algoritmics

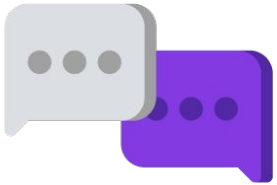
Module 4. Lesson 3.

The Easy Editor app Part 2

Link to the methodological guidelines

Discussion:

Project planning

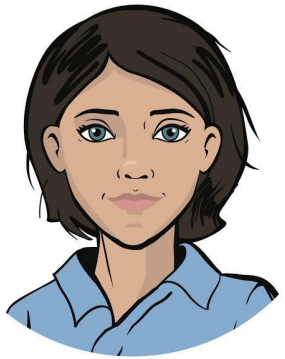


Still working on the order!

Last time, a representative of the Ministry for Social Development turned to ProTeam specialists.
He is making a software package for the elderly people.

One of the apps should be an **Easy Editor photo editor**.

Highlight today's work tasks!



Emily,
Project Manager

Discussion of tasks

General technical exercise

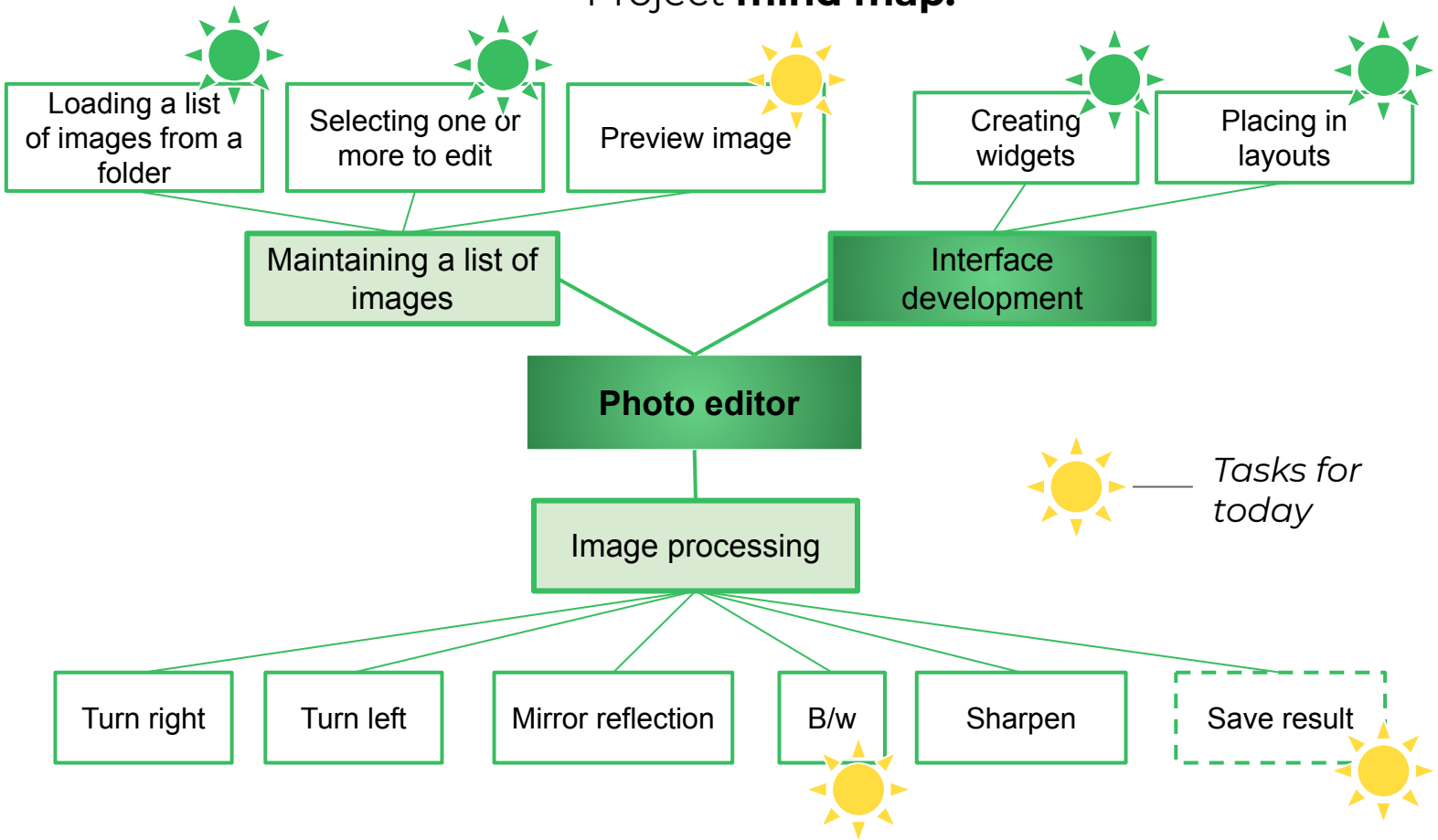
The **goal** is to program the Easy Editor app.

Expected app view:

Discussion of tasks

Planning work on the project

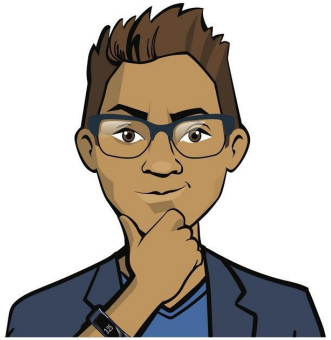
Project **mind map:**

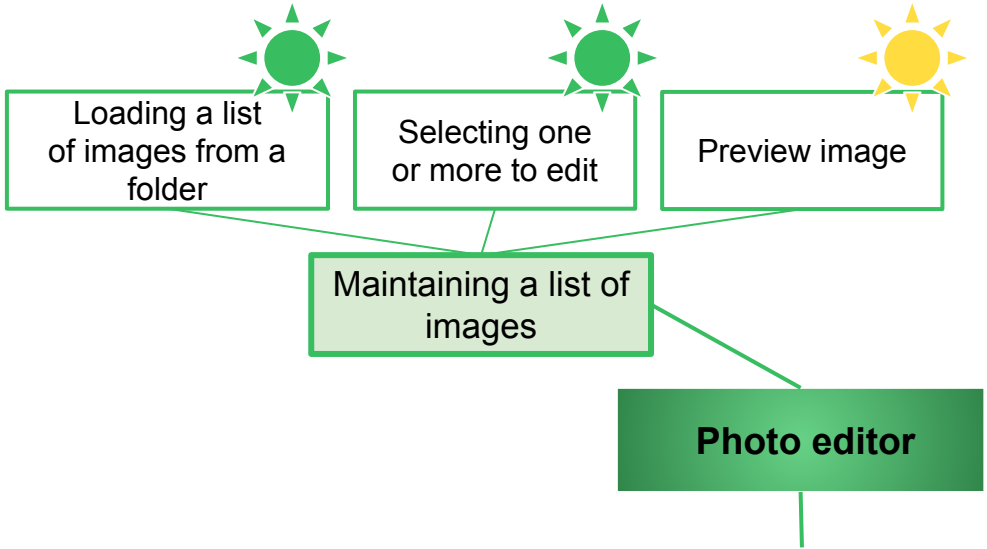


Discussion of tasks

Previewing an image is a more complex task, than you might think.

- The image must adapt to the size of the application window.
- *When you switch* between images, the preview should change.
- *A preview* of the processed copy should appear during processing.





Discussion of tasks

Planning work on the project

Checklist based on the **mind map**:

✅ 1. *Create an interface for* the app.

✅ 2. *Ensure loading* images from the required folder.

Today

3. *Show a preview* of the image selected in the list.

4. *Program editing* of a photo:

- creating a modified copy;
- showing a preview of the modified copy;
- saving to the Modified subfolder.

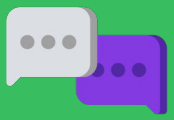
Discussion of tasks

The goal of the working day is

will program the image previews and start processing them.

Today you:

- Remember working with objects like Image and the os module.
- Find out how to solve the complex task of fitting the image into the app window.
- Program a preview of the images and process the first image.



Discussion of tasks

Qualifications

Demonstrate knowledge

of working with images and computer files.

Qualifications

What formats of graphic files do you know?

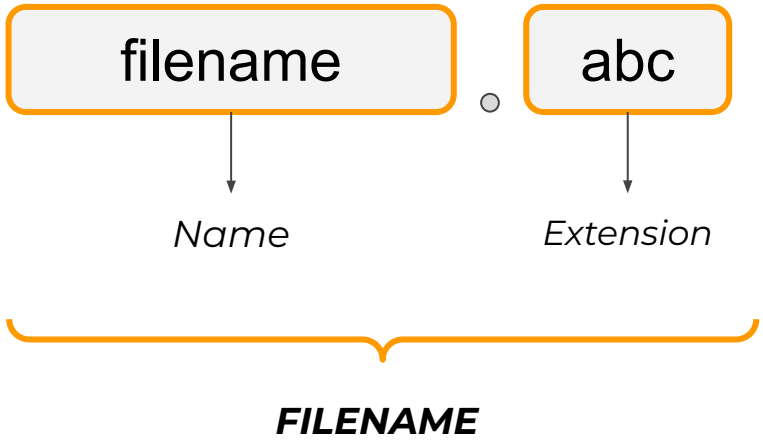
How does the file format differ from the extension ?

Qualifications

Graphic file formats:

- JPG-file
- PNG-file
- BMP-file
- SVG-file
- EPS-file

etc.



Qualifications

What is the path to a folder ?

And a path to a file ?

What will be the value of the workdir variable after the programme has run?

```
workdir = ''
def chooseWorkdir():
    global workdir
    workdir = QFileDialog.getExistingDirectory()
btn_dir.clicked.connect(chooseWorkdir)
```

Qualifications

A folder path

– is a sequence of folder (directory) names and additional characters specifying the path to the folder.

A file path

– is a sequence of folder names, symbols, and the name of the file you are looking for, giving the route to the file.

C:\User\Sasha\School\IT – “IT” folder path

C:\User\Sasha\School\IT\project.py — file path **project.py** in the “IT” file

Qualifications

A folder path

– is a sequence of folder (directory) names and additional characters specifying the path to the folder.

A file path

– is a sequence of folder names, symbols, and the name of the file you are looking for, giving the route to the file.

```
workdir = ''
def chooseWorkdir():
    global workdir
    workdir = QFileDialog.getExistingDirectory()
btn_dir.clicked.connect(chooseWorkdir)
```

The value of workdir is the path to the folder selected by the user.

Qualifications

Why do I need the os module?

Which function from os do you know?

Qualifications

os module

is located in the Python standard library and contains functions for working with the operating system.

Command	*Purpose*
`import os`	Connecting the os module
`files = os.listdir(<file>)`	Retrieve a list of file names from a specified folder

Qualifications

The PIL library includes two modules: Image and ImageFilter .

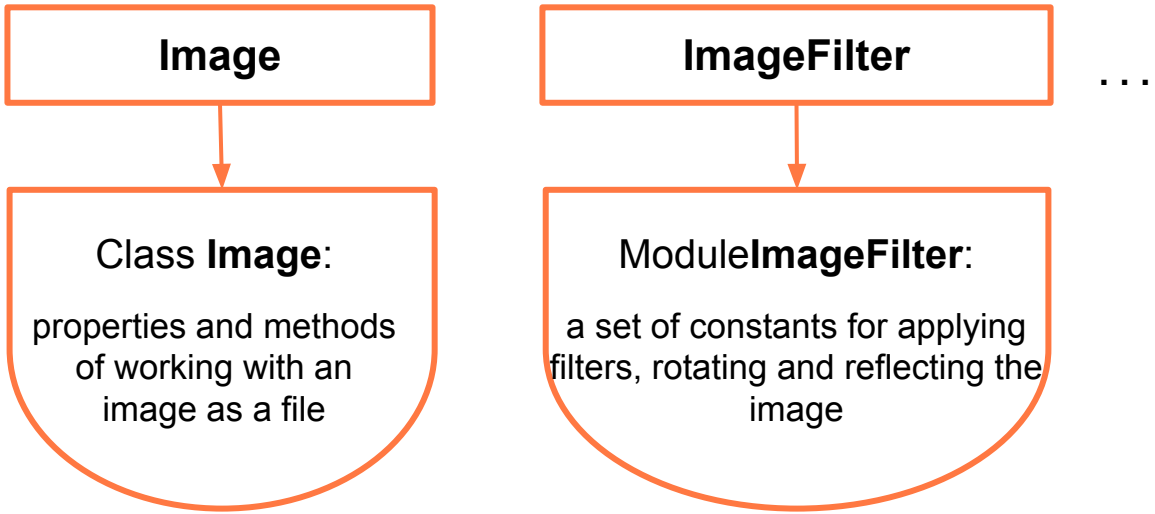
What purpose is each of them intended for?

Qualifications

Python Imaging Library (PIL) —
is a library for working with bitmap graphics

The PIL library has a hierarchical structure.

Two modules from the base structure will be useful: **Image** and **ImageFilter.**



Qualifications

How do we open an image and create an Image type object?

How do we save an image?

Qualifications

Open and save the image

Command	*Purpose*
`from PIL import Image`	From the PIL library, connect the Image module
`cur_image = Image.open('photo.jpg')`	Open an image file from the project folder
`cur_image = Image.open(<full path>)`	Open an image located according to the given path
`cur_image.save('new_photo.jpg')`	Save the image in the project folder
`cur_image.save(<full path>)`	Save the image to a random location on your computer

Qualifications

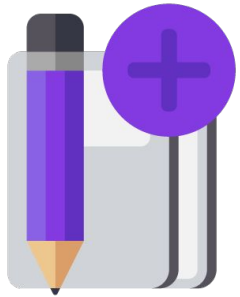
Qualifications confirmed!

Great, you are ready to brainstorming and complete your work task!

Qualifications

Brainstorm:

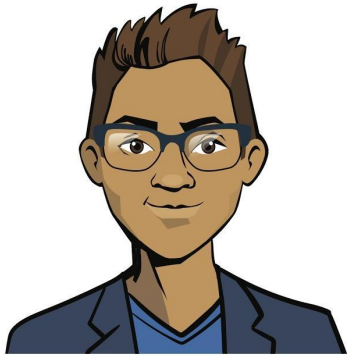
The ImageProcessor class



Working tasks

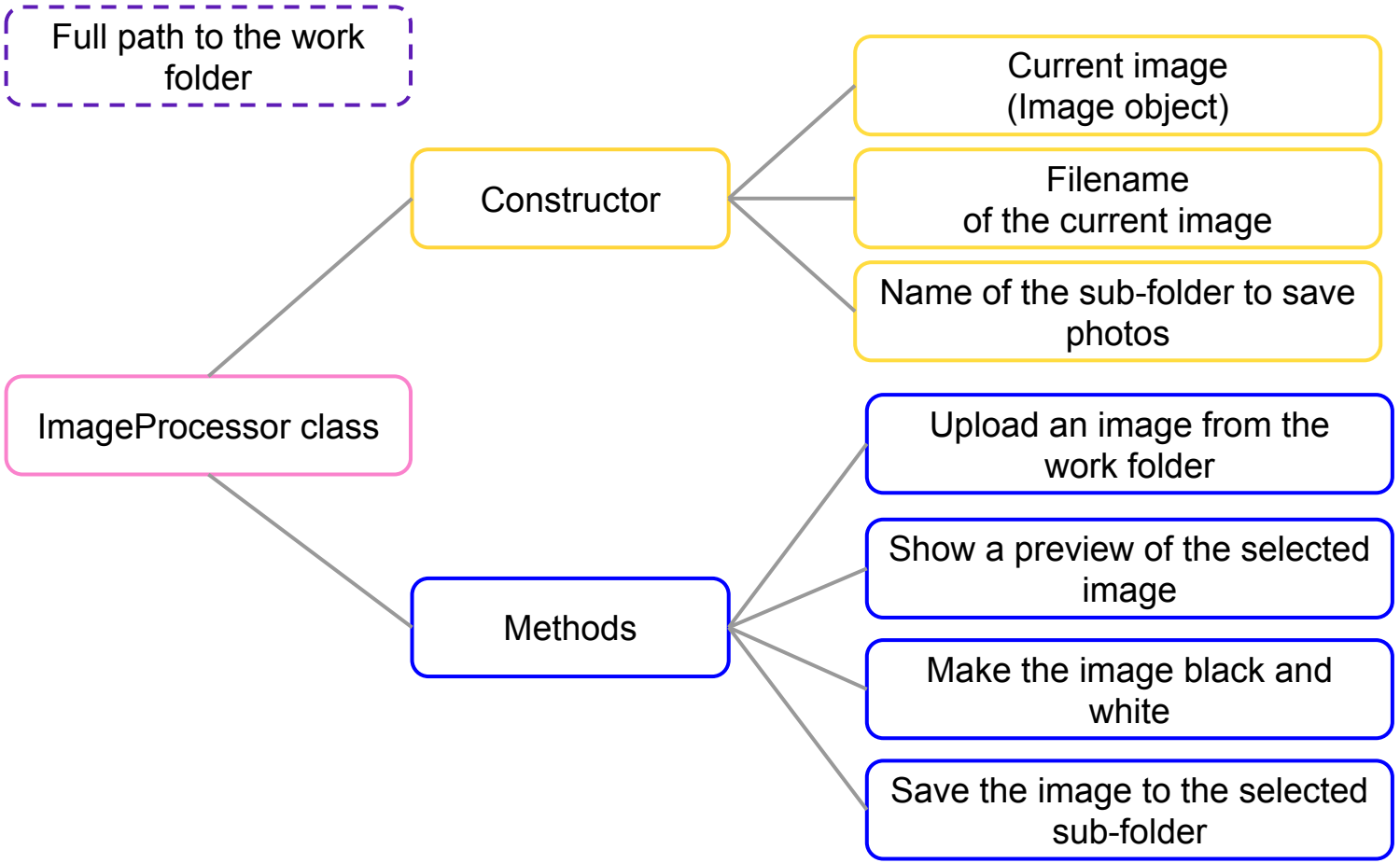
Let’s program the **ImageProcessor** class.
With its help, we will be able to:

— upload photos from a random folder;
— display a preview of the photo (both original and processed);
— process photos;
— save the result to the subfolder Modified in the selected folder.



Brainstorming

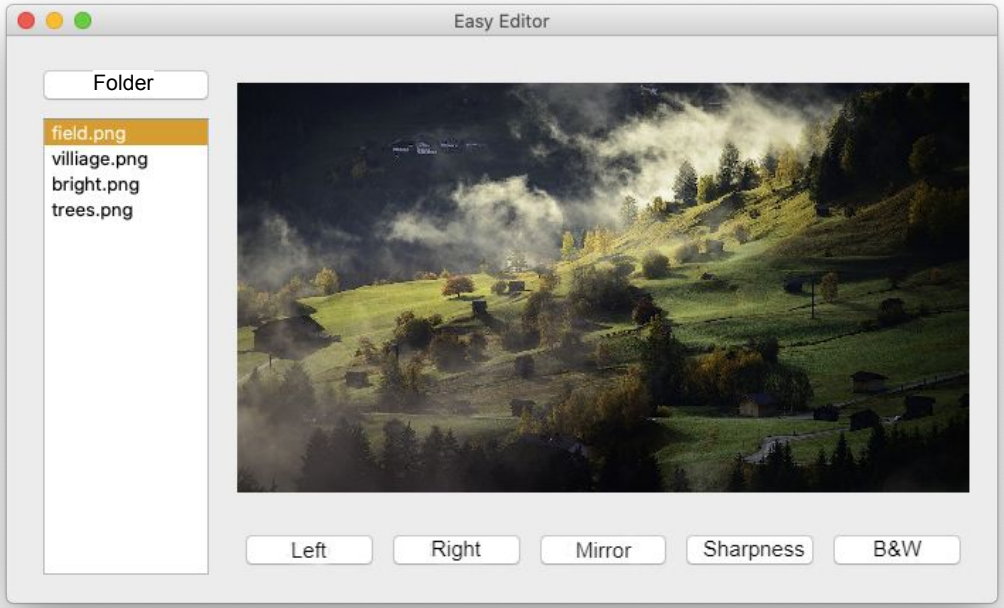
Mind map of the class:



Brainstorming

Expected result

After implementing the methods for loading and displaying images.



Brainstorming

The ImageProcessor class: current tasks

```
class ImageProcessor():
```

constructor

- current **images** (defaults to None);
- current **filename** (defaults to None);
- **sub-folder name** to save the modified images;

methods

- **Load an image** from a folder from the selected name in the image list.
- **display the current image in** the app window

A sample can be created immediately after displaying the folder file names.

Learning new functions from the os to address the image by full path is required.

Introductory lesson from the senior developer Cole.

Brainstorming

1. The loadImage() method – loading an image

How do we load an image from a work folder if we know the file name?

```
def loadImage(self, filename):
```

Save filename in the filename properties of the ImageProcessor class sample

↓

From the path to the work folder and the file name, **form the path to the image** → Use the new function from the os module.

↓

Open an image (Image object) following the full path

↓

Save the resulting Image object in the image properties of the ImageProcessor sample

Brainstorming

The os module: the join() function

Command	*Purpose*
`import os`	Connecting the os module
`file_path = os.path.join(workdir, filename)`	Obtaining the full path to a file by combining the path to folder and the file name

Path to the work folder

C:\User\Granny\Vacation2020

Name of the desired file

puppies.jpg

Result of work join()

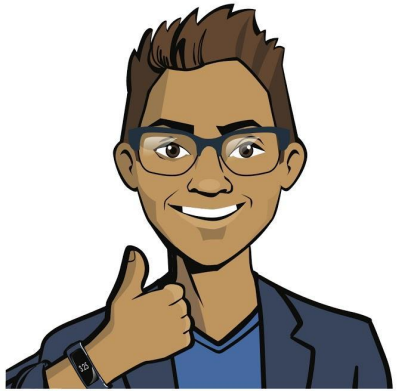
C:\User\Granny\Vacation2020\puppies.jpg

Brainstorming

1. The loadImage() method – loading an image

How do we load an image from a working folder if we know the file name?

```
def loadImage(self, filename):
    self.filename = filename
    image_path = os.path.join(workdir, filename)
    self.image = Image.open(image_path)
```



Brainstorming

2. The showImage() method — show image

Showing an image in a window is not as easy as it seems.
Our senior developer Cole will demonstrate this difficult trick from beginning to end.

Brainstorming

Note.
The image to be displayed is loaded by accessing the file via its full path.

The full path may vary depending on whether we are showing the original or a processed image.

2. The showImage() method — show image

Showing an image in a window is not as easy as it seems.
Our senior developer Cole will demonstrate this difficult trick from beginning to end.

```
def showImage(self, path):
    lb_image.hide()
    lb_image.show()
```

Hide the widget for the duration of the “technical work”.

Display the modified widget.

Brainstorming

2. The showImage() method — show image

Showing an image in a window is not as easy as it seems.
Our senior developer Cole will demonstrate this difficult trick from beginning to end.

```
def showImage(self, path):
    lb_image.hide()
    pixmapimage = QPixmap(path)


    lb_image.show()
```

Using the full path of the file, create a QPixmap object specifically for displaying graphics in the UI.

Brainstorming

2. The showImage() method — show image

Showing an image in a window is not as easy as it seems.
Our senior developer Cole will demonstrate this difficult trick from beginning to end.

```
def showImage(self, path):
    lb_image.hide()
    pixmapimage = QPixmap(path)
    w, h = lb_image.width(), lb_image.height()


    lb_image.show()
```

Find out the dimensions of the field for placing the image.

Brainstorming

2. The showImage() method — show image

Showing an image in a window is not as easy as it sounds.
This sophisticated trick will be demonstrated in its entirety by our developer Kostya.

```
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.show()
```

Adapting the image according to the dimensions of the field.

Brainstorming

2. The showImage() method — show image

Showing an Image object in a window is not as easy as it sounds.
This sophisticated trick will be demonstrated in its entirety by our developer Kostya.

```
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()
```

Posting the image
in the lb_image widget.

Brainstorming

2. The showImage() method — show image

Showing an Image object in a window is not as easy as it sounds.
This sophisticated trick will be demonstrated in its entirety by our developer Kostya.

```
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()
```

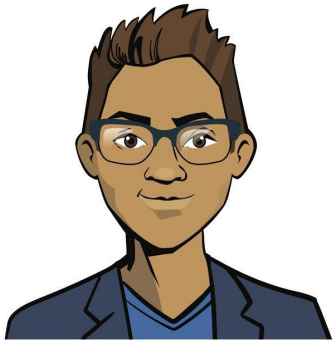
All done!
Display the widget!

Brainstorming

3. The showChosenImage() function

To apply the written functionality to the program, create a click handler function according to the element of the list of image names.

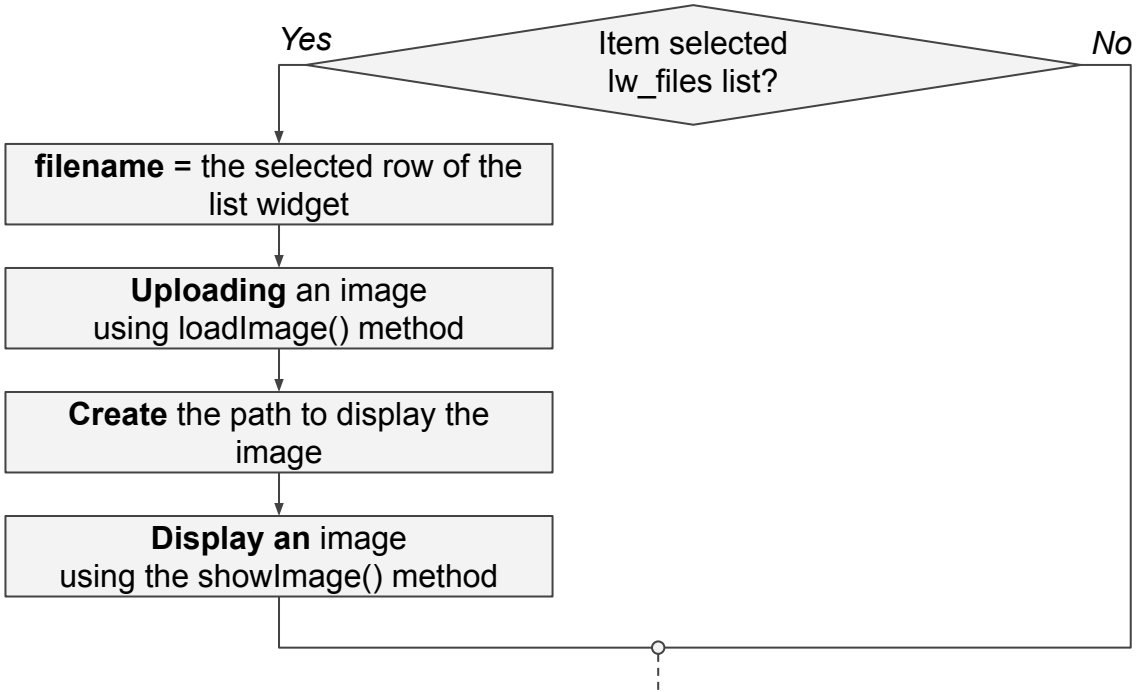
The methods for working with the list widget are already familiar to you from the Smart Notes project.



Brainstorming

3. The showChosenImage() function

To apply the written functionality to the program, create a click handler function according to the element of the list of image names.



Brainstorming

3. The showChosenImage() function

To apply the written functionality to the program, create a click handler function according to the element of the list of image names.

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


lw_files.currentRowChanged.connect(showChosenImage)
```

The modified images will be in a different folder, so you will need a different path in order to display them.

Brainstorming

Implementing the solution in the project:

```
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QPixmap
```

The described interface elements

Implemented in the previous session.

```
workdir = ''
```

Reading and displaying file names

```
ImageProcessorclass ():
```

Class description

Description of the ImageProcessor class and creating a class sample.

```
workimage = ImageProcessor()
```

```
def showChosenImage():
```

Function body

```
lw_files.currentRowChanged.connect(showChosenImage)
```

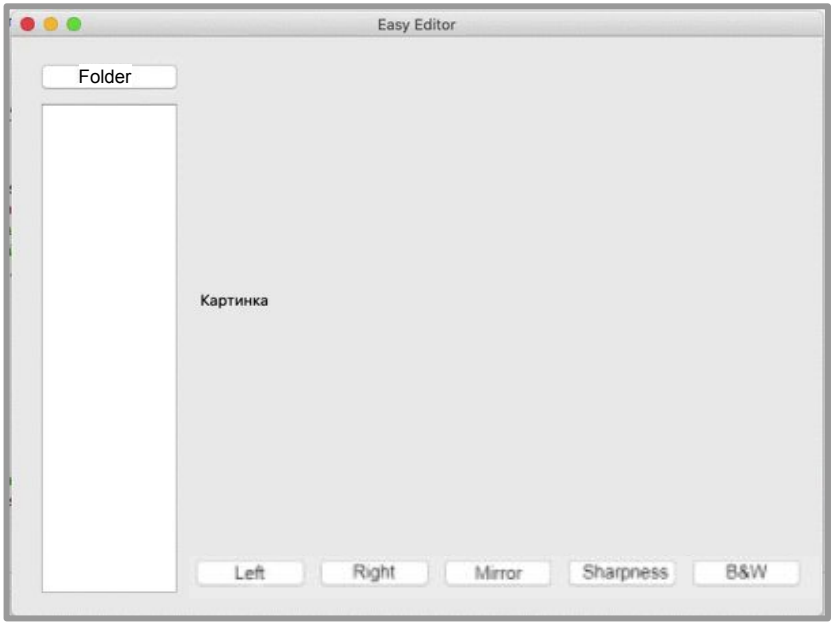
Apply ImageProcessor methods to display a preview image.

Brainstorming

Your task is:

To program displaying the image preview .

Write a part of the ImageProcessor class. Use class methods to display the image preview with a handler function.



Kostya,
Senior Developer

Brainstorming

The vs code: Easy Editor app

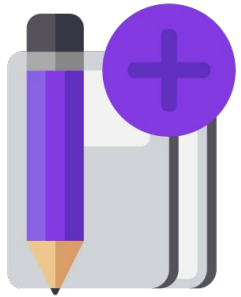
Complete task 3 in VS Code

The Easy Editor app

Break

Brainstorming:

The ImageProcessor class



Moving on to the long-awaited image processing!

By the way, you have already done a similar task in the training program with the ImageEditor class!

Let’s examine the processing using a black-and-white filter as an example.

Brainstorming

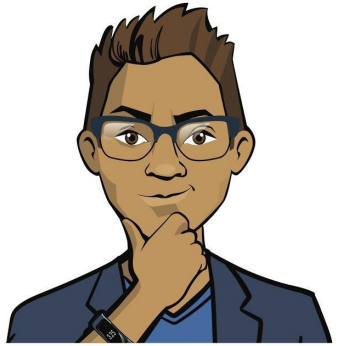
Expected result

If you press “B&W”, the image becomes black and white.
The processed copy is saved in the Modified subfolder of the work folder.

Brainstorming

The ImageProcessor class should to be supplemented with the following methods:

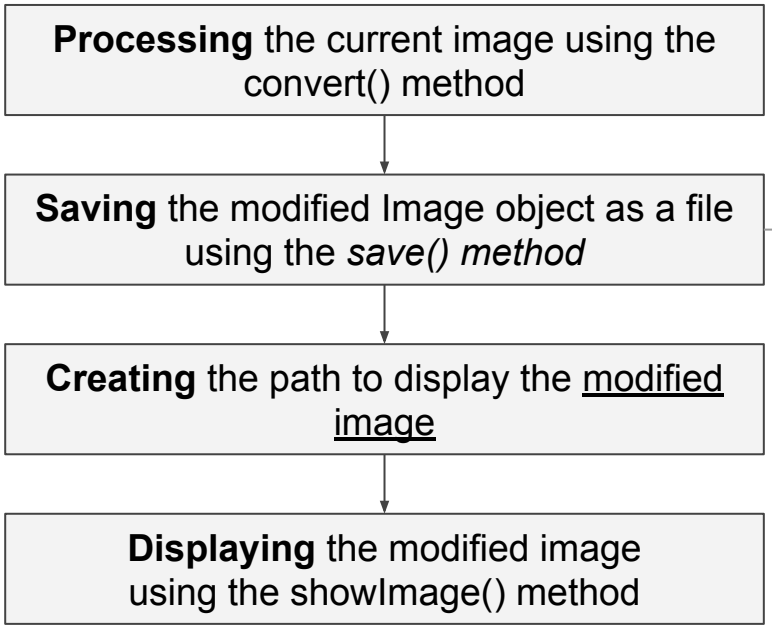
- ➔ *do_bw()* — a method that applies a black and white filter to the image;
- ➔ *saveImage()* — a method that saves the modified Image to the Modified subfolder.



Brainstorming

1. The do_bw() method – to make b/w

```
def do_bw(self):
```

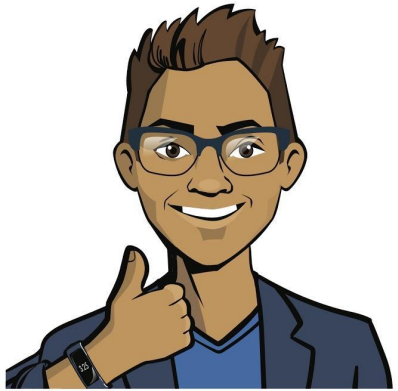


Programming at the same stage of the working day

Brainstorming

1. The do_bw() method – to make b/w

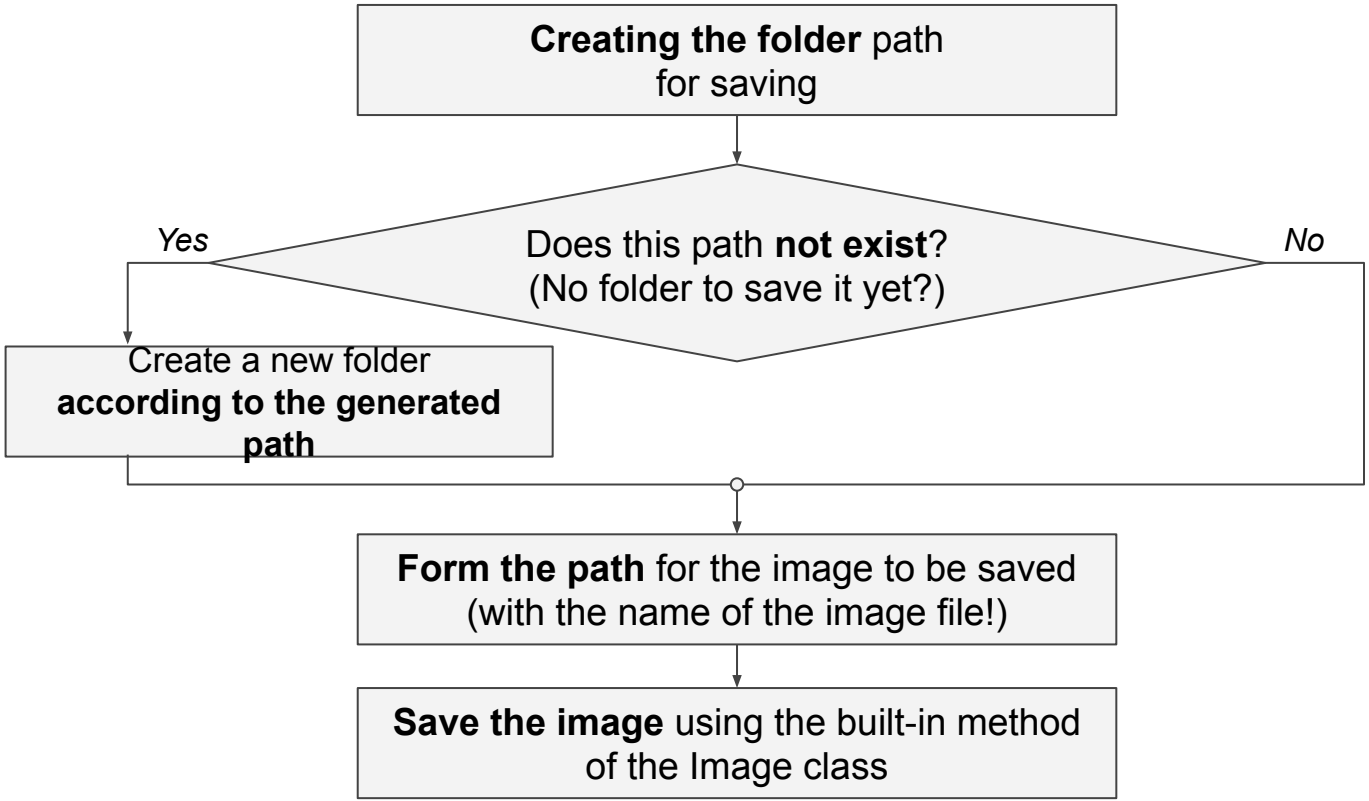
```
def do_bw(self):
    self.image = self.image.convert("L")
    self.saveImage()
    image_path = os.path.join(workdir, self.save_dir, self.filename)
    self.showImage(image_path)
```



Brainstorming

2. The saveImage() method — to save image

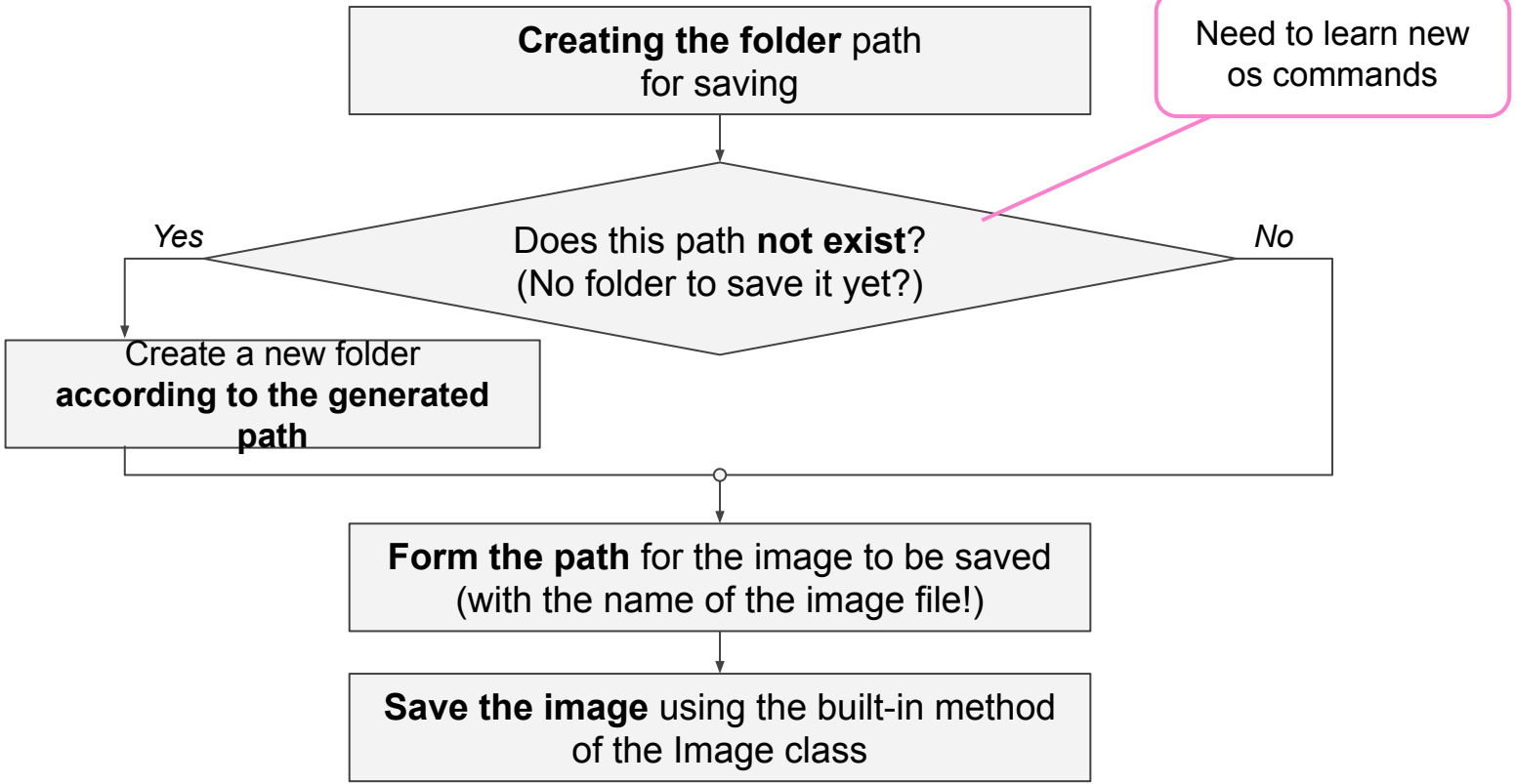
```
def saveImage(self):
```



Brainstorming

2. The saveImage() method — to save image

```
def saveImage(self):
```



Brainstorming

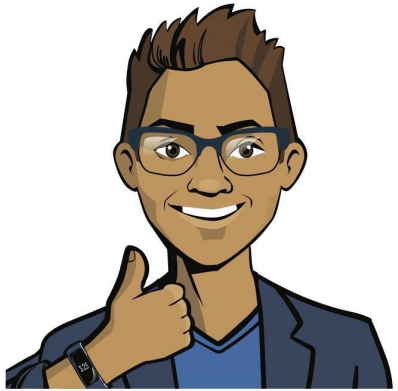
The os module – new commands

Command	*Purpose*
`import os`	Connect the os module
`os.mkdir(path)`	Create a new folder according to the specified path (the name of the folder to be created is part of the path!)
`os.path.exists(path)` `os.path.isdir(path)`	Check if something in this path already exists (e.g. a folder)

Brainstorming

2. The saveImage() method — to save image

```
def saveImage(self):
    '''saves a copy of the file in a sub-folder'''
    path = os.path.join(workdir, self.save_dir)
    if not(os.path.exists(path) or os.path.isdir(path)):
        os.mkdir(path)
    image_path = os.path.join(path, self.filename)
    self.image.save(image_path)
```



Brainstorming

Implementing the solution in the project:

The described interface elements

Reading and displaying file names

```
ImageProcessor class ():
```

Class description

SaveImage() method

Method do_bw()

We add new methods for processing and saving the image.

```
workimage = ImageProcessor()
```

```
def showChosenImage():
```

Function body

```
lw_files.currentRowChanged.connect(showChosenImage)
btn_bw.clicked.connect(workimage.do_bw)
```

Handle pressing “B&W” with do_bw().

Brainstorming

Task:

Supplement the ImageProcessor class with do_bw() and saveImage() methods.

Handle clicking the “B&W” button with them.

Brainstorming

VS Code:

The Easy Editor app

Complete task 4 in VS Code

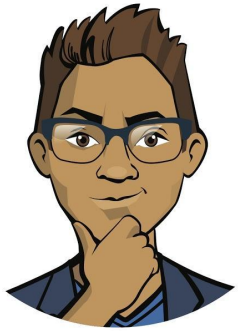
The Easy Editor app

Wrapping up the workday

To complete, pass a technical interview:

1. What is the purpose of the ImageProcessor class? Which methods can be used to supplement it next time?

1. What does the path to a file consist of? What new features of the os module have you learned?



Cole,
Senior Developer

Emily,
Project Manager

Wrapping up the workday

Great job!

Dear colleagues!

You’ve done a great job today.

During the next working day, we will finalize the Easy Editor application and program the remaining filters!

Wrapping up the workday