

Methodological guidelines.

The Easy Editor app P. 1

STORYLINE:

A representative from the Ministry of Social Development is preparing a special software package for elderly people and asked the specialists at ProTeam for help. It should include simple and useful apps for users with poor computer skills. One of the apps should be an Easy Editor photo editor.

To implement the Easy Editor app, developers program a comprehensive solution using the PyQt5, os, PIL libraries and modules, as well as their own classes.

SUMMARY:

The lesson **goal** is to program the app interface to be able to display a list of graphic files.

In the first half of the lesson, students program the Easy Editor interface. In the second half, they supplement the class with methods for loading and displaying a list of images to be processed.

Technical note. All work on the Easy Editor is organized in one task in the VS Code.









LINKS AND ACCESSORIES:

- ☐ [Presentation](#),
- ☐ Lesson tasks: [the Easy Editor app](#) (Visual Studio Code).

LEARNING RESULTS

| <i>After the lesson, students will:</i> | <i>The result is achieved when students:</i> |
|---|---|
| <ul style="list-style-type: none"> • program the app interface based on the technical specification; • understand the difference between local and global variables; • know and use the os module command to call the file explorer window to select the working directory; • program functions to work with files and interface elements (e.g., for selecting graphic files and displaying a list of filenames). | <ul style="list-style-type: none"> • have participated in the discussions and asked clarifying questions; • have used concepts related to accessing a folder along the path; • have programmed the Easy Editor app interface; • have configured the loading of a list of images from an arbitrary folder to the application widget; • have answered the teacher's questions during the review stage. |

RECOMMENDED LESSON STRUCTURE

| Time | Stage | Stage aims |
|--|---|---|
| 10 min  | Storyline. Discussion: Project planning | <ul style="list-style-type: none"> ❑ Remind the storyline task: develop an Easy Editor app. ❑ Build the project mind map and highlight the tasks for this workday. |
| 10 min  | Qualification | <ul style="list-style-type: none"> ❑ Organize qualification confirmation for the developers by topic: <ul style="list-style-type: none"> ❑ Creating classes. ❑ Interface development (PyQt5). |
| 5 min  | Brainstorming: The Easy Editor interface | <ul style="list-style-type: none"> ❑ Name the Easy Editor interface elements. ❑ Make a diagram for placing elements in layouts |
| 20 min  | Platform: “VSC: The Easy Editor app” | <ul style="list-style-type: none"> ❑ Organize completion of task 1 in the exercise “VSC. The Easy Editor app”. |
| 5 min  | Break | <ul style="list-style-type: none"> ❑ Do a warm-up or change students’ activity. |
| 15 min  | Brainstorming: Loading graphic files | <ul style="list-style-type: none"> ❑ Demonstrate that it is required to create functions for obtaining and analyzing a list of file names from an arbitrary folder on a computer. ❑ Mention that it is necessary to use the os module command and explain how it works. ❑ Describe how functions work and how they relate ❑ Demonstrate the expected operation of the app after the functions have been introduced. |
| 20 min  | Platform: “VSC: The Easy Editor app” | <ul style="list-style-type: none"> ❑ Organize completion of task 2 in the exercise “VSC. The Easy Editor app”. |
| 5 min  | Wrapping up the lesson. Reflection | <ul style="list-style-type: none"> ❑ Conduct a technical interview based on the brainstorming material. ❑ Announce the content of the next workday. |

Storyline. Discussion: Project planning

(10 min)

Open the presentation. The developers do not need computers yet.

"Hello, colleagues! Today, we begin developing the Easy Editor app for photo editing. Our product must be easy to understand even for inexperienced users, so we need to carefully plan how to develop it.

Let's start the project by analyzing the customer's technical specification."

Study the project specification together with developers. Draw attention to the interface details (e.g., the window for selecting a folder with files to work with), to the photo processing tools, to the ability to process several images at once, to the automatic saving of the result to the Modified subfolder.

Recall and name familiar tools for working on the project. Suggest building the project mind map and a checklist to implement it.

Technical specification

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

Requirements:

- ❑ The interface must be like in the picture.
- ❑ Ability to **select a folder with images on a computer**.
- ❑ Processing tools:
 - "Make it black and white".
 - "Rotate left (90°)".
 - "Rotate right (90°)".
 - "Sharpen".
 - "Mirror (left to right)".
- ❑ **Saving edited photos** (copies of the original) to a new Modified subfolder.

Planning work on the project

Project **mind map**:

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.
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.

Specify the tasks to be implemented during the current workday. Then formulate its goal and content.

Qualification

(10 min)

Use the presentation to organize confirmation of the developers' qualifications before they start working. This time it covers the following topics: Object-Oriented Programming and Interface Development Using PyQt5.

Show and name all the **widgets in the picture:**

Confirmation of qualifications

Show and name all the **widgets in the picture:**

Qualification

How do I **create and position widgets in an empty window like in the picture?**

Qualification

If you are confused, use the theoretical documentation!

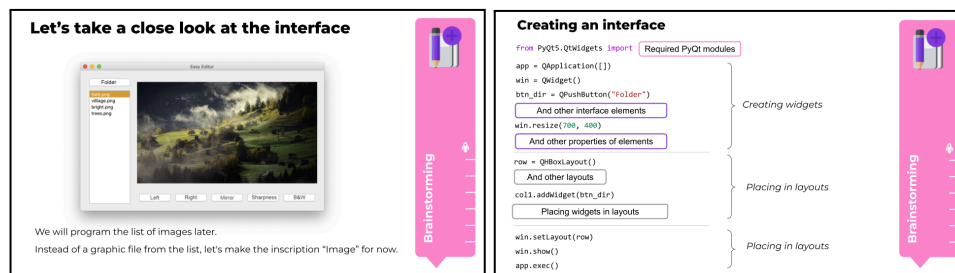
Brainstorming: The Easy Editor interface

(5 min.)

Use the presentation to analyze how the Easy Editor app interface is arranged. Note that displaying an image preview is possible only after loading a list of graphic files from a computer folder, so the “Image” inscription is enough to start.

- What widgets are there in the screenshot of the expected app interface? Which widgets need additional properties, such as size or label (don’t forget that the app window is also a widget)?
- How to place widgets in layouts? Describe at least two ways to place them.
- How to launch and evaluate the app interface?

Developers have already got experience developing interfaces, so there is no need to discuss all the details of the app window.



Wrap up the discussion and begin working on the VS Code.

Platform: “VSC. The Easy Editor app”

(20 min)

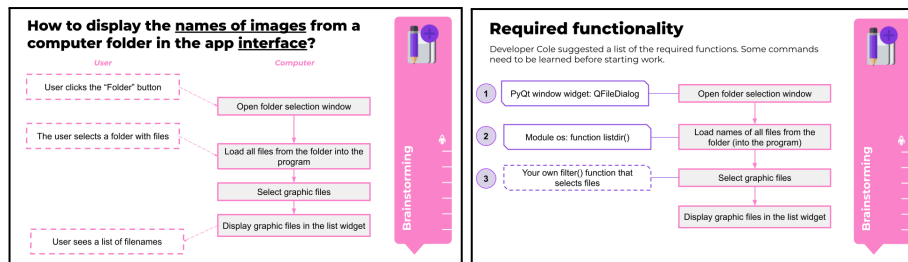
Arrange the work in the VS Code for the first task of the Easy Editor project. Remind developers that all the technical documentation on the interface development is available on the platform, and they can view any forgotten stuff there.

A link to the archive with the solution to task 1 is available at the end of the methodological guidelines.

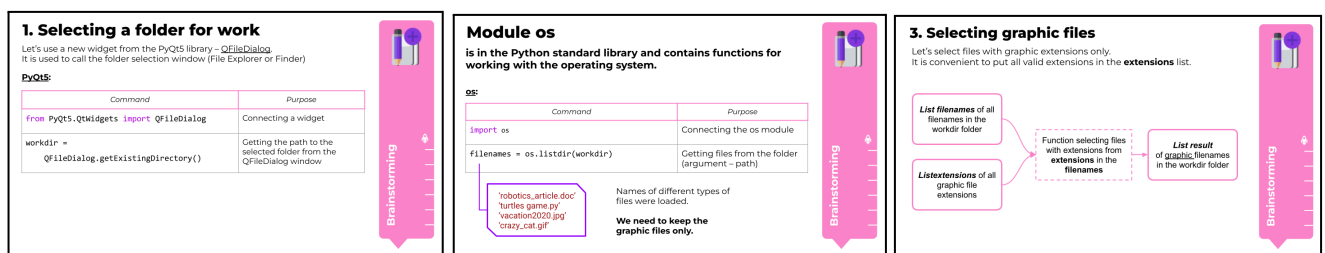
Brainstorming: Loading graphic files

(15 min)

On behalf of senior developer Cole, ask the question: “How to display the names of images from a computer folder (arbitrary) in the app interface?” Proceed to the diagram relating the actions a user should take and the execution of commands by a computer. Emphasize that to implement some functions, such as folder selection; kids need to learn new commands.



- Tell them how to program the selection of a work folder. The PyQt5 QFileDialog widget can call the File Explorer window and allow the user to select a directory. The `getExistingDirectory()` method returns the path to the selected folder to the program.
- Loading names of all files in the folder into the program is possible using the `os` module command. Explain briefly that `os` is a built-in module in the Python standard library. To work, they will need commands to call files and folders along the path. Show the `listdir()` function and mention the need to sort and keep only graphic files.
- Filtering files by extensions can be wrapped in a separate function `filter()`. Emphasize to the developers: they can get part of the filename with the extension using both a slice and `endswith()` function.



Technical comment. Python has a standard filter function, but writing your own function to filter files by extension is a good learning task. This task helps better understand when a function needs to be defined as a class method and when not.

The specified technical solutions can be combined in the `showFilenamesList()` handler function. The process of displaying the finished list of strings in the widget is similar to the task from the previous Smart Notes project.

Show implementation of the described functions in the program. Emphasize to the developers that they need to declare the `workdir` variable as global. Otherwise, after the handler function finishes, the path to the current work folder will be lost.

Technical comment. If you want, you can make the `workdir` variable a field of the `ImageProcessor` class in the next lesson. In the base version of the program, `workdir` will remain a global variable.

State the task for the next stage and proceed to programming.

Platform: “VSC. The Easy Editor app”

(20 min)

Arrange the work in the VS Code for the second task of the Easy Editor project.

A link to the archive with the solution to task 2 is available at the end of the methodological guidelines.

Wrapping up the lesson

(5 min.)

Have the developers turn away from the computers, then organize a technical interview about the brainstorming material.

Suggest that the developers complete the additional exercises to improve their skills and provide learning materials.

Answers for tasks

Excesrise “VSC. The Easy Editor app”.

Task 1. Creating an interface.

[Link to the archive.](#)

```
from PyQt5.QtWidgets import (
    QApplication, QWidget,
    QFileDialog, # Dialogue for opening files (and folders)
    QLabel, QPushButton, QListWidget,
    QHBoxLayout, QVBoxLayout
)

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

btn_left = QPushButton("Left")
btn_right = QPushButton("Right")
btn_flip = QPushButton("Mirror")
btn_sharp = QPushButton("Sharpness")
btn_bw = QPushButton("B/W")

row = QHBoxLayout()          # Main line
col1 = QVBoxLayout()         # divided into two columns
col2 = QVBoxLayout()
col1.addWidget(btn_dir)      # in the first - directory selection button
col1.addWidget(lw_files)     # and file list
col2.addWidget(lb_image, 95) # in the second - image
```

```

row_tools = QHBoxLayout()    # and button bar
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()
app.exec()

```

Task 2. Displaying a list of graphic file names.

[Link to the archive.](#)

```

import os
from PyQt5.QtWidgets import (
    QApplication, QWidget,
    QFileDialog, # Dialogue for opening files (and folders)
    QLabel, QPushButton, QListWidget,
    QHBoxLayout, QVBoxLayout
)

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

btn_left = QPushButton("Left")
btn_right = QPushButton("Right")
btn_flip = QPushButton("Mirror")
btn_sharp = QPushButton("Sharpness")
btn_bw = QPushButton("B/W")

row = QHBoxLayout()          # Main line
col1 = QVBoxLayout()         # divided into two columns
col2 = QVBoxLayout()
col1.addWidget(btn_dir)      # in the first - directory selection button
col1.addWidget(lw_files)     # and file list
col2.addWidget(lb_image, 95) # in the second - image

```

```

row_tools = QHBoxLayout()    # and button bar
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()

workdir = ''

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

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)
app.exec()

```