Methodological guidelines.

Smart Notes app P. 1

🚀**STORYLINE:**

The Scientific Institute of Theoretical Physics has addressed to ProTeam specialists. A group of theoretical physicists requires an app for managing scientific notes. The scientists should be able to create, delete, edit, tag, and search for scientific notes.

During the working day, developers who design the interface are forced to store notes in json files.

⚠ **SUMMARY:**

The **goal** is to apply knowledge and skills in storing data in files and developing applications using PyQT to develop a Smart Notes app.

During the lesson, students will start developing Smart Notes in the VS Code software development environment. The work on this app will take three lessons. During this lesson, students will design the Smart Notes interface, come up with a structure for storing tagged notes, and save the first note to a json file.

💾 **LINKS AND ACCESSORIES:**

- ❏ Presentation,
- ❏ Lesson tasks: Smart Notes app (Visual Studio Code).

Note. During the next three lessons, the students will work on the same exercise.

Technical comment. For project files to be displayed correctly on different operating systems, the files are set to utf-8 encoding. For correct processing of data by your computer, use the extra parameter encoding='utf-8'.

🎯 LEARNING RESULTS

After the lesson, the students will:	*The result is achieved when the students:*
• know that json is a data type for storing data in a file with a ready-to-use structure; • know and apply commands for working with json files; • know the attributes for saving data to a json file (e.g., sort_keys); • be able to retrieve data from a json file using keys.	• can explain in their own words that it is better to solve simple tasks using ordinary text files, and json files are better suited for data with complex structures; • know all the commands for opening a json file, and for reading and writing data; • have developed the basic interface of Smart Notes and added one note to the json file of the project; • have answered the teacher's questions at the review stage.

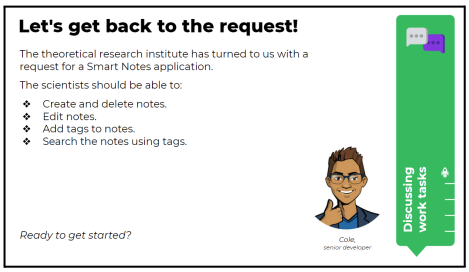
RECOMMENDED LESSON STRUCTURE

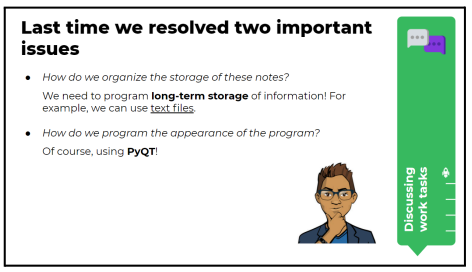
Time	Stage	Tasks of the stage
5 min 	**Storyline. Discussion: Smart Notes**	❏ Create a friendly atmosphere. ❏ Motivate students to develop a large and interesting new app. ❏ Encourage students to review the app development tools and data structures.
10 min 	**Qualification**	❏ Arrange repetition of the following using the presentation: ❏ Reading and writing to a file ❏ PyQT widgets, layouts
10 min 	**Discussion: Smart Notes interface**	❏ Discuss the Smart Notes interface: ❏ list the required widgets ❏ list the useful methods ❏ distribute widgets to layouts using the presentation
20 min 	**Visual Studio Code: VSC. Smart Notes App**	❏ Organize interface development using VSC in the VSC. Smart Notes App exercise
5 min 	**Break**	❏ Do a warm-up or change students' activity.
5 min 	**Review: Data structures**	❏ Use the presentation to review lists and dictionaries with the students (give examples).
10 min 	**Brainstorming: Storing Data in json Files**	❏ Tell about json files as a new way of storing information. ❏ Demonstrate functions for reading and writing information to/from a json file. ❏ Demonstrate possible storage of data in Smart Notes using a json file.
20 min 	**Visual Studio Code: VSC. Smart Notes App**	❏ Organize programming for storing notes in the VSC. Smart Notes App exercise.
5 min 	**Wrapping up the lesson. Reflection**	❏ Repeat the already learned lesson.

Storyline. Discussion: Smart Notes

(5 min)

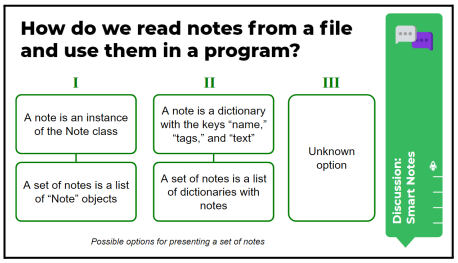
Welcome the developers. Remind them that in the previous lesson they studied theory material on long-term data storage.

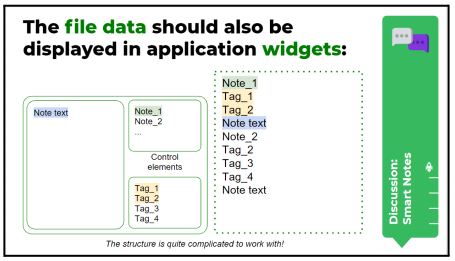


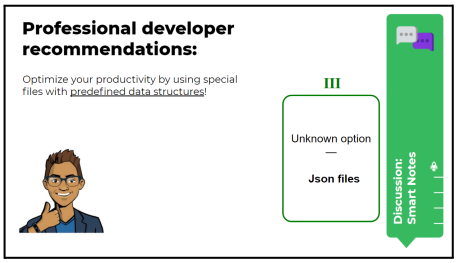


Ask them if storing their notes in a text file (.txt) is optimal? If not, what tool do they think would optimize data handling?

> *"Suppose you have a file for storing notes. How will this data be presented? How will the title, text, and tags of a note be related? How can we handle this data in the program and in what form can we write it to a file?*
>
> *That's correct - firstly, the notes data can be organized in the same way as the students in the task from the previous lesson. That is, you can create a Note class with a title, tags, and note text. In this case, the set of notes from the file is a list of Note objects.*
>
> *Secondly, you can present a note as a dictionary. We discussed dictionaries in the Data Structures module. In dictionaries, items are stored in the "key : value" pairs. In this case, one note could be a dictionary with the "title", "tags" and "text" keys.*







Thirdly, there are other ways that we do not know about yet."

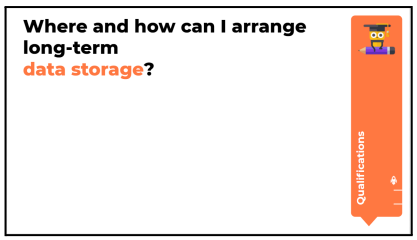
After considering various options, justify the need to use a file with a ready-to-use internal structure, that is, a json file.

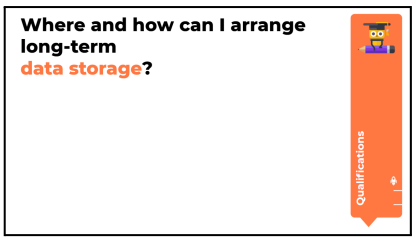
Set the goal of the working day and highlight its contents. In the first half, the interface of the Smart Notes app will be programmed, and in the second one, the storage of notes in a json file will be studied and implemented.

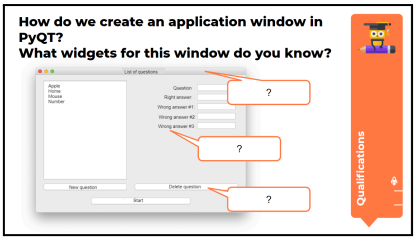
Qualification

(10 min)

Qualify the developer skills.





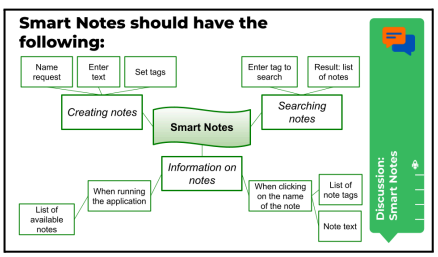


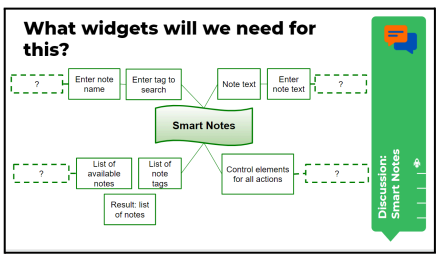
Discussion: Smart Notes interface

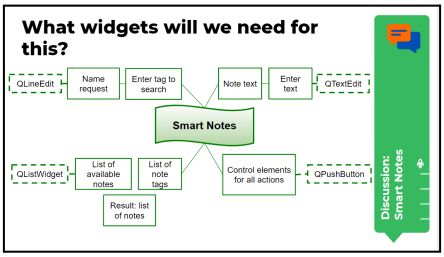
(10 min)

Return to the discussion of Smart Notes. Discuss the first task of this exercise: create an app interface. Consider the mind map of the project created by the senior developer Cole.

List the widgets required to implement this functionality. Already at the stage of considering the mind map, you can notice similar interface elements for which the same types of widgets can be used.

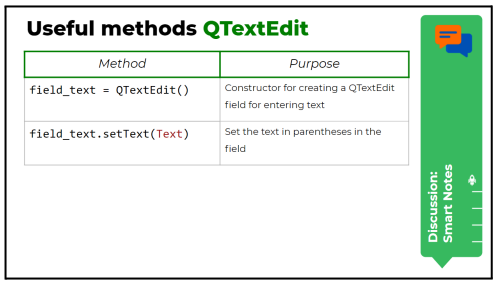


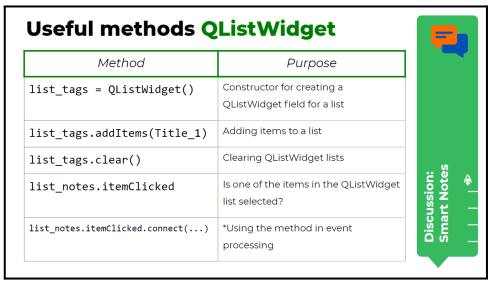




Discuss new types of widgets for text input fields (*QLineEdit* string field, *QTextEdit* large field, *QListWidget* clickable list).

> *"Please note that some end elements of the mind map are repeated, for example, List of Existing Notes and List of Note Tags. Perhaps they can be implemented using a single widget such as QListWidget.*
>
> *Let us consider the Entering Text Notes and Entering Tag for Search elements. Which widget(s) will help us implement them?* (Listen to the students' answers). *In fact, the input field for a tag can be small, since a tag is one word. QLineEdit will suffice. It is best to use a large field to enter text for your note. The QTextEdit widget is suitable for this, which we have not studied before, but it is similar to the previous one."*





List the names of the desired widgets and their methods. After discussion, tell them about the programming tasks for the next stage of the workday.

Visual Studio Code: VSC. Smart Notes App

(20 min)

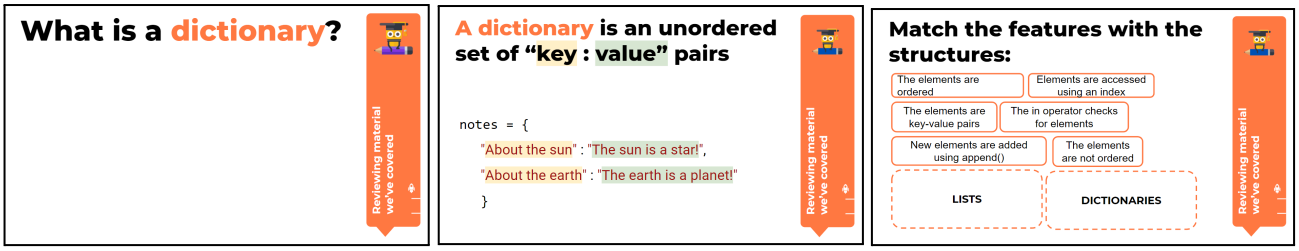
Arrange the work in the VS Code for the first task of the exercise titled VSC. Smart Notes App. When programming, it is allowed to consult with a neighbor.

The goal of the assignment is to design the application interface. You will find the answer to the exercise at the end of the methodological guidelines.

Qualification

(5 min)

Qualify the developer skills once again using the presentation. Make sure you get confident answers to questions about lists and strings. This will help developers learn how to use json files.



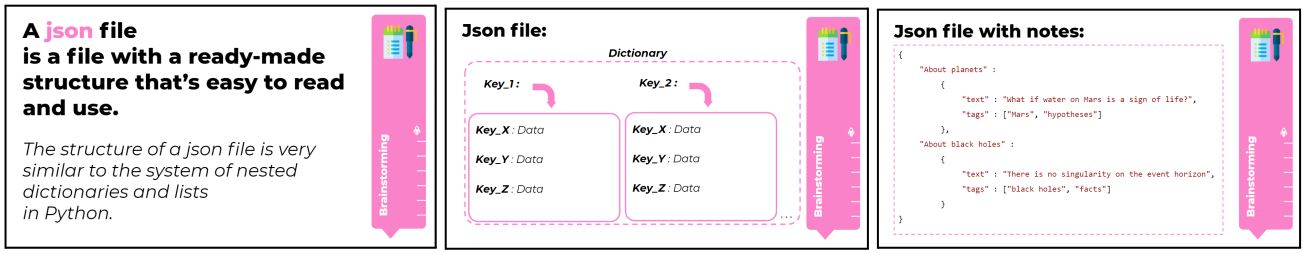
Brainstorming: Storing Data in json Files

(10 min)

Using the presentation, remind students that they need to come up with a convenient structure that can be displayed seamlessly in PyQT widgets and can be written and read to/from a file.

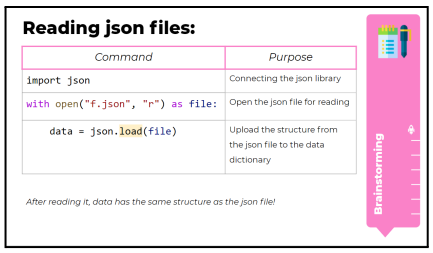
Demonstrate a possible option similar to a dictionary of dictionaries. Show that this note structure is fairly clear and convenient to use. Discuss how difficult it is to store such a dictionary in a file? In fact, this is not difficult, because this is exactly how json files are arranged, which many software developers use to store data. Actually, json files provide methods that allow you to read and write data in one line, but the main thing is to stick to the existing structure.

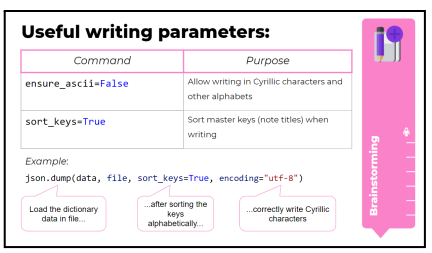
In general, a json file can be considered as a text file with a strictly defined structure.

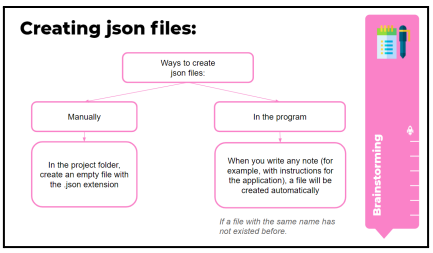


Demonstrate methods for reading and writing information to/from a json file. Pay attention to some useful parameters that will be useful later. For example, the sort_keys parameter, which allows you to sort the keys of a json file alphabetically. It will be very helpful when adding new notes.

Also note that the students can create a json file using the VS Code; for this, they need to create a new blank file with the json extension in the project folder.



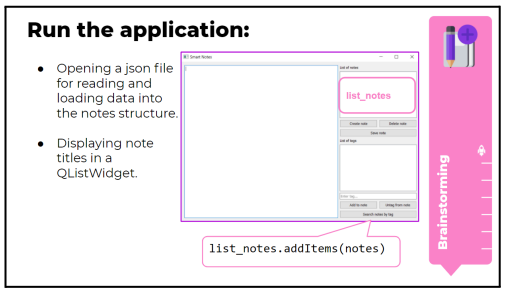




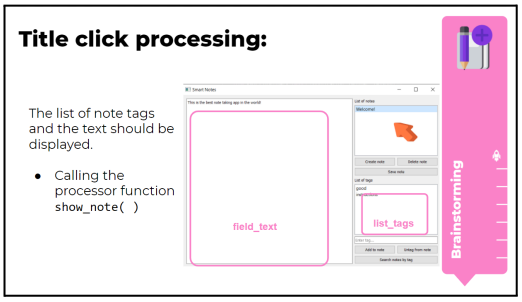
Go to solving current tasks using json files. Suggest the following way to define the structure of notes in a file: define the Notes dictionary of dictionaries in the program, create one greeting note in the Notes and write Notes to a json file.

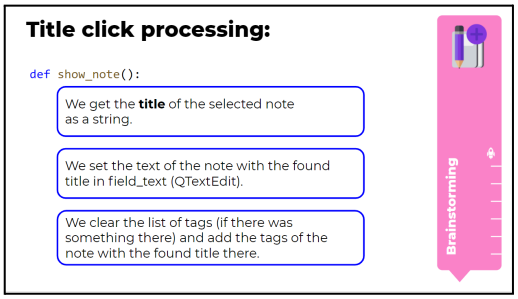
"We need to make sure that the data of the json file is read and displayed in the desired widgets. That is, firstly, when launching the app, a list of available notes should be displayed. Secondly, when you click on the title of the note, the tags and text of the note should be displayed in the respective widgets.

When you launch the program, you need to read all notes from the file into the Notes structure and display the list of keys (keys are the names of Notes items) using the list_notes.addItems (notes) method.



To handle clicking on the title of a note, a handler function should be created to display the elements of the note in the respective widgets. We will use this function time and time again in the future, so we need to make it universal."





Tell your students what the handler function should look like. Demonstrate the slide with code if needed. At the end of the discussion, tell students about the tasks for the next stage in the lesson.

Visual Studio Code: VSC. Smart Notes App

(20 min)

Arrange the work in the VS Code. Tell the students to open the development environment and complete the second task of the exercise titled VSC. Smart Notes App.

The task is to organize the storage of notes in a json file and their presentation in the app widgets. You will find the answer to the exercise at the end of the methodological guidelines.

Wrapping up the lesson

(5 min)

Briefly summarize the working day. When reviewing, focus on the purpose of the json file and methods for reading and writing data.

Answers to tasks

Task “VSC. Smart Notes application”

Task 1. Creating an interface for the Smart Notes application

```
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QLabel, QListWidget, QLineEdit, QTextEdit,
QInputDialog, QHBoxLayout, QVBoxLayout, QFormLayout

import json

app = QApplication([])

'''Application interface'''
#application window parameters
notes_win = QWidget()
notes_win.setWindowTitle('Smart Notes')
notes_win.resize(900, 600)

#application window widgets
list_notes = QListWidget()
list_notes_label = QLabel('List of notes')

button_note_create = QPushButton('Create note') #a window appears with the field "Enter note name"
button_note_del = QPushButton('Delete note')
button_note_save = QPushButton('Save note')

field_tag = QLineEdit('')
field_tag.setPlaceholderText('Enter tag...')
field_text = QTextEdit()
button_add = QPushButton('Add to note')
button_del = QPushButton('Untag from note')
button_search = QPushButton('Search notes by tag')
list_tags = QListWidget()
```

```
list_tags_label = QLabel('List of tags')

#arranging widgets by layout
layout_notes = QHBoxLayout()
col_1 = QVBoxLayout()
col_1.addWidget(field_text)

col_2 = QVBoxLayout()
col_2.addWidget(list_notes_label)
col_2.addWidget(list_notes)
row_1 = QHBoxLayout()
row_1.addWidget(button_note_create)
row_1.addWidget(button_note_del)
row_2 = QHBoxLayout()
row_2.addWidget(button_note_save)
col_2.addLayout(row_1)
col_2.addLayout(row_2)

col_2.addWidget(list_tags_label)
col_2.addWidget(list_tags)
col_2.addWidget(field_tag)
row_3 = QHBoxLayout()
row_3.addWidget(button_add)
row_3.addWidget(button_del)
row_4 = QHBoxLayout()
row_4.addWidget(button_search)

col_2.addLayout(row_3)
col_2.addLayout(row_4)

layout_notes.addLayout(col_1, stretch = 2)
layout_notes.addLayout(col_2, stretch = 1)
notes_win.setLayout(layout_notes)

#run the application
notes_win.show()
app.exec_()
```

Task 2. Creating a json file

```
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QLabel, QListWidget, QLineEdit, QTextEdit,
QInputDialog, QHBoxLayout, QVBoxLayout, QFormLayout

import json

app = QApplication([])

'''Notes in json'''
notes = {
    "Welcome!" : {
        "text" : "This is the best note taking app in the world!",
        "tags" : ["good", "instructions"]
    }
}
with open("notes_data.json", "w") as file:
    json.dump(notes, file, ensure_ascii=False)


'''Application interface'''
```

```
#application window parameters
notes_win = QWidget()
notes_win.setWindowTitle('Smart Notes')
notes_win.resize(900, 600)

#application window widgets
list_notes = QListWidget()
list_notes_label = QLabel('List of notes')

button_note_create = QPushButton('Create note') #a window appears with the field "Enter note name"
button_note_del = QPushButton('Delete note')
button_note_save = QPushButton('Save note')

field_tag = QLineEdit('')
field_tag.setPlaceholderText('Enter tag...')
field_text = QTextEdit()
button_add = QPushButton('Add to note')
button_del = QPushButton('Untag from note')
button_search = QPushButton('Search notes by tag')
list_tags = QListWidget()
list_tags_label = QLabel('List of tags')

#arranging widgets by layout
layout_notes = QHBoxLayout()
col_1 = QVBoxLayout()
col_1.addWidget(field_text)

col_2 = QVBoxLayout()
col_2.addWidget(list_notes_label)
col_2.addWidget(list_notes)
row_1 = QHBoxLayout()
row_1.addWidget(button_note_create)
row_1.addWidget(button_note_del)
row_2 = QHBoxLayout()
row_2.addWidget(button_note_save)
col_2.addLayout(row_1)
col_2.addLayout(row_2)

col_2.addWidget(list_tags_label)
col_2.addWidget(list_tags)
col_2.addWidget(field_tag)
row_3 = QHBoxLayout()
row_3.addWidget(button_add)
row_3.addWidget(button_del)
row_4 = QHBoxLayout()
row_4.addWidget(button_search)

col_2.addLayout(row_3)
col_2.addLayout(row_4)

layout_notes.addLayout(col_1, stretch = 2)
layout_notes.addLayout(col_2, stretch = 1)
notes_win.setLayout(layout_notes)

'''Application functionality'''
def show_note():
    #get the text from the note with the title highlighted and display it in the edit field
    key = list_notes.selectedItems()[0].text()
    print(key)
    field_text.setText(notes[key]["text"])
    list_tags.clear()
```

```
    list_tags.addItems(notes[key]["tags"])

'''Run the application'''
#connecting event handling
list_notes.itemClicked.connect(show_note)

#run the application
notes_win.show()

with open("notes_data.json", "r") as file:
    notes = json.load(file)
list_notes.addItems(notes)

app.exec_()
```

```
start to create smart notes app
```