algorithmics

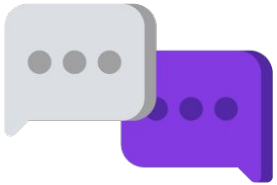
Module 2. Lesson 4.

Memory Card Application P.2

Methodological guidelines

Discussion:

Memory Card Application



Let’s keep working on that big job

Last time, the ProTeam developers were hired by the “Citizen of the World” Cultural Center.

To sharpen their specialists’ knowledge of cultures and languages all over the world, the Center has asked us to create a **Memory Card application**.

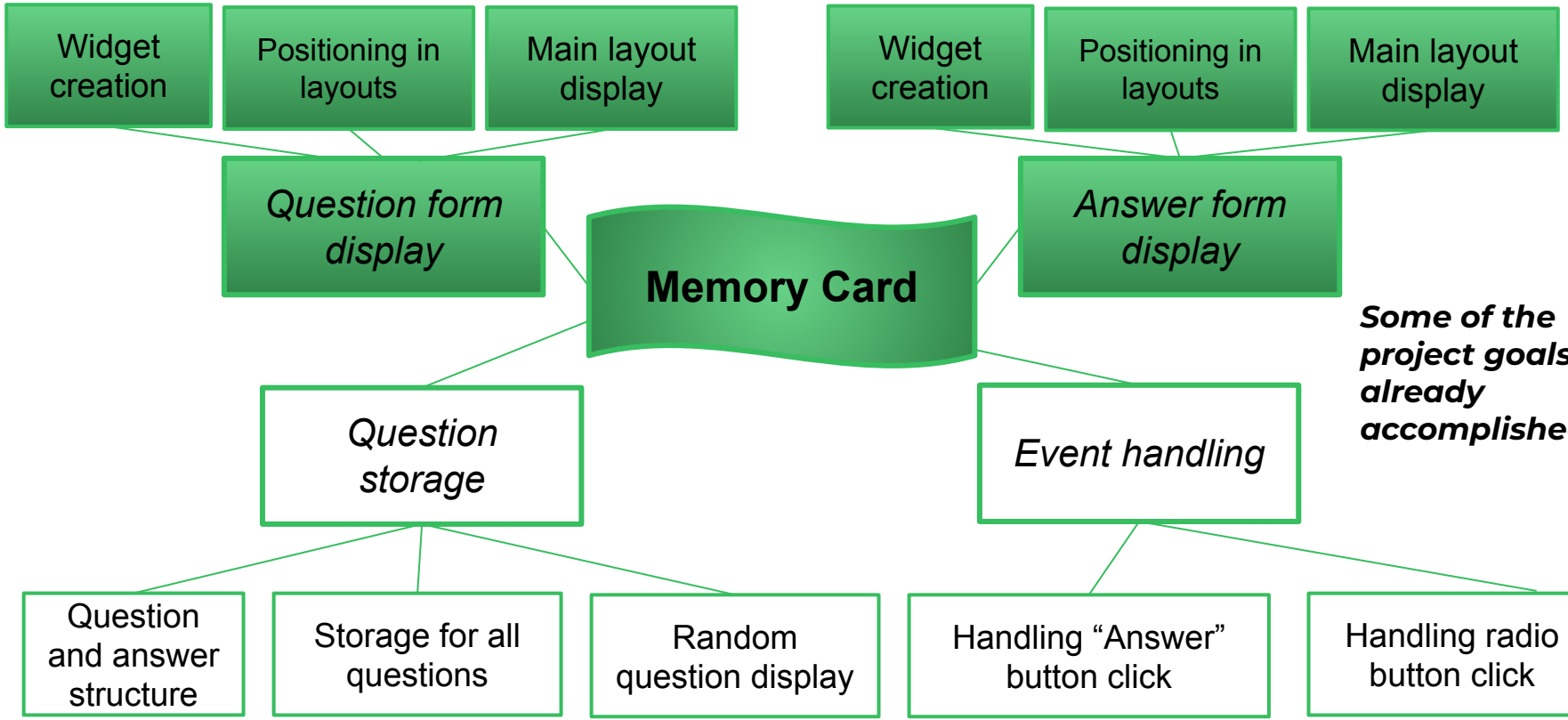
We’ve already planned our work and programmed the basic interface for this application.

Ready to continue our work?

Emily,
Project Manager

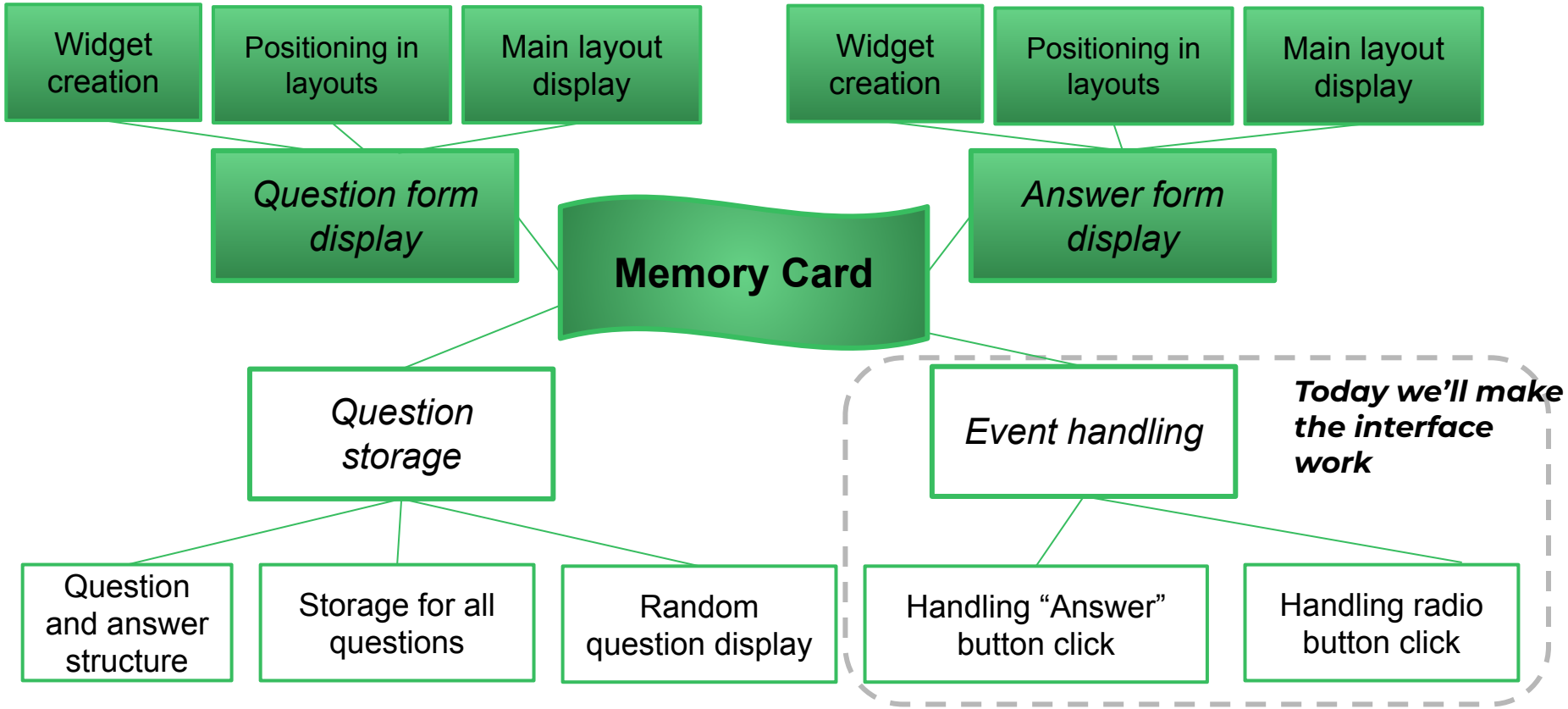
Discussion of tasks

Let’s look at our project’s mind map



Some of the project goals are already accomplished

Let’s look at our project’s mind map



The goal of this workday is:

program event handling for the Memory Card application.

Today you will :

- review what a group of widgets is, and several parameters for setting a widget's location
- learn the specifics of handling radio buttons
- program event handling with specially made handler functions!

Discussion of tasks

Qualification

Show your knowledge of the PyQt library

Qualification

How do you <u>create</u> an application?
What widgets do you know?

Qualification

PyQt5 is

a cross-platform library for creating windowed applications.

```
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QLabel, QVBoxLayout
```

Object	*Designation*
Application	`QApplication`
Application window	`QWidget`
Label	`QLabel`
Button	`QPushButton`
Vertical guide line	`QVBoxLayout`

Qualification

What is event handling ?
How do we handle a mouse event ?

Qualification

Event handling (button click)

1. *Describe the operations* of an individual handler function when a button is clicked.
2. *Apply a command* for linking the function to the widget.

```
def show_result():
```

```
button.clicked.connect(show_result)
```

THE ACTUAL FUNCTION IS WRITTEN SEPARATELY

NAME OF THE HANDLER FUNCTION CALLED BY THE BUTTON

Proficiency Testing

How do you create a widget group?

How do you position the widgets inside the group?

Qualification

To create a group of widgets and assign widget positions inside the group:

- ❑ Create a QGroupBox group (from QtWidgets).
- ❑ Position the relevant widgets in the layouts separately
- ❑ Set the main widget layout in the group.

If the widgets we need have already been created, then:

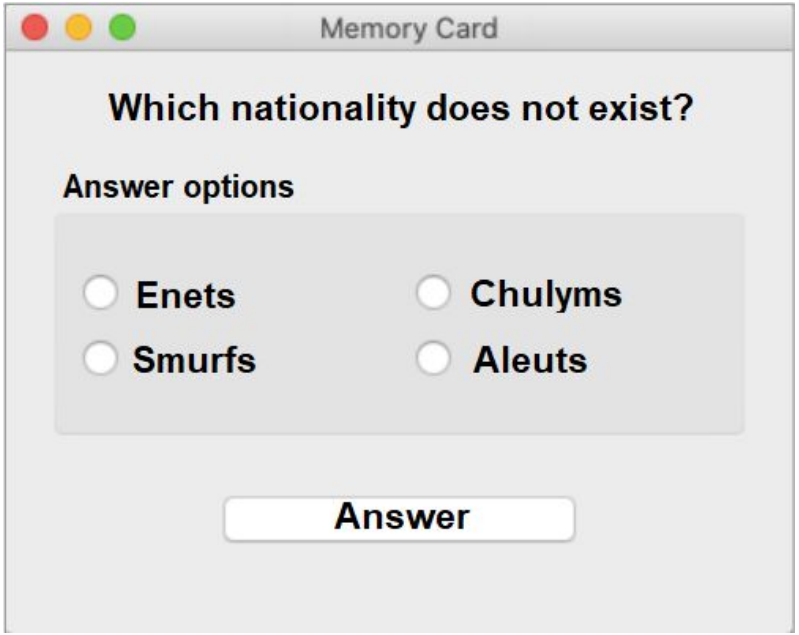
`RadioGroupBox = QGroupBox(‘Options’)`	A constructor for creating the group

`rbtn_1 = QRadioButton('Enets')`
`layout_quest = QHBoxLayout()`
`layout_quest.addWidget(rbtn_1)`
	Create radio button,
	Create layout line,
	Add radio button to it

`RadioGroupBox.setLayout(layout_quest)`	Create main layout for the group

Qualification

```
RadioGroupBox = QGroupBox("Answer options")
rbtn_1 = QRadioButton('Enets')
rbtn_2 = QRadioButton('Smurfs')
rbtn_3 = QRadioButton('Chulyms')
rbtn_4 = QRadioButton('Aleuts')
layout_ans1 = QHBoxLayout()
layout_ans2 = QVBoxLayout()
layout_ans3 = QVBoxLayout()

layout_ans2.addWidget(rbtn_1)
layout_ans2.addWidget(rbtn_2)
layout_ans3.addWidget(rbtn_3)
layout_ans3.addWidget(rbtn_4)
layout_ans1.addLayout(layout_ans2)
layout_ans1.addLayout(layout_ans3)

RadioGroupBox.setLayout(layout_ans1)
```



Qualification

What additional parameters for positioning widgets do you know?

Qualification

Parameters for positioning widgets

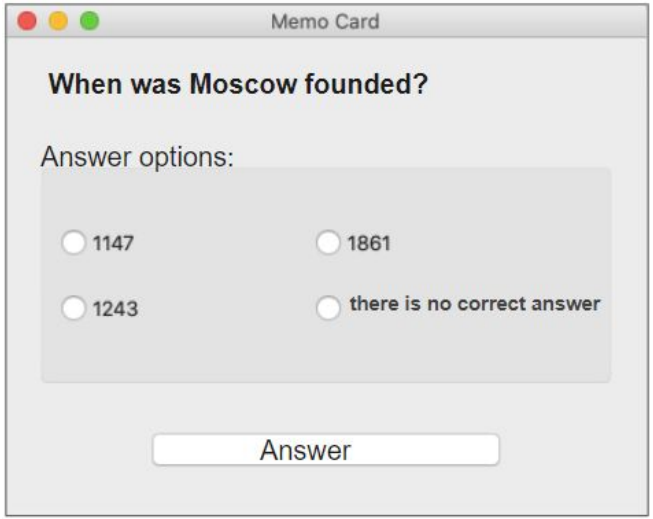
Command	*Designation*

`alignment=Qt.AlignHCenter`
 	
`alignment=Qt.AlignVCenter`
	Centering
	(horizontally)
	Centering
	(vertically)

`stretch=2`	Stretch the widget (for example, a button)
`layout_card.setSpacing(5)`	Set spacing between contents of layout (for example, between horizontal lines)

Qualification

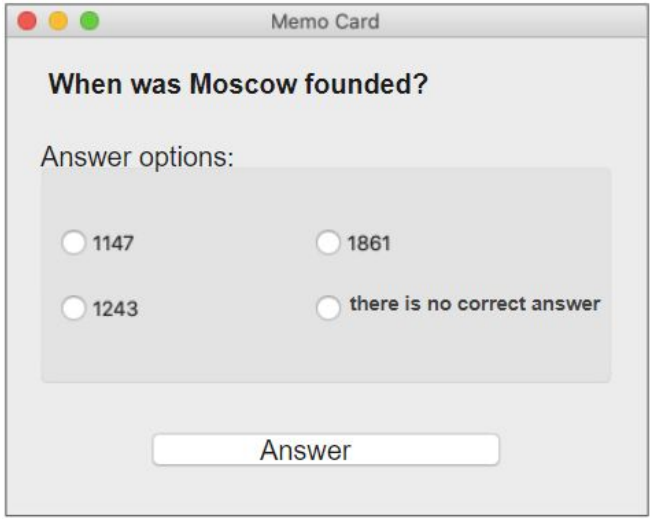
How do you stretch the “Answer” button?



Qualification

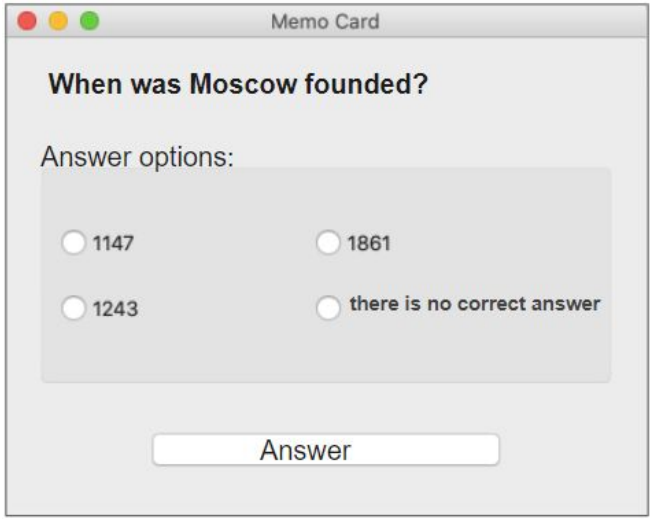
The Stretch parameter

`layout_line3.addWidget(btn_OK, stretch=3)`	Stretch the widget button three times



Qualification

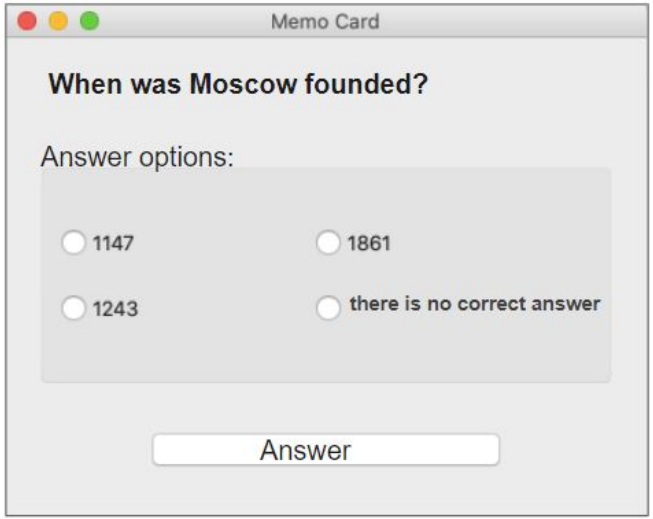
How do we align the question to the left?



Qualification

The Alignment parameter:

`layout_line1.addWidget(lb_Question, alignment=Qt.AlignLeft)`	Align left



Qualification

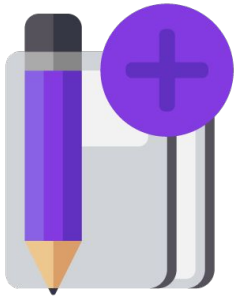
Qualification confirmed!

Excellent, you're ready for brainstorming and the task ahead!

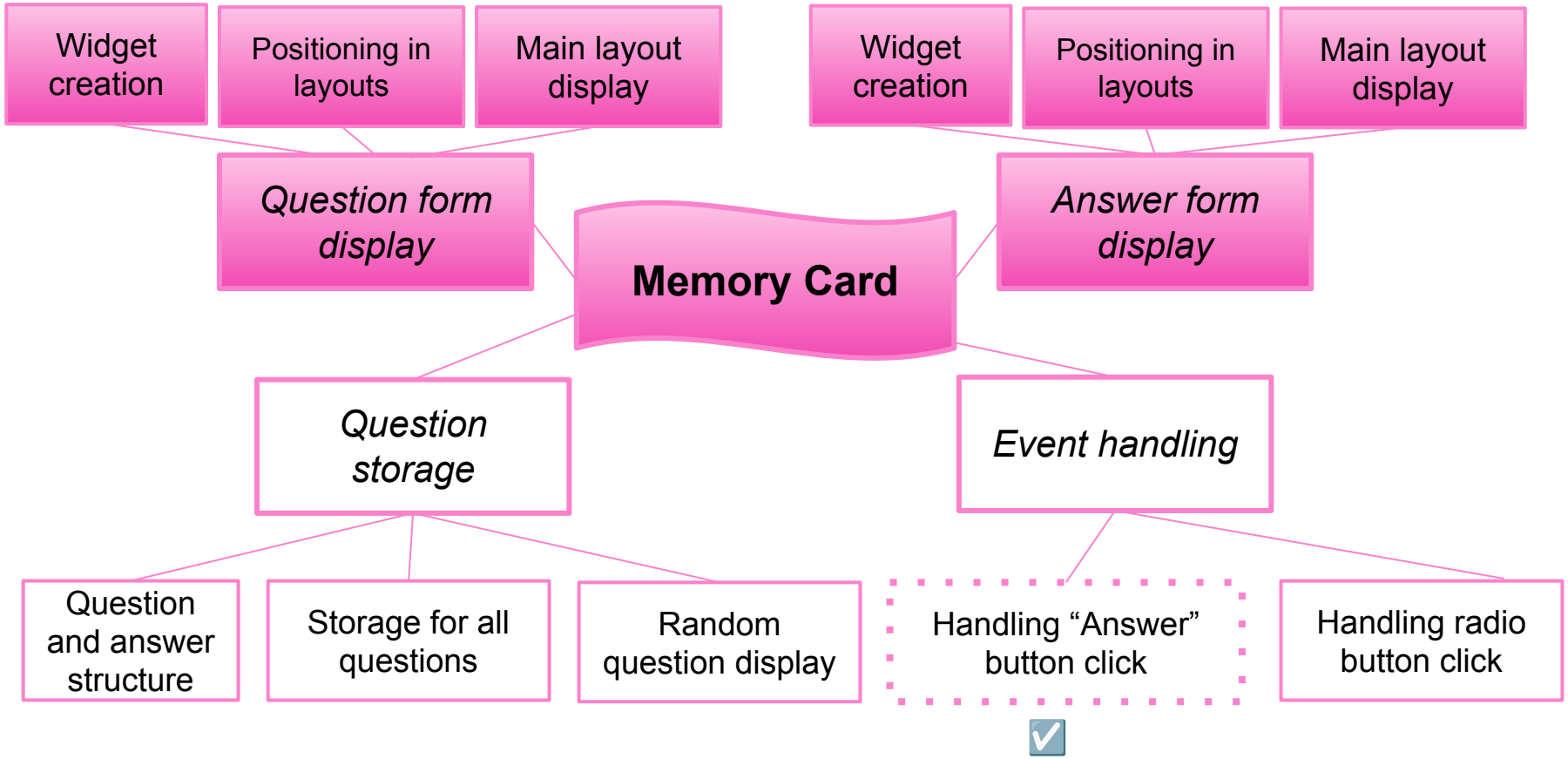
Qualification

Brainstorming

Event handling



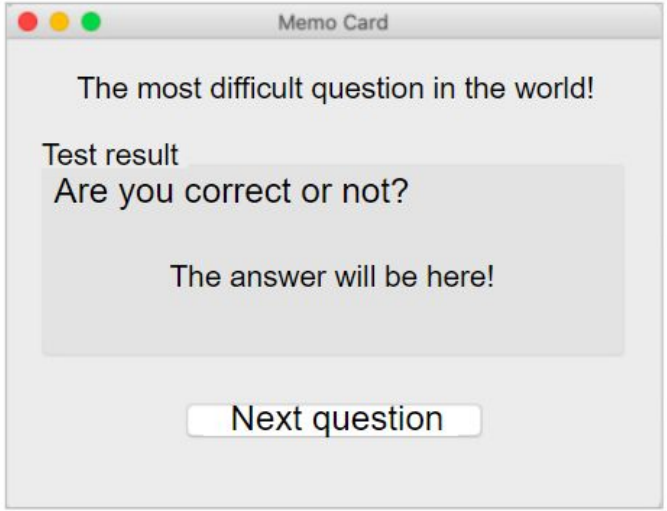
Project mind map:



1. Displaying the <u>answer to a question</u>

Clicking “Answer” in the question form:

- ❑ ?
- ❑ ?
- ❑ ?



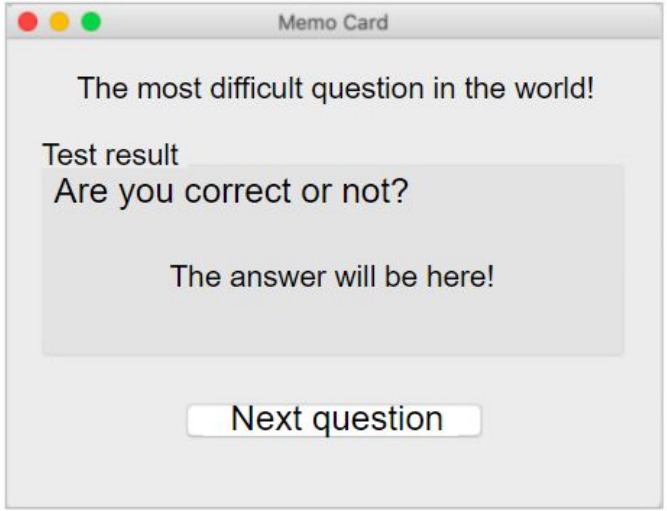
Which parts of the interface must be changed? How do we change them?

Brainstorming

1. Displaying the <u>answer to a question</u>

Clicking “Answer” in the question form:

- ❑ hide question form
- ❑ show answer form
- ❑ Change label from “Answer” to “Next question”



How do we program this?

Brainstorming

1. Displaying the answer to a question

```
def show_result():
```

- ❑ hide question form,
- ❑ show answer form,
- ❑ Change label from “Answer” to “Next question”

```
#...

btn_OK.clicked.connect(show_result)
```

The handler function that displays the answer form.

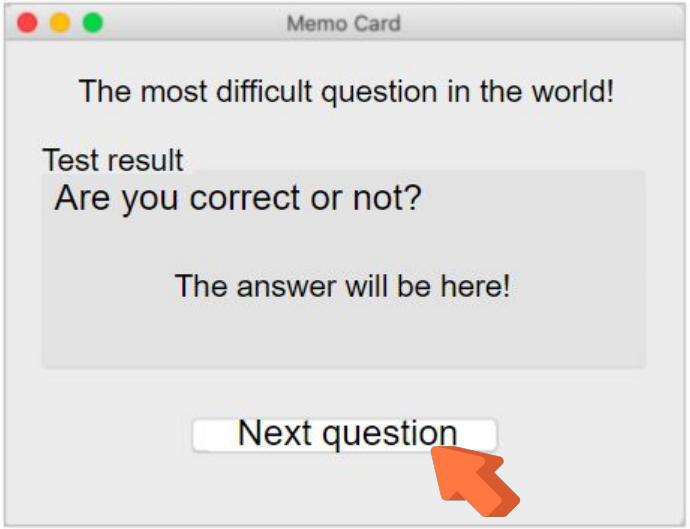
Calling the function when the “Answer” button is clicked.

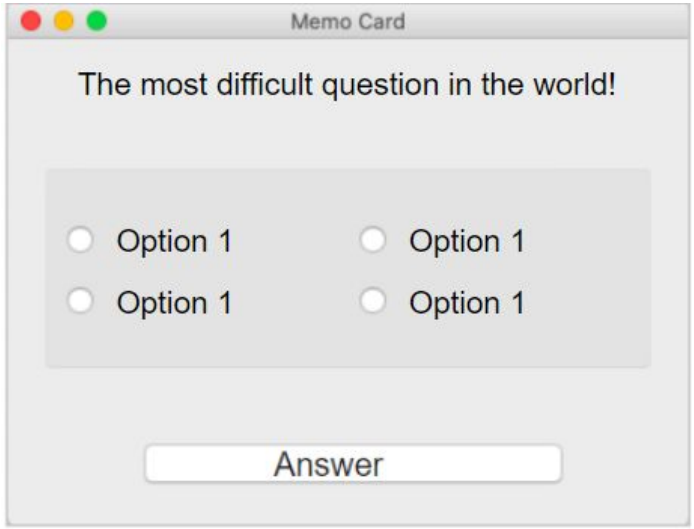
Brainstorming

2. Displaying the question and answer options

Answer form, click “Next question”:

- ❑ ?
- ❑ ?
- ❑ ?



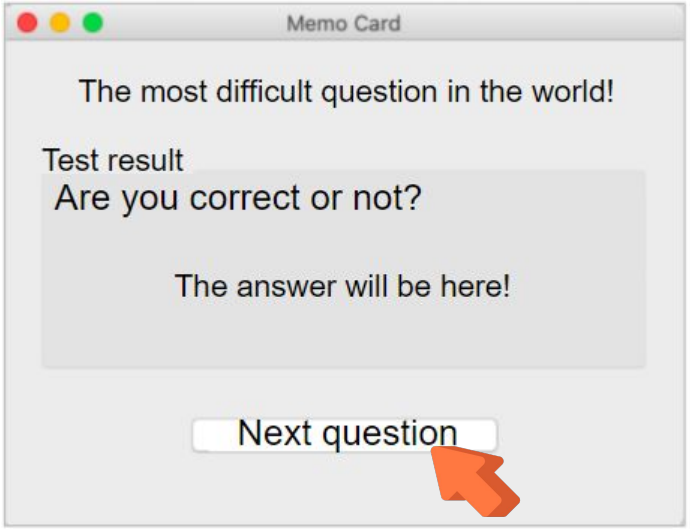


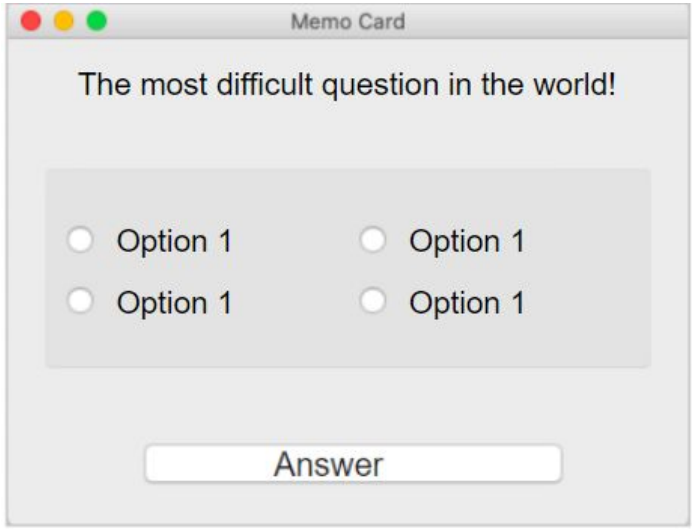
Brainstorming

2. Displaying the question and answer options

Answer form, click “Next question”:

- ❑ hide the answer form,
- ❑ show the question form,
- ❑ change the inscription "Next question" to "Answer".





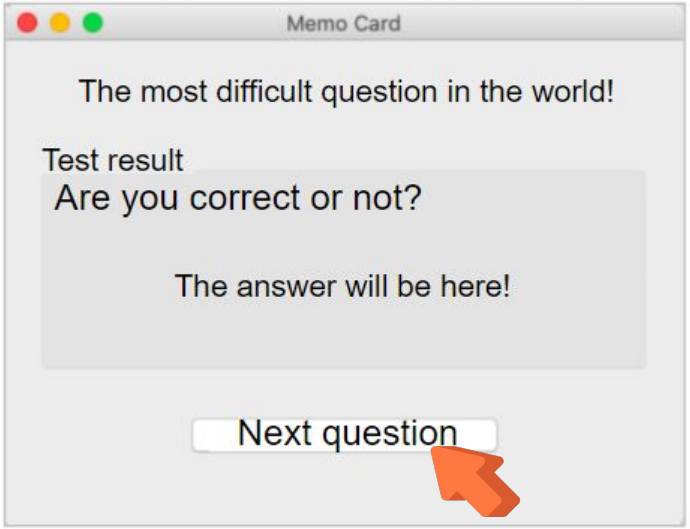
Is it enough?

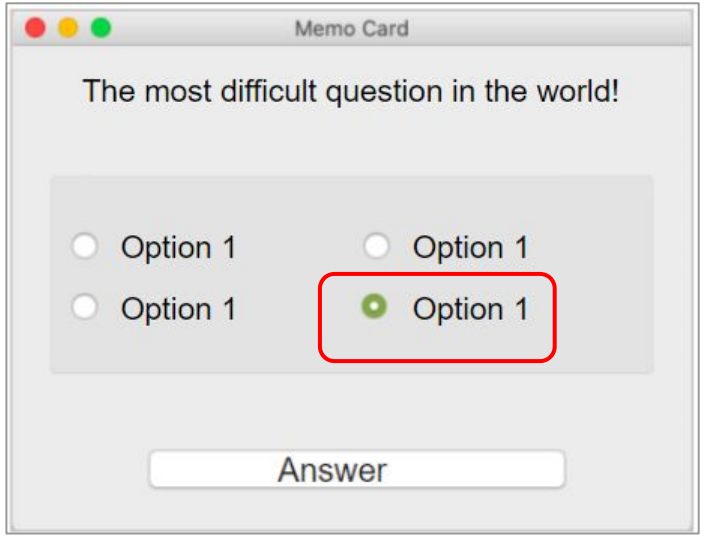
Brainstorming

2. Displaying the question and answer options

Answer form, click “Next question”:

- ❑ hide answer form,
- ❑ show question form,
- ❑ change the label “Next question” to “Answer”





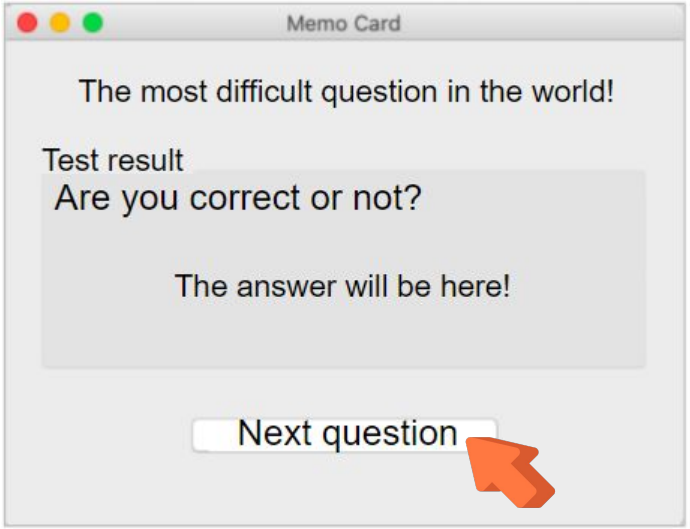
The previous choice will remain!

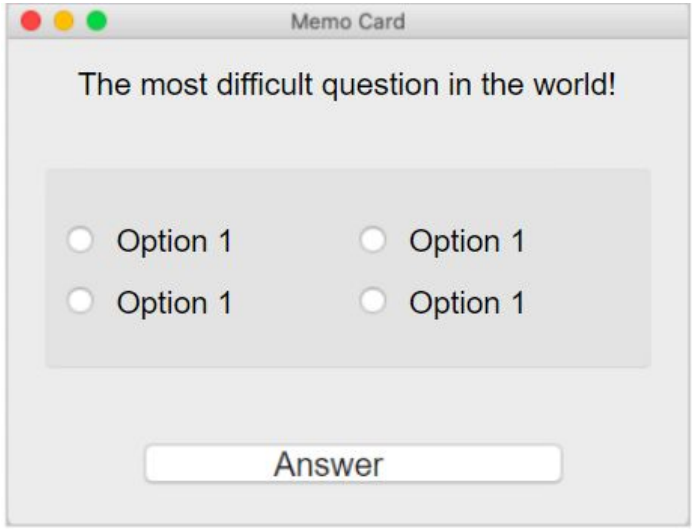
No! The radio buttons won’t actually reset!

Brainstorming

2. Displaying the question and answer options, click “Next question”:

- ❑ hide answer form,
- ❑ show question form,
- ❑ change the label “Next question” to “Answer”
- ❑ reset all radio buttons





How do we program this?

Brainstorming

2. Displaying the question and answer options

```
RadioGroup = QButtonGroup()
RadioGroup.addButton(rbtn_1)
RadioGroup.addButton(rbtn_2)
RadioGroup.addButton(rbtn_3)
RadioGroup.addButton(rbtn_4)
```

interface

We are **uniting** all the radio buttons in a special group.

Now only one of them can be selected at a time.

```
#...

RadioGroup.setExclusive(False)
rbtn_1.setChecked(False)
rbtn_2.setChecked(False)
rbtn_3.setChecked(False)
rbtn_4.setChecked(False)
RadioGroup.setExclusive(True)
```

show_question()

Let's remove the limits for the choice reset.

Reset the choice for all radio buttons.

Bring back the limits.

Brainstorming

2. Displaying the question and answer options

```
def show_question():
```

- ❑ hide answer form,
- ❑ show question form,
- ❑ change the label “Next question” to “Answer”,
- ❑ reset all radio buttons.

```
#...
```

Handler function that displays the question form.

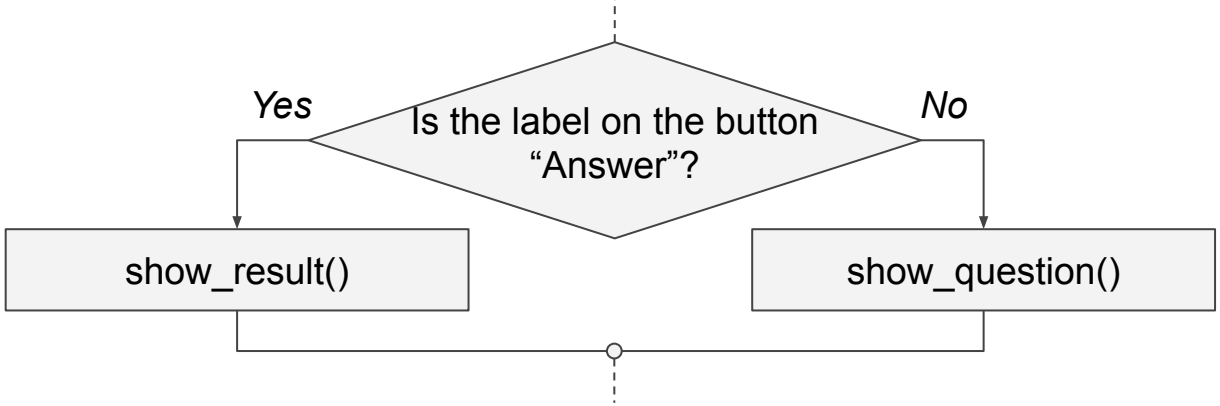
But the button click is already being handled by how_result()!

What do we do?

How do we choose the correct handler function after a click?

Brainstorming

3.Function that chooses the handler function




```
def start_test():
```

- If the label on the button says "Answer," call function show_result().
- Otherwise, call show_question().

Brainstorming

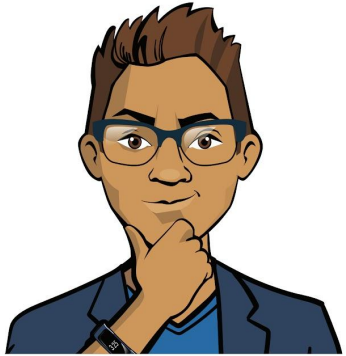
Let’s bring it all together:

```
#uniting the radio buttons into a special group

def show_question():
    Function body

def show_result():
    Function body

def start_test():
    Function body

btn_OK.clicked.connect(start_test)
```

Brainstorming

Before we continue, let's clarify :

1. Why unite the radio buttons into a special group? How should we link it from PyQt?
2. How do we use different functions to handle a click of the same button?
3. What will the program display if you do the following in order:
   - ❑ launch the program,
   - ❑ choose an answer option and click "Answer"
   - ❑ look at the correct answer and click "Next question"?



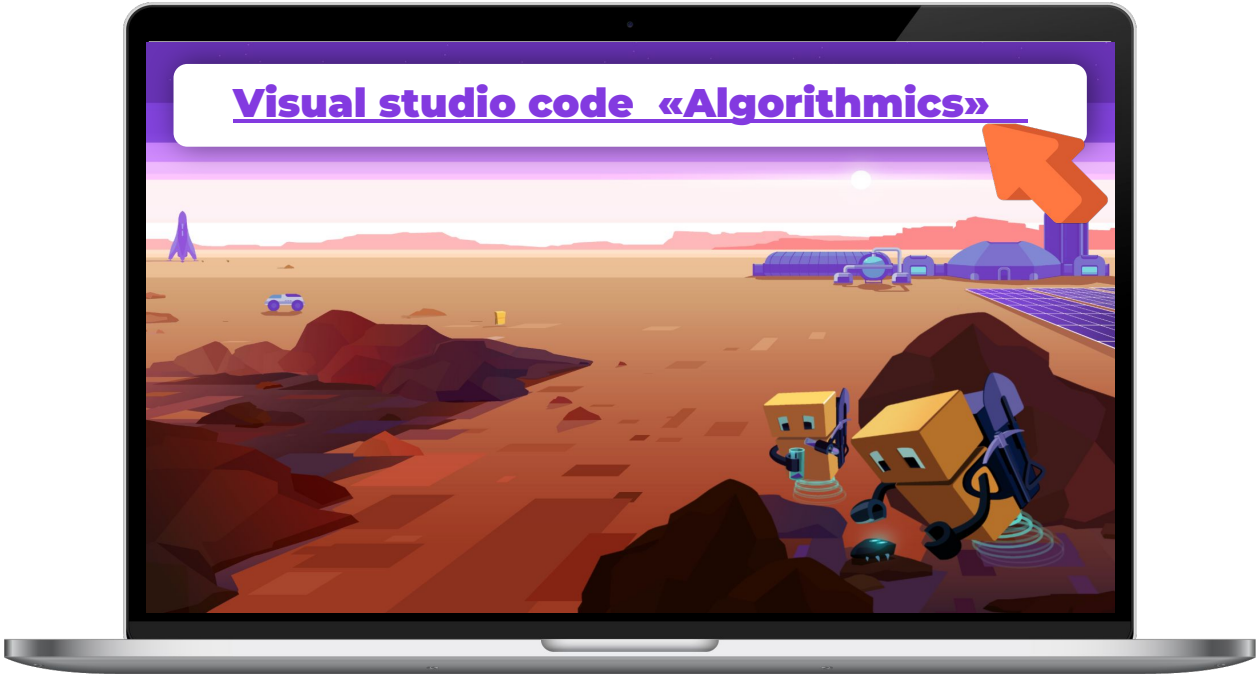
Brainstorming

Visual Studio Code:

Memory Card Application

Complete the task in VS Code

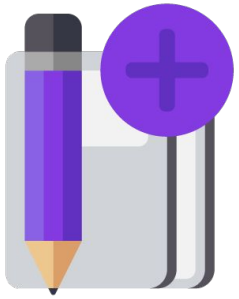
“VSC. PyQt. Memory Card”



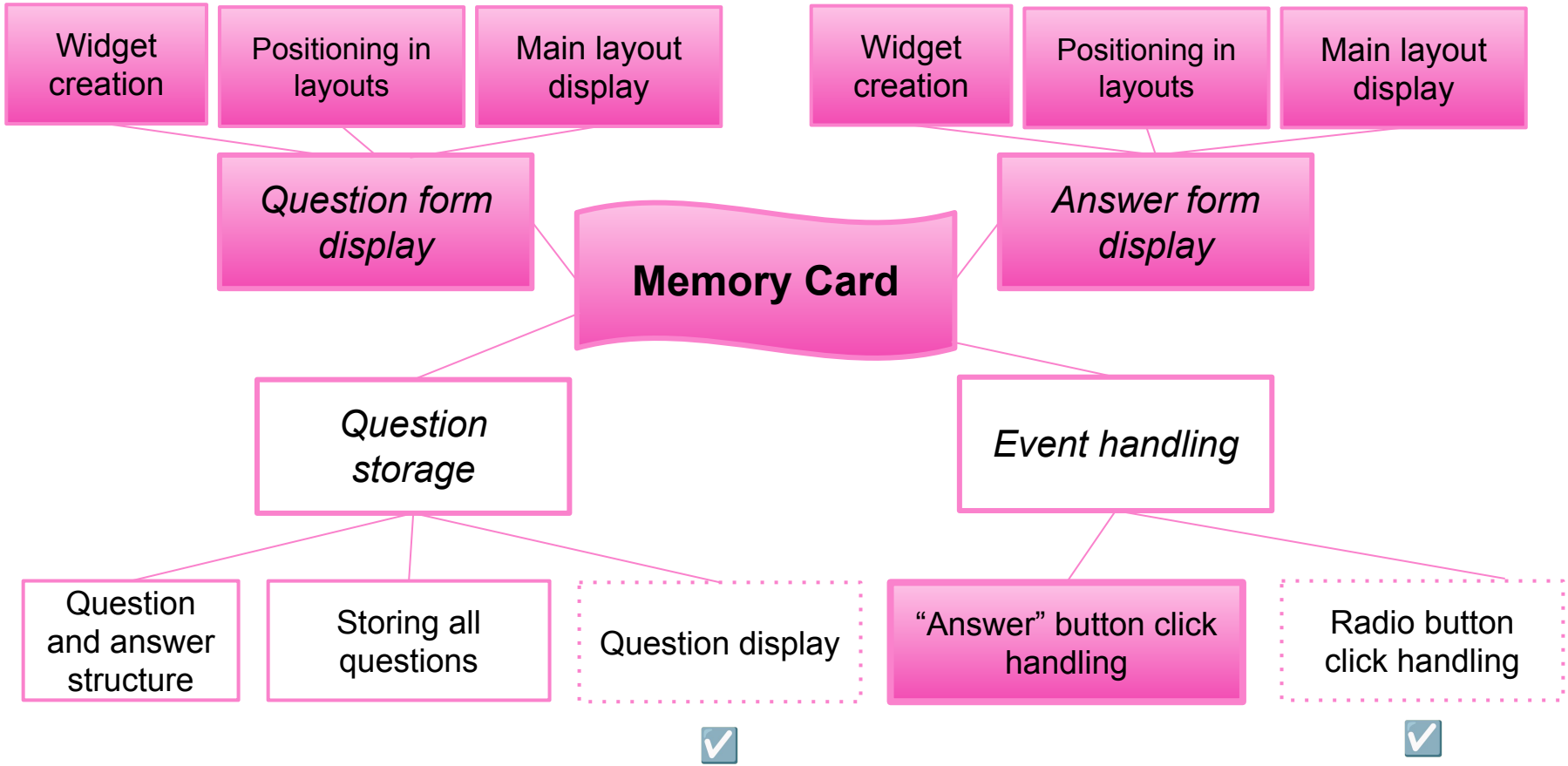
Application content

Brainstorming:

Question display



Project mind map:

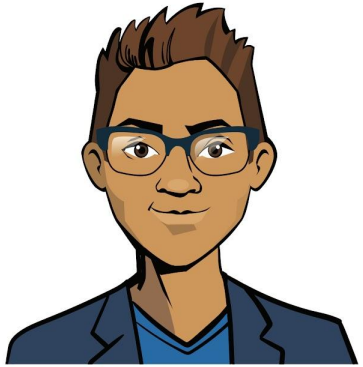


How do we ask a question?

We used to display the question by initially putting the necessary labels on widgets.

Now we'll try to describe an *ask()* function that **asks a question** and a *check_answer()* that **checks the answer**.

If we can do this for one question, then, next time, we can expand this solution for the entire set of questions.

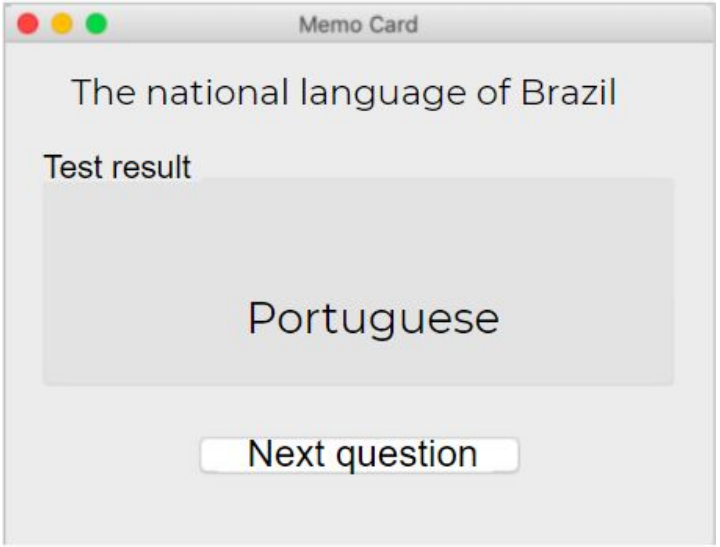


Brainstorming

1. Displaying the question in the form

Let the question and answer options be given in lines. Let's describe the ask() function:

```
def ask(question, right_answer, wrong1, wrong2, wrong3)
```



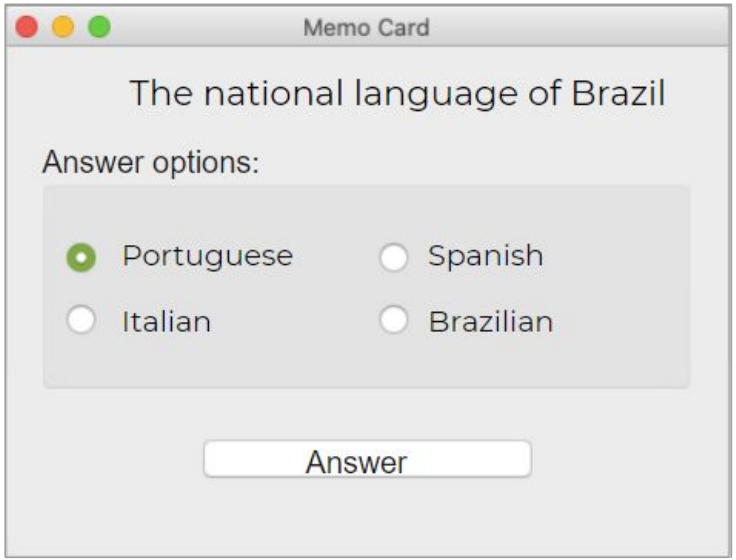
Brainstorming

1. Displaying the question in the form

Let the question and answer options be given in lines. Let's describe the ask() function:

```
def ask(question, right_answer, wrong1, wrong2, wrong3)
```

How do we put the data lines in the widgets?



Brainstorming

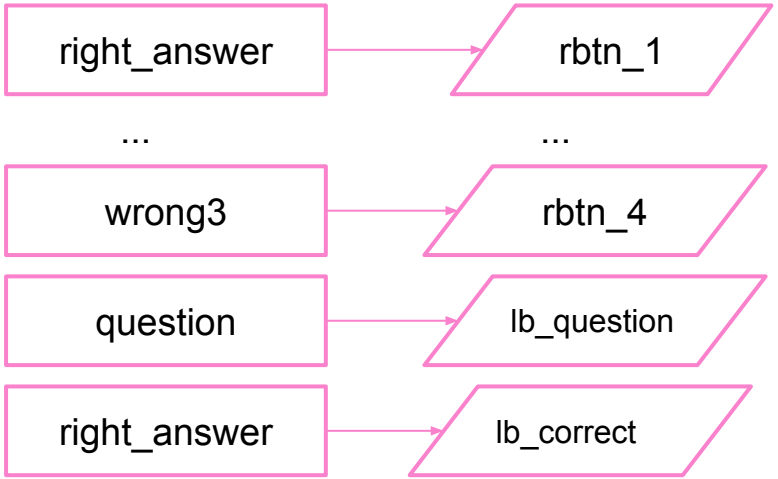
1. Displaying the question in the form

Let the question and answer options be given in lines. Let's describe the ask() function:

```
def ask(question, right_answer, wrong1, wrong2, wrong3)
:
```

How do we put the data lines in the widgets?

Here's one possibility: put the ith answer option into the ith radio button.



Should we use it?

Brainstorming

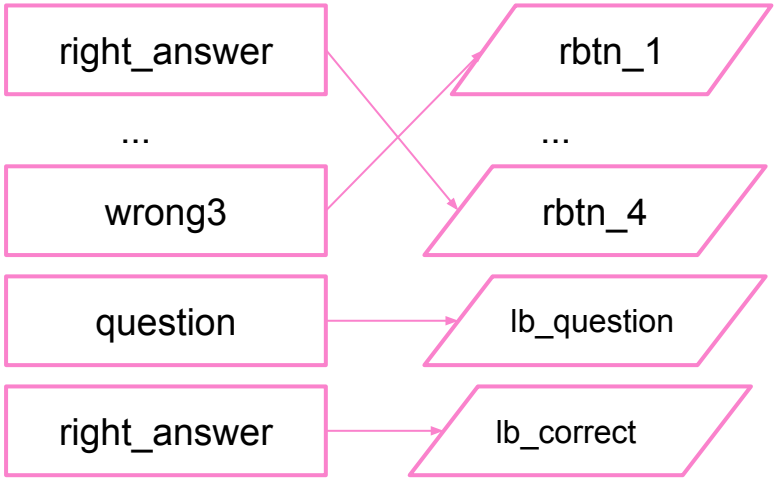
1. Displaying the question in the form

Let the question and answer options be given in lines. Let's describe the ask() function:

```
def ask(question, right_answer, wrong1, wrong2, wrong3)
:
```

How do we put the data lines in the widgets?

Here's one possibility: put the ith answer option into the ith radio button.



No! If we do that, the correct answer will always be the first button.

The answers need to be shuffled.

Brainstorming

1. Displaying the question in the form

```
#The function writes the value of the question

def ask(question, right_answer, wrong1, wrong2, wrong3):
```

- ❑ shuffle the answer options
- ❑ give the correct answer as a random button while the rest are incorrect
- ❑ give the question text and give the correct answer in the answer form
- ❑ display the question form

Brainstorming

1. Displaying the question in the form

```
#The function writes the value of the question

def ask(question, right_answer, wrong1, wrong2, wrong3):
```

- ❑ shuffle the answer options
- ❑ give the correct answer as a random button while the rest are incorrect
- ❑ give the question text and give the correct answer in the answer form
- ❑ display the question form

How do we shuffle the answer options?

Brainstorming

1. Displaying the question in the form

```
#The function writes the value of the question

def ask(question, right_answer, wrong1, wrong2, wrong3):
```

- ❑ shuffle the answer options
- ❑ give the correct answer as a random button while the rest are incorrect
- ❑ give the question text and give the correct answer in the answer form
- ❑ display the question form

```
from random import shuffle

answers = [rbtn_1, rbtn_2, rbtn_3, rbtn_4]

shuffle(answers)
```

We can **shuffle the buttons**!

Create a list of radio buttons and mix its elements.

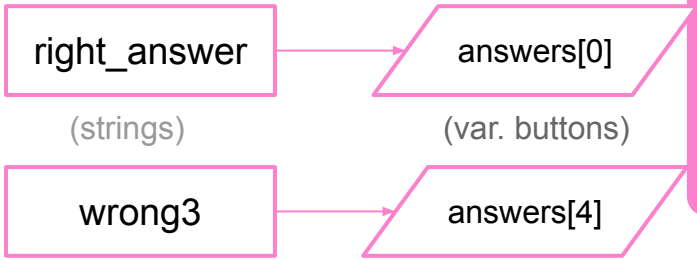
Brainstorming

1. Displaying the question in the form

```
#The function writes the value of the question

def ask(question, right_answer, wrong1, wrong2, wrong3):
```

- ❑ shuffle the answer options
- ❑ give the correct answer as a random button while the rest are incorrect
- ❑ give the question text and give the correct answer in the answer form
- ❑ display the question form

```
from random import shuffle

answers = [rbtn_1, rbtn_2, rbtn_3, rbtn_4]
shuffle(answers)
```



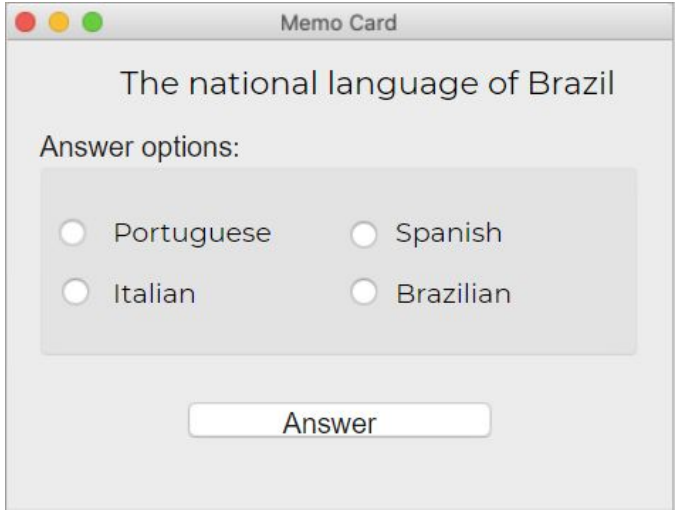
Brainstorming

2. Checking the selected answer

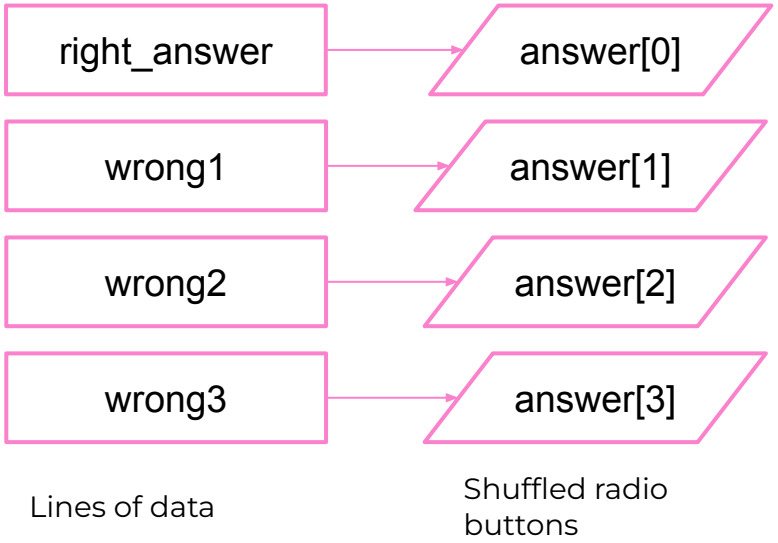
Let's describe the check_answer() function that checks the answer:

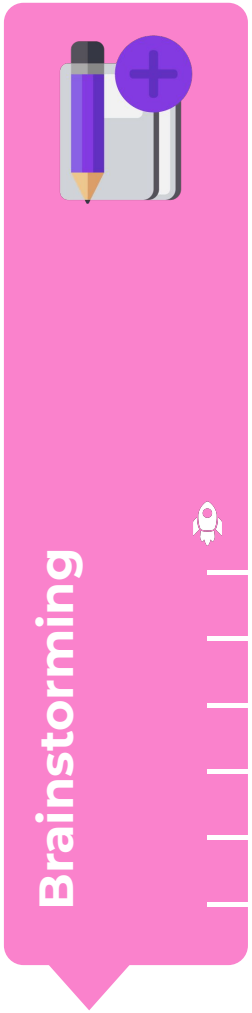
```
def check_answer()
```

How do we check a given answer and display the result?



A link appeared in the previous function:





2. Checking the selected answer

#function that checks the answer to the question and displays the result

```
def check_answer():
```

- If the first radio button, answer[0], is clicked, then show the message: “Correct”!
- If any other radio button is clicked, show the message: “Incorrect!”
- Display the answer form and show the correct answer.

The radio button method **rbtn.isChecked()** checks if the radio button is clicked.

The operations are performed when “Answer!” is clicked.

Check_answer() replaces the current start_test() function!

Brainstorming

Let’s unite them :

```
def ask(question, right_answer, ...wrong3):
```

Shuffle the button, linking answer options (answer[0] is correct).

Display the question in the question form and the correct answer in the answer form.

```
def check_answer():
```

Check the answer. If the radio button answer[0] is clicked, then the answer is correct. If any other answer is clicked, then incorrect.

Call **show_correct**(), passing the line with the result.

```
def show_correct(res):
```

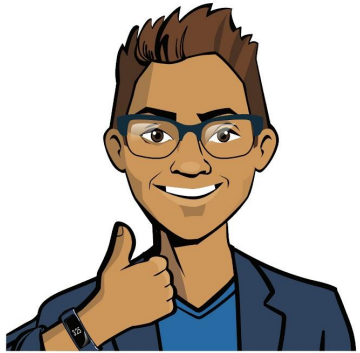
Display the answer form with the correct answer and a res mark (“Correct”/”Incorrect”).

```
ask('The national language of Brazil', 'Portuguese', ... 'Italian')

btn_OK.clicked.connect(check_answer)
```

Brainstorming

Expected results:

- The start_test() will **stop working**.
- The application **now has the functions** ask(), check_answer() and show_correct() for asking a question and checking the given answer.
- Now the application knows how to ask one question, check the answer and display the result.



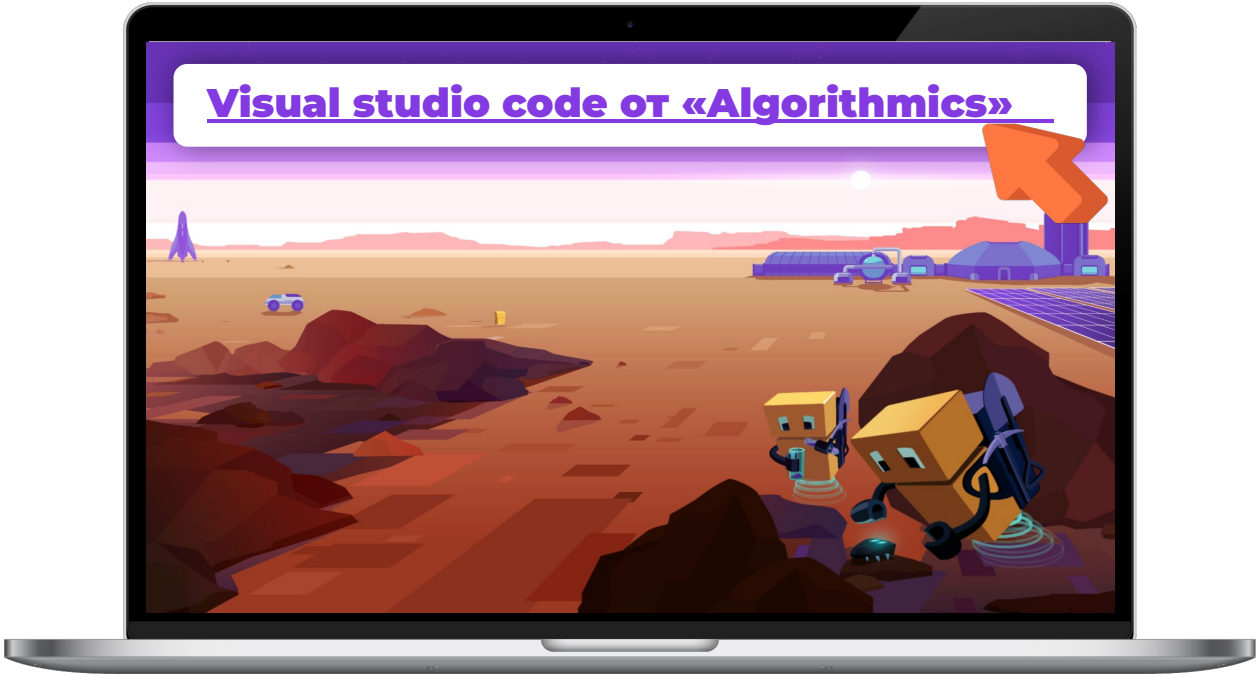
Brainstorming

Visual Studio Code:

Memory Card Application

Cmoplete the task in VS Code

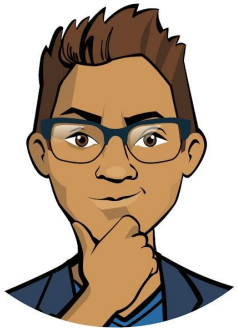
“VSC. PyQt. Memory Card”



Wrapping up the workday

To wrap up the workday, complete this technical interview:

1. Which method allows us to check whether a radio button has been selected?
2. How do we create the logical expression: “At least one of three radio buttons has been clicked”?
3. How do we shuffle the elements in a list? How is this function used in Memory Card?



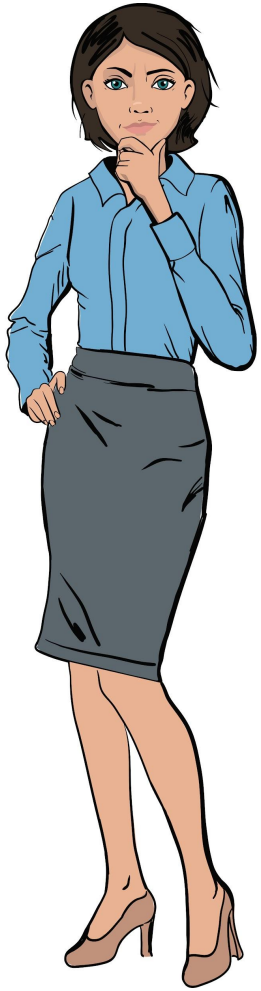
Cole,
Senior Developer

Emily,
Project Manager

How effective was our work?

Together with your colleagues, answer these questi

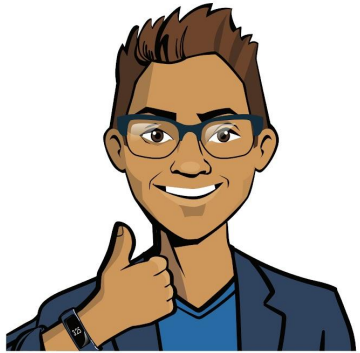
1. What went especially well?
2. What didn’t go as well as you expected?
3. What can you do next time to ensure success?



Wrapping up the workday

Additional tasks

- ❑ Look at the code you've written one more time.
- ❑ Finish writing the code if necessary.
- ❑ **Add comments to the code to explain** which part of the code does what.



Wrapping up the workday