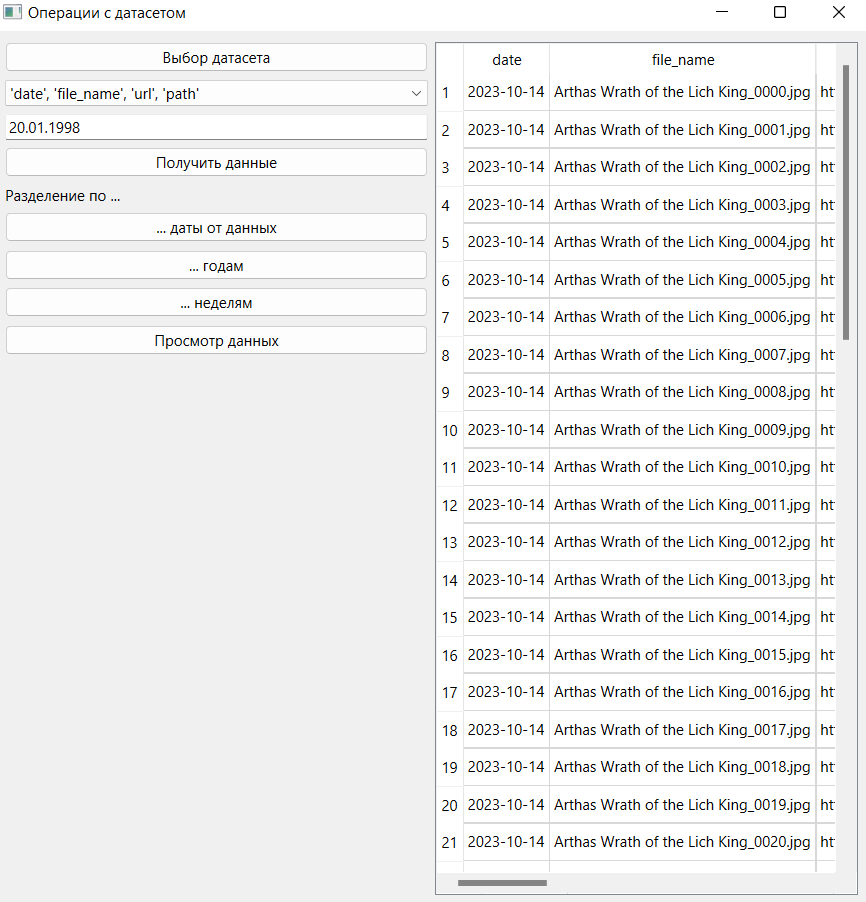


Работа итератора

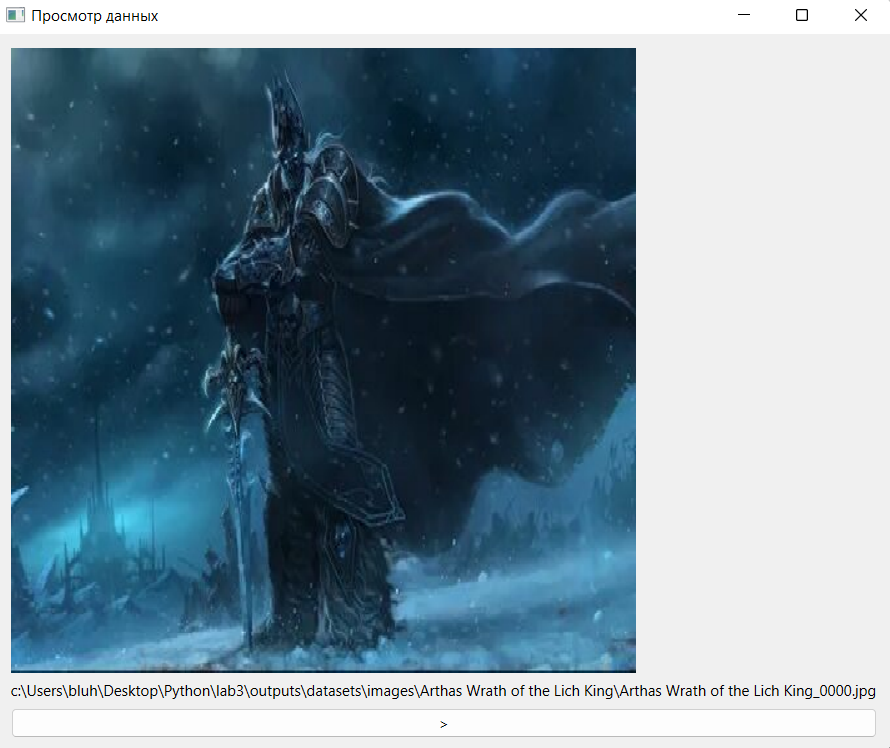


Загрузка нового датасета с картинками





Просмотр данных картинки



Работа итератора

