Checking qualifications

Prove that you are ready for a brainstorm!

Demonstrate your knowledge of the basics of creating games in Pygame.

Checking qualifications

What is Pygame ?

What capabilities of Pygame do you know about?

Checking qualifications

Pygame is
a library used to create games

We used to work only with the Python Standard Library.

Pygame has ***modules*** with ready-made tools for:

- ➔ handling in-game events;
- ➔ handling external events;
- ➔ configuring game timers;
- ➔ configuring game interfaces and sound effects,

and more.

Command	*Purpose*
`import pygame`	Enables all features of the Pygame library

Checking qualifications

How do you create a game scene ?

How do you set a colored background for it?

Checking qualifications

A game scene with a colored background

Note. The color can be set using the RGB palette.

Command	*Purpose*
`window = pygame.display.set_mode((500, 500))`	Creates a window with the following size: (width, length).
`window.fill(<color>)`	Fills the background with the specified color.
`pygame.display.update()`	Updates the content of the game window.
`clock = pygame.time.Clock()`	Creates a game timer.
`clock.tick(40)`	Sets the scene refresh rate to ~40 FPS.

Checking qualifications

What is the RGB palette ?

How do we get different colors from it?

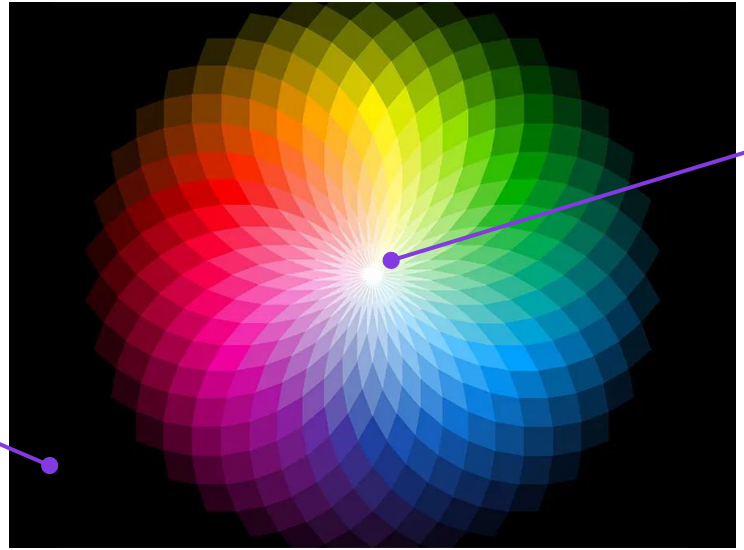
Checking qualifications

RGB color palette (red, green, blue)

We can use the RGB color palette.

We get the palette's colors by mixing red, green and blue. The absence of color appears as black.

All of the three basic colors are missing = black.



Maximum saturation of all three basic colors = white.

Link to the RGB color calculator

Checking qualifications

What is a game loop ?

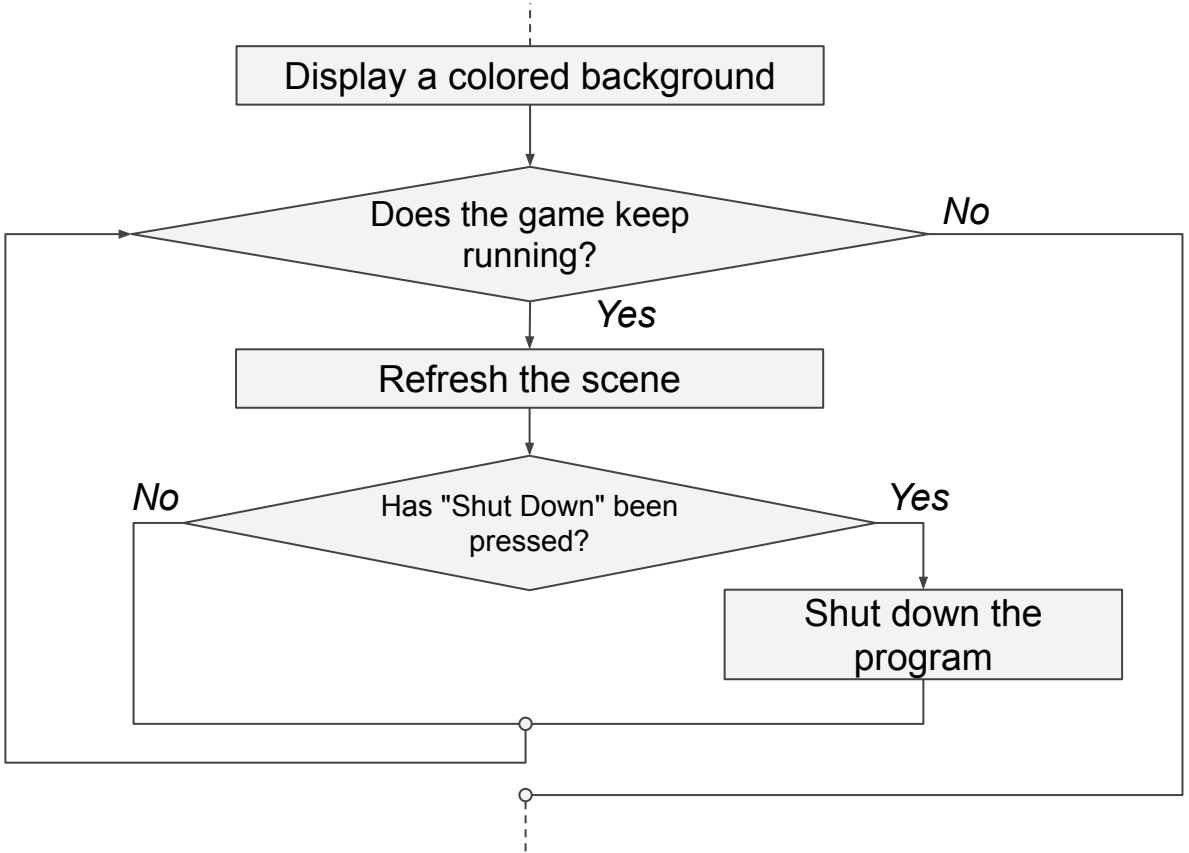
How do we create it?

What will be the condition for its termination ?

Checking qualifications

The flowchart for a game loop

The loop ends when you click on the "Shutdown" button



Checking qualifications

A game loop for a scene with a background

A very simple game template for displaying the scene without sprites:

- Connect Pygame modules
- Create a scene object
- Fill the scene with color ← The background can be white
- Create a game timer
- Game loop:
  - Set the frame rate to ~40 FPS
  - Update the scene (next frame of the game loop)

Checking qualifications

You already know two types of ready-made objects from pygame. State what they are.

How do we create them?

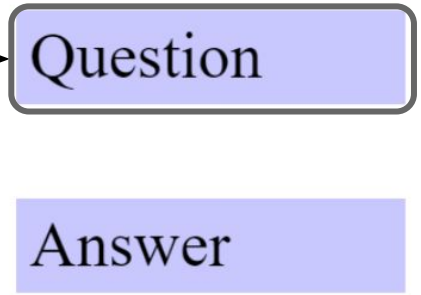
What are they used for?

Checking qualifications

Rectangular area (rect)

Command	*Purpose*
`rect = pygame.Rect(x, y, width, height)`	Creates a rectangle at the point (x, y) with a certain width and height
`pygame.draw.rect(mw, fill_color, rect)`	Draws a rectangle (rect) in the mw window and fills it with fill_color

We used rect to design the question and answer blocks.



Checking qualifications

Font and text

First, the ability to use Pygame objects is enabled, then the "font" object is created, and after that, we get the writing itself.

Command	*Purpose*
`pygame.init()`	Enables the ability to use commands for Pygame objects.
`font1 = font.Font(None, 70)`	Sets the font / Creates a font object with the parameters: font — default, size — 70.

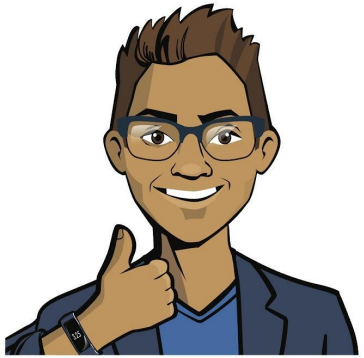
`question = font1.render(`
`  text, True, (255, 215, 0)`
`)`
	Creates a question with text, drawn using font1, color (255, 215, 0).

`mw.blit(question, (x, y))`	Displays the text at point (x, y) in the mw window.

Checking qualifications

Qualifications confirmed!

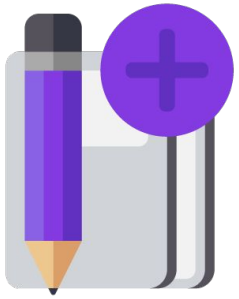
Great, you are ready to brainstorm and to do your work tasks!



Confirmation of qualifications

Brainstorm:

Lists



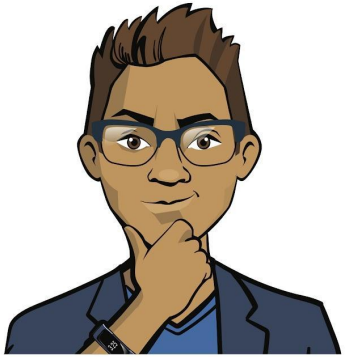
Training for how to work with lists

A list is another Python data type. It is used to store ordered datasets.

To work with lists, it is important to be able to:

1. Create a list.
2. Manage the contents of a list
   (get values, add elements, etc.).

Before you start using lists in your projects, you will need to pass a special training course.



Brain storm

A list is a structure for storing different types of data in a well-ordered manner .

Example. A list of results from a tournament of the online game "Space Shooter".

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

List name

List items

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

List item numbers

Brain storm

A list is a structure for storing different types of data in a well-ordered manner .

Example. A list of results from a tournament of the online game "Space Shooter".

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

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

→ Best result: 181

Get a list item by its number (index)

Brain storm

Methods of working with lists

What do you need to do?	*Which method should you use?*
Data storage and printing in whole and in parts	**Create a** list, **get a** list item, **print a** list
Add new data and delete unnecessary data	**Add** and **remove** an item, **clear** the entire list
Search for the necessary data in a set	**Search** for occurrences of an item in a list
Data processing for different purposes	**Sort** list items, **iterate** through items in a loop, determine the **length** of the list

Brain storm

Methods of working with lists

What do you need to do?	*Which method should you use?*
Data storage and printing in whole and in parts	**Create a** list, **get a** list item, **print a** list
Add new data and delete unnecessary data	**Add** and **remove** an item, **clear** the entire list
Search for the necessary data in a set	**Search** for occurrences of an item in a list
Data processing for different purposes	**Sort** list items, **iterate** through items in a loop, determine the **length** of the list

Let's take a look at methods for specific tasks.

Brain storm

Working with lists

Task 1a. You need to register your team to participate in the online tournament. Write a program for recording. You need to:

- request input of the participants' names and add them to the group.
- after completing the input, print the list for the group.

Possible view of the program:

```
Enter the participant's last name (0 – stop entering)
>>> Smith
Enter the participant's last name (0 – stop entering)
>>> Jones
Enter the participant's last name (0 – stop entering)
>>> Ray
Enter the participant's last name (0 – stop entering)
>>> 0
The group is typed: ['Smith', 'Jones', 'Ray']
```

What do you need to know how to do to create this kind of program?

Brain storm

Working with lists

Task 1a. You need to register your team to participate in the online tournament. Write a program for recording. You need to:

- request input of the participants' names and add them to the group.
- after completing the input, print the list for the group.

Functions and methods that will be useful to us:

Function	*Purpose*
`participants = list()`	Declare an empty list
`participants.append('Smith')`	Add an item to the end of the list
`print(participants)`	Print the entire list

Brain storm

Working with lists

Task 1a. You need to register your team to participate in the online tournament. Write a program for recording. You need to:

- request input of the participants' names and add them to the group.
- after completing the input, print the list for the group.

```
participants = list()
participant = input("Enter the participant's last name (0 - stop entering)")
while participant != '0':
    participants.append(participant)
    participant = input("Enter the participant's last name (0 - stop entering)")
print('The group is typed:', participants)
```

Brain storm

Working with lists

Task 1b. The tournament organizer has asked us to finalize the program.

- Each team's list should initially have a moderator.
- A safeguard against carelessness: if the entered last name is already in the list, then do not add it again.
- Once input is finished, sort the last names alphabetically.

Possible view of the program:

```
Enter the participant's last name (0 – stop entering)
>>> Smith
Enter the participant's last name (0 – stop entering)
>>> Jones
Enter the participant's last name (0 – stop entering)
>>> Ray
Enter the participant's last name (0 – stop entering)
>>> 0
The group is typed: ['(Moderator)', 'Jones', 'Ray', 'Smith']
```

What do you need to know how to do to create this kind of program?

Brain storm

Working with lists

Task 1b. The tournament organizer has asked us to finalize the program.

- Each team's list should initially have a moderator.
- A safeguard against carelessness: if the entered last name is already in the list, then do not add it again.
- Once input is finished, sort the last names alphabetically.

We will need these tools:

Command	*Purpose*
`participants = ['(Moderator)']`	Declare a list with elements
`'Johnson' in participants`	Search for the occurrence of an element in the list (returns True or False)
`participants.sort()`	Sort the list in lexicographic order (by ascending numbers and letters of the alphabet)

Brain storm

Working with lists

Task 1b. The tournament organizer has asked us to finalize the program.

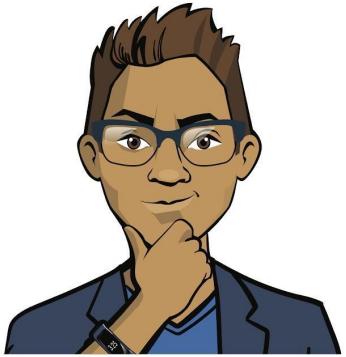
- Each team's list should initially have a moderator.
- A safeguard against carelessness: if the entered last name is already in the list, then do not add it again.
- Once input is finished, sort the last names alphabetically.

```
participants = ['(Moderator)']
participant = input("Enter the participant's last name (0 - stop entering)")
while participant != '0':
  if participant in participants:
      print('This participant has already been recorded!')
  else:
      participants.append(participant)
  participant = input("Enter the participant's last name (0 - stop entering)")
participants.sort()
print('The group is typed:', participants)
```

Brain storm

Before moving on to another task:

1. Last names were entered by the user: "Williams", "Brown", "Davis", "Smith". What will the program print?
2. How can we refine the program and limit the possible input to ten surnames?



Brain storm

Working with lists

Task 2. The results of all the teams are analyzed at the end of the tournament. The analysts group should:

- calculate and print the average result for the tournament;
- determine the result of the winning team.

Possible view of the program:

What do you need to know how to do to create this kind of program?

Brain storm

Working with lists

Task 2. The results of all the teams are analyzed at the end of the tournament. The analysts group should:

- calculate and print the average result for the tournament;
- determine the result of the winning team.

We will need these tools:

Command	*Purpose*

`for result in results:`
`    Team1`
`    Team2`
	Iterate through the `results` list items.
	
	“For each item (`result`) of the list (`results`), execute Command1, Command2”

`len(results)`	Determining the length of the `results` list

Brain storm

Working with lists

Task 2. The results of all the teams are analyzed at the end of the tournament. The analysts group should:

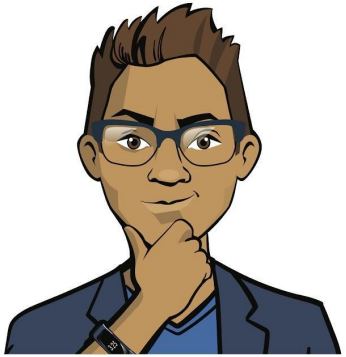
- calculate and print the average result for the tournament;
- determine the result of the winning team.

```
results = [225, 220, 199, 263, 259, 225, 226]
average_result = 0
max_result = 0
for result in results:
  average_result += result
  if max_result < result:
      max_result = result
average_result = average_result / len(results)
print('Results:', results)
print('Average result:', average_result)
print('Maximum result:', max_result)
```

Brain storm

Before moving on to another task:

1. What will the program print if you load another data set: [200, 204, 202]? What will be the value of the max_result variable at each stage of the loop?
2. How can we change the program so that it prints the minimum result instead of the maximum?



Brain storm

Planning work tasks

Task 3. Write a program that requests the input of an arbitrary number of round results, saves them to the archive (as a list), and prints the number of teams that have received more than 200 points.

How do we write a program like this? You know all the required methods.

Brain storm

Planning work tasks

Task 3. Write a program that requests the input of an arbitrary number of round results, saves them to the archive (as a list), and prints the number of teams that have received more than 200 points.

```
results = list()
amount_200 = 0
result = int(input('Enter result (0 - stop):'))
while result != 0:
  if result > 200:
     amount_200 += 1
  results.append(result)
  result = int(input('Enter result (0 - stop):'))
print('Round results:', results)
print('Advanced to next round:', amount_200)
```

```
Enter result (0 – stop):
>>> 220
Enter result (0 – stop):
>>> 211
Enter result (0 – stop):
>>> 198
Enter result (0 – stop):
>>> 196
Enter result (0 – stop):
>>> 0
Round results: [220, 211, 198, 196]
Advanced to next round: 2
```

Brain storm

Planning work tasks

Task 3. Write a program that requests the input of an arbitrary number of round results, saves them to the archive (as a list), and prints the number of teams that have received more than 200 points.

```
results = list()
amount_200 = 0
result = int(input('Enter result (0 - stop):'))
while result != 0:
  if result > 200:
     amount_200 += 1
  results.append(result)
  result = int(input('Enter result (0 - stop):'))
print('Round results:', results)
print('Advanced to next round:', amount_200)
```

The solution is correct, but the data entry could be simplified.

By the way, you know the method for simplifying it from another topic.

Brain storm

Planning work tasks

Task 3. Write a program that requests the input of an arbitrary number of round results, saves them to the archive (as a list), and prints the number of teams that have received more than 200 points.

```
results_list = results.split('')
```

This is the method that divides a string into parts with the specified separator. A list is compiled from the received parts.

Program	*Program will print*

`results = '220 211 198 185'`
`results = results.split(' ')`
`print(results)`
	['220', '211', '198', '185']

Brain storm

Planning work tasks

Task 3. Write a program that requests the input of an arbitrary number of round results, saves them to the archive (as a list), and prints the number of teams that have received more than 200 points.

```
results_list = results.split('')
```

This is the method that divides a string into parts with the specified separator. A list is compiled from the received parts.

Program	*Program will print*

`results = '220 211 198 185'`
`results = results.split(' ')`
`print(results)`
	['220', '211', '198', '185']

How can we redo the previous program?

Brain storm

Planning work tasks

Task 3. Write a program that requests the input of an arbitrary number of round results, saves them to the archive (as a list), and prints the number of teams that have received more than 200 points.

```
results = input('Enter the marks separated by a
space:')
results = results.split(' ')
amount_200 = 0
for result in results:
  if int(result) > 200:
     amount_200 += 1
print('Round results:', results)
print('Advanced to next round:', amount_200)
```

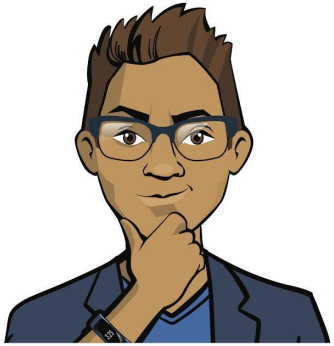
```
Enter the marks separated by a space:
>>> 300 350 289 140 135
Round results: ['300', '350', '289', '140', '135']
Advanced to next round: 3
```

Brain storm

Strings and lists

You may have noticed that strings and lists have a lot in common.

Thanks to the Python Interpreter, we can now use the same operators for working with multiple data types.



Brain storm

Strings and lists

The work for some methods is very similar:

Command	*Strings*	*Lists*
`marks1 + marks2`	Merging strings into one	Merging lists into one
`marks * 3`	Repeat string n-times	Repeat items in the list n-times
`len(marks)`	Calculating the length of a string (number of characters)	Calculating the length of a list (number of elements)
`marks[i]`	Getting a character by number	Getting an item by number
`marks.find('5')`	Searching for the occurrence of a substring in a string (returns the input number)	-
`'A' in marks`	Searching for the occurrence of a character in a string	Search for the occurrence of an item in a list

Brain storm

Strings and lists

There are no direct analogues for some methods:

Command	*Strings*		*Lists*	
Add an element	-		`marks = [5, 4]` `marks.append(3)`	`[5, 4, 3]`
Delete an element by value	-		`marks = [5, 4, 3]` `marks.remove(5)`	`[4, 3]`
Replace an element with another element	`marks = '545'` `m = marks.replace('4','5')`	'555'	-	

The current string does not change, a new one is created.

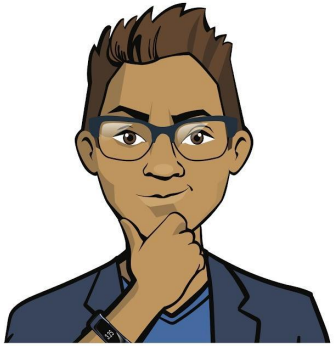
The current list changes.

Brain storm

Your tasks:

➔ Log on to the platform and go through the practical part of the training. Use the documentation if you need to.

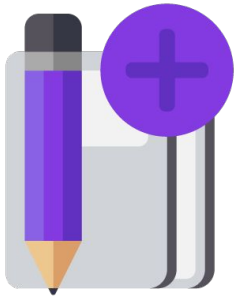
If you have any time left over, study the additional tasks "Lists: additional tasks".



Brain storm

Brainstorm:

Storing data



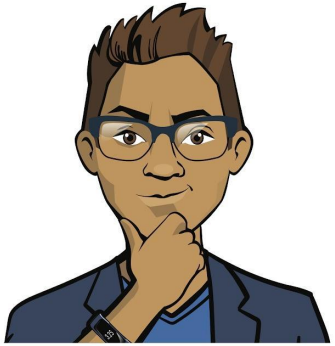
Revising the "Questions and Answers" game

We can use lists to optimize the storage of questions and answers in the game. You need to understand:

1. Where will the question and answer sets be stored?
2. How will a random question/answer be displayed in the game loop?

There are different solutions available.

Let's choose one of them and analyze how it's implemented.

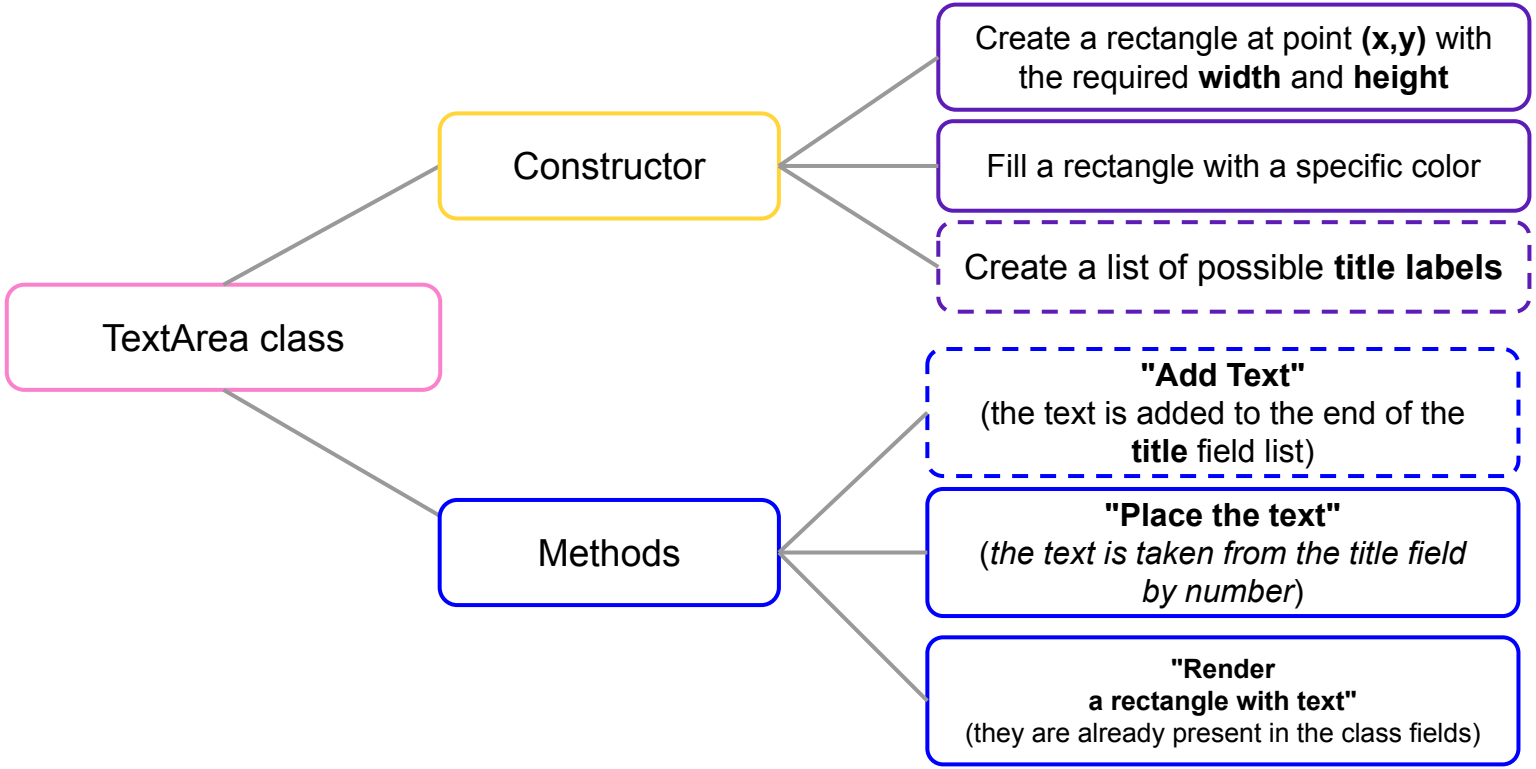


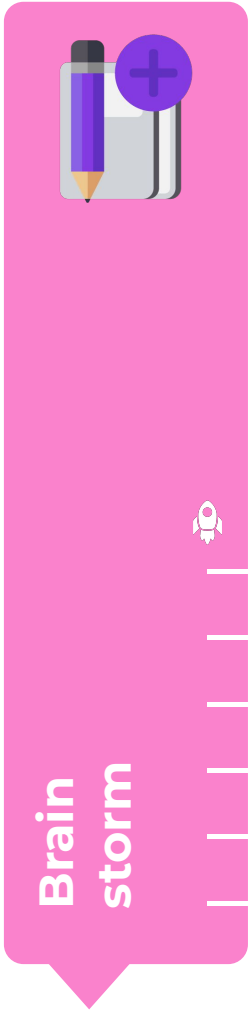
Brain storm

TextArea class

Let's add a list of possible block labels to the class.

To maintain the list, add the add_text() method and change set_text().

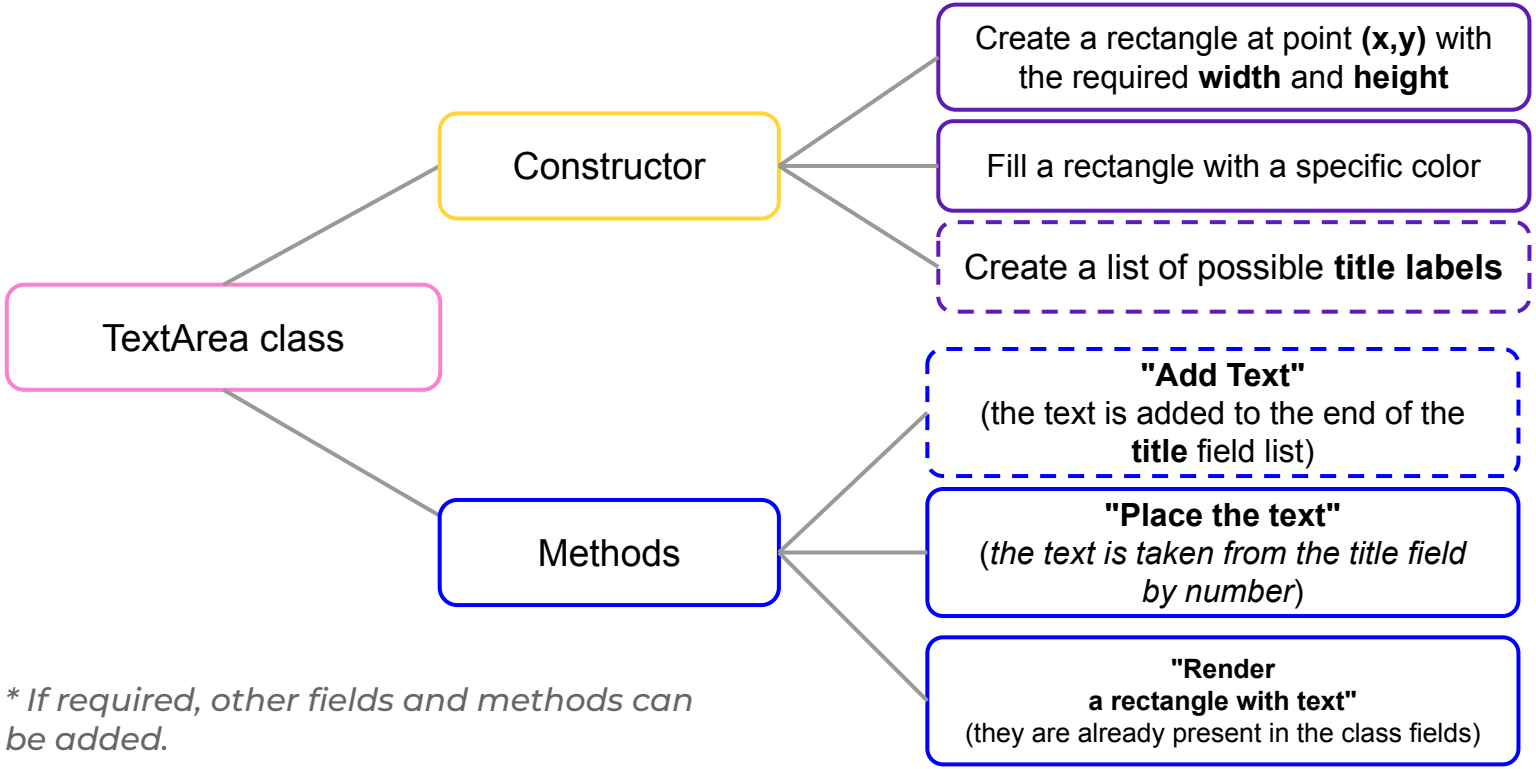




TextArea class

Let's add a list of possible block labels to the class.

To maintain the list, add the add_text() method and change set_text().



** If required, other fields and methods can be added.*

Brain storm

1. Filling in the list of formulations

When creating a TextArea instance for a block with text, you can add possible formulations straightaway:

Creating a TextArea instance

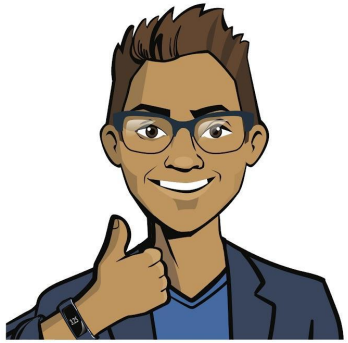
Adding formulation 1

Adding formulation 2

. . .

Adding formulation n

Let's have the starting "Question" template in 0 place.

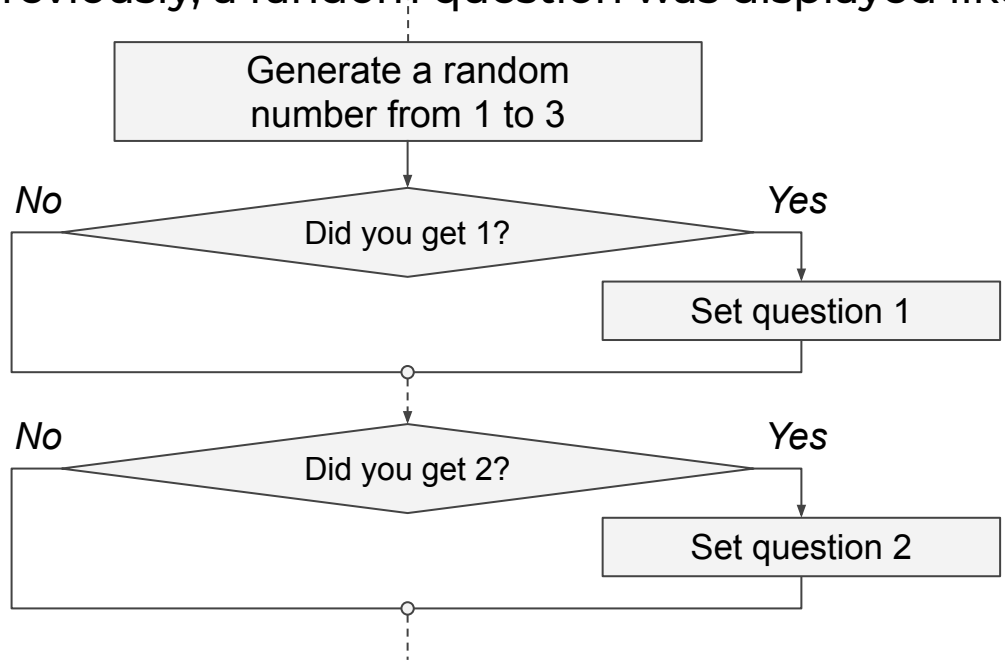


Brain storm

2. Displaying a random formulation

Let's make it so that a question needs to be shown when the Q key is pressed.

Previously, a random question was displayed like this:

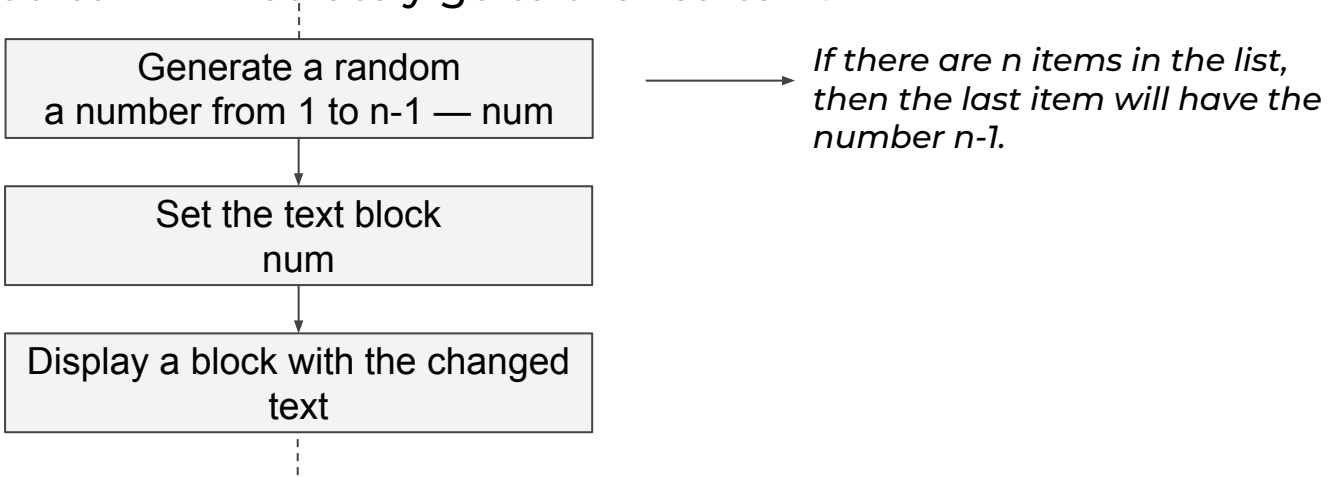


Brain storm

2. Displaying a random formulation

Let's make it so that a question needs to be shown when the Q key is pressed.

Now you can immediately go to the list item.



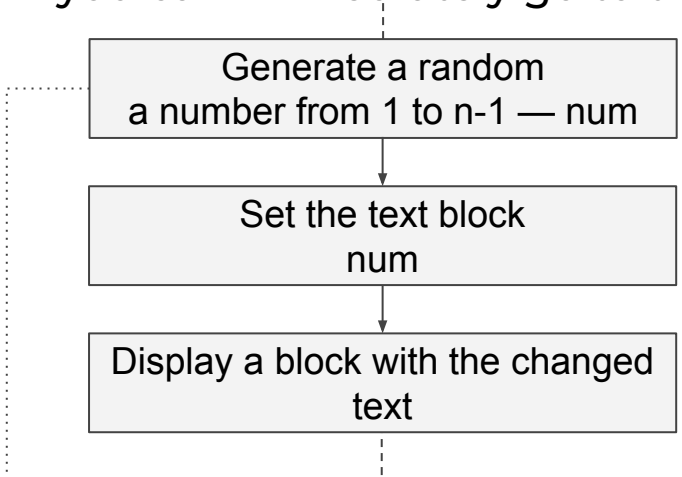
How do you generate a number from 1 to n-1, where n is the length of the list?

Brain storm

2. Displaying a random formulation

Let's make it so that a question needs to be shown when the Q key is pressed.

Now you can immediately go to the list item.



If there are n items in the list, then the last item will have the number n-1.

```
num = randint(1, len(quest_card.titles)-1)
```

Brain storm

Your tasks:

- ➔ Change the way data is stored in the "Questions and Answers" game. To do this, add a field with a list and methods for working with it to the TextArea class.
- ➔ Add at least 5 formulated questions and answers to the game. *Try to be original!*
- ➔ Launch and test the game.

Brain storm