Methodological guidelines

The Basics of Image Processing

🚀**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. The package should have helpful and easy-to-use apps that can be used by both experienced tech experts and people who barely know how to use a computer. One of the apps should be a photo editor.

To create the Photo Editor app, the developers need to become familiar with the PIL library for image processing tools and recall their previous work with classes.

⚠ **SUMMARY:**

Lesson goal: Learn about the Python Imaging Library (PIL) for programmable image processing and put their knowledge into practice.

During the lesson, the developers will learn about the practicality of using Python tools to process graphic files. The students will learn how to open, edit, and save images. They will then use that knowledge to complete tasks related to the storyline.

💾 **LINKS AND ACCESSORIES:**

❏ Presentation,

❏ Exercises for the lesson: processing 1, processing 2 (Visual Studio Code).

🎯 EDUCATIONAL OUTCOMES

After the lesson, students will:	*The result is achieved when students:*
• list the capabilities of PIL; • know about the Image and ImageFilter modules in the PIL library; • list the parameters for an image (size, mode, color); • open, edit, and save a copy of an image in the project folder; • make a photo black and white, rotate it, and blur it; • understand that it is more convenient to create classes when processing several images.	• have participated in the discussions and asked clarifying questions; • can confidently recall the commands for working with images from a project file; • have processed images using filters from the library • have completed the task for linear image processing; • have completed the task to create an ImageEditor image handler class; • have answered the teacher's questions during the review stage.

RECOMMENDED LESSON STRUCTURE

Time	Stage	Stage aims
5 min	Storyline. Discussion: “Image Processing”	❏ Set the task: develop an app for simple photo editing. ❏ Arrive to the idea of studying programmable image processing (PIL library).
10 min	Qualification	❏ Organize confirmation of developers’ qualification by the following topics: ❏ Commands for working with files. ❏ Objects, properties, methods, classes.
15 min	Brainstorming: Image processing with PIL	❏ Select the Image and ImageFilter PIL modules. ❏ List the methods that can be used for image processing (Cropping, filters). ❏ Go through the tasks that require loading and processing image files.
20 min	Platform: “VSC: Graphics: Basics”	❏ Have the students complete the task “Graphics: Basics”.
5 min	Break	❏ Do a warm-up or change activities.
10 min	Brainstorming: “Processing images using classes”	❏ Demonstrate the need to create an ImageEditor class for easily processing several files with different filters. ❏ Describe what the class can do and the expected instance of the class, and then list its fields and methods. ❏ Go through the tasks using the ImageEditor class
20 min	Platform: “VSC: Graphics: Classes”	❏ Have the students complete the task “VSC: Graphics: Classes”.
5 min	End of the lesson. Reflection	❏ Conduct a technical interview based on the material of the brainstorming stage. ❏ Suggest that the students complete the bonus tasks on the VS Code for additional practice.

Storyline. Discussion: “Working with files”

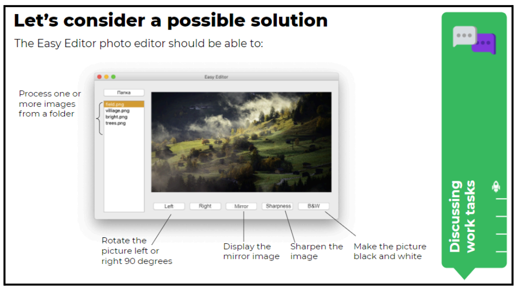
(5 min.)

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

> *“Hello, colleagues! We have a new order from the Ministry of Social Development. The Ministry representative is preparing a special software package for elderly people. The package should have helpful and easy-to-use apps that can be used by both experienced tech experts and people who barely know how to use a computer. One of the apps in the package is a photo editor.”*

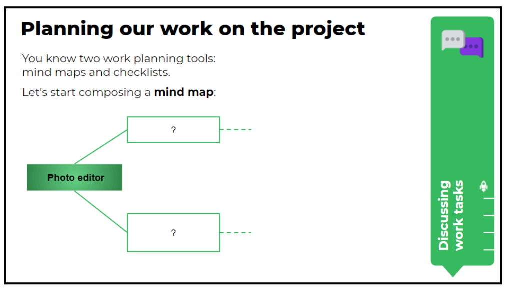
Show the developers the slide with the expected interface for the program. Note that the interface works with graphics. Study the different elements of the interface and list the elements of the app’s functionality.

Work with the developers to create a mind map of the project. Note that they will need to learn about additional libraries if they want to do more than just program the interface.



Let’s consider a possible solution

The Easy Editor photo editor should be able to:



Planning our work on the project

You know two work planning tools: mind maps and checklists.

Let’s start composing a **mind map**:

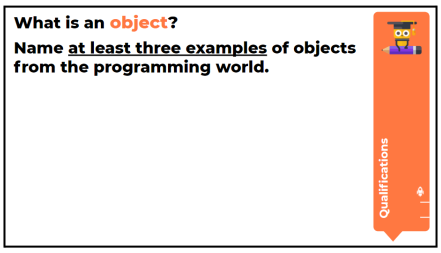
> *“Python has different libraries with the tools needed to work with raster graphics. Some of these tools even allow you to program things like computer vision for self-driving cars. The PIL (Python Imaging Library) will have all of the tools that we’ll need to complete the Photoeditor project and process photos.”*

Set the goal for the day and announce what will need to be done.

Qualification

(10 min)

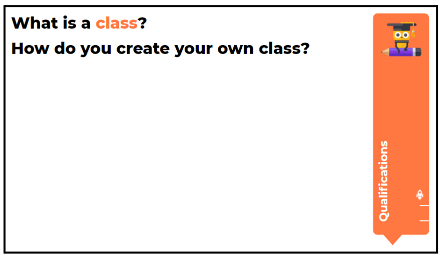
Using the presentation, organize the confirmation of the developers’ qualifications before the start of work. This time it will be organized by topics: working with files and object-oriented programming.



What is an object?

Name at least three examples of objects from the programming world.

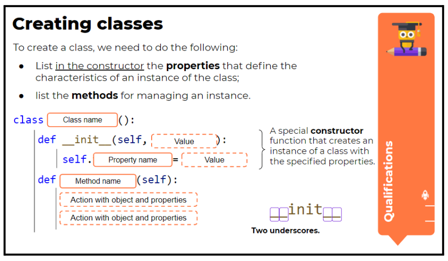
Qualifications



What is a class?

How do you create your own class?

Qualifications



Creating classes

To create a class, we need to do the following:

- List <u>in the constructor</u> the **properties** that define the characteristics of an instance of the class;
- list the **methods** for managing an instance.

```
class Class name():
    def __init__(self, Value):
        self.Property name = Value
    def Method name(self):
        Action with object and properties
        Action with object and properties
```

A special **constructor** function that creates an instance of a class with the specified properties.

__init__

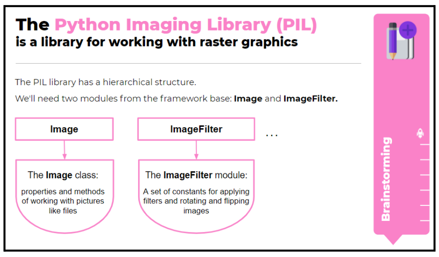
Two underscores.

Brainstorm: “Image processing with PIL”

(10 min)

Using the presentation, explain the different ways of processing images with PIL. Note that the library tools work with raster graphics.

Tell the students that the Image and ImageFilter modules are important parts of PIL. When loading and processing images, programmers work with Image objects. List the fields of the Image class that correspond to the image parameters.

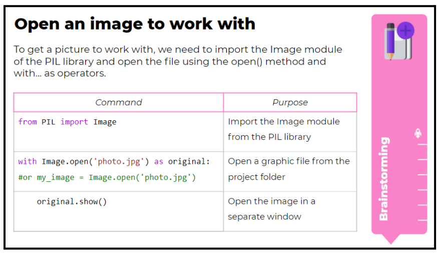


The Python Imaging Library (PIL)
is a library for working with raster graphics

The PIL library has a hierarchical structure.

We'll need two modules from the framework base: **Image** and **ImageFilter.**

Image
ImageFilter
...
The Image class: properties and methods of working with pictures like files
The ImageFilter module: A set of constants for applying filters and rotating and flipping images
Brainstorming



Open an image to work with

To get a picture to work with, we need to import the Image module of the PIL library and open the file using the open() method and with… as operators.

Command	*Purpose*
`from PIL import Image`	Import the Image module from the PIL library
`with Image.open('photo.jpg') as original:` `#or my_image = Image.open('photo.jpg')`	Open a graphic file from the project folder
`    original.show()`	Open the image in a separate window

Brainstorming

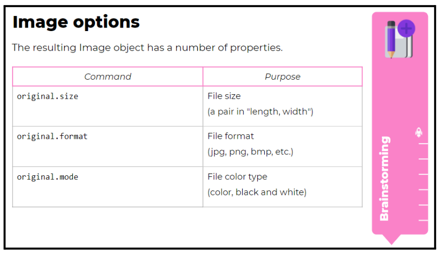


Image options

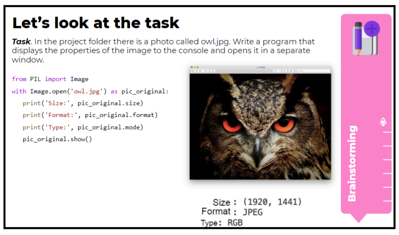
The resulting Image object has a number of properties.

Command	*Purpose*
`original.size`	File size (a pair in "length, width")
`original.format`	File format (jpg, png, bmp, etc.)
`original.mode`	File color type (color, black and white)

Brainstorming

Technical comment. For this lesson, students should only work with the images in the project folder. When students being the Photo Editor project in the next lesson there will be an arbitrary computer folder.

Discuss the task of loading an image, displaying it in a separate window, and typing the image parameters in the console. Then list the commands for processing images. Tell the students that several methods use constants.



Let’s look at the task

Task. In the project folder there is a photo called owl.jpg. Write a program that displays the properties of the image to the console and opens it in a separate window.

```
from PIL import Image
with Image.open('owl.jpg') as pic_original:
   print('Size:', pic_original.size)
   print('Format:', pic_original.format)
   print('Type:', pic_original.mode)
   pic_original.show()
```

```
 Size : (1920, 1441)
Format : JPEG
Type: RGB
```

Brainstorming

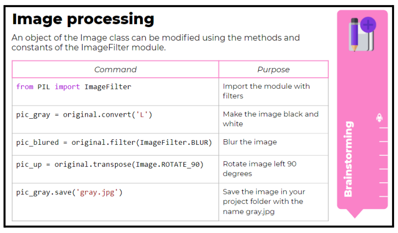
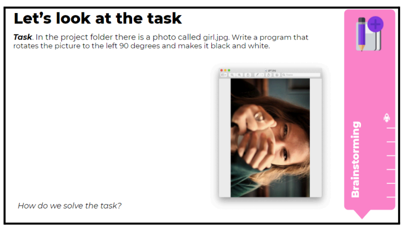


Image processing

An object of the Image class can be modified using the methods and constants of the ImageFilter module.

Command	*Purpose*
`from PIL import ImageFilter`	Import the module with filters
`pic_gray = original.convert('L')`	Make the image black and white
`pic_blured = original.filter(ImageFilter.BLUR)`	Blur the image
`pic_up = original.transpose(Image.ROTATE_90)`	Rotate image left 90 degrees
`pic_gray.save('gray.jpg')`	Save the image in your project folder with the name gray.jpg

Brainstorming



Let’s look at the task

Task. In the project folder there is a photo called girl.jpg. Write a program that rotates the picture to the left 90 degrees and makes it black and white.

How do we solve the task?

Brainstorming

Discuss one more task for processing photos that need to be rotated or changed into black and white. Note that it’s best to save the resulting work as a new file. This is done automatically when trying to save an image as a file with a non-existent name.

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

Platform: “VSC. Graphics: Basics”

(20 min)

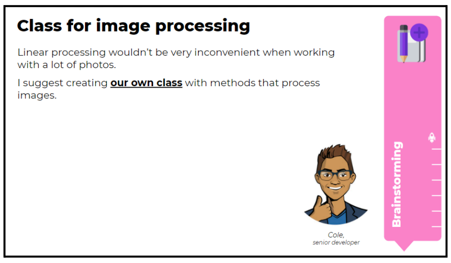
Arrange the work on the VS Code. The exercise is designed as a set of tasks for processing images. The answers for the required and bonus tasks can be found by following the links to the project archives at the bottom of the methodological guidelines.

Brainstorm: “Processing images using classes”

(15 min)

On behalf of the senior developer Cole, note that it is best to create your own class when processing several photos at a time. Actually, linearly processing a single image with several filters will make it look 3D.

So developers need to create an ImageEditor class that can load, process, and save its instances. Demonstrate the content of the class on the mind map.

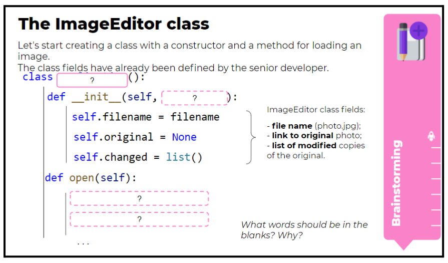


Class for image processing

Linear processing wouldn't be very inconvenient when working with a lot of photos.

I suggest creating **our own class** with methods that process images.

Cole,
senior developer
Brainstorming



The ImageEditor class

Let's start creating a class with a constructor and a method for loading an image.
The class fields have already been defined by the senior developer.

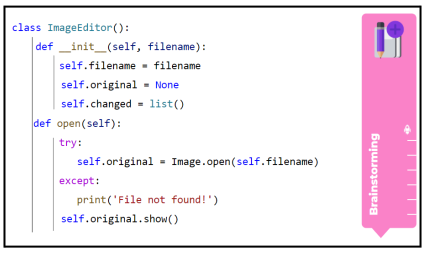
```
class    ?    ():
    def __init__(self,    ?    ):
        self.filename = filename
        self.original = None
        self.changed = list()
    def open(self):
            ?
            ?
        ...
```

ImageEditor class fields:
- **file name** (photo.jpg);
- **link to original** photo;
- **list of modified** copies of the original.

What words should be in the blanks? Why?

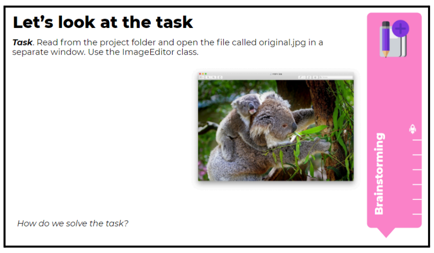
Make a technical note to the “Upload Photo” block and demonstrate the alternative way of opening pictures from the project folder. A special feature of this method is the exception handling when there is no file with the associated name (otherwise the program crashes).

Discuss the setup of the ImageEditor class. Suggest that the developers fill in the blanks using information from the previous slides. Then discuss the two tasks about applying and modifying a class.



```
class ImageEditor():
    def __init__(self, filename):
        self.filename = filename
        self.original = None
        self.changed = list()
    def open(self):
        try:
            self.original = Image.open(self.filename)
        except:
            print('File not found!')
        self.original.show()
```

Brainstorming

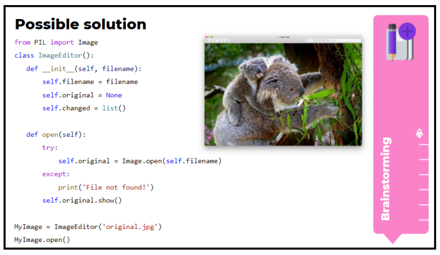


Let's look at the task

Task. Read from the project folder and open the file called original.jpg in a separate window. Use the ImageEditor class.

How do we solve the task?

Brainstorming



Possible solution

```
from PIL import Image
class ImageEditor():
    def __init__(self, filename):
        self.filename = filename
        self.original = None
        self.changed = list()

    def open(self):
        try:
            self.original = Image.open(self.filename)
        except:
            print('File not found!')
        self.original.show()

MyImage = ImageEditor('original.jpg')
MyImage.open()
```

Brainstorming

Ask questions to make sure they understand the program and then move on to the programming exercise.

Platform: “VSC. Graphics: Classes”

(20 min)

Arrange the work in the VS Code. The exercise is designed as a set of tasks for processing images. The answers for the required and bonus tasks can be found by following the links to the project archives at the bottom 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. Announce that they will begin developing the Photo Editor app on the next workday.

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

Exercises answers

Exercise “VSC. Graphics: Basics”

Link to the archive with solutions

```
from PIL import Image
from PIL import ImageFilter
#for bonus task
from PIL import ImageEnhance

with Image.open('original.jpg') as pic_original:
    print('Image is open\nSize:', pic_original.size)
    print('Format:', pic_original.format)
    print('Type:', pic_original.mode) #цветное
    pic_original.show()

    pic_gray = pic_original.convert('L')
    pic_gray.save('gray.jpg')
    print('Image is created\nSize:', pic_gray.size)
    print('Format:', pic_gray.format)
    print('Type:', pic_gray.mode) #bw
    pic_gray.show()

    pic_blured = pic_original.filter(ImageFilter.BLUR)
    pic_blured.save('blured.jpg')
    pic_blured.show()

    pic_up = pic_original.transpose(Image.ROTATE_180)
    pic_up.save('up.jpg')
    pic_up.show()


    #bonus 1. Mirror reflection
    pic_mirrow = pic_original.transpose(Image.FLIP_LEFT_RIGHT)
    pic_mirrow.save('mirrow.jpg')
    pic_mirrow.show()

    #bonus 2. Contrast enhancing
    pic_contrast = ImageEnhance.Contrast(pic_original)
    pic_contrast = pic_contrast.enhance(1.5)
    pic_contrast.save('contr.jpg')
    pic_contrast.show()
```

Exercise “VSC. Graphics: Classes”

Link to the archive with solutions

```
from PIL import Image
from PIL import ImageFilter

class ImageEditor():
    def __init__(self, filename):
        self.filename = filename
        self.original = None
        self.changed = list()

    def open(self):
        try:
            self.original = Image.open(self.filename)
        except:
            print('File not found!')
        self.original.show()

    def do_left(self):
        rotated = self.original.transpose(Image.FLIP_LEFT_RIGHT)
        self.changed.append(rotated)

        #bonus. Automatic naming for edited images
        temp_filename = self.filename.split('.')
        new_filename = temp_filename[0] + str(len(self.changed)) + '.jpg'

        rotated.save(new_filename)

    #bonus. Crop the image of baby koala
    def do_cropped(self):
        box = (250, 100, 600, 400) #left, up, right, down
        cropped = self.original.crop(box)
        self.changed.append(cropped)

        #bonus. Automatic naming for edited images
        temp_filename = self.filename.split('.')
        new_filename = temp_filename[0] + str(len(self.changed)) + '.jpg'

        cropped.save(new_filename)

MyImage = ImageEditor('original.jpg')
MyImage.open()

MyImage.do_left()
MyImage.do_cropped()

for im in MyImage.changed:
    im.show()
```