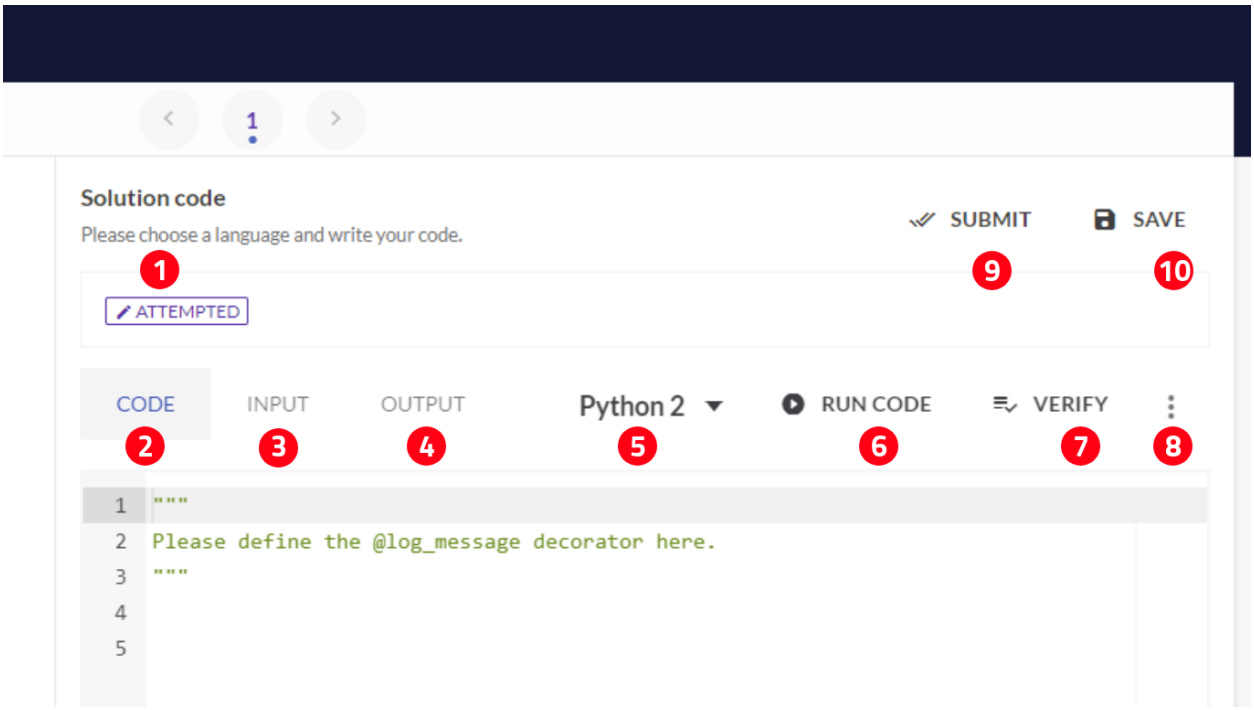
General Guidelines (Coding Environment)

Once you have gone through the process of starting the test and clicking on some Coding problem, the given window shows up on your screen. The various functionalities have been marked as such:



1. **Evaluation Status of the Solution:** This displays the evaluation status of your solution to the given problem. Possible statuses are:

    a. Unsolved
    b. Submitted
    c. Partially Correct
    d. Accepted
    e. Rejected

2. **CODE:** This tab is where you write your code in the DoSelect IDE.

3. **INPUT:** Before Submitting your code for evaluation, you can try out your solution by providing custom test cases under this tab.

4. **OUTPUT:** For the custom test cases provided by you under the INPUT tab, this OUTPUT tab lists out the output for the same.

5. **List of Languages:** This drop-down list contains all the languages that have been enabled for the mentioned Coding problem. If it is a generic problem, you will see all the languages that are supported by the DoSelect IDE.

6. **RUN CODE:** This is the button that you click on if you want to evaluate your solution against your own custom test cases.

7. **VERIFY:** If you are satisfied with your custom test cases, you can go ahead and click this button to test your solution against the sample test case, which has been provided with the problem.

8. **More Options:** Contains two options:

    a. **Load Language Stub:** Loads the default boilerplate code template and replaces any text written in the IDE.
    b. **Toggle night mode:** Used to toggle between night mode and normal mode.

9. **SUBMIT:** After you are satisfied with running the solution against your custom test cases and the sample test cases, you can Submit your code to go for evaluation against actual test cases. The evaluation of your solution based on the actual test cases is what produces your score.

10. **SAVE:** You can save your code to save all changes that you have made to your solution for the given problem.

How to Use the IDE

Here, we take an example and demonstrate how to use the DoSelect IDE.

Problem Description

You are given a list of an employee working in a daily-wages firm. In this firm, employees aren't permanent i.e. they keep changing their job profile every day (because of daily-wages).
Because of this, it is becoming difficult for the manager to track the progress of each project/employee.

After some brainstorming, the manager resolves that if he can replace the name of the employee associated with each project then it would be easy for him to track. Since manager comes from a non-programming background so he needs your help.

All you need to do is, given a list consisting of the names of employees, you need to replace each occurrence of given employee name with another employee name.

To fulfill the requirement, you have to complete the body of ***replaceEmployee*** method in the class given in the editor.

Input Format

List of employees, Name of the employee to be replaced, Name of the employee coming after replacement.

Output Format

The final list of employees.

Sample Input

["Samar", "Reyansh", "Parinaaz", "Vanya"], Samar, Faiyaz

Sample Output

["Faiyaz", "Reyansh", "Parinaaz", "Vanya"]

Explanation

In the given list of four employees, Faiyaz comes in place of Samar.

Note: The Input and Output are just for the understanding purpose, you have to complete the ***replaceEmployee*** function.

Code Editor:

```
import java.util.*;
class Source
{
	public ArrayList<String> replaceEmployee(ArrayList<String> employee, String replaceThis, String
replaceWith)
	{

		// Implement your code here

	}
}
```

- As has been demonstrated through the above, you need to write your code in place of // Implement your code here.
- When writing in the DoSelect IDE, the main class should always be named as **Source.**
- If it has not been explicitly mentioned in the **Problem Description**, you SHOULD NOT write the main() function; as it is implicitly implemented in the DoSelect IDE.
- A **Sample Test Case** is given along with the **Input/Output Format**. This is for your understanding to get a grasp of what logic you need to implement in order to reach the desired output.
- The below given **Sample Solution** demonstrates all the above points.

Sample Solution

```
import java.util.*;
class Source
{

    public ArrayList<String> replaceEmployee(ArrayList<String> employee, String replaceThis, String
replaceWith)
    {
        int index = employee.indexOf(replaceThis);    // Code Implementation Starts
        while (index != -1)
        {
                employee.set(index, replaceWith);
                index = employee.indexOf(replaceThis);
        }
        return employee;   //Code Implementation Ends
    }
}
```