Methodological guidelines

The Smart Notes App. P. 3

🚀**STORYLINE:**

The Scientific Institute of Theoretical Physics has turned to the ProTeam specialists for help. A team of theoretical physicists needs an app for managing their scientific notes. The scientists need to create, delete, edit, tag, and search for these notes.

During the working day, the developers will assess the optimality of the proposed technical solution and program an alternative demo-version of Smart Notes.

⚠ **SUMMARY:**

Lesson goal: Evaluate the optimality and convenience of working with data from the Smart Notes program and compare text files and json files.

In class, students will finish developing the Smart Notes App. For the first half of the lesson, students will summarize their knowledge of working with files and complete test work on the platform. In the second half of the lesson, students will organize the storage of Smart Notes in text files.

Technical comment. For project files to be displayed correctly on different operating systems, the files are set to utf-8 encoding. Use the extra parameter encoding='utf-8' to ensure that your computer correctly processes the data.

💾 **LINKS AND ACCESSORIES:**

❏ Presentation,

❏ Lesson tasks: qualification, demo-version (Visual Studio Code).

🎯 EDUCATIONA OUTCOMES

After the lesson, the students will:	*The result is achieved when the students:*
• understand what a parser is and how to read data from a text file; • know the advantages and disadvantages of text files; • know the advantages and disadvantages of json files; • can write and read data in a text file; • can write and read data in a json file.	• can explain in their own words the advantages of using text files for solving simple tasks and using json files for data with complex structures; • can confidently list the commands for opening a json file, as well as for reading and writing data; • have rogrammed a demo-version of Smart Notes with text files; • have answered the teacher's questions at the review stage.

RECOMMENDED LESSON STRUCTURE

Time	Stage	Stage tasks
5 min	**Storyline. Discussion: The Optimal Solution**	❏ Review the material of the previous lessons. ❏ Set a story-related task: explain the value of the technical solution to the customer ❏ Arrive to the idea of programing a demo-version of the app using text files, and then compare it with the program's json files.
10 min	**Brainstorming: Json files or text files?**	❏ Using the presentation, discuss: ❏ How is data stored in a json file? In a text file? ❏ How many ways can a json file be structured? And a text file? ❏ How can we use a program to see if a project has a file with a certain name?
15 min	**Platform: Qualification test**	❏ Organize qualification confirmation on the platform ❏ Answer any questions that may arise.
15 min	**Discussion: Smart Notes in text files**	❏ Discuss the storage of data using the "one note, one file" approach in Smart Notes: ❏ Develop an algorithm for reading data from the file with notes; ❏ Develop an algorithm for the creation and storage of notes in a file.
5 min	**Break**	❏ Do a warm-up or change activities
20 min	**Visual Studio Code: "VSC: Smart Notes Analysis"**	❏ Have the students complete the "VSC: Smart Notes Analysis" exercise to create a demo version of the app using text files.
10 min	**Discussion: Optimal Data Storage**	❏ Discuss which tasks are suitable to each method of data storage.
5 min.	**End of the lesson.**	❏ Announce the release of the project to production. ❏ Conduct a technical interview based on the material of brainstorming and answer any questions.

Storyline. Discussion: The Optimal Solution

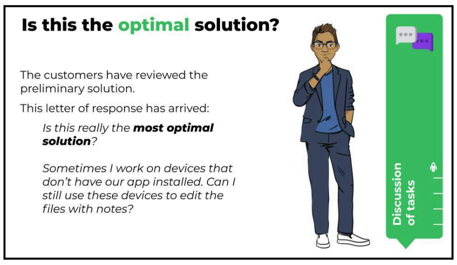
(5 min.)

Welcome the developers. Remind them that they have completed most of the work for the Smart Notes project.

Explain the lesson tasks for the day. Tell them that the customer would like to know whether json files are better for storing notes. Is it convenient to use json with notes on a computer that does not have the app? Is it easy to share your ideas with other scientists by typing out notes or by sending them over e-mail?

Move on to the task.

> *"To understand if json is truly the optimal choice, we will discuss everything we know about json files and text files. After that, we will create a new demo-version of Smart Notes using text files to simulate an alternative solution.*
>
> *As you might have noticed earlier, many functions in the app are very similar, so we are just going to limit the task to just reading data from a file and creating and saving notes.*
>
> *After that, we should be able to discuss:*



Is this the optimal solution?

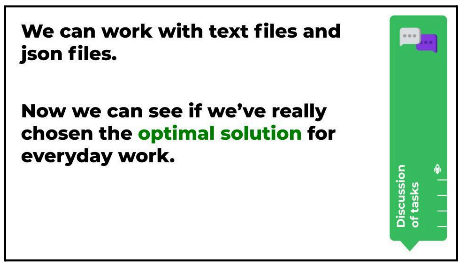
The customers have reviewed the preliminary solution.

This letter of response has arrived:

*Is this really the **most optimal solution**?*

Sometimes I work on devices that don't have our app installed. Can I still use these devices to edit the files with notes?

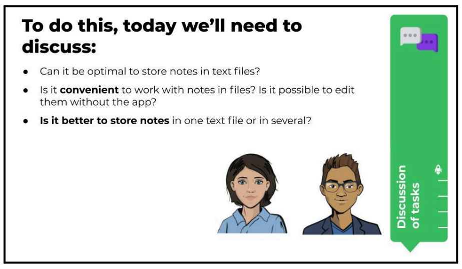
Discussion of tasks



We can work with text files and json files.

Now we can see if we've really chosen the optimal solution for everyday work.

Discussion of tasks



To do this, today we'll need to discuss:

- Can it be optimal to store notes in text files?
- Is it **convenient** to work with notes in files? Is it possible to edit them without the app?
- **Is it better to store notes** in one text file or in several?

Discussion of tasks

> *and make our final recommendations: which situations are best suited for json files and which are best for text files".*

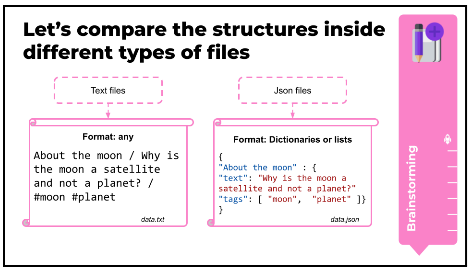
After setting the task, move on to the discussion.

Brainstorming: Json Files or Text Files?

(10 min.)

Lead the discussion using the presentation. Compare the structure of a json file and a text file. During the discussion, address the problem: text files are not machine-readable and cannot recognize data.

To receive a list of notes like the one in the main version of the app, the notes should be supplemented with line-by-line file reading by selecting pieces of information from the general data block (parsing).



Let's compare the structures inside different types of files

Text files

Format: any

```
About the moon / Why is
the moon a satellite
and not a planet? /
#moon #planet
```

data.txt

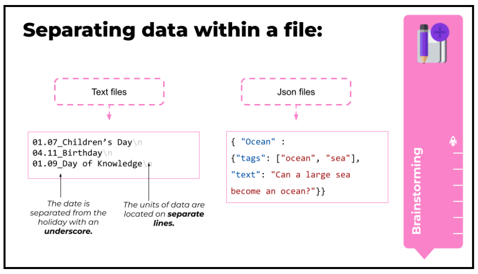
Json files

Format: Dictionaries or lists

```
{
"About the moon" : {
"text": "Why is the moon a
satellite and not a planet?"
"tags": [ "moon",  "planet" ]}
}
```

data.json

Brainstorming



Separating data within a file:

Text files

```
01.07_Children's Day\n
04.11_Birthday\n
01.09_Day of Knowledge\n
```

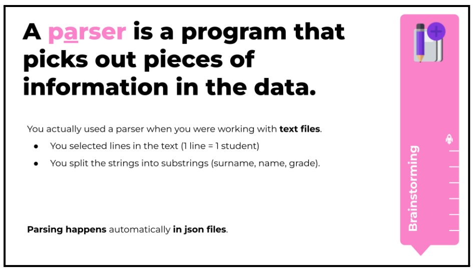
*The date is separated from the holiday with an **underscore**.*

*The units of data are located on **separate lines**.*

Json files

```
{ "Ocean" :
{"tags": ["ocean", "sea"],
"text": "Can a large sea
become an ocean?"}}
```

Brainstorming



A parser is a program that picks out pieces of information in the data.

You actually used a parser when you were working with **text files**.

- You selected lines in the text (1 line = 1 student)
- You split the strings into substrings (surname, name, grade).

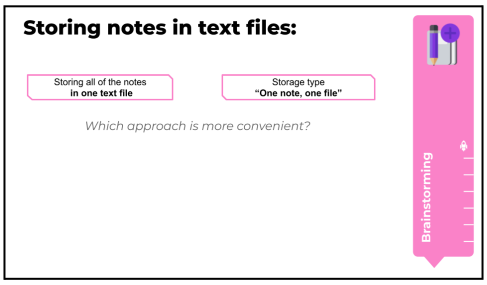
Parsing happens automatically **in json files**.

Brainstorming

Separately, discuss the different ways notes can be stored in text files.

"There are several ways to store notes in text files. Highlight the two main approaches: store all of the notes in one file (as we have done with json) or store notes using the "one note, one text file" method".

One file is more convenient for storing data entries. However, the "one note, one text file" method is more convenient for the user. This approach makes it easier to open files outside of the application and send them to others or print them out.



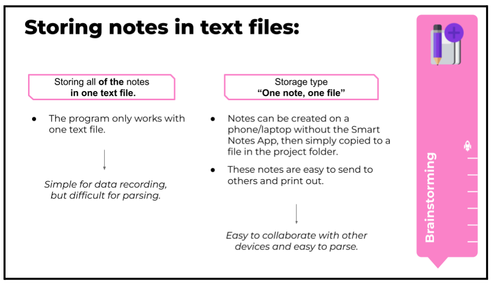
Storing notes in text files:

Storing all of the notes **in one text file**

Storage type **"One note, one file"**

Which approach is more convenient?

Brainstorming



Storing notes in text files:

Storing all **of the** notes **in one text file.**

- The program only works with one text file.

Simple for data recording, but difficult for parsing.

Storage type **"One note, one file"**

- Notes can be created on a phone/laptop without the Smart Notes App, then simply copied to a file in the project folder.
- These notes are easy to send to others and print out.

Easy to collaborate with other devices and easy to parse.

Brainstorming

To the joy of programmers, it is much easier to parse these types of notes. Programmers do not need to divide a file into notes because all of the notes are already stored in different files.

As a result of the discussion, decide to program the demo-version of Smart Notes using the "one note, one file" method of data storage.

Platform: Test: Working with Files

(15 min)

Arrange the work on the platform. Tell the students to complete "Test: Working with Files" Please note that the exercise will only be scored once the correct answer is given.

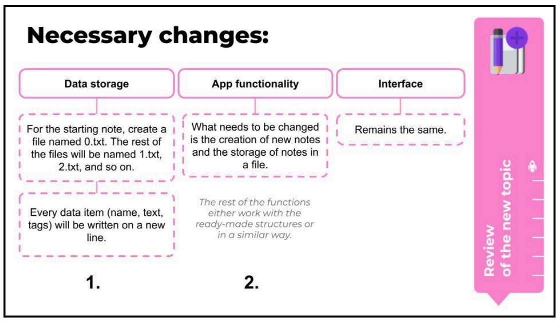
After completing the test, discuss any questions that arise.

Discussion: Smart Notes in text files

(15 min.)

Once again, explain that notes should be stored in different text files. Engage the developers in a discussion of what the demo version of the app should include and what functions need to be redone. Divide the task into two sub-tasks: changing the data storage and changing certain functions.

Make sure the students realize that they do not have to redo the entire app. To understand the point of the new method, they should only change the parts related to the reading of the notes, the creation of new notes, and the storage of typed text from the notes.



Necessary changes:

Data storage

For the starting note, create a file named 0.txt. The rest of the files will be named 1.txt, 2.txt, and so on.

Every data item (name, text, tags) will be written on a new line.

1.

App functionality

What needs to be changed is the creation of new notes and the storage of notes in a file.

The rest of the functions either work with the ready-made structures or in a similar way.

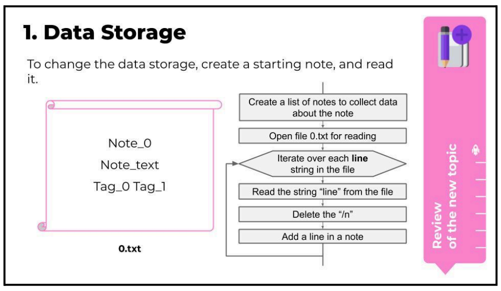
2.

Interface

Remains the same.

Move on to a discussion of the selected subtexts.

“Let’s start with changes to data storage. Imagine that the app only works with one file and that file only has one note. The file structure is displayed on the slide. How can we read the data — the name, text, and tags — from this file? (Students answer)



1. Data Storage

To change the data storage, create a starting note, and read it.

Note_0
Note_text
Tag_0 Tag_1

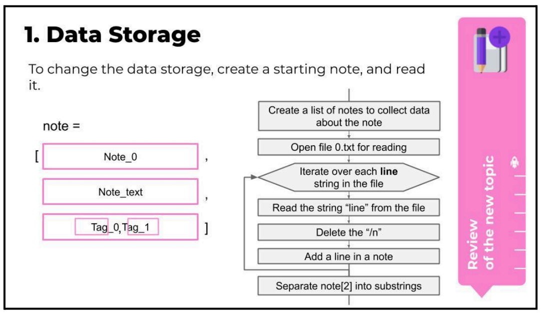
0.txt

Create a list of notes to collect data about the note
Open file 0.txt for reading
Iterate over each **line** string in the file
Read the string “line” from the file
Delete the “/n”
Add a line in a note

That is true. Generally, the application can be described using the following algorithm. Just like with the old version of Smart Notes, a structure is created — the list of notes — to store data about the notes. Then the file opens for reading and three strings of the file are recorded in a note: item 0 — name, item 1 — text, item 2 — tags.

Note that each item ends in a newline character. It must be deleted. Now, are we having any issues with the data? (Students answer)

Yes, actually. The tags that were on the same line were read as a single string. These need to be separated. To do this, the algorithm needs to be improved.



1. Data Storage

To change the data storage, create a starting note, and read it.

note =
[Note_0 ,
Note_text ,
Tag_0, Tag_1]

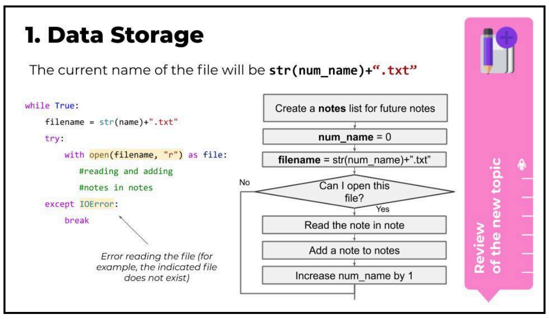
Create a list of notes to collect data about the note
Open file 0.txt for reading
Iterate over each **line** string in the file
Read the string “line” from the file
Delete the “/n”
Add a line in a note
Separate note[2] into substrings

Let us go back to the initial task. But what if the program is working with more than one note? How do we iterate over files with notes? (Students answer) *There are a few ways to do this. Let us look at the easiest way.*

Let us give the files with notes similarly numbered names. For example, 0.txt, 1.txt or note0.txt, note1.txt, etc. Then we will get the name of the file for reading by adding strings in the loop. We will add the data findings to the notes list.

The loop will run until the files with the specified names can be opened. Which construct will help you try to open the file and if unsuccessful (IOerror) will stop the loop? (Students answer)

Yes, the data will be read in the try...except blocks. The loop will stop when the program fails to open a file for reading".



1. Data Storage

The current name of the file will be `str(num_name)+".txt"`

```
while True:
    filename = str(name)+".txt"
    try:
        with open(filename, "r") as file:
            #reading and adding
            #notes in notes
    except IOError:
        break
```

Error reading the file (for example, the indicated file does not exist)

Discuss changes to the app's functionality in a similar way.

Tell the students about the tasks for the next stage of the lesson and move on to programming.

VS Code: VSC. Smart Notes Analysis

(20 min.)

Arrange the work in the VS Code. Tell the students to open the development environment and complete the exercise "VSC. Smart Notes Analysis" When programming, it is allowed to consult with a neighbor.

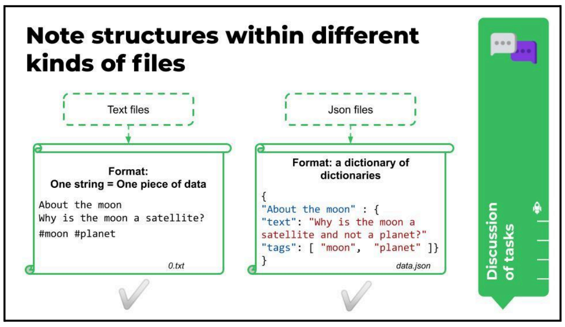
Technical comment. When reading a file, use the encoding='utf-8' parameter. This encoding (Unicode) ensures that the data in the text file is correctly displayed in the widgets on any platform. Otherwise, when opening a text file on Windows, the system might try to read it in Win-1251 (ansi) encoding.

The goal of the exercise is to create a demo-version of Smart Notes using another data storage.

Discussion: “VSC. Smart Notes Analysis”

(10 min.)

Sum up your work in the VS Code. Ask the developers which version of the app they find most convenient. Explain the advantages and disadvantages of both json files and text files. Ask for examples of the types of data that best suit each of the methods.



Note structures within different kinds of files

Text files

Format:
One string = One piece of data

```
About the moon
Why is the moon a satellite?
#moon #planet
```

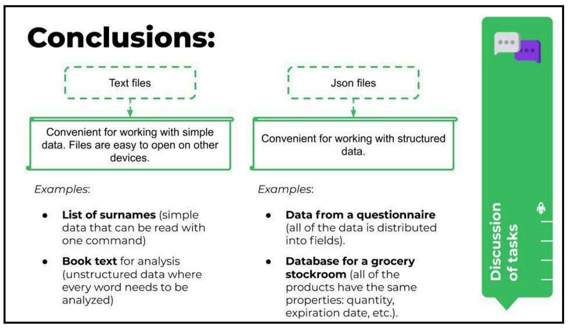
0.txt

Json files

Format: a dictionary of dictionaries

```
{
"About the moon" : {
"text": "Why is the moon a
satellite and not a planet?"
"tags": [ "moon",  "planet" ]}
}
```

data.json



Conclusions:

Text files

Convenient for working with simple data. Files are easy to open on other devices.

Examples:

- **List of surnames** (simple data that can be read with one command)
- **Book text** for analysis (unstructured data where every word needs to be analyzed)

Json files

Convenient for working with structured data.

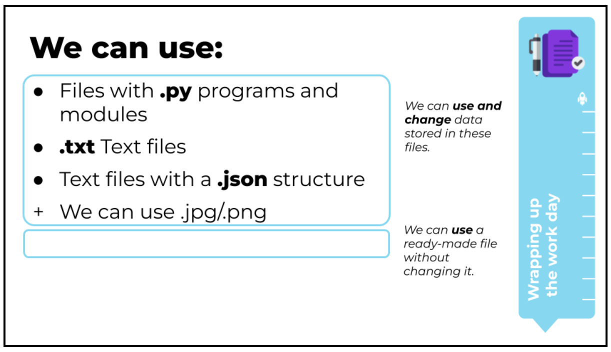
Examples:

- **Data from a questionnaire** (all of the data is distributed into fields).
- **Database for a grocery stockroom** (all of the products have the same properties: quantity, expiration date, etc.).

Wrapping up the lesson

(5 min.)

Summarize the lesson. With the developers, list every kind of file that they know how to work with over Python.



We can use:

- Files with **.py** programs and modules
- **.txt** Text files
- Text files with a **.json** structure

+ We can use .jpg/.png

*We can **use and change** data stored in these files.*

*We can **use** a ready-made file without changing it.*

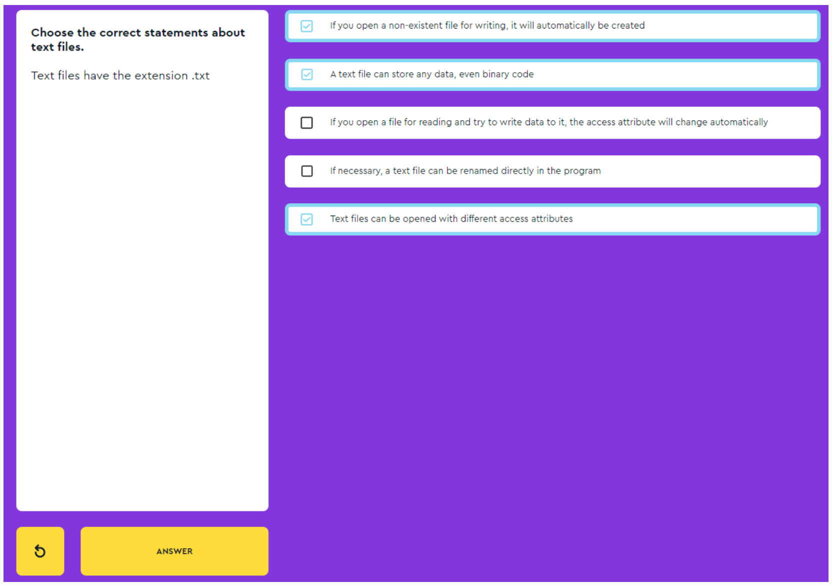
Wrapping up the work day

Mention that there are many more things that they can do with the language. For example, if they continue to study programming, in the future they may be able to create and change graphic files.

Exercises answers

Exercise: Test: Working with Files

1.

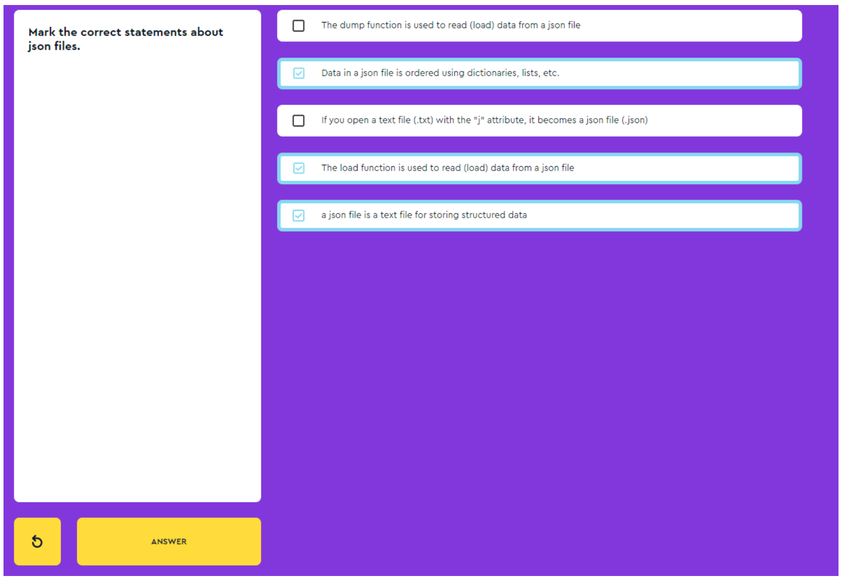


Choose the correct statements about text files.

Text files have the extension .txt

- [✓ If you open a non-existent file for writing, it will automatically be created]
- [✓ A text file can store any data, even binary code]
- If you open a file for reading and try to write data to it, the access attribute will change automatically
- If necessary, a text file can be renamed directly in the program
- [✓ Text files can be opened with different access attributes]

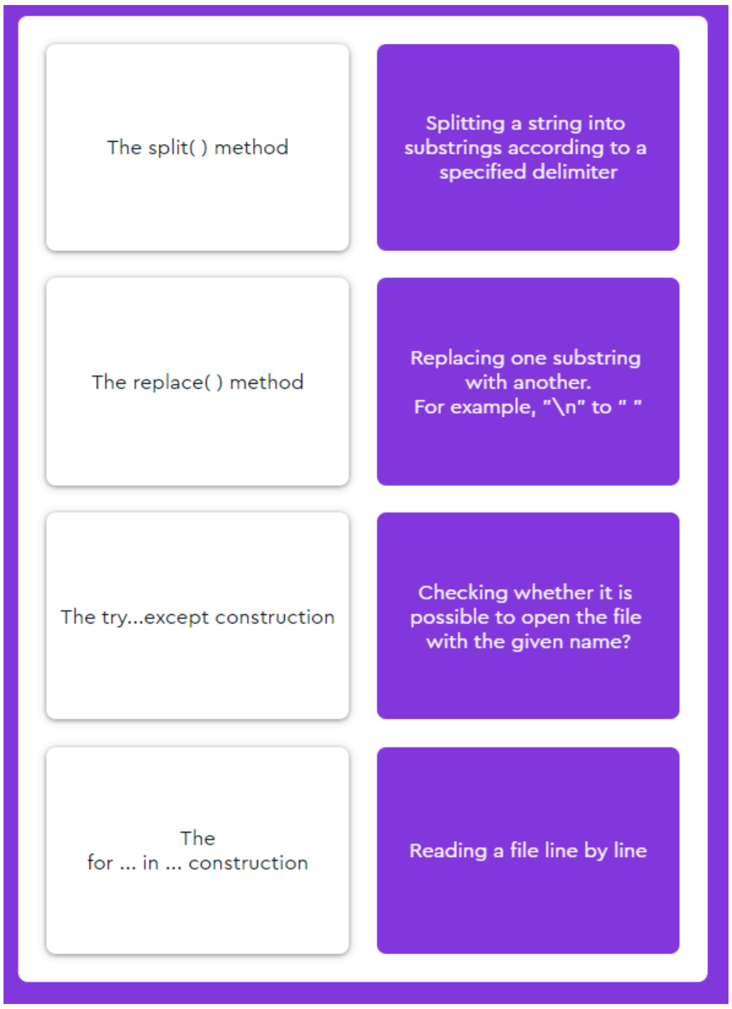
2.



Mark the correct statements about json files.

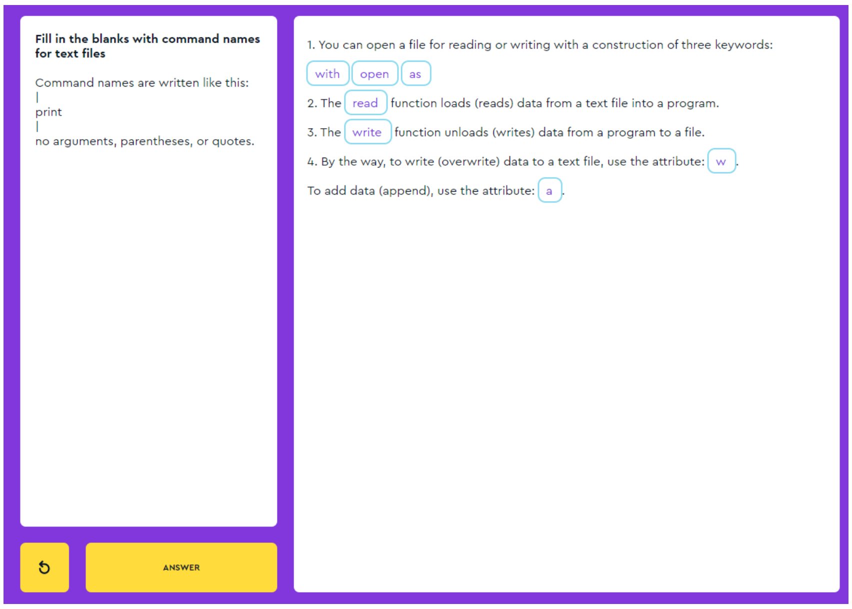
- The dump function is used to read (load) data from a json file
- [✓ Data in a json file is ordered using dictionaries, lists, etc.]
- If you open a text file (.txt) with the "j" attribute, it becomes a json file (.json)
- [✓ The load function is used to read (load) data from a json file]
- [✓ a json file is a text file for storing structured data]

3.



The split( ) method	Splitting a string into substrings according to a specified delimiter
The replace( ) method	Replacing one substring with another. For example, "\n" to " "
The try...except construction	Checking whether it is possible to open the file with the given name?
The for ... in ... construction	Reading a file line by line

4.



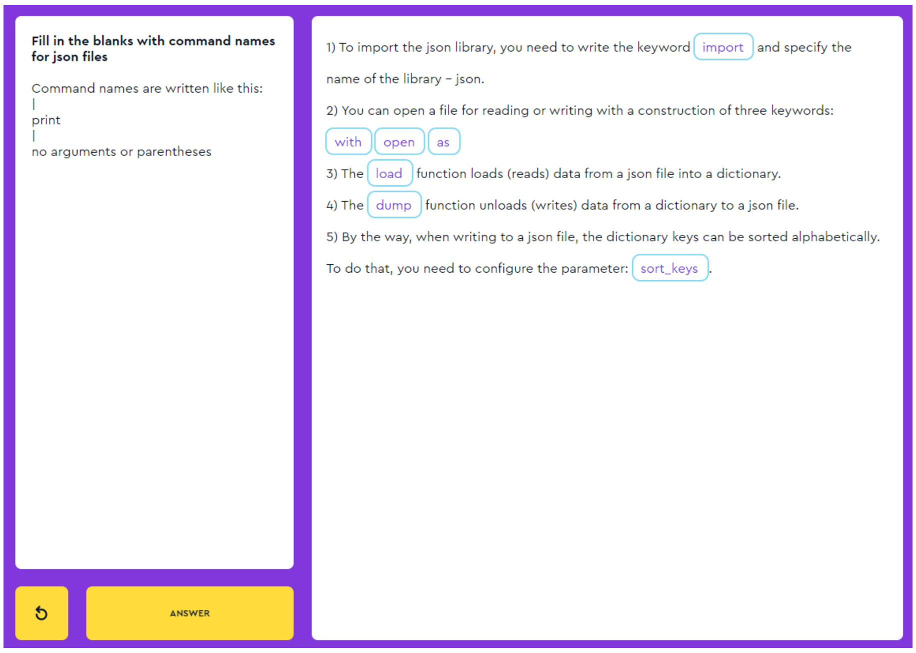
Fill in the blanks with command names for text files

Command names are written like this:
|
print
|
no arguments, parentheses, or quotes.

1. You can open a file for reading or writing with a construction of three keywords: with open as
2. The read function loads (reads) data from a text file into a program.
3. The write function unloads (writes) data from a program to a file.
4. By the way, to write (overwrite) data to a text file, use the attribute: w.

To add data (append), use the attribute: a.

5.



Fill in the blanks with command names for json files

Command names are written like this:
|
print
|
no arguments or parentheses

1) To import the json library, you need to write the keyword [import] and specify the name of the library – json.

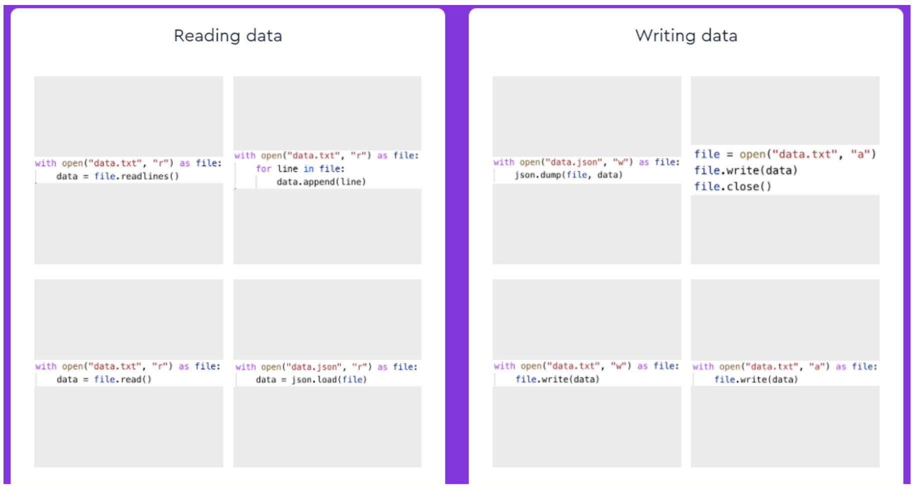
2) You can open a file for reading or writing with a construction of three keywords: [with] [open] [as]

3) The [load] function loads (reads) data from a json file into a dictionary.

4) The [dump] function unloads (writes) data from a dictionary to a json file.

5) By the way, when writing to a json file, the dictionary keys can be sorted alphabetically. To do that, you need to configure the parameter: [sort_keys].

6.



Reading data

```
with open("data.txt", "r") as file:
    data = file.readlines()
```

```
with open("data.txt", "r") as file:
    for line in file:
        data.append(line)
```

```
with open("data.txt", "r") as file:
    data = file.read()
```

```
with open("data.json", "r") as file:
    data = json.load(file)
```

Writing data

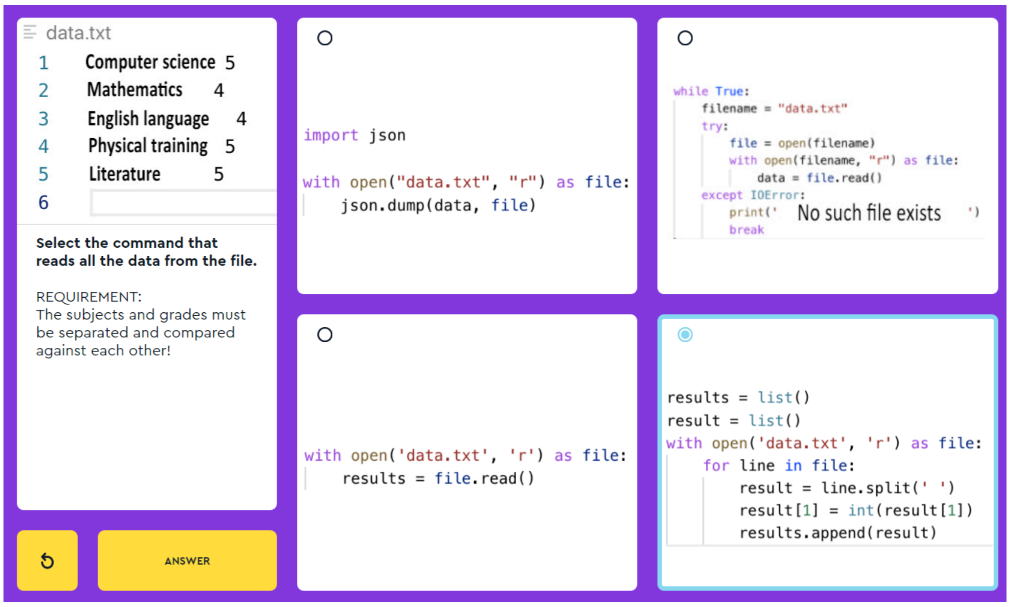
```
with open("data.json", "w") as file:
    json.dump(file, data)
```

```
file = open("data.txt", "a")
file.write(data)
file.close()
```

```
with open("data.txt", "w") as file:
    file.write(data)
```

```
with open("data.txt", "a") as file:
    file.write(data)
```

7.



data.txt

```
Computer science  5
Mathematics     4
English language    4
Physical training  5
Literature        5
```

Select the command that reads all the data from the file.

REQUIREMENT:
The subjects and grades must be separated and compared against each other!

○
```
import json

with open("data.txt", "r") as file:
    json.dump(data, file)
```

○
```
while True:
    filename = "data.txt"
    try:
        file = open(filename)
        with open(filename, "r") as file:
            data = file.read()
    except IOError:
        print(' No such file exists ')
        break
```

○
```
with open('data.txt', 'r') as file:
    results = file.read()
```

●
```
results = list()
result = list()
with open('data.txt', 'r') as file:
    for line in file:
        result = line.split(' ')
        result[1] = int(result[1])
        results.append(result)
```

Exercise "VSC: Smart Notes Analysis"

File 0.txt (starting note):

```
Welcome!
These are the smartest notes
#tutorial #smart
```

File notes_txt.py:

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

app = QApplication([])
notes = []

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

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

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

field_tag = QLineEdit('')
field_tag.setPlaceholderText('Enter tag...')
field_text = QTextEdit()
button_tag_add = QPushButton('Add to note')
button_tag_del = QPushButton('Unpin from note')
button_tag_search = QPushButton('Search notes by tag')
list_tags = QListWidget()
list_tags_label = QLabel('List of tags')

#arrangement of widgets by layouts
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_tag_add)
row_3.addWidget(button_tag_del)
row_4 = QHBoxLayout()
row_4.addWidget(button_tag_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)

App functionality
def show_note():
    key = list_notes.selectedItems()[0].text()
    print(key)
    for note in notes:
        if note[0] == key:
            field_text.setText(note[1])
            list_tags.clear()
            list_tags.addItems(note[2])

def add_note():
    note_name, ok = QInputDialog.getText(notes_win, "Add note", "Note name: ")
    if ok and note_name != "":
        note = list()
        note = [note_name, '', []]
        notes.append(note)
        list_notes.addItem(note[0])
        list_tags.addItems(note[2])
        print(notes)
        with open(str(len(notes)-1)+".txt", "w") as file:
            file.write(note[0]+'\n')

def save_note():
    if list_notes.selectedItems():
        key = list_notes.selectedItems()[0].text()
        index = 0
        for note in notes:
            if note[0] == key:
                note[1] = field_text.toPlainText()
                with open(str(index)+".txt", "w") as file:
                    file.write(note[0]+'\n')
                    file.write(note[1]+'\n')
                    for tag in note[2]:
                        file.write(tag+' ')
                    file.write('\n')
            index += 1
        print(notes)
    else:
        print("Note to save is not selected!")

#event handling
list_notes.itemClicked.connect(show_note)
button_note_create.clicked.connect(add_note)
button_note_save.clicked.connect(save_note)

#app startup
notes_win.show()

name = 0
note = []
while True:
    filename = str(name)+".txt"
    try:
        with open(filename, "r", encoding='utf-8') as file:
            for line in file:
                line = line.replace('\n', '')
                note.append(line)
        tags = note[2].split(' ')
        note[2] = tags

        notes.append(note)
        note = []
        name += 1

    except IOError:
        break

print(notes)
for note in notes:
    list_notes.addItem(note[0])

app.exec_()
```