algorithmics

Memory Card Application. Part 1

🚀**STORYLINE:**

The ProTeam developers have been hired by the “Citizen of the World” cultural center.

Center employees organize lectures, master classes, plays, and performances to celebrate the beauty and variety of cultures and languages around the world. To keep their employees well-versed in the various cultures and languages, the center has commissioned a Memory Card application. This application should ask the user questions with answer options, which are stored in the program’s memory.

Breaking this task into sub-tasks, the developers realize they must begin by planning out the application’s interface.

⚠️ **SUMMARY:**

The goal for this lesson is to apply knowledge of the main widgets and interface design to create the Memory Card application.

In the first half of the lesson, students plan and program a question form with answer options. In the second half of the lesson, they finish writing the result display form.

Technical note. After this, all students will spend four lessons working on the same task: “VSC. Memory Card Application”

💾 **LINKS AND ESSENTIALS:**

- lesson presentation
- task: Memory Card (Visual Studio Code).

🎯 LESSON EDUCATIONAL GOALS

After the lesson, the students:	*The result is achieved when the students:*
• list the widgets and their names in PyQt; • link the necessary modules and their components • work with widgets: changing labels, setting the desired sizes, align to center or to an edge, regulate the space between them; work with the attached layouts • work with the attached layouts • unite several widgets into a group (QGroupBox) and position them in the layouts • create an application, window, and set up the necessary layout	• have participated in discussions and asked clarifying questions • confidently give the correct names for the widgets • confidently suggest placements for the widgets in the layouts • created an application in PyQt • answered the teacher’s questions during the reinforcement stage

RECOMMENDED LESSON STRUCTURE

Time	Stage	Stage goals
5 min	**Storyline. Discussion: “Memory Card”**	❏ Set a story-related goal: to make an application for memorizing information. ❏ Come to realize the necessity of planning your work on a complex task and breaking it into several parts.
10 min	**Qualification**	❏ Organize review using a presentation. ❏ The PyQT library, the main widgets ❏ positioning widgets in layouts
15 min	**Brainstorming: “Memory Card Interface”**	❏ Make a mind map of the goals associated with the Memory Card application. ❏ List the necessary widgets to create an application window and a question form with answer options on it. ❏ Suggest a way of uniting answer options into a group and positioning widgets in layouts.
20 min	**Visual Studio Code: “VSC. PyQt. Memory Card”**	❏ Organizing the completion of the task “VSC. PyQt. Memory Card” in the Visual Studio Code environment.
5 min	**Break**	❏ Help restore concentration through a game.
10 min	**Brainstorming: “Displaying forms”**	❏ Check off completed tasks on the mind map. ❏ List the widgets necessary to create the correct answer form. ❏ Suggest hiding the question form and displaying the correct answer form when the “Answer” button is pressed. ❏ Suggest a way to program this solution.
20 min	**Visual Studio Code: “VSC. PyQt. Memory Card”**	❏ Organize the completion of the task “VSC. PyQt. Memory Card” in the Visual Studio Code environment.
5 min	**Lesson wrap-up. Reflection.**	❏ Complete a technical interview on the topics: groups of widgets and their use in layouts. ❏ Suggest “documentation” on the platform and additional goals.

Storyline. Discussion: “Memory Card”

(5 min.)

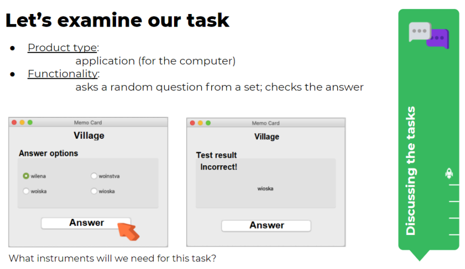
Open the presentation. The developers do not need computers for now.

> *“Hello, colleagues! I’m glad to see you as part of our ProTeam development team. We have been hired by the “Citizen of the World” cultural center. Center employees organize lectures, master classes, plays, and performances to celebrate the beauty and variety of cultures and languages around the world. To keep their employees well-versed in the various cultures and languages, the center has commissioned a Memory Card application.”*

Explain that the application must:

- ❏ store a set of questions and answer options
- ❏ in random order, ask questions and suggest answer options
- ❏ after the user has chosen an answer and pressed the “Answer” button, display a result (“Correct” or “Incorrect” and the correct answer)
- ❏ when the “Next question” button is pressed, go on to the next question

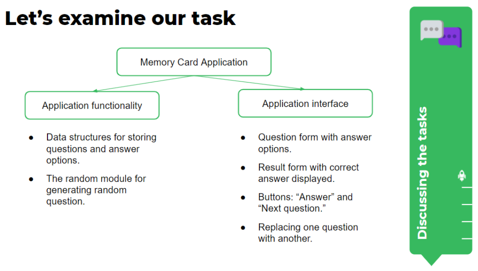
List the instruments for realizing this commission. Show that a program of this kind requires the developers to work with the basic Python tools (random numbers and data structures) and be well-versed in PyQt.



Let’s examine our task

- Product type:
  application (for the computer)
- Functionality:
  asks a random question from a set; checks the answer

What instruments will we need for this task?



Let’s examine our task

Memory Card Application

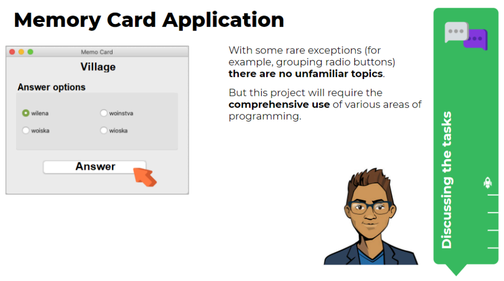
Application functionality

- Data structures for storing questions and answer options.
- The random module for generating random question.

Application interface

- Question form with answer options.
- Result form with correct answer displayed.
- Buttons: “Answer” and “Next question.”
- Replacing one question with another.

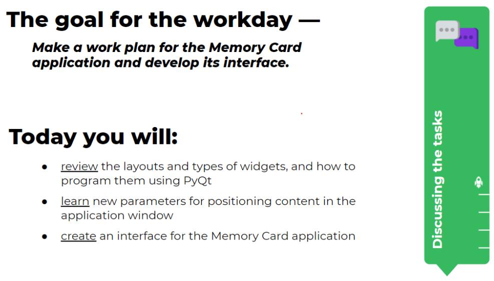
Mention that such a large project requires the cohesive use of different areas of programming and a work plan; for example, breaking the tasks into subtasks and making a mind map of the project.



Memory Card Application

With some rare exceptions (for example, grouping radio buttons) **there are no unfamiliar topics**.

But this project will require the **comprehensive use** of various areas of programming.



The goal for the workday —

Make a work plan for the Memory Card application and develop its interface.

Today you will:

- review the layouts and types of widgets, and how to program them using PyQt
- learn new parameters for positioning content in the application window
- create an interface for the Memory Card application

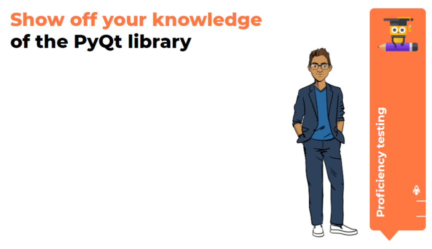
Discussing the tasks

State the goal for the workday and announce its content.

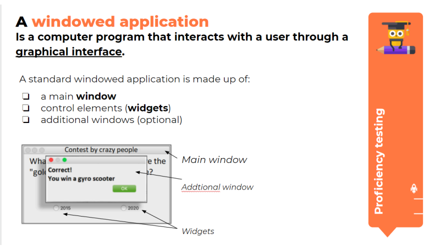
Qualification

(10 min.)

Check the developers' proficiency in the required skills. This time, the proficiency check will involve the brainstorming topics from the previous two workdays.



Show off your knowledge
of the PyQt library

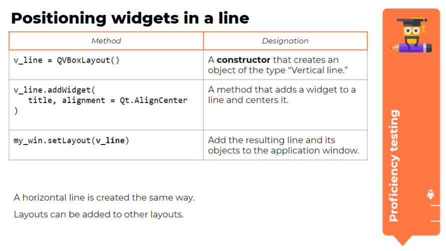


A windowed application
Is a computer program that interacts with a user through a graphical interface.

A standard windowed application is made up of:

- ❑ a main **window**
- ❑ control elements (**widgets**)
- ❑ additional windows (optional)

Contest by crazy people
Correct!
You win a gyro scooter
OK
2015
2020
Main window
Addtional window
Widgets
Proficiency testing



Positioning widgets in a line

Method	*Designation*
`v_line = QVBoxLayout()`	A **constructor** that creates an object of the type "Vertical line."

`v_line.addWidget(`
`    title, alignment = Qt.AlignCenter`
`)`
	A method that adds a widget to a line and centers it.

`my_win.setLayout(v_line)`	Add the resulting line and its objects to the application window.

A horizontal line is created the same way.

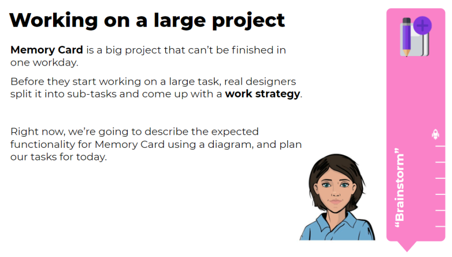
Layouts can be added to other layouts.

...

Brainstorming: "The Memory Card Interface"

(15 min.)

Begin the brainstorming by dividing the project into tasks. Explain to the developers that this is not just a quiz application with one question, but a serious program with a dataset, in which the switch between question forms and the correct answer must happen in the sme window.



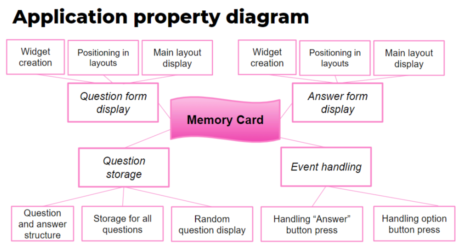
Working on a large project

Memory Card is a big project that can't be finished in one workday.

Before they start working on a large task, real designers split it into sub-tasks and come up with a **work strategy**.

Right now, we're going to describe the expected functionality for Memory Card using a diagram, and plan our tasks for today.

"Brainstorm"



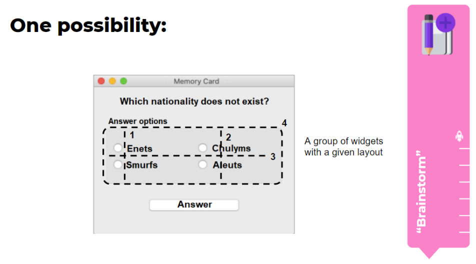
Application property diagram

Discuss your work on the Memory Card project:

- ❑ Ask the developers to identify the important goals for this project. These could be: question interface, correct answer interface, switching between forms, storing data, etc.
- ❑ Break each task into subtasks. For example, for the question form these could be: creating the necessary widgets, positioning the widgets in the layouts, displaying the top-layer layout.
- ❑ Ask the developers to put the tasks in order and select several for the current workday.

The result of this discussion should be a mind map with a highlighted goal for the first half of the workday.

Next, discuss the interface of the application form.



One possibility:

Memory Card
Which nationality does not exist?
Answer options
1
Enets
2
Chulyms
3
Smurfs
Aleuts
4
Answer
A group of widgets with a given layout
"Brainstorm"

......

1. Ask the developers to list the necessary widgets. Remind them that these are imported from the QtWidgets module of the PyQt library.

2. Demonstrate how several widgets are united into a group. To do that, create a group of type QGroupBox, position the necessary widgets in the layouts (how? - ask the developers) and establish the main layout in the group.

3. Draw their attention to the parameters that influence the positioning of the content: aligning to the center or an edge, stretching or compressing the widgets, adding spaces between widgets or layouts. These parameters are imported from the QtCore module of the PyQt library.

   Demonstrate the anticipated result, state the current goals and move on to the task in VSC.

Visual Studio Code: “VSC. PyQt. Memory Card”

(20 min.)

Have the students complete the task on the question interface form for the Memory Card attachment. The task does not have an auto-checker. A solved problem must be checked against a screenshot and shown to the mentor.

You will find the anticipated solution at the end of this methodological plan.

Break

(5 min.)

Give the developers a break from their computers. The goal of this break is to switch their attention to something else and let them stretch. Organize one of the recommended physical activities.

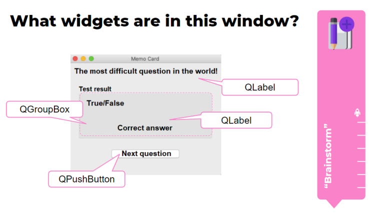
Brainstorming: “Displaying forms”

(10 min.)

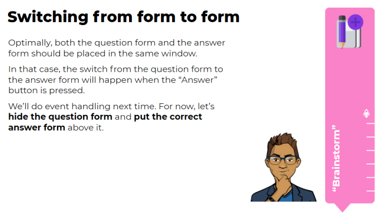
Together with the developers, mark off the completed task on the project mind map. Then, discuss the correct answer form:

“Note that both the question form and the answer form are in the same window, and that they switch off. Creating that functionality is not a priority for this workday so, for now, we should just hide the group of radio buttons with the answer options (the question and button remain).

❏ List the necessary widgets. Ask the developers: does this form have a group of widgets like before? What widgets should be included?
❏ Let the developers suggest their own positioning for the widgets in the layout. Then, show them your solution.



What widgets are in this window?



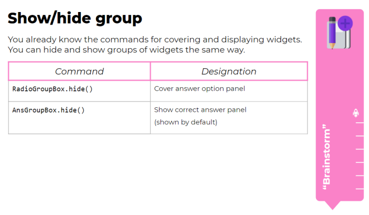
Switching from form to form

Optimally, both the question form and the answer form should be placed in the same window.

In that case, the switch from the question form to the answer form will happen when the "Answer" button is pressed.

We'll do event handling next time. For now, let's **hide the question form** and **put the correct answer form** above it.

"Brainstorm"



Show/hide group

You already know the commands for covering and displaying widgets. You can hide and show groups of widgets the same way.

Command	*Designation*
`RadioGroupBox.hide()`	Cover answer option panel
`AnsGroupBox.hide()`	Show correct answer panel (shown by default)

"Brainstorm"

Wrap up the discussion, answer any questions and go on to the task.

Visual Studio Code: "VSC. PyQt. Memory Card"

(20 min.)

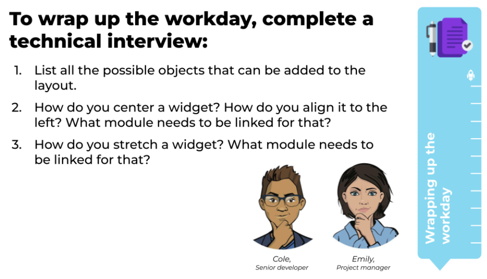
Begin working on the next task for the Memory Card application.

The expected solution to both tasks for this lesson is given at the end of this methodological plan.

Wrapping up the workday. Reflections

(10 min.)

Use a presentation to wrap up the workday. Conduct a technical interview with questions about the material from the brainstorming stage.



To wrap up the workday, complete a technical interview:

1. List all the possible objects that can be added to the layout.
2. How do you center a widget? How do you align it to the left? What module needs to be linked for that?
3. How do you stretch a widget? What module needs to be linked for that?

Cole,
Senior developer
Emily,
Project manager
Wrapping up the workday

Suggest an additional task: look through the code one more time and add comments. Soon the program will get even bigger, and thelayou code will become unreadable without comments.

Answers to the tasks

Task «VSC. PyQt. Memory Card»

1. Program text:

```
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import (QApplication, QWidget, QHBoxLayout, QVBoxLayout, QGroupBox,
QRadioButton, QPushButton, QLabel)grolay

app = QApplication([])

window = QWidget()
window.setWindowTitle('Memo Card')

btn_OK = QPushButton('Answer') # answer button
lb_Question = QLabel('----------------------') # your question

RadioGroupBox = QGroupBox("Answer options") # group on the screen for radio buttons with answers
rbtn_1 = QRadioButton('---------')
# 3 buttons
rbtn_1 = QRadioButton('---------')
rbtn_1 = QRadioButton('---------')
rbtn_1 = QRadioButton('---------')


layout_ans1 = QHBoxLayout()
layout_ans2 = QVBoxLayout() # the vertical ones will be inside the horizontal ones
layout_ans3 = QVBoxLayout()
layout_ans2.addWidget(rbtn_1) # two answers in the first column
layout_ans2.addWidget(rbtn_2)
layout_ans3.addWidget(rbtn_3) # two answers in the second column
layout_ans3.addWidget(rbtn_4)

layout_ans1.addLayout(layout_ans2)

layout_ans1.addLayout(layout_ans3) # columns are in the same line

RadioGroupBox.setLayout(layout_ans1) # “panel” with answer options is ready

layout_line1 = QHBoxLayout() # question
layout_line2 = QHBoxLayout() # answer options or test results
layout_line3 = QHBoxLayout() # “Answer” button

layout_line1.addWidget(lb_Question, alignment=(Qt.AlignHCenter | Qt.AlignVCenter))
```

```
layout_line2.addWidget(RadioGroupBox)

layout_line3.addStretch(1)
layout_line3.addWidget(btn_OK, stretch=2) # the button sho

uld be large
layout_line3.addStretch(1)

# Now let’s put the lines we’ve created one under one another:
layout_card = QVBoxLayout()

layout_card.addLayout(layout_line1, stretch=2)
layout_card.addLayout(layout_line2, stretch=8)
layout_card.addStretch(1)
layout_card.addLayout(layout_line3, stretch=1)
layout_card.addStretch(1)
layout_card.setSpacing(5) # spaces between the content

window.setLayout(layout_card)
window.show()
app.exec()
```

2. Program text:

```
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import (
        QApplication, QWidget,
        QHBoxLayout, QVBoxLayout,
        QGroupBox, QRadioButton,
        QPushButton, QLabel)

app = QApplication([])

btn_OK = QPushButton('Answer')

lb_Question = QLabel('The most difficult question in the world!')
RadioGroupBox = QGroupBox("Answer options")

rbtn_1 = QRadioButton('Option 1')
rbtn_2 = QRadioButton('Option 2')
rbtn_3 = QRadioButton('Option 3')
rbtn_4 = QRadioButton('Option 4')

layout_ans1 = QHBoxLayout()
```

```
layout_ans2 = QVBoxLayout()
layout_ans3 = QVBoxLayout()
layout_ans2.addWidget(rbtn_1) # two answers in the first column
layout_ans2.addWidget(rbtn_2)
layout_ans3.addWidget(rbtn_3) # two answers in the second column
layout_ans3.addWidget(rbtn_4)

layout_ans1.addLayout(layout_ans2)
layout_ans1.addLayout(layout_ans3)

RadioGroupBox.setLayout(layout_ans1)

# Create a results panel
AnsGroupBox = QGroupBox("Test result")
lb_Result = QLabel('Are you correct or not?') # “Correct” or “Incorrect” text will be here
lb_Correct = QLabel('the answer will be here!') # correct answer text will be written here

layout_res = QVBoxLayout()
layout_res.addWidget(lb_Result, alignment=(Qt.AlignLeft | Qt.AlignTop))
layout_res.addWidget(lb_Correct, alignment=Qt.AlignHCenter, stretch=2)
AnsGroupBox.setLayout(layout_res)

# Place all the widgets in the window:
layout_line1 = QHBoxLayout() # question
layout_line2 = QHBoxLayout() # answer options or test result
layout_line3 = QHBoxLayout() # “Answer” button

layout_line1.addWidget(lb_Question, alignment=(Qt.AlignHCenter | Qt.AlignVCenter))
# Put both panels in the same line; one of them will be hidden and the other will be shown:
layout_line2.addWidget(RadioGroupBox)
layout_line2.addWidget(AnsGroupBox)
RadioGroupBox.hide() # We’ve already seen this panel; let’s hide it and see how the answer panel
turned out

layout_line3.addStretch(1)
layout_line3.addWidget(btn_OK, stretch=2) # the button should be large
layout_line3.addStretch(1)

# Now let’s put the lines we’ve created one under one another:
layout_card = QVBoxLayout()

layout_card.addLayout(layout_line1, stretch=2)
layout_card.addLayout(layout_line2, stretch=8)
layout_card.addStretch(1)
layout_card.addLayout(layout_line3, stretch=1)
layout_card.addStretch(1)
layout_card.setSpacing(5) # spaces between content

window = QWidget()
window.setLayout(layout_card)
window.setWindowTitle('Memory Card')
window.show()
app.exec()
```