algorithmics

International school of programming for kids

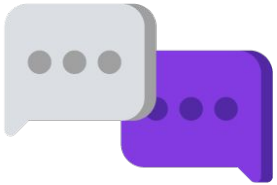
Module 2. Lesson 5.

Memory Card Application P. 3

MG

Discussion:

Memory Card Application



Continuing to work on our project

We are continuing to work on our project for the “Citizen of the World” cultural center.

The Center has ordered a **Memory Card application** to sharpen their specialists’ knowledge of world cultures and languages.

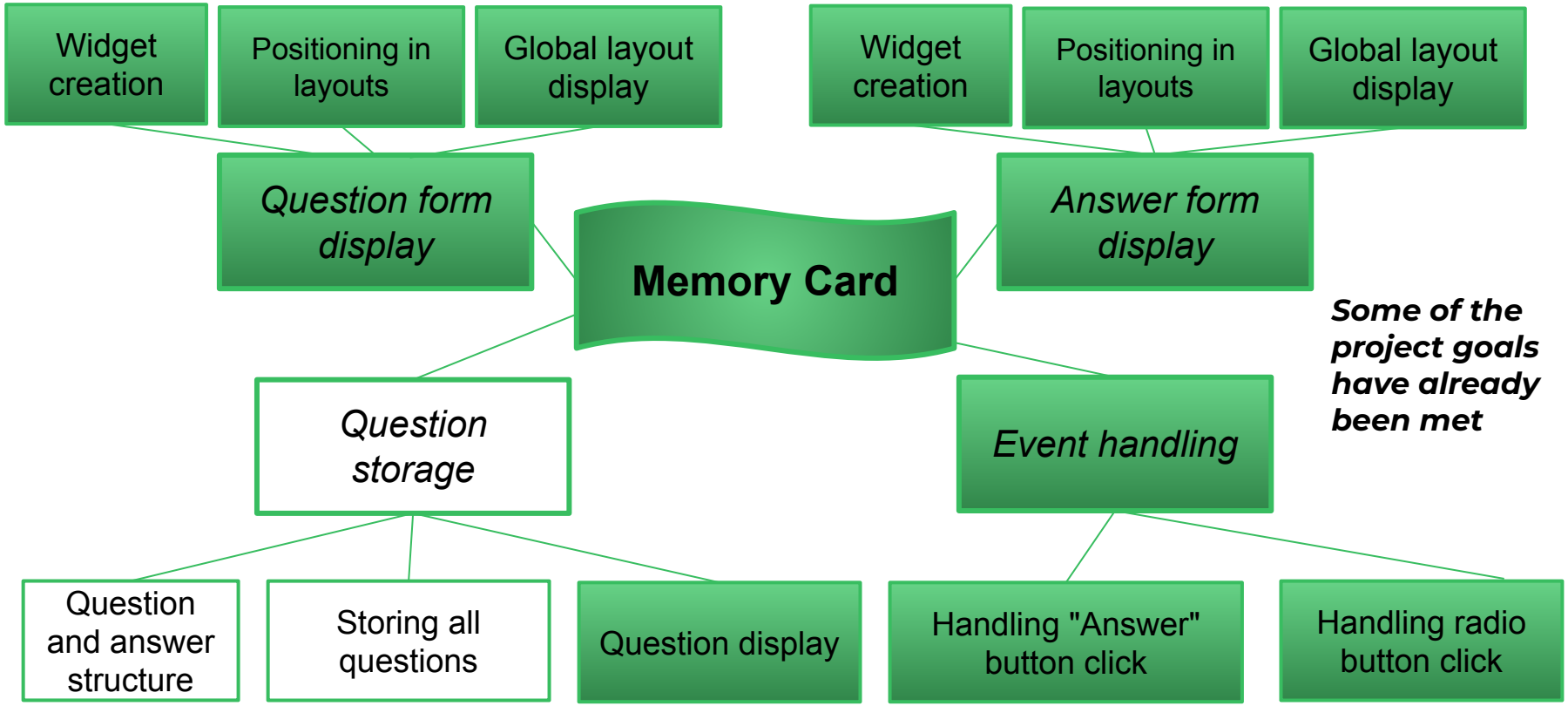
We have already programmed the basic interface for this application and learned to ask one question with several answer options.

Ready to keep working?

Emily,
Project Manager

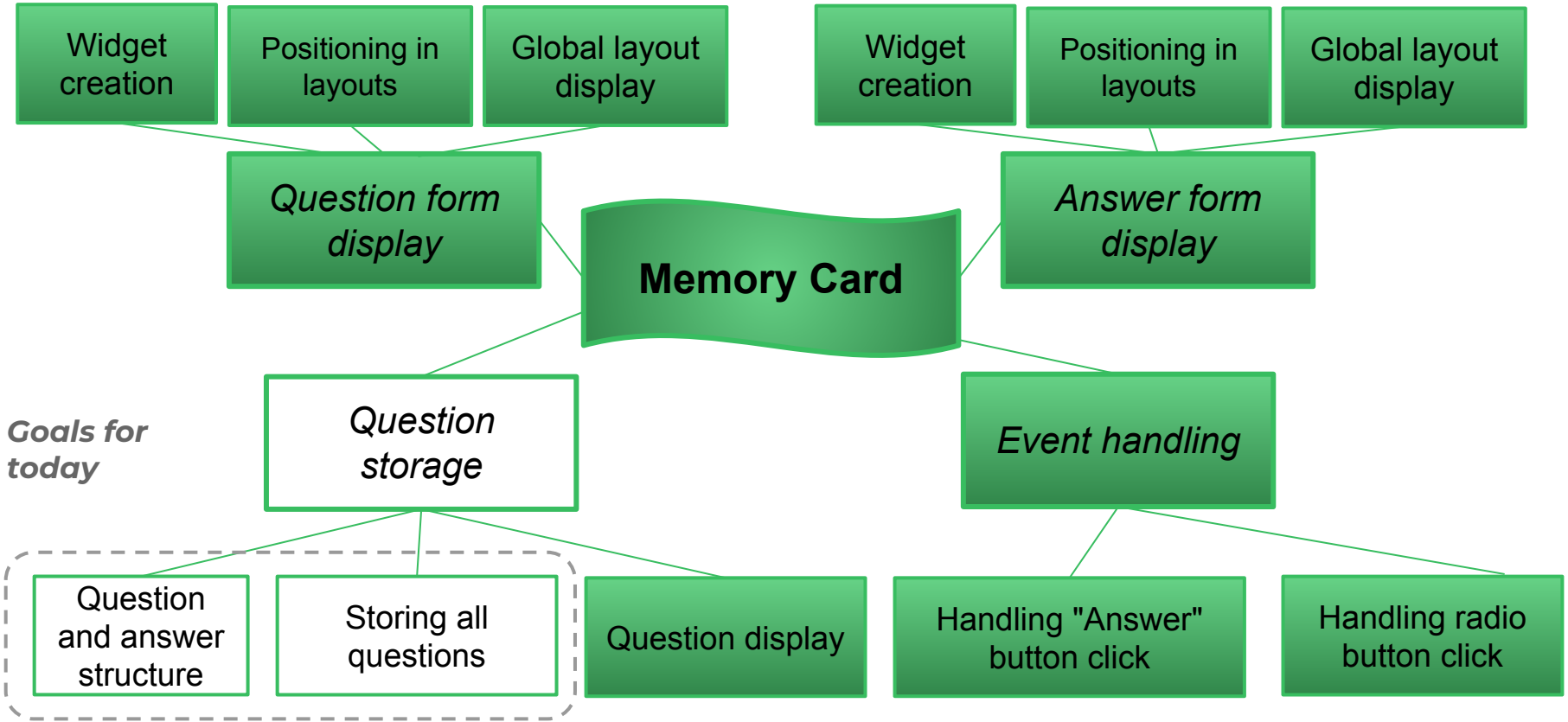
Discussing Project Tasks

Let’s look at a mind map of the project



Some of the project goals have already been met

Let’s look at a mind map of the project



The goal for the work day

is to program the storage of a set of questions and the move from one question to another.

Today you will :

- review what a class is and program your own class
- choose a data structure for storing the questions
- implement a system of questions and answers in your program!

Discussing Project Tasks

Qualification

Show your knowledge of data structures and object-oriented programming

Qualification

What is a list?
What methods for working with lists do you know?

Qualification

A list is a structure for the ordered storage of various types of data.

```
results = list()

results = [181, 176, 160, 178, 171, 179, 165]
```

181	176	160	178	171	179	165
0	1	2	3	4	5	6

```
print('Best result:', results[0])
```

Best result: 181

Get an element from the list using its number (index)

The program will print

Qualification

A list is a structure for the ordered storage of various types of data.

Let's say we have a list:

```
students = ['Smith', 'Garcia', 'Patel']
```

Add the last name 'Tanner' to the list	`students.append('Tanner')`
Remove the last name 'Smith' from the list	`students.remove('Smith')`
Check if the list contains the last name 'Hanz'	`'Hanz' in students`
Sort the list in alphabetical order	`students.sort()`

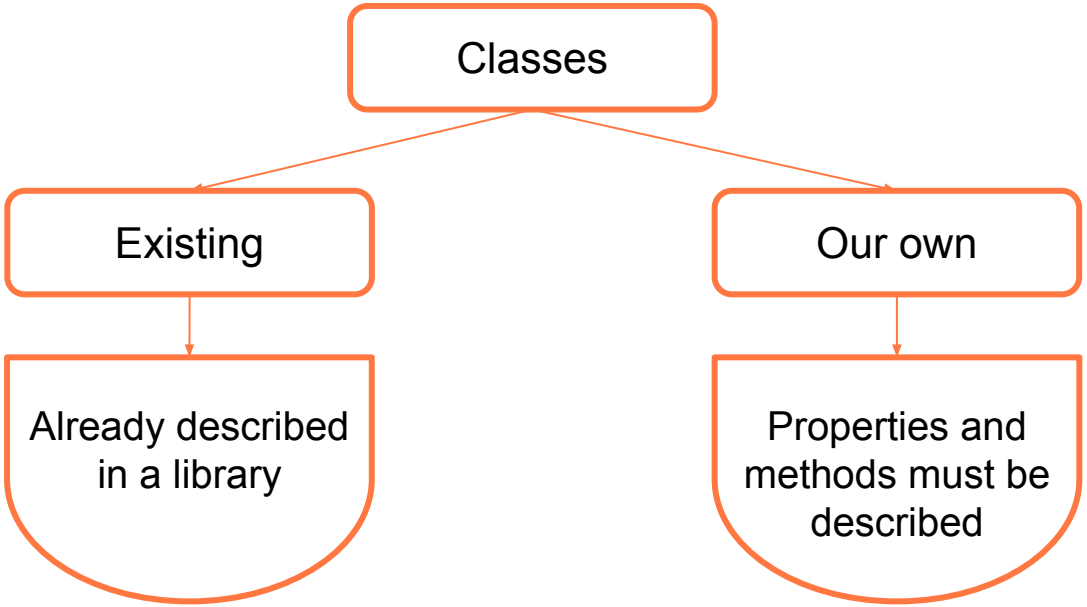
Qualification

What is a class?
Are there any existing classes?

Qualification

A class is

- a shared name for multiple objects
- (in programming) a shared description of how these objects should work



Qualification

How do we <u>create</u> our own <u>class</u>?

Qualification

Creating classes

To create a class, we must:

- list the **properties** that determine the traits of an instance of the class
- list the **methods** for working with an instance of the class

```
class [Class name]():
    def __init__(self, [Value]):
        self.[Property name] = [Value]
    def [Method name](self):
        [Action with object and properties]
        [Action with object and properties]
```

Qualification

What is a constructor ?
What does the self parameter do?

Qualification

Creating classes

A constructor is a special function that creates an instance of the class with the indicated properties.

```
__init__
```

Two underscores.

```
class [Class name]():
    def __init__(self, [Value]):
        self.[Property name] = [Value]
    def [Method name](self):
        [Action with object and properties]
        [Action with object and properties]
```

self is a parameter indicating the object to which the method is applied.

self.property is a property of the object to which the method is applied.

Qualification

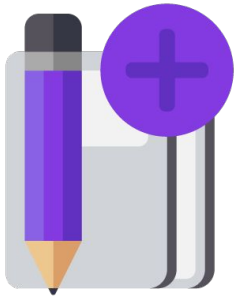
Qualification confirmed!

Excellent! You're ready to brainstorm a solution and meet today's goals!

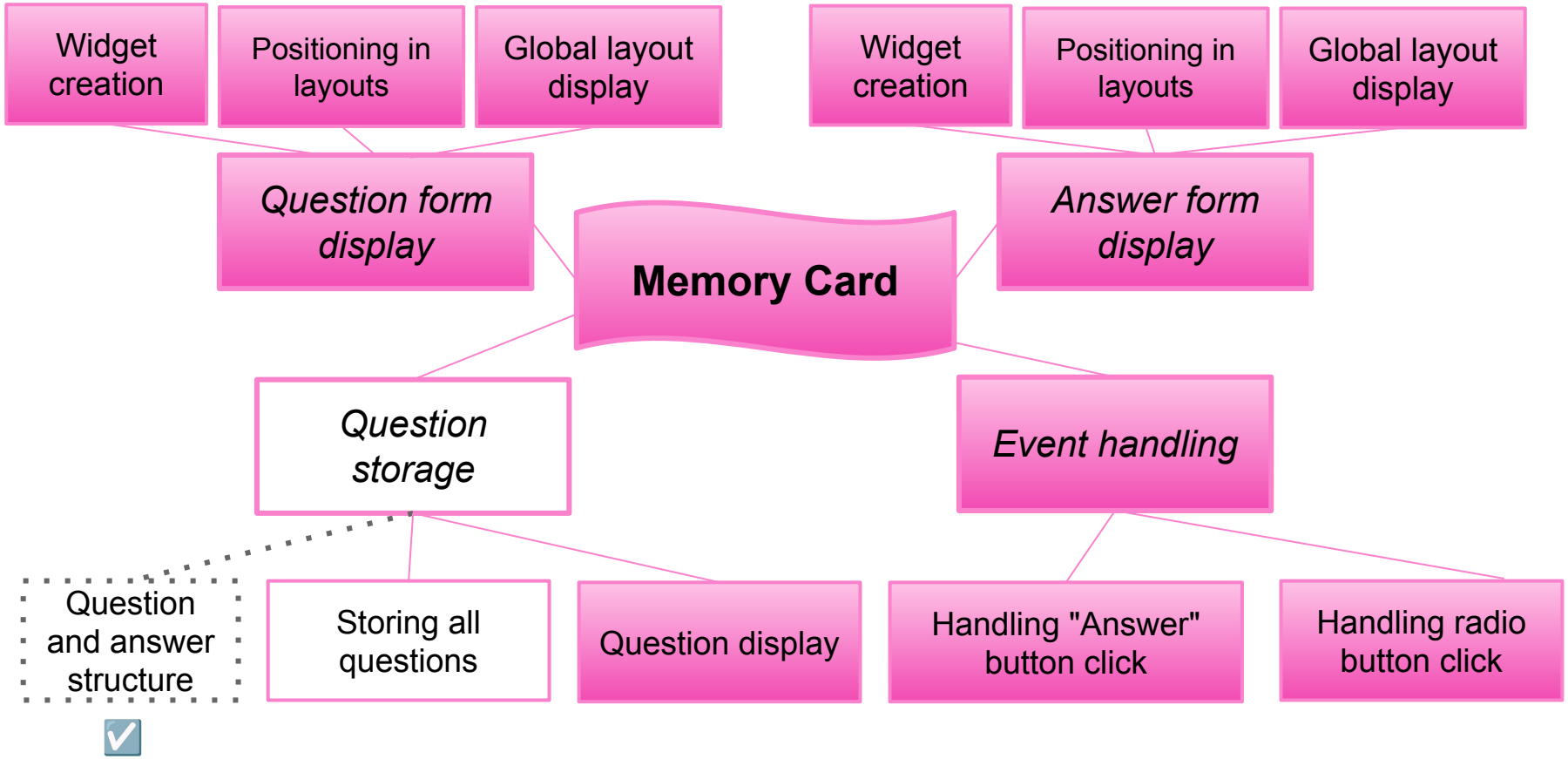
Qualification

Brainstorming:

Data Storage



Project mind map:



A question and the information about it

Question-related actions:

- ❑ storing a question
- ❑ displaying a question in an application window
- ❑ reading and checking the user's answer
- ❑ displaying the correct answer

Data we need about the question:

- ❑ ?
- ❑ ?
- ❑ ?

What data do we need for this functionality?

Brainstorming

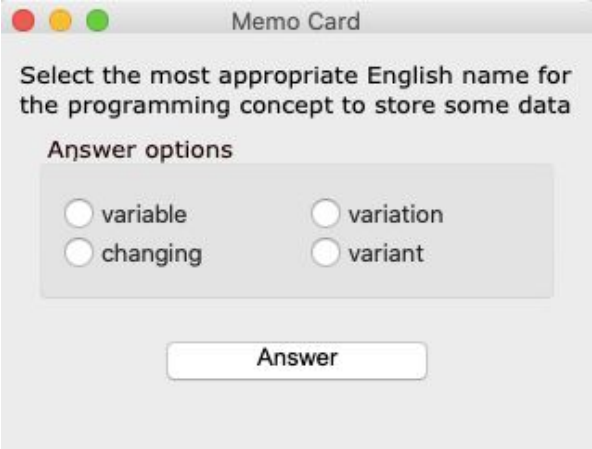
A question and the information about it

Question-related actions:

- ❑ storing a question
- ❑ displaying a question in an application window
- ❑ reading and checking the user's answer
- ❑ displaying the correct answer

Data we need about the question:

- ❑ the text of the question
- ❑ the correct answer option
- ❑ three incorrect answer options



Brainstorming

A question and the information about it

?

Data we need about the question:

- ❑ the text of the question
- ❑ the correct answer option
- ❑ three incorrect answer options

To ask multiple questions, we need a structure that can store a lot of data. ***What structure should we use***?

Brainstorming

A question and the information about it

```
class Question():
    def __init__(
     self, question, right_answer,
     wrong1, wrong2, wrong3):
        self.question = question
        self.right_answer = right_answer
        self.wrong1 = wrong1
        self.wrong2 = wrong2
        self.wrong3 = wrong3
```

The Question class

The class **constructor**, which gives the properties for an instance of Question

Properties:
– question text
– correct answer
– incorrect answer 1
– incorrect answer 2
– incorrect answer 3

It's convenient to "wrap up" the question data in the Question class.

Brainstorming

Implementing the Question class

To implement Question, we must make some changes to our program.

What	*Before*	*After*
Adding a new question	Data were stored in variables	?
Displaying a question	Widgets displayed data from variables: `answers[0].setText(right_answer)`	?
Asking a question (calling ask())	The entry parameters were the constants with the question data	?

Brainstorming

Implementing the Question class

To implement Question, we must make some changes to our program.

What	*Before*	*After*
Adding a new question	Data were stored in variables	Data are the properties of an instance of the class
Displaying a question	Widgets displayed data from variables `answers[0].setText(right_answer)`	Widgets display fields of the class `answers[0].setText(q.right_answer)`
Asking a question (calling ask())	The entry parameters were the constants with the question data	The entry parameter is an instance of the Question class

Brainstorming

Let’s put it all together:

```
class Question():
```

Class description

Application interface

Functions that display the question

```
def ask(q: Question):
```

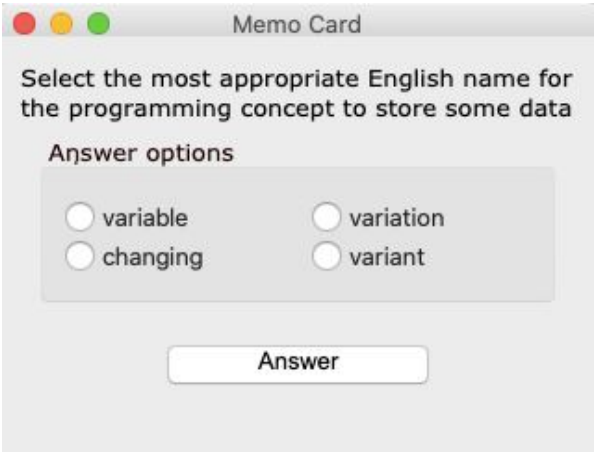
Changed function body with the properties of instance q

Creating a window, launching the application

Creating **instance q** of Question

Calling **ask with argument q**

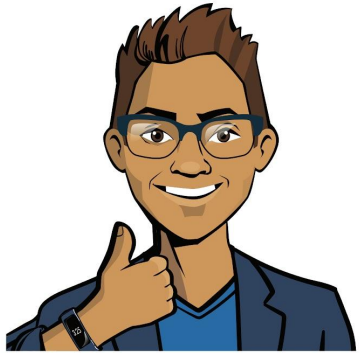
```
window = QWidget()
window.setLayout(layout_card)
window.setWindowTitle('Memo Card')
q = Question('Select the most appropriate English name for the programming concept to store some data',
'variable', 'variation', 'variant', 'changing')
ask(q)
btn_OK.clicked.connect(check_answer) # remove the test, we need to check the answer here
window.show()
app.exec()
```



Brainstorming

Expected results:

- “Inside” the application, the data storage system shall change. Instead of storing question data in variables, we’ll have the Question class and its instances.
- The application will not change externally and its functionality will not increase, but it will work differently.



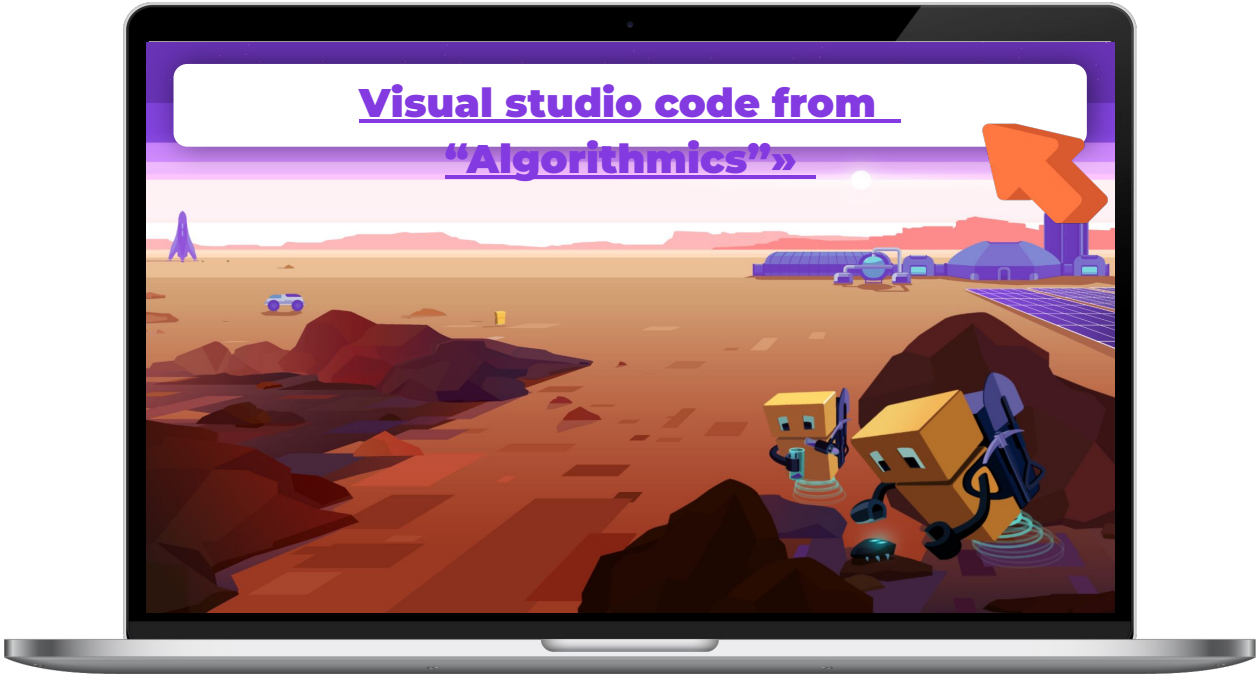
Brainstorming

Visual Studio Code:

Memory Card Application

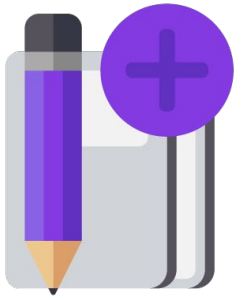
Do the task in VS Code

“VSC. PyQt. Memory Card”

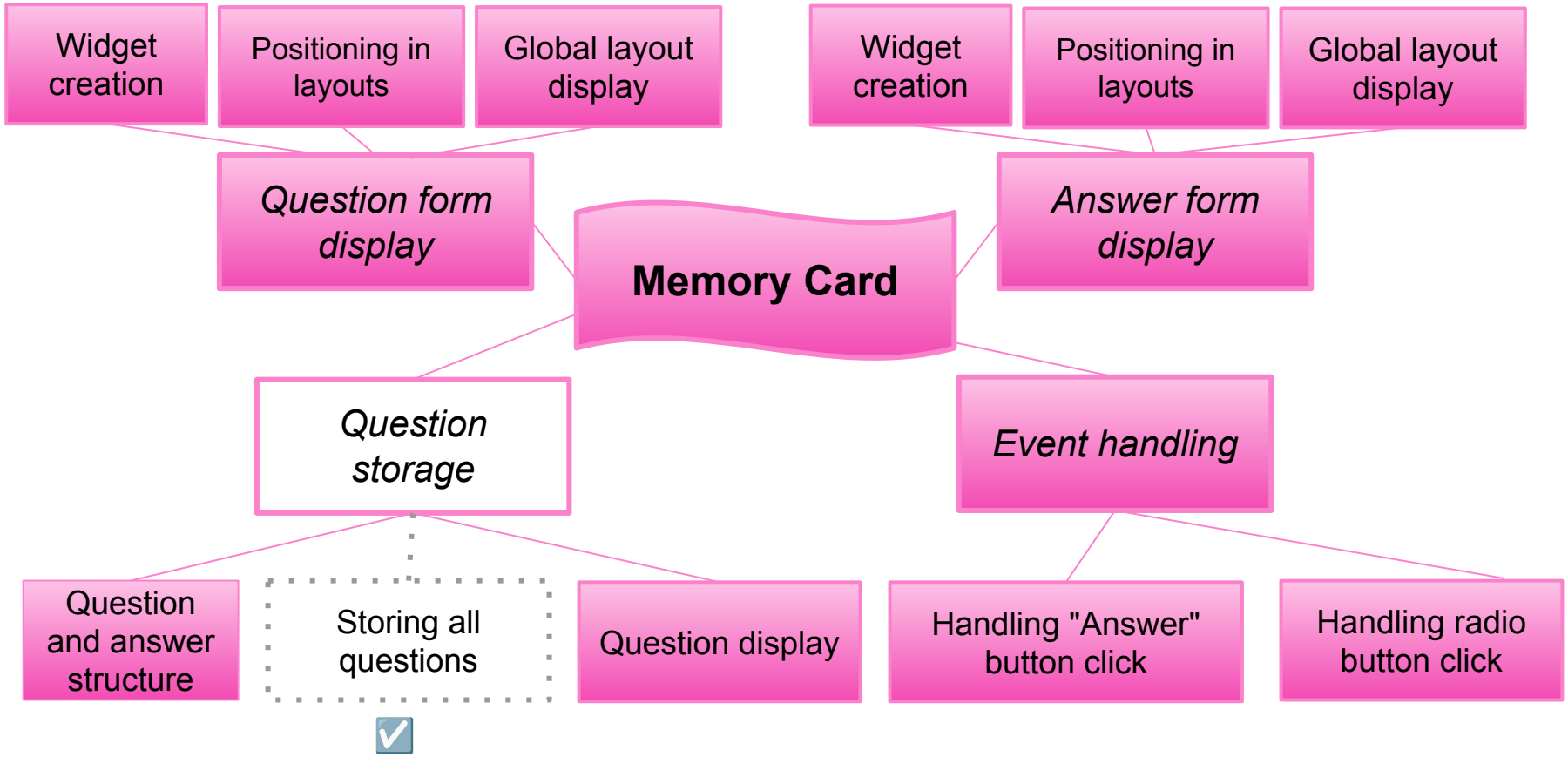


Brainstorming:

A System for Working with Questions



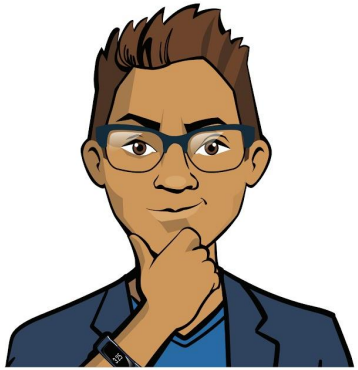
Project mind map:



How do we work with multiple questions?

To move from one question to multiple questions, we'll need to come up with a technical solution to the following issues:

- What structure will be used to store the set of questions, and how will it be filled in?
- How will the next question be displayed?
- How do we switch between checking an answer and transitioning to the next question?



Brainstorming

1. Storing a set of questions

One question is created as an instance of the class.

How do we create multiple instances of Question and organize their storage?

```
questions_list = []
q1 = Question(
      'The state language of Portugal', 'Portuguese',
      'English', 'Spanish', 'French')
questions_list.append(q1)
```

List of questions

Instance of the Question class

Adding an instance to the list of questions

Brainstorming

1. Storing a set of questions

One question is created as an instance of the class.

How do we create multiple instances of Question and organize their storage?

```
questions_list = []
q1 = Question(
      'The state language of Portugal', 'Portuguese',
      'English', 'Spanish', 'French')
questions_list.append(q1)
```

List of questions

Instance of the Question class

Adding an instance to the list of questions

```
questions_list = []
questions_list.append(
      Question('The state language of Portugal',
      'Portuguese', 'English', 'Spanish',
      'French'))
```

Can be shortened by nesting

✅

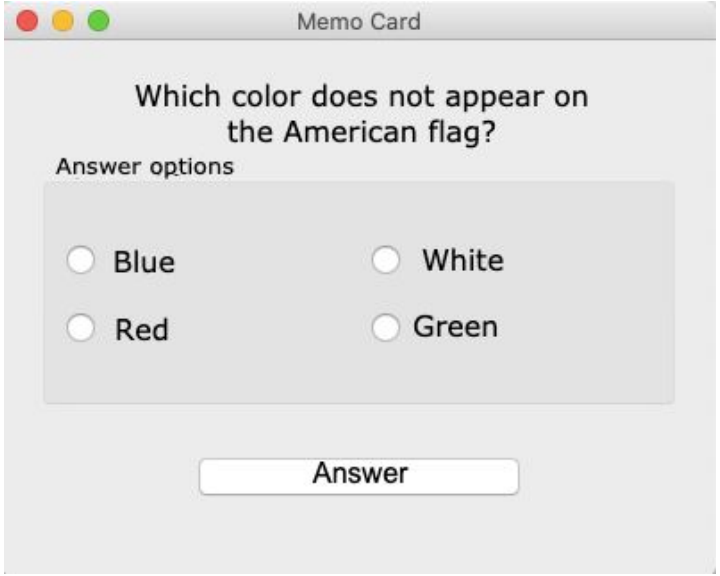
Brainstorming

2. Displaying the next question

Let's describe the function next_question() , which asks the next question.

To do that, we need to figure out:

How do we ask the first question? How do we check and move on to the next question?



Brainstorming

2. Displaying the next question

Let's say the function next_question() has been written.

next_question()	check_answer()
Displays the next question in the list	Checks the answer to the current question (displays answer form)

How do we determine which of the two functions to call at any given moment?

Brainstorming

2. Displaying the next question

Let's say the function next_question() has been written.

New function

next_question()

Displays the next question in the list

check_answer()

Checks the answer to the current question (displays answer form)

click_OK()

Calls next_question() or check_answer()

Use this function to handle the button click

Brainstorming

2. Displaying the next question

next_question()

Displays the next question in the list

check_answer()

Checks the answer to the current question (displays answer form)

click_OK()

Calls next_question() or check_answer()

In it, choose the function to call

Use this function to handle the button click

Brainstorming

2. Displaying the next question

Let's describe the function next_question() , which asks the next question

```
def next_question()
```

To switch to the next question, we need to implement a **question counter.**

1

```
window.cur_question = -1
```

Set a **property of the window** — the number of the question displayed. Give it the value -1 upon creation.

Brainstorming

2. Displaying the next question

Let's describe the function next_question() , which asks the next question.

```
def next_question()
```

To switch to the next question, we need to implement a **question counter.**

1

```
window.cur_question = -1
```

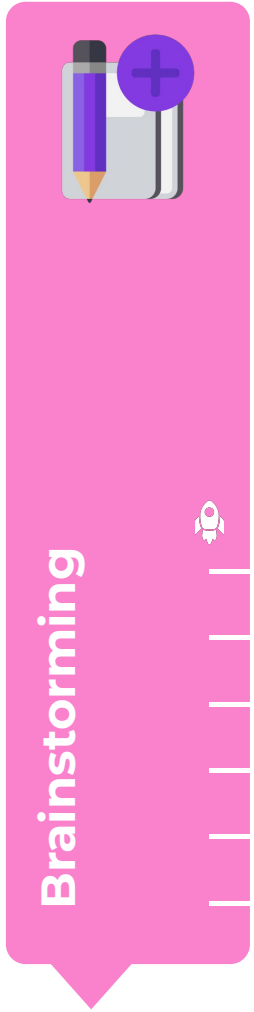
Set a **property of the window** — the number of the question displayed.

Give it the value -1 upon creation.

2

```
window.cur_question += 1
```

When next_question() is called, **increase** the counter **by one**.



2. Displaying the next question

Let's describe the function next_question() , which asks the next question.

```
def next_question()
```

To switch to the next question, we need to implement a **question counter.**

1

```
window.cur_question = -1
```

Set a **property of the window** — the number of the question displayed.

Give it the value -1 upon creation.

2

```
window.cur_question += 1
```

When next_question() is called, **increase** the counter **by one**.

3

If the list of questions has ended, **reset the counter** and start over.

Brainstorming

2. Displaying the next question

Let's put it all together into a function:

```
def next_question():
```

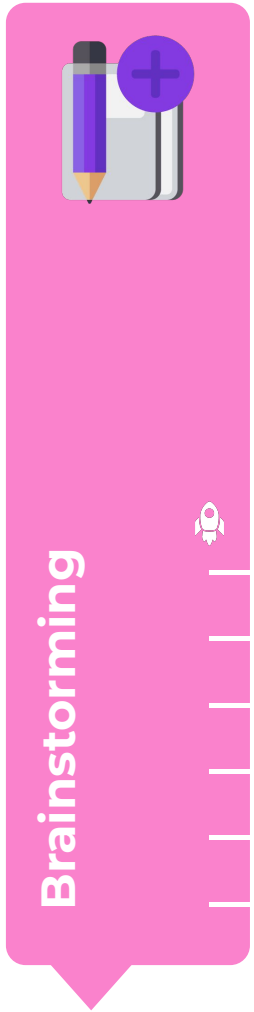
- ❑ When the function is called, increase the counter by 1.
- ❑ If the number of the current question is equal to the length of the list, reset the counter.
- ❑ Get a question from the list using a number.
- ❑ Ask the question using the ask() function.

Brainstorming

3. Checking and moving on

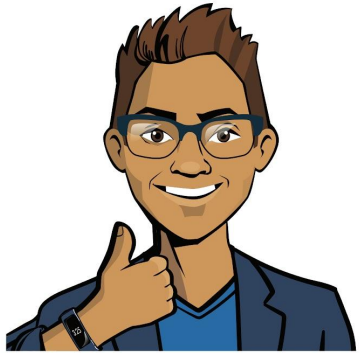
Let's implement our functionality in the program:

- The **Question** class and the set of questions
- The application interface
- The **ask()** function and accompanying elements
- The **check_answer()** function and accompanying elements (1)
- The **next_question()** function and accompanying elements (2)
- The **click_ok()** function, which calls (1) or (2)
- Application launch, **question counter = -1**
- *Button handling* by the function **click_ok()**



Expected result:

- The click_ok() function regulates the processes of displaying new questions and checking answers.
- The click_ok() function calls next_question() or check_answer().
- Questions from the set are asked in order. If the list has ended, the first question is asked again.



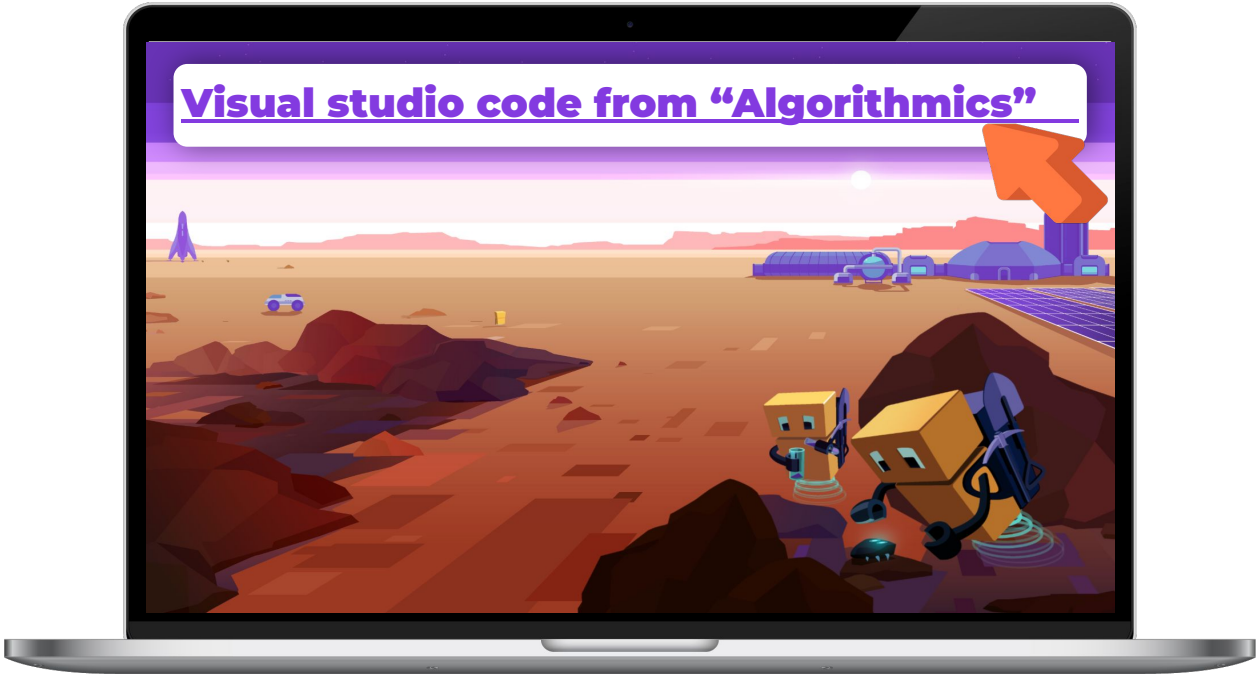
Brainstorming

Visual Studio Code:

Memory Card Application

Do the task in VS Code

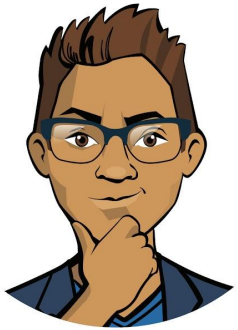
“VSC. PyQt. Memory Card”



Wrapping up the Work Day

To finish off the work day, complete this technical interview:

1. We need to program a new type of object and give it some properties. How do we do that?
2. Can we endlessly switch between different forms within the same interface? How is this issue resolved in Memory Card?



Cole,
Senior Developer

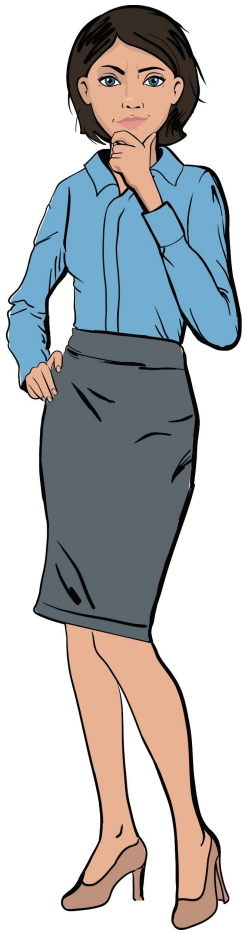
Emily,
Project Manager

Work Day Wrap-Up

Evaluating the effectiveness of today's work

With your colleagues, answer the questions:

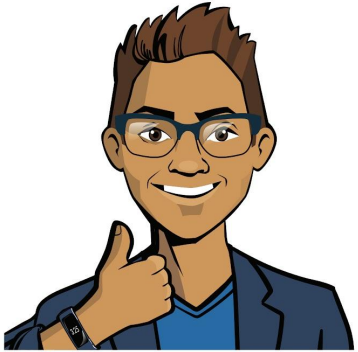
1. What went especially well?
2. What didn't go as planned?
3. What can you do to ensure success next time?



Work Day Wrap-Up

Additional tasks

- ❑ Take another look at the code you wrote.
- ❑ If necessary, finish writing the code.
- ❑ **Add comments to the code to explain** which part of the code does what.



Work Day Wrap-Up