Макойда Максим

```
import os

from PyQt5.QtWidgets import (
   QApplication, QWidget,
   QFileDialog,
   QLabel, QPushButton, QListWidget,
   QHBoxLayout, QVBoxLayout
)
```

Для реалізації нашого проєкту і програмування зовнішнього вигляду програми скористаємося бібліотеко PyQT. Тому імпортуємо основні віджети які будуть використані в програмі: QApplication-Додаток, QWidget- Вікно додатка, QLabel – Напис, QPushButton-Кнопка, Вертикальна і горизонтальна направляюча лінія- QHBoxLayout, QVBoxLayout, QListWidget - Список з можливістю вибору елементів(клікабельний список)

Також оскільки робота програми передбачає вибір зображень з комп'ютера скористаємося новим віджетом бібліотеки PyQt5 - QFileDialog.
З його допомогою викликається вікно вибору папки (Провідник або Finder)
import os - Підключення модуля os, що **містить функції для роботи з операційною системою**

Кучак Ярослав

```
from PyQt5.QtCore import Qt # потрібна константа Qt.KeepAspectRatio для зміни розмірів із збереженням пропорцій
from PyQt5.QtGui import QPixmap # оптимізована для показу на екрані картинка

from PIL import Image
from PIL.ImageQt import ImageQt # Для перенесення графіки з Pillow до QT
from PIL import ImageFilter
from PIL.ImageFilter import (
   BLUR, CONTOUR, DETAIL, EDGE_ENHANCE, EDGE_ENHANCE_MORE,
   EMBOSS, FIND_EDGES, SMOOTH, SMOOTH_MORE, SHARPEN,
   GaussianBlur, UnsharpMask
)
```

Клас **Image**:властивості і методи роботи з картинкою як з файлом, Модуль **ImageFilter**:

Набір констант для накладення фільтрів, повороту і відображення картинки

Черчель Юрій

```
app = QApplication([])
win = QWidget()
win.resize(700, 500)
win.setWindowTitle('Easy Editor')
lb_image = QLabel("Картинка")
btn_dir = QPushButton("Папка")
lw_files = QListWidget()

btn_left = QPushButton("Вліво")
btn_right = QPushButton("Вправо")
btn_flip = QPushButton("Відзеркалити")
btn_sharp = QPushButton("Різкість")
btn_bw = QPushButton("Ч/Б")
```

Створимо інтерфейс програми: app = QApplication([])- **Конструктор**, створює об'єкт типу «Додаток»
win = QWidget()- **Конструктор**, створює об'єкт типу «Вікно». win.resize -розмір вікна програми, win.setWindowTitle -встановлює заголовок вікна, а саме 'Easy Editor'
lb_image = QLabel("Картинка") -конструктор, створює об'єкт типу напис, в подальшому тут буде завантажена картинка
btn_dir = QPushButton("Папка")- конструктор, створює об'єкт типу кнопка для вибору зображень з комп'ютера, QListWidget() - конструктор, створює об'єкт типу клікабельний список; наспупні елементи QPushButton -кнопки

Влад І Руслан Керчуни

```
row = QHBoxLayout()          # Головна лінія
col1 = QVBoxLayout()         # ділиться на два стовпці
col2 = QVBoxLayout()
col1.addWidget(btn_dir)      # в першому - кнопка вибору каталогу
col1.addWidget(lw_files)     # і список файлов
col2.addWidget(lb_image, 95) # в другому - картинка
row_tools = QHBoxLayout()    # і ряд кнопок
row_tools.addWidget(btn_left)
row_tools.addWidget(btn_right)
row_tools.addWidget(btn_flip)
row_tools.addWidget(btn_sharp)
row_tools.addWidget(btn_bw)
col2.addLayout(row_tools)

row.addLayout(col1, 20)
row.addLayout(col2, 80)
win.setLayout(row)

win.show()
```

Розташування віджетів по лінії
row = QHBoxLayout() головна горизонтальна лінія, що ділиться на 2 стовпці
# в першому - кнопка вибору каталогу і список файлів
# в другому – картинка і ряд кнопок
addWidget - Метод, додає віджет до лінії і розташовує по центру.
Команда setLayout -додати отриману лінію і її об'єкти у вікно програми

main_win.show() - Зробити вікно видимим

app.exec()- Залишати додаток відкритим, поки не буде натиснута кнопка виходу

Тут треба запустити код на виконання і показати зовнішній вигляд програми

Яснiковський Святослав

```
workdir = ''
```

Наступним ми запрограмовували кнопку **Папка**, при натисненні на яку зявлятиметься діалогове вікно і можна буде вибрати на комп'ютері папку із файлами та завантажити список імен файлів у ListWidget. Створимо глобальну змінну workdir = ''. Її значення можна отримувати та змінювати з будь-якої частини програми, тут буде зберігатися шлях до вибраної папки з вікна QFileDialog

```
def filter(files, extensions):
    result = []
    for filename in files:
        for ext in extensions:
            if filename.endswith(ext):
                result.append(filename)
    return result
```

Виберемо файли лише з графічними розширеннями.

- Усі допустимі розширення зручно розмістити у списку **extensions**. Для цього створимо функцію def filter у якій необхідно Створити пустий список result для імен файлів.
- *Для кожного імені файлу зі списку filenames*:
  - *І кожного розширення зі списку extensions*:
    якщо ім'я закінчується на це розширення,
    то додати його до списку результатів.
- Повернути список result.

Метод **endswith(ext)** Поверне **True**, якщо ім'я файлу закінчується на ext і **False**, якщо ні.

Марцинюк Михайло

```
def chooseWorkdir():
    global workdir
    workdir = QFileDialog.getExistingDirectory()
```

```
def showFilenamesList():
    extensions = ['.jpg','.jpeg', '.png', '.gif', '.bmp']
    chooseWorkdir()
    filenames = filter(os.listdir(workdir), extensions)

    lw_files.clear()
    for filename in filenames:
        lw_files.addItem(filename)

btn_dir.clicked.connect(showFilenamesList)
```

Запрограмуємо відображення списку імен у віджеті в функції def showFilenamesList():
Вибираємо робочу папку (workdir).
Задаємо список допустимих розширень.
Завантажуємо імена файлів папки та залишаємо лише з розширеннями extensions.
Очищаємо віджет-список (у випадку, якщо там залишилися імена файлів іншої папки).
По одному додаємо у віджет відібрані імена файлів.
clicked.connect – обробка події натиснення на кнопку

Зауваження:

Для роботи знадобиться не лише список файлів (є у віджеті), а й робоча папка.

Наразі ім'я папки видаляється після завершення роботи showFilenamesList()

Щоб зберегти папку, визначимо її як ***глобальну змінну***.

Тут треба запустити код на виконання, натиснути на кнопку Папка, вибрати папку з графічними файлами та показати, що їх назви відображаються в списку

Булавков Олександр

```
class ImageProcessor():
    def __init__(self):
        self.image = None
        self.dir = None
        self.filename = None
        self.save_dir = "Modified/"
```

Наступне завдання запрограмувати попередній перегляд картинки, тобто щоб при натисненні на картинку у списку назв файлів, вона відображалася у вікні програми.

- Це завдання складніше ніж може здатися , тому що Зображення має адаптуватися до розміру вікна програми.
- При перемиканні між картинками прев'ю має змінюватися.
  Під час обробки має з'являтися прев'ю обробленої копії.
- Для цього запрограмуємо клас ImageProcessor() в конструкторі якого запишемо поточне **зображення** (за замовчуванням None);
- поточне **ім'я файлу** (за замовчуванням None);

```
    def loadImage(self, filename):
        ''' під час завантаження запам'ятовуємо шлях та
        self.filename = filename
        fullname = os.path.join(workdir, filename)
        self.image = Image.open(fullname)
```

Метод loadImage() — завантаження зображення
Збережемо filename у властивості filename екземпляра класу ImageProcessor
Зі шляху до робочої папки та імені файлу **сформуємо шлях до картинки**
Відкриємо картинку (об'єкт Image), звернувшись за повним шляхом
fullname = os.path.join(workdir, filename) - Отримання повного шляху до файлу об'єднанням шляху до папки та імені файлу

Строкош Євген

```
    def showImage(self, path):
        lb_image.hide()
        pixmapimage = QPixmap(path)
        w, h = lb_image.width(), lb_image.height()
        pixmapimage = pixmapimage.scaled(w, h, Qt.KeepAspectRatio)
        lb_image.setPixmap(pixmapimage)
        lb_image.show()
```

Метод showImage() — показати зображення
Приховаємо віджет на час «технічних робіт».
Повним шляхом до файлу створюємо об'єкт QPixmap спеціально для відображення графіки в інтерфейсі.
Повним шляхом до файлу створюємо об'єкт QPixmap спеціально для відображення графіки в інтерфейсі.
Адаптуємо картинку під розміри поля.
Відображаємо віджет!

```
    def showChosenImage():
        if lw_files.currentRow() >= 0:
        filename = lw_files.currentItem().text()
        workimage.loadImage(filename)
        workimage.showImage(os.path.join(workdir, workimage.filename))
```

Щоб застосувати написаний функціонал у програмі, створимо функцію-обробник **showChosenImage** натискання по елементу списку імен картинок.

Якщо вибрано елемент списку lw_files:

filename = вибраний рядок віджету-списку

Завантажуємо картинку методом loadImage()
Формуємо шлях для відображення картинки
Відображаємо картинку методом showImage()
Далі команда **`lw_files.currentRowChanged.connect(showChosenImage)` підключення функції до віджета `QListWidget`**

Тут треба клацнути по картинках і показати що вони відображаються у вікні програми

Сверида Тимур

```
def saveImage(self):
    ''' зберігає копію файлу у підпапці '''
    path = os.path.join(workdir, self.save_dir)
    if not(os.path.exists(path) or os.path.isdir(path)):
        os.mkdir(path)
    fullname = os.path.join(path, self.filename)

    self.image.save(fullname)
```

Метод saveImage() — зберегти картинку
Формуємо шлях до папки для збереження
Якщо цей шлях **не існує**?
(Папки для збереження ще немає?) За сформованим шляхом **створити нову папку**
Формуємо шлях до картинки, що зберігається (з ім'ям файлу картинки!)
Зберігаємо картинку вбудованим методом класу Image

Кучак Ярослав

Переходимо до довгоочікуваної обробки зображень!
Тут розповісти про імпорт бібліотеки Pillow, код на початку

```
def do_bw(self):
    self.image = self.image.convert("L")
    self.saveImage()
    image_path = os.path.join(workdir, self.save_dir, self.filename)
    self.showImage(image_path)
```

Функція опрацювання події натиснення на кнопку ч/б
- `convert("L")`- перетворює зображення вбудованим методом у відтінки сірого (чорно-біле)
- **Зберегти** змінений об'єкт Image як файл методом save()
- **Формуємо** шлях для відображення зміненої картинки
- **Відображаємо** змінену картинку методом showImage()
+ сказати за
`btn_bw.clicked.connect(workimage.do_bw)`

Левицький Дмитро

```
def do_left(self):
    self.image = self.image.transpose(Image.ROTATE_90)
    self.saveImage()
    image_path = os.path.join(workdir, self.save_dir, self.filename)
    self.showImage(image_path)

def do_right(self):
    self.image = self.image.transpose(Image.ROTATE_270)
    self.saveImage()
    image_path = os.path.join(workdir, self.save_dir, self.filename)
    self.showImage(image_path)
```

Функції опрацювання події натиснення на кнопки вліво-вправо
Обробляємо поточну картинку вбудованим методом (ROTATE_90) – праворуч; (ROTATE_270) – лівоворуч
Зберігаємо змінений об'єкт Image як файл методом *saveImage*()
Формуємо шлях для відображення зміненої картинки
Відобразимо змінену картинку методом *showImage*()
+ сказати за
`btn_left.clicked.connect(workimage.do_left)`
`btn_right.clicked.connect(workimage.do_right)`

Корнієнко Олександр

```
def do_flip(self):
    self.image = self.image.transpose(Image.FLIP_LEFT_RIGHT)
    self.saveImage()
    image_path = os.path.join(workdir, self.save_dir, self.filename)
    self.showImage(image_path)

def do_sharpen(self):
    self.image = self.image.filter(SHARPEN)
    self.saveImage()
```

Функції опрацювання події натиснення на кнопки дзеркало і різкість
Обробляємо поточну картинку вбудованим методом `transpose(Image.FLIP_LEFT_RIGHT)`- дзеркальневідображення
`filter(SHARPEN)`-для різкості
Зберігаємо змінений об'єкт Image як файл методом *saveImage*()
Формуємо шлях для відображення зміненої картинки
Відобразимо змінену картинку методом *showImage*()
+ сказати за

```
        image_path = os.path.join(workdir, self.save_dir, self.filename)
        self.showImage(image_path)




workimage = ImageProcessor() #поточне робоче зображення для роботи
lw_files.currentRowChanged.connect(showChosenImage)

btn_bw.clicked.connect(workimage.do_bw)
btn_left.clicked.connect(workimage.do_left)
btn_right.clicked.connect(workimage.do_right)
btn_sharp.clicked.connect(workimage.do_sharpen)
btn_flip.clicked.connect(workimage.do_flip)

app.exec()
```