algorithmics

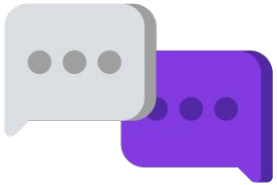
Module 3. Lesson 3.

The Smart Notes Application . P. 2

Link to methodological guidelines

Discussion:

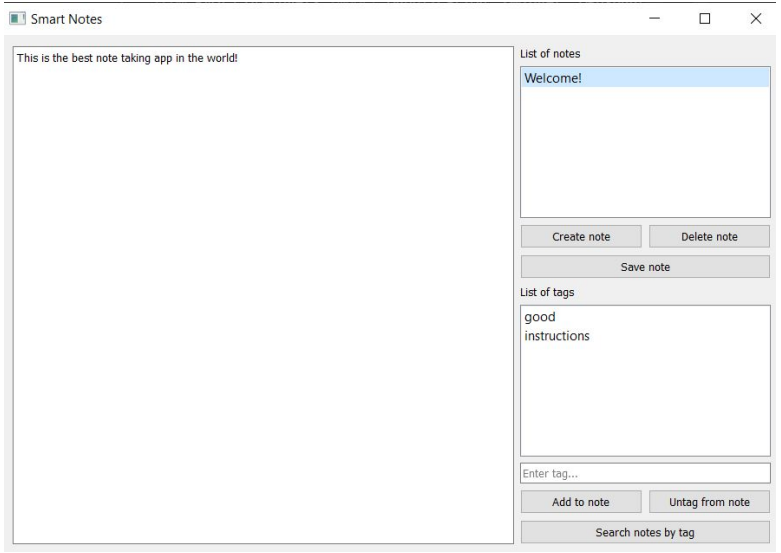
The Smart Notes Application



Let’s continue working on our order!

The Scientific Institute of Theoretical Physics has made an order for a “Smart Notes” application.

Last time, we:



created the “Smart Notes” **interface**

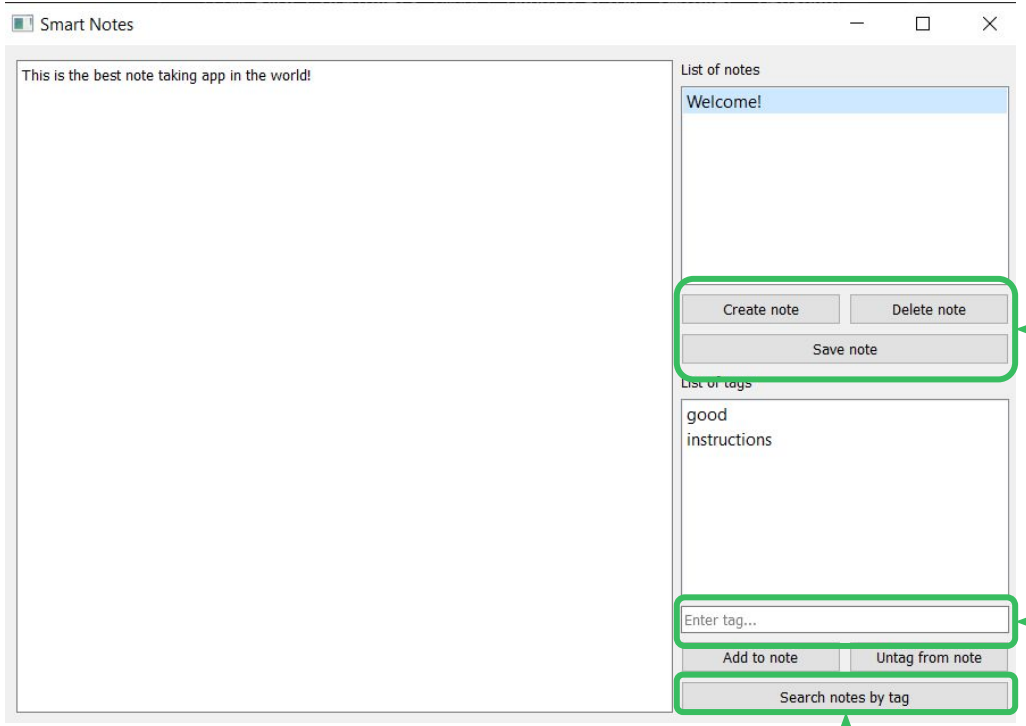
created the **json file** notes_data.json

added the first note to the json file

Cole,
Senior Developer

Discussion of work tasks

Tasks for today



Creating, saving, deleting notes.

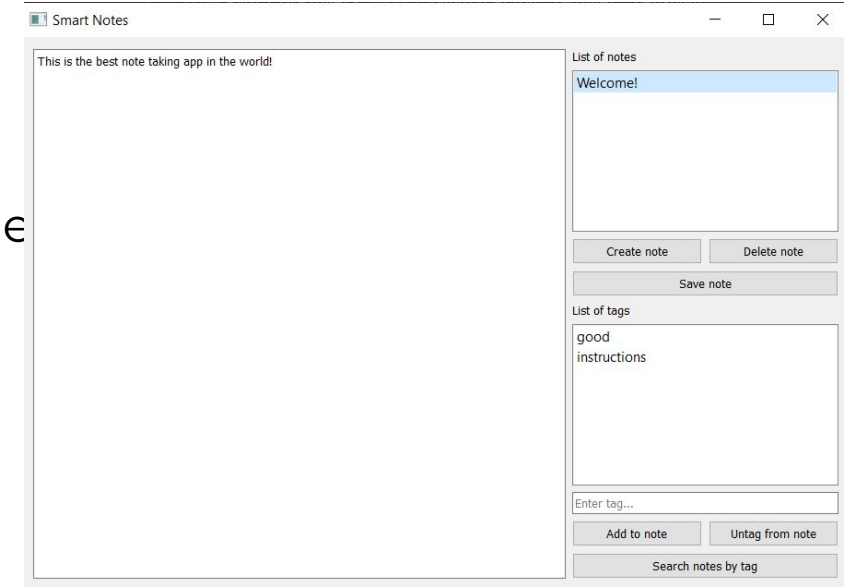
Adding a tag to or unclipping it from a note.

Searching for notes by entered tag.

Discussion of work tasks

To accomplish these tasks, we need to know :

- how to display the data from a selected note;
- how to organize the making of changes to the notes dictionary and the json file;
- how to clear list widgets and fields;
- how to search for notes by tag.



Discussion of work tasks

The goal of this workday is

to program the application interface and organize the storage of notes in a json file.

Today you will :

- Learn about the structure of a json file — a file with a built-in data structure.
- Program the application interface.
- Upload your first smart note.

Discussion of work tasks

Qualification

Demonstrate your knowledge of the PyQt library and how to work with text files

Qualification

What is a json file?

Qualification

A file in the json format is a text file for storing structured data.

The structure of a json file is very similar to a dictionary of dictionaries in Python.

Qualification

Which command opens a json file for reading ? How about for writing ?

Qualification

Opening a json file for reading and writing:

Command	*Purpose*
`with open("f.json", "r") as file:`	Open a json file for reading
`with open("f.json", "w") as file:`	Open a json file for writing

Qualification

Which command loads data from a json file?

Which command records data in a json file?

Qualification

Loading and recording data in a json file:

Command	*Purpose*
`data = json.load(file)`	Load a structure from a json file into data dictionary
`json.dump(data, file)`	Load a structure from data into a json file

Qualification

Writing the notes dictionary into a json file:

Command	*Purpose*
`with open("f.json", "w") as file:`	Open a json file for writing
`  json.dump(notes, file)`	Load a structure from data into a json file

Qualification

How can we sort a dictionary's keys as we write it into a json file?

Qualification

Sorting a dictionary's keys while writing it into a file:

```
with open("notes_data.json", "w") as file:
    json.dump(notes, file, sort_keys=True)
```

```
{
    "About space" : {
        "text" : "Is there extraterrestrial life?",
        "tags" : ["life", "space"]
    },
    "Astronauts" : {
        "text" : "Gagarin's courage is an example to us all!",
        "tags" : ["astronauts"]
    }
}
```

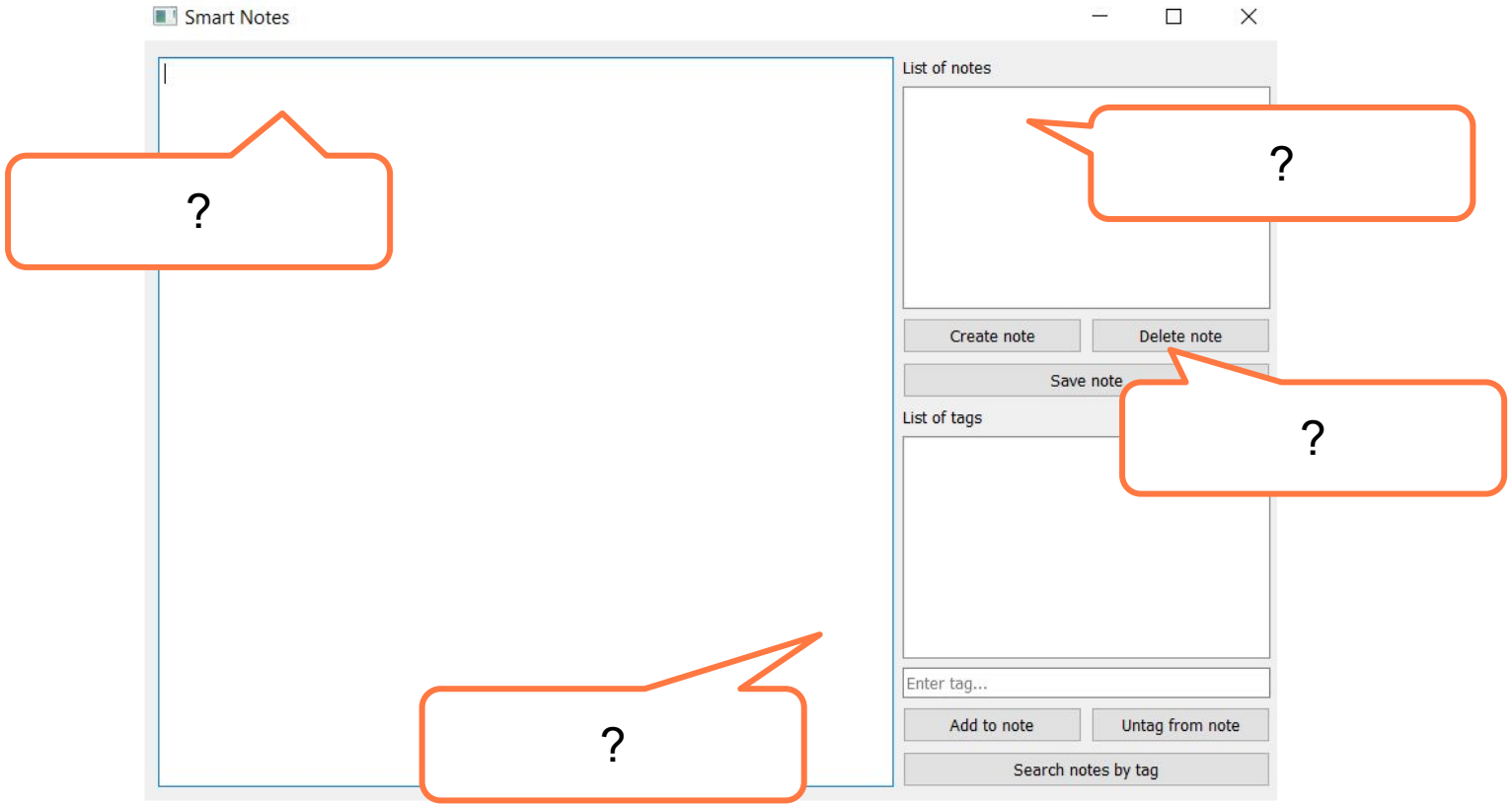
notes dictionary

```
{
    "Astronauts" : {
        "text" : "Gagarin's courage is an example to us all!",
        "tags" : ["astronauts"]
    },
    "About space" : {
        "text" : "Is there extraterrestrial life?",
        "tags" : ["life", "space"]
    }
}
```

file: notes_data.json

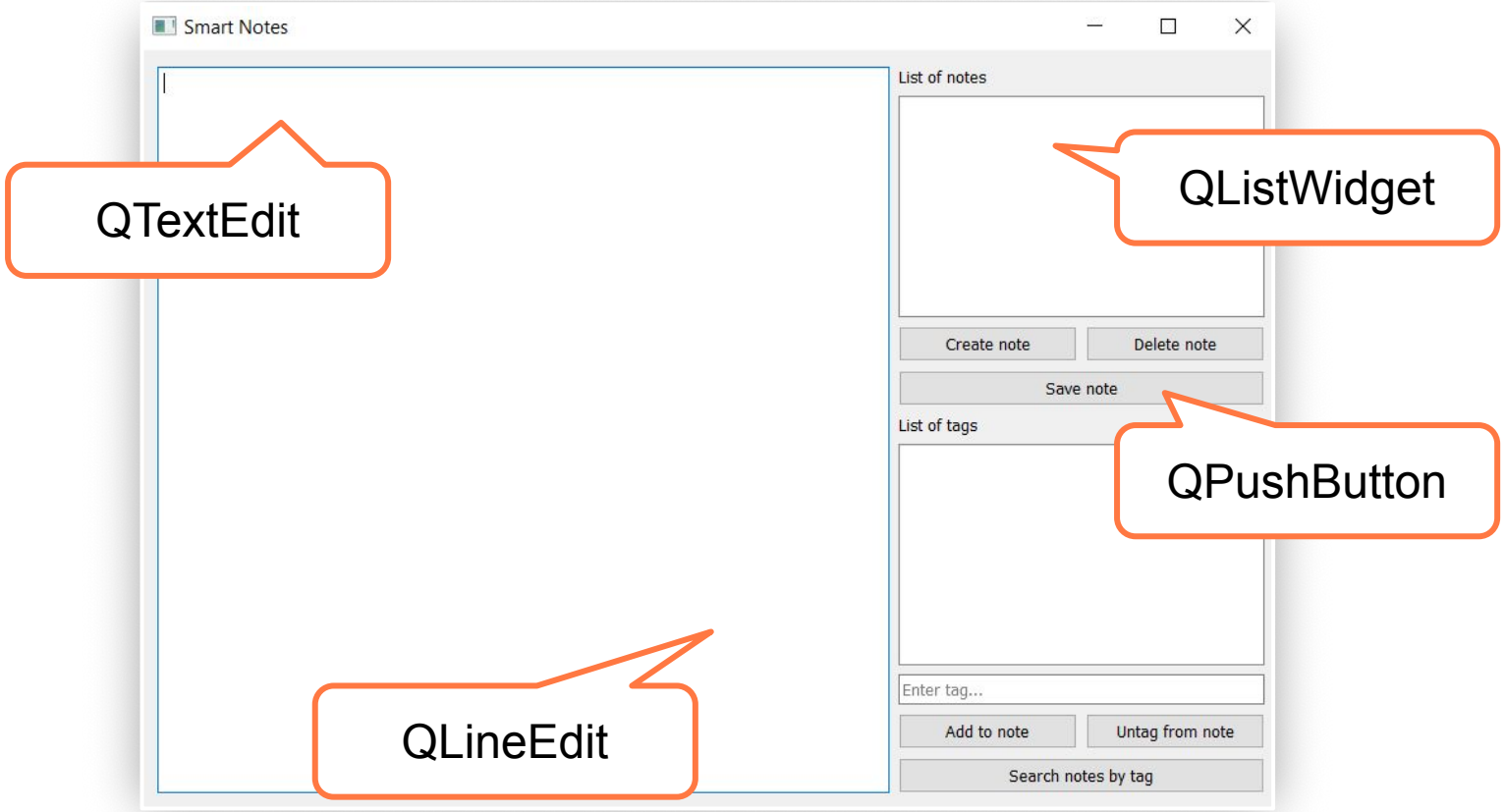
Qualification

Name these widgets:



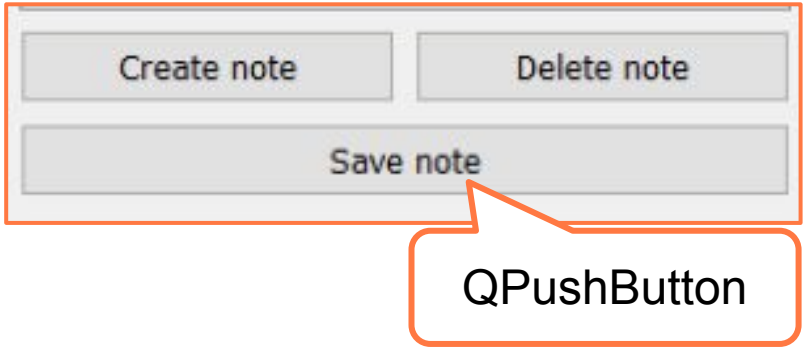
Qualification

Name these widgets:



Qualification

Which command will change the text in this widget?

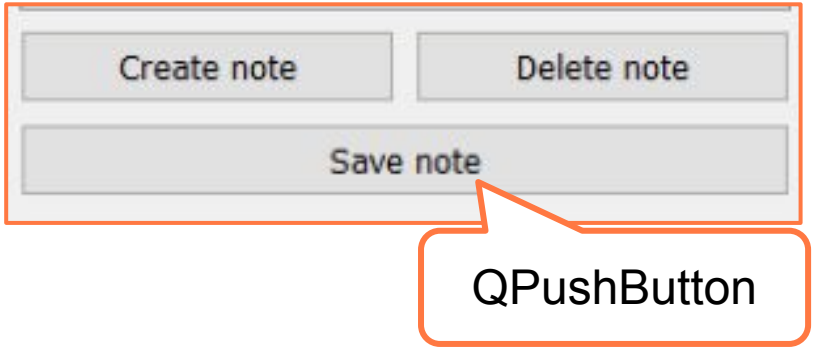



```
#creating a button without text
button_save = QPushButton(
    'Save note'
)

#write the label
#“Save text”...
```

Qualification

The command to change the text in this widget:



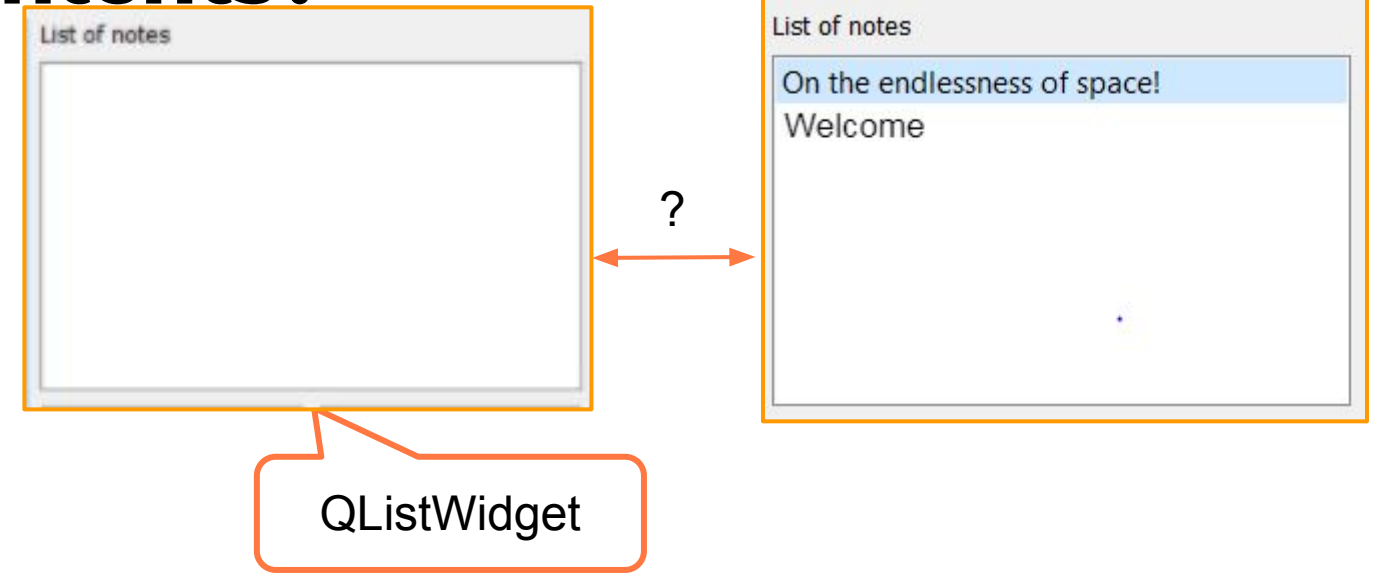
By the way, the `button_save.text()` method returns a line with the widget's label.

```
#creating a button without text
button_save = QPushButton(
    'Save note'
)

#the label “Save text”
button_save.setText(
    'Save text'
)
```

Qualification

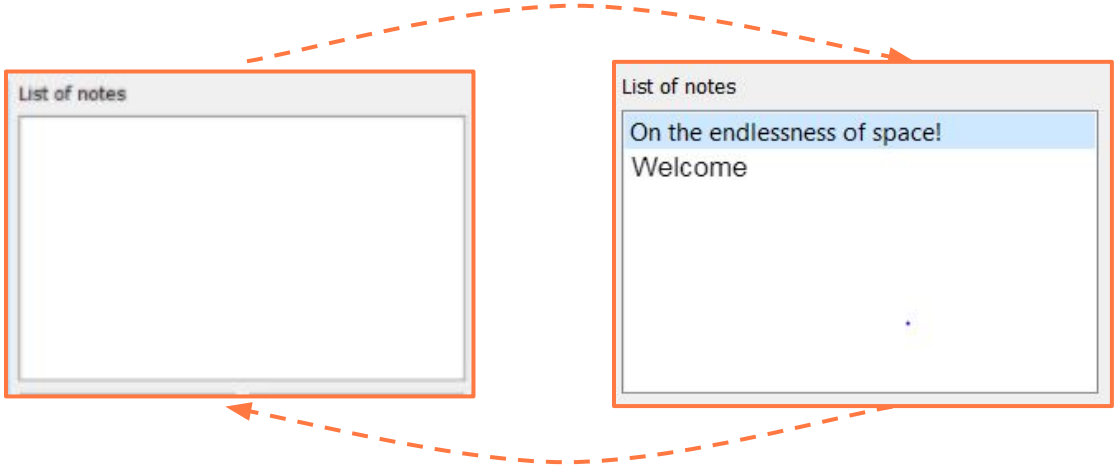
Which command will add elements to a widget? Which one will clear a widget's contents?



Qualification

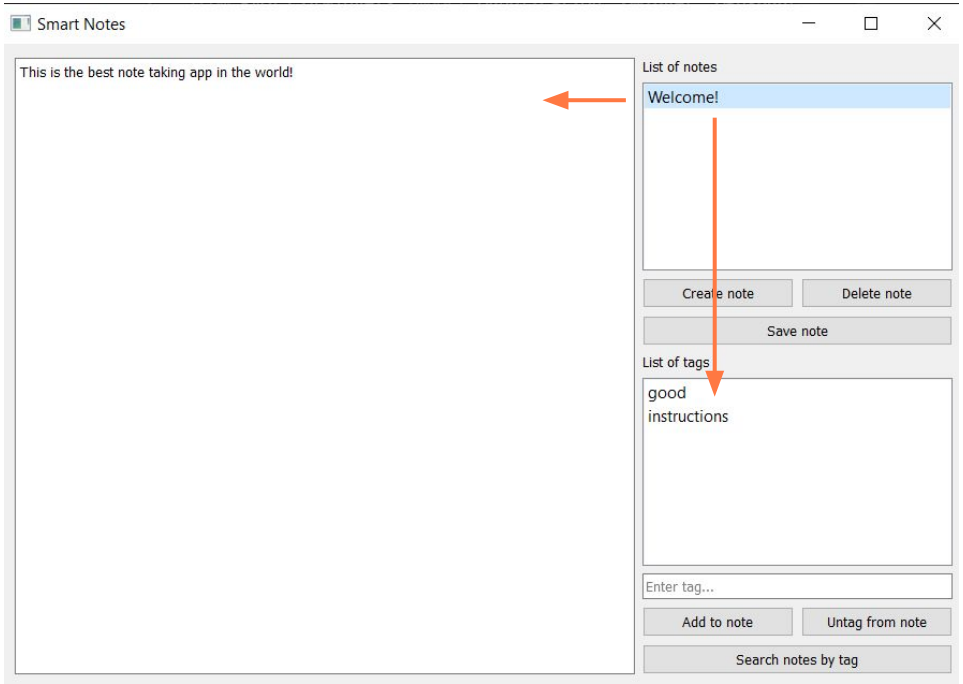
Adding and removing a set of elements:

```
list_notes.addItems(notes)
```




```
list_notes.clear()
```

How do we handle the mouse click of an element in the list?



In “Smart Notes”, clicking the name of a note should cause the text and tags to be displayed.

Qualification

Handling the mouse click of an element in the list:

```
def show_note():
```

```
list_notes.itemClicked.connect(show_note)
```

THE FUNCTION ITSELF IS DESCRIBED SEPARATELY

THE NAME OF THE HANDLER FUNCTION CALLED BY THE BUTTON

*In the same way, each button needs its own handler functions.

Qualification

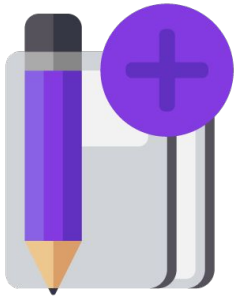
Qualification confirmed!

Excellent, you are ready to brainstorm and tackle today’s task!

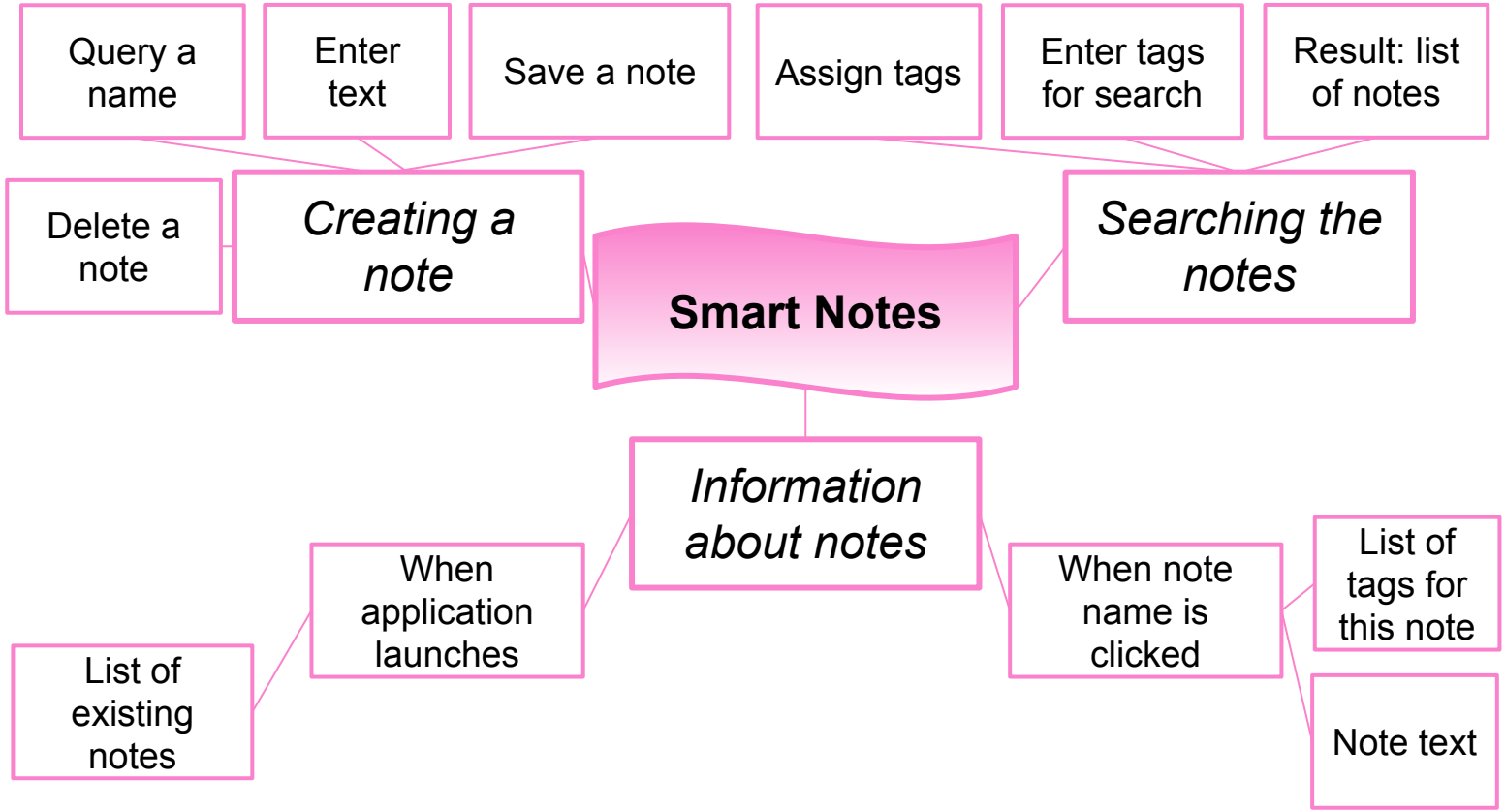
Qualification

Brainstorming:

Editing the Smart Notes

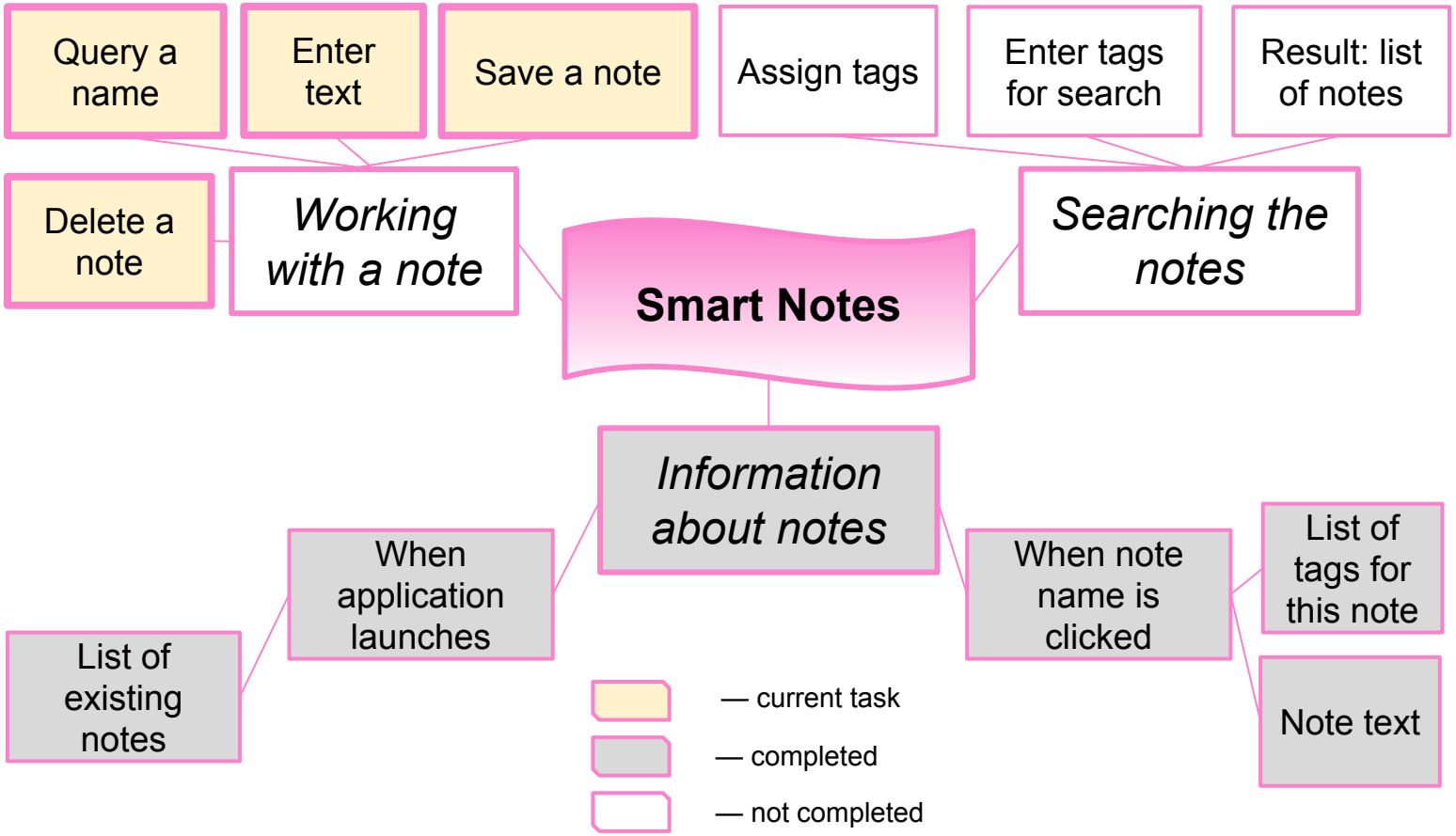


All the tasks for Smart Notes:



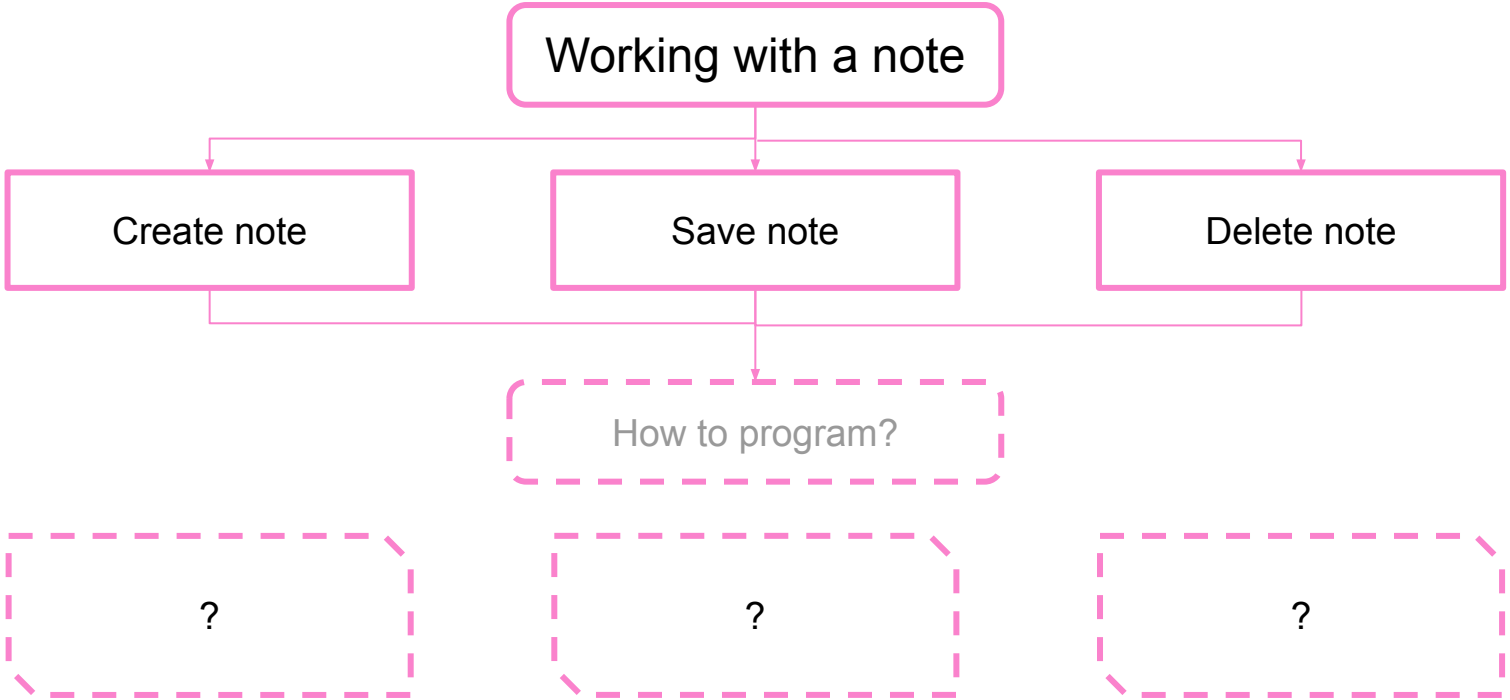
Brainstorming

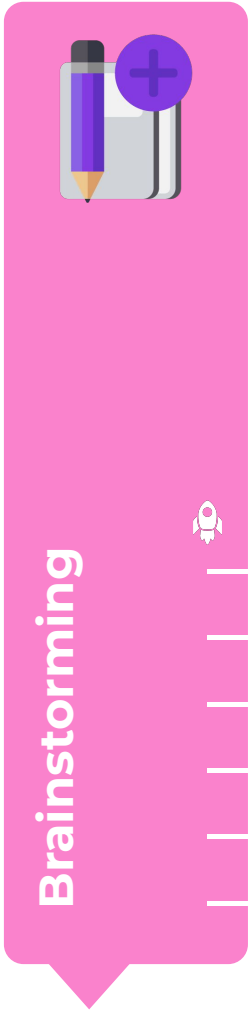
Current tasks:



Brainstorming

Current tasks:





Current tasks:

A button is clicked...

“Create note”

The widget `button_add_note` calls the function `add_note`

1.

“Save note”

The widget `button_save_note` calls the function `save_note`

2.

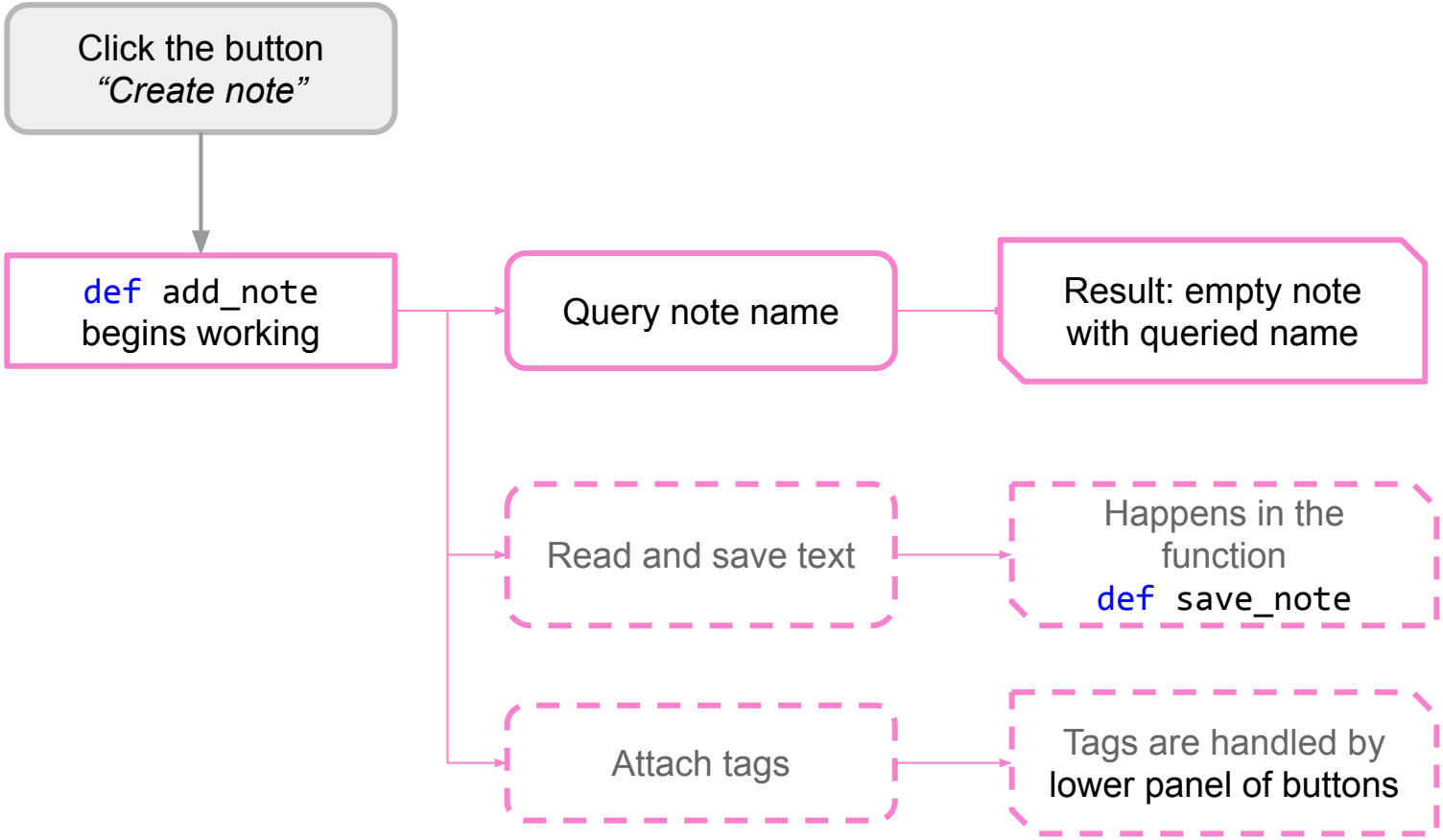
“Delete note”

The widget `button_del_note` calls the function `del_note`

3.

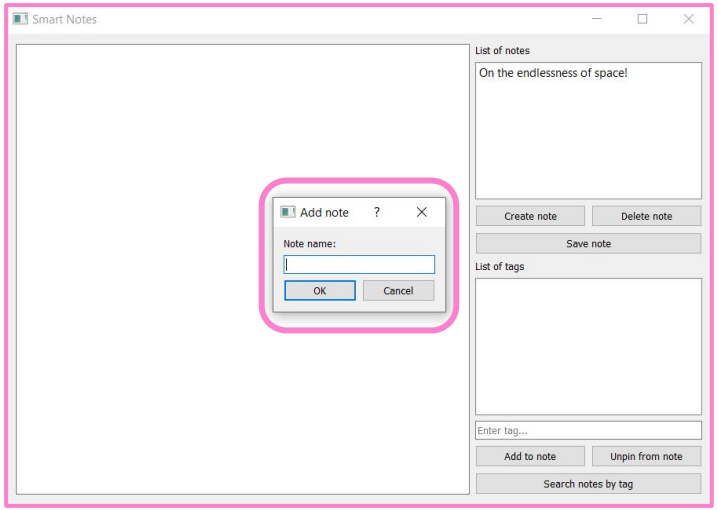
Brainstorming

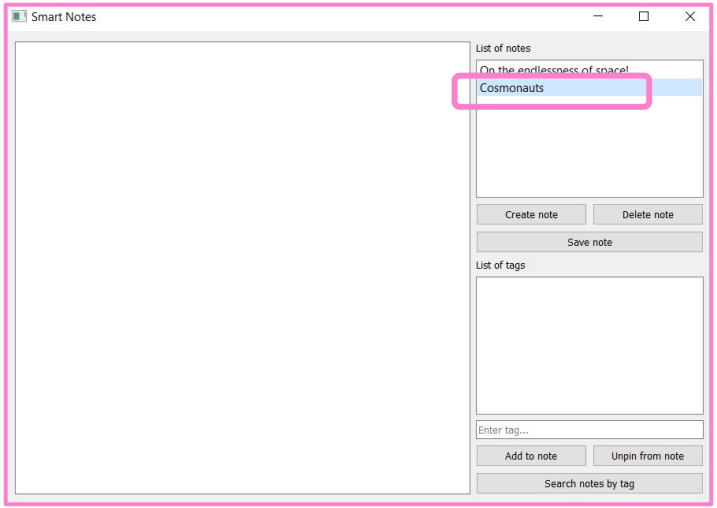
1. Let’s look at creating a note:



Brainstorming

The add_note function:





1. The name of the note is queried in a separate window `QInputDialog`.

2. An empty note with the queried name is created and appears in the list of notes.

Brainstorming

The QInputDialog window:

Command	*Purpose*
`from PyQt5.QtWidgets import QInputDialog`	Import a widget
`note_name, ok = QInputDialog.getText(...)`	Create a QInputDialog window with the name note_name and read the text from the entry field

Note:
QInputDialog is similar to a widget you already know: QMessageBox. The new widget not only shows the window with the text, but prompts the user to Enter text.

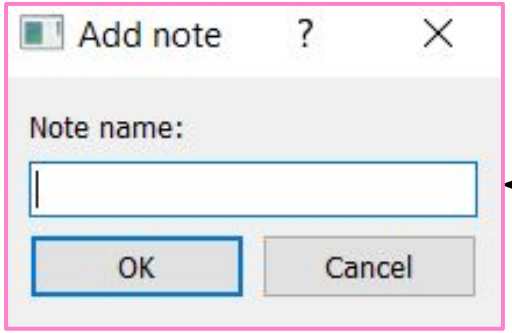
Brainstorming

The QInputDialog window:

Example:

```
note_name, result = QInputDialog.getText(
    notes_win, "Add note", "Note name:"
)
```

Result:



The input text will end up in the variable `result`

Brainstorming

We need to program:

```
def add_note():
    '''Asks for the name of the new note and
    creates an empty note with this name'''
```

- **asks for the name of the note** through the `QInputDialog` window.
- **creates a notes element** with the given name. The text and tags for this note are empty.
- the **name** of the new note is **displayed in the** `list_notes` **list.**

Brainstorming

We need to program:

```
def add_note():
    note_name, ok = QInputDialog.getText(
         notes_win, "Add note", "Note name: "
    )
    if ok and note_name != "":
        notes[note_name] = {"text" : "", "tags" : []}
        list_notes.addItem(note_name)
        #list_tags.addItems(notes[note_name]["tags"])
```

Exploring
a new topic

2. Let's consider deleting a note:

Select a note from the list_notes list

The button *"Delete note"* is clicked

`def del_note` begins working

Delete a note from data

Delete an element from the notes dictionary

Write updated notes dictionary to file

Delete a note from widgets

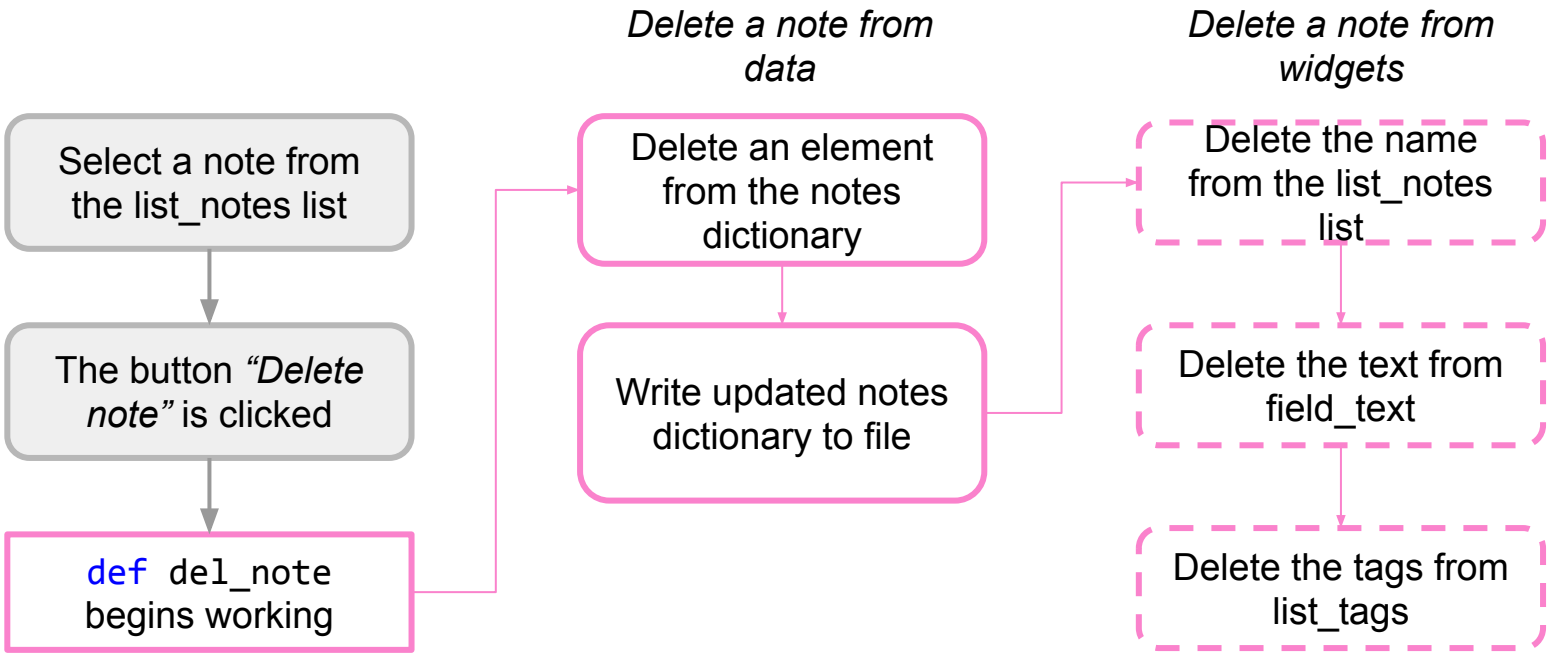
Delete the name from the list_notes list

Delete the text from field_text

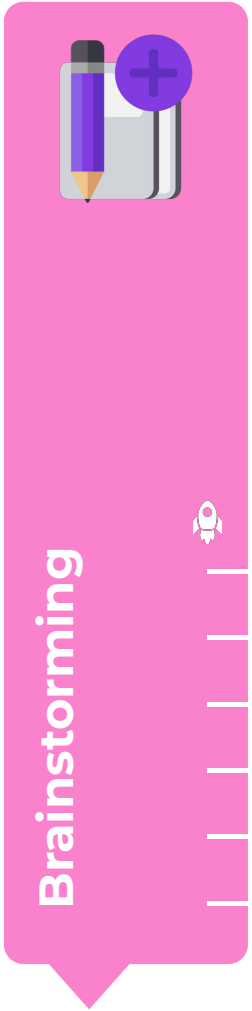
Delete the tags from list_tags

Brainstorming

2. Let's consider deleting a note:



***Note**: if a note to be deleted has not been selected, we can display a message to the console: "No note selected!"*



We need to program:

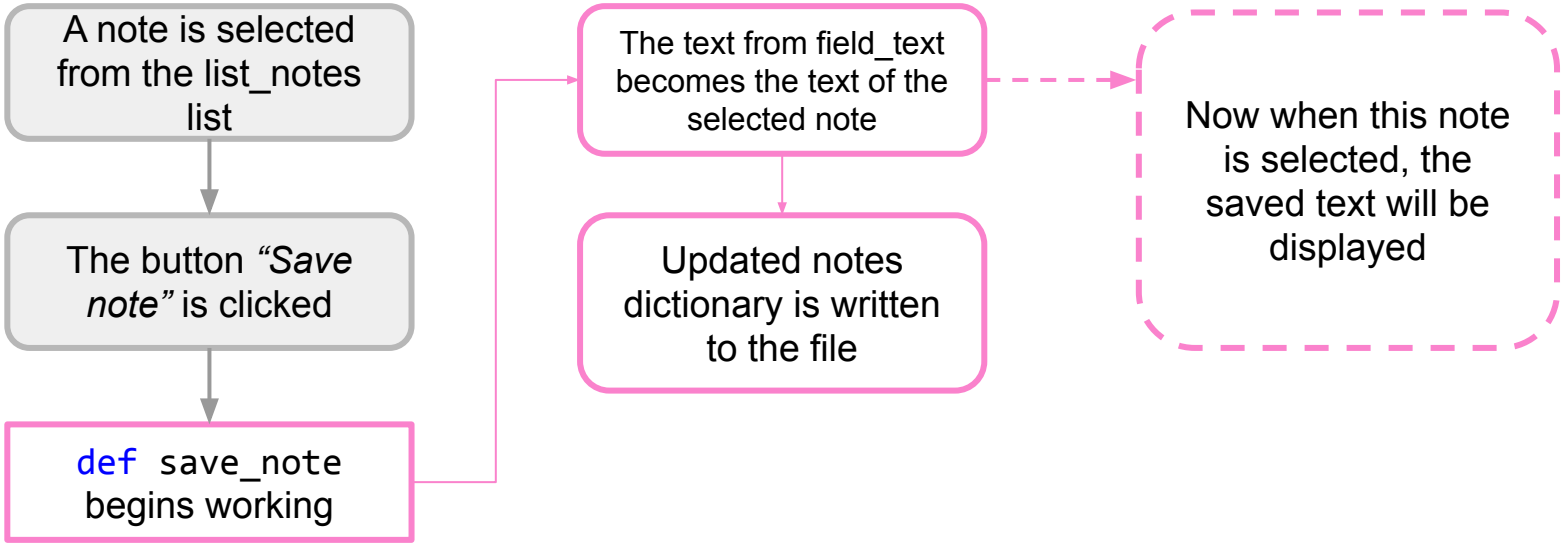
```
def del_note():
    '''Deletes the selected note from the notes
    dictionary, from the file and widgets.'''
```

If a note is selected from list_notes, **then:**

- **remember the name** of the selected note;
- **delete** the note **from the** notes **dictionary**;
- **rewrite** notes to the **file** with the data;
- **delete** the data about the note **from the widgets** (from the list of names, from the list of tags, from the text fields).

Brainstorming

3. Let’s consider saving a note:



***Note**: if a note to be saved has not been selected, we can display a message to the console: “No note selected!”.*

Brainstorming

3. Let’s consider saving a note:

A note is selected from the list_notes list

The button *“Save note”* is clicked

`def save_note` begins working

Save the data

The text from field_text becomes the text of the selected note

Updated notes dictionary is written to the file

Display on interface

Now when this note is selected, the saved text will be displayed

Note:
the `field_text.toPlainText()` method returns entered text from the field_text field

Brainstorming

We need to program:

```
def save_note():
    '''Saves text to selected note from the notes
    dictionary and updates data file'''
```

If a note is selected from list_notes, **then:**

- **get text from the widget** field_text (toPlainText);
- **record the text in the** notes **note** with the selected name;
- **rewrite** notes to the data **file**.

Brainstorming

Tasks:

- Create functions for creating, saving and deleting notes.
- Handle clicks of the buttons “Create note”, “Save note” and “Delete note” using these functions.

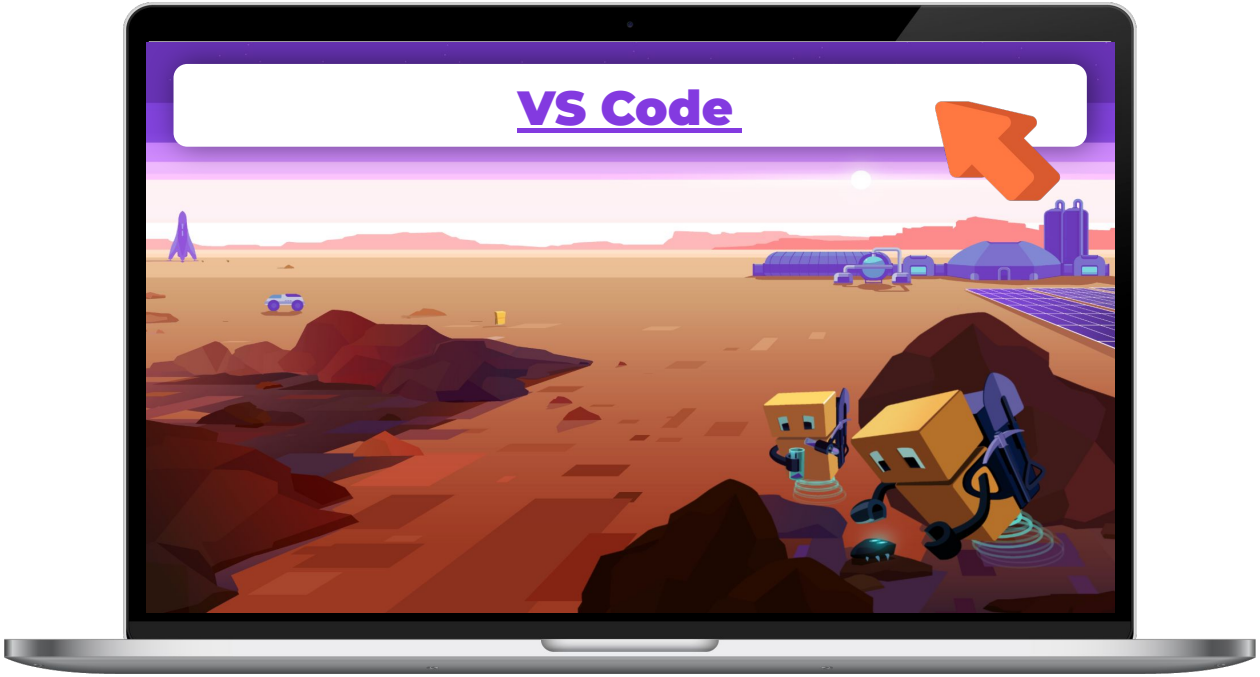
Brainstorming

Visual Studio Code:

The Smart Notes Application

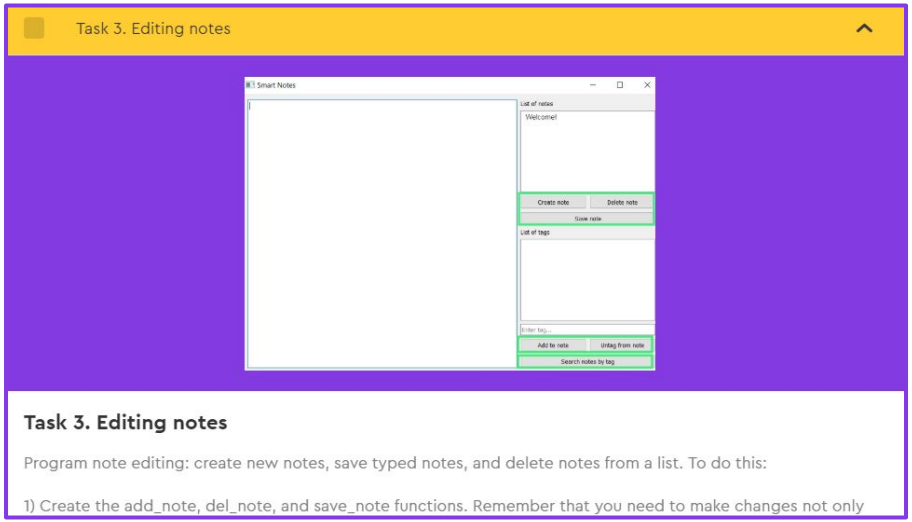
Do the task in VS Code

“VSC. The Smart Notes Application”



Do the tasks in VS Code

“VSC. The Smart Notes Application”

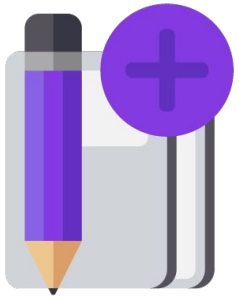


Complete “Task 3. Editing notes”.

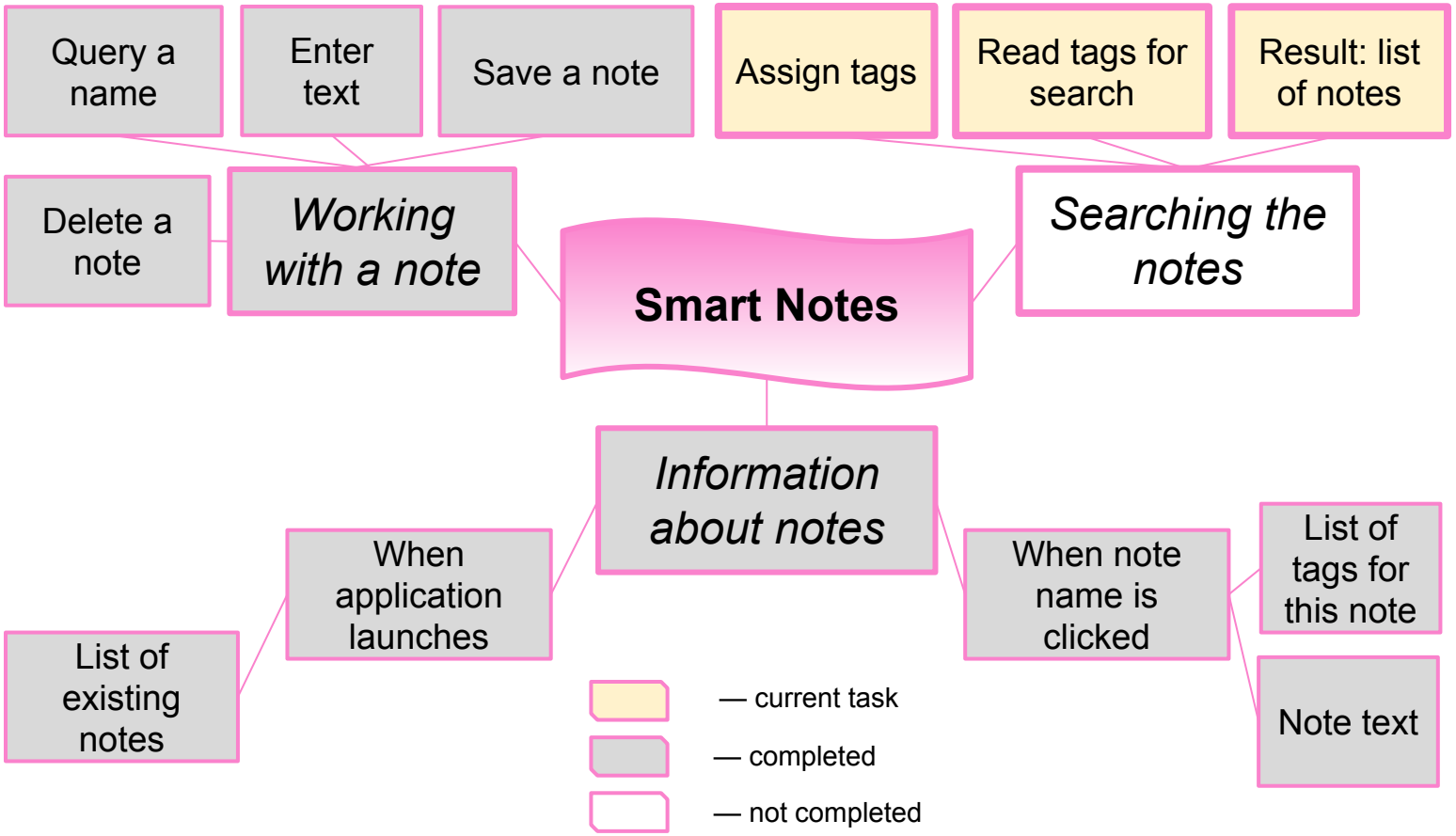
Break

Brainstorming:

Working with tags in Smart Notes



Current tasks:



Brainstorming

Current tasks:

Searching for notes by tag

- Attaching a tag to a note
- Unclipping a tag from a note
- Searching for notes with the given tag

How to program?

- ?
- ?
- ?

Brainstorming

Current tasks:

Searching for notes by tag

- Attaching a tag to a note
  - The `add_tag` function *adds* the tag to the list of tags for this note and updates the data file
- Unclipping a tag from a note
  - The `del_tag` function *deletes* the tag from the list of tags for this note and updates the data file
- Searching for notes with the given tag
  - The `search_tag` function leaves *only the notes with the given tag* in the list

Brainstorming

Current tasks:

Searching for notes by tag

- Attaching a tag to a note
  - The `add_tag` function *adds* the tag to the list of tags for this note and updates the data file
- Unclipping a tag from a note
  - The `del_tag` function *deletes* the tag from the list of tags for this note and updates the data file
- Searching for notes with the given tag
  - The `search_tag` function leaves *only the notes with the given tag* in the list

***Note**: these functions are handler functions for a click of the widget button "Add to note" and others.*

"Brainstorm"

1. Let's consider adding a tag:

A note is selected in list_notes

There's a tag in the field (not empty)

The "*Add to note*" button is clicked

`def add_tag` begins working

Adding the tag to data

The tag is recorded in the list of tags for this note

Updated notes dictionary is written to the data file

Adding the tag to interface

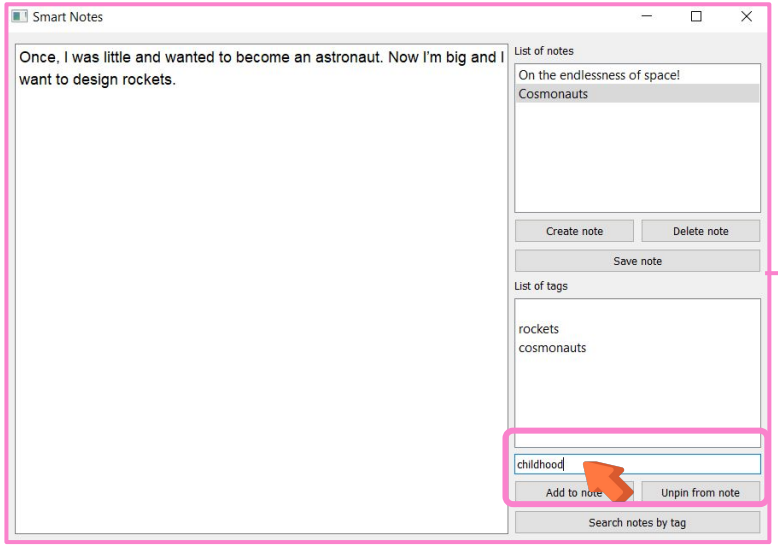
Data for widget with list of tags is updated

Entry field for new tags is cleared

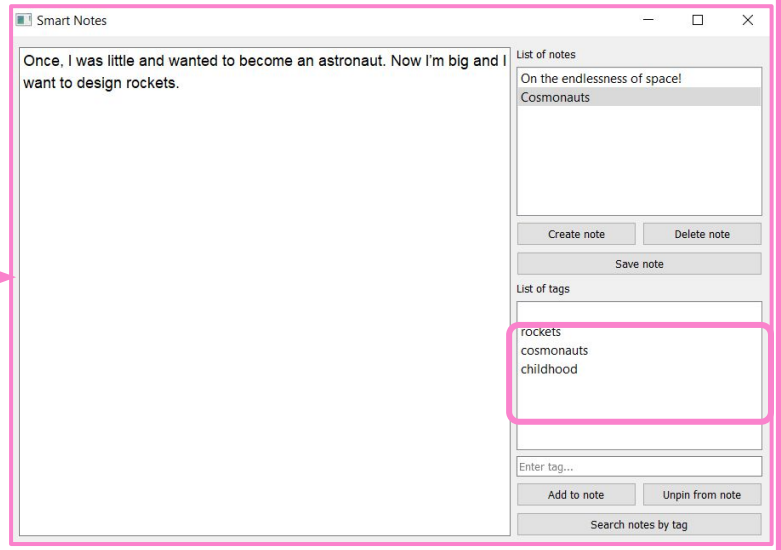
Brainstorming

1. Let’s consider adding a tag:

Before



After



When “Add to note” is clicked, the entered tag is displayed in the list of tags. The tag entry field is cleared so new data can be entered.

Brainstorming

We need to program:

```
def add_tag():
    '''Adds the entered tag to the list of tags
    for the selected note'''
```

If a note is selected in list_notes, then:

- **read the tag** from the field_tag field.

If there is no such tag among the note's tags, then:

- add **tag to** `notes[note]["tags"]`;
- add **tag to the** tag list **widget**;
- clear the tag entry field;
- **write the** notes **dictionary** to the file again.

Brainstorming

We need to program:

```
def add_tag():
    if list_notes.selectedItems():
        key = list_notes.selectedItems()[0].text()
        tag = field_tag.text()
        if not tag in notes[key]["tags"]:
            notes[key]["tags"].append(tag)
            list_tags.addItem(tag)
            field_tag.clear()
        with open("notes_data.json", "w") as file:
            json.dump(notes, file, sort_keys=True)
    else:
        print("No note selected to add tag!")
```

Exploring a new topic

2. Let’s consider deleting a tag:

A note is selected in list_notes

A tag is selected in the list of tags

The “*Unclip from tag*” button is clicked

`def del_tag` begins working

Deleting the tag from data

The tag is deleted from the list of tags for this note

Updated notes dictionary is written into the data file

Deleting the tag from interface

Data for widget with list of tags is updated

Brainstorming

We need to program:

```
def del_tag():
    '''Deletes selected tag from list of tags
    for selected note'''
```

If a note is selected in list_notes:

- **delete the tag** selected in the list of tags from the widget `list_tags`;
- delete this tag from the list of tags for the current note `notes[note]["tags"]`;
- rewrite the updated notes dictionary to the data file.

Brainstorming

3. Let's consider searching by tag:

Tag for search is entered in the field

The "*Search by tag*" button is clicked

`def search_tag` begins working

Searching for notes by the tag

Search for notes with the entered tag

Create the dictionary `notes_filtered` with search results

Display `notes_filtered` in `list_notes` instead of the full `notes` list

Clearing results for a new search

Clear the tag entry field

Display full list of notes in list_notes

Brainstorming

3. Let's consider searching by tag:

Tag for search is entered in the field

The "*Search by tag*" button is clicked

`def search_tag` begins working

Searching for notes by tag

Search for notes with the entered tag

Create the dictionary `notes_filtered` with search results

Display `notes_filtered` in `list_notes` instead of the full `notes` list

Clearing results for new search

Clear tag entry field

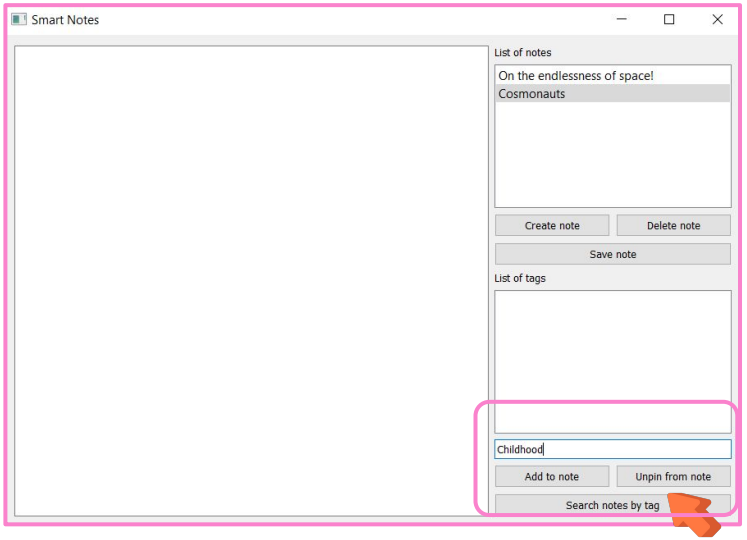
Display full list of notes in list_notes

How do we prompt the user to clear the search?

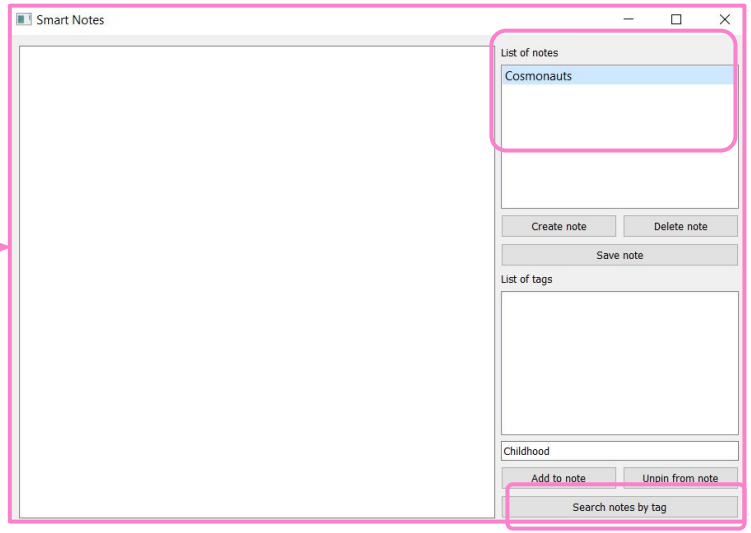
Brainstorming

3. Let’s consider searching by tag:

Before



After



After searching for notes by tag, the user is prompted to clear the search results (clear the entry field and display the full list of notes).

Brainstorming

3. Let's consider searching by tag:

Label on the button changes to *"Search by tag"*

Tag for search is entered into field

The button *"Search by tag"* is clicked

`def search_tag` begins working

Searching for notes by tag

Search for notes with selected tag

The dictionary `notes_filtered` is created with the search results

Results displayed in `list_notes` (instead of the full `notes` list)

Clearing results for new search

Label on the button changes to *"Clear search"*

The button *"Clear search"* is clicked

Tag entry field is cleared

Full list of notes is displayed in list_notes

Brainstorming

We need to program:

```
def search_tag():
    '''Creates a list of notes that have the
    selected tag'''
```

If the “Search by tag” button is clicked:

- **read the tag** from the entry field;
- **select from the `notes` dictionary** only notes with the entered tag (`notes_filtered`);
- **display the resulting dictionary** in `list_notes`;
- change the name of the button to “Clear search”.

Brainstorming

We need to program:

```
def search_tag():
    '''Creates a list of notes that have the
    selected tag'''
```

If the “Clear search” button is clicked:

- **clear** the entry field;
- **show the full** notes **dictionary** in list_notes;
- change the name of the button to “Search by tag”.

Brainstorming

```python
def search_tag():
    tag = field_tag.text()
    if button_tag_search.text() == "Search for notes by tag" and tag:
        notes_filtered = {} #notes with selected tag will be here
        for note in notes:
            if tag in notes[note]["tags"]:
                notes_filtered[note]=notes[note]
        button_tag_search.setText("Clear search")
        list_notes.clear()
        list_tags.clear()
        list_notes.addItems(notes_filtered)
    elif button_tag_search.text() == "Clear search":
        field_tag.clear()
        list_notes.clear()
        list_tags.clear()
        list_notes.addItems(notes)
        button_tag_search.setText("Search for notes by tag")
    else:
        pass
```

Exploring a new topic

Tasks:

- Create functions for adding and deleting tags, and for searching for notes by tag.

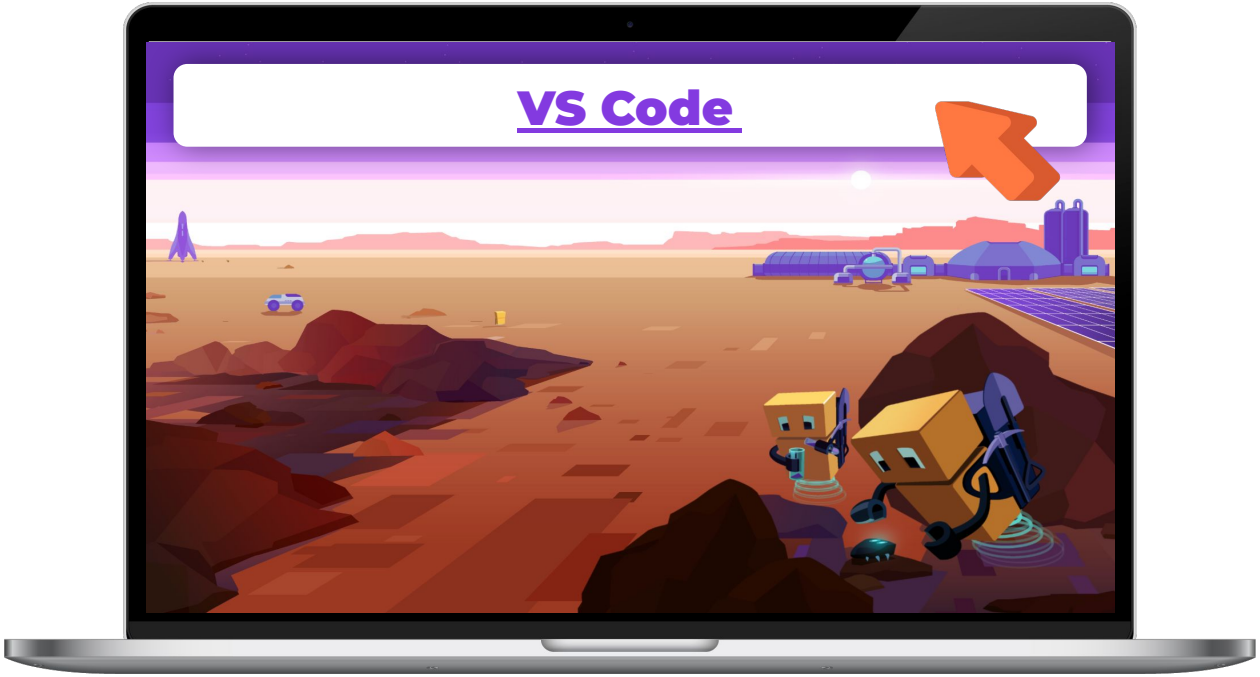
- Handle clicks of the buttons “Add to note”, “Unclip from note” and “Search for note by tag” using these functions.

Brainstorming

Working in VS Code

Complete the task in VS Code

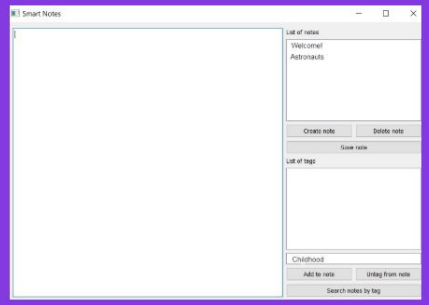
“VSC. The Smart Notes Application”



Do the task in VS Code

“VSC. The Smart Notes Application”

Task 4. Working with tags



Task 4. Working with tags

Program note searching by tag. To do this, create the search_note function. Implement branching in the function:

1) If the "Search by tag" button is clicked, notes with this tag should be executed. The result—a dictionary of filtered notes—should appear instead of the full dictionary in list_notes. After the search, the name of the button should change to "Reset Search."
2) If the "Reset Search" button is clicked, the tag input field should be cleared and the full notes dictionary

Complete “Task 4. Searching for notes by tag.”

Wrapping up the work day

Additional task:

Create two notes about outer space
In Smart Notes and
attach tags to them.

Test your application independently or with a partner:

- *add one more note,*
- *delete it,*
- *search by tag.*