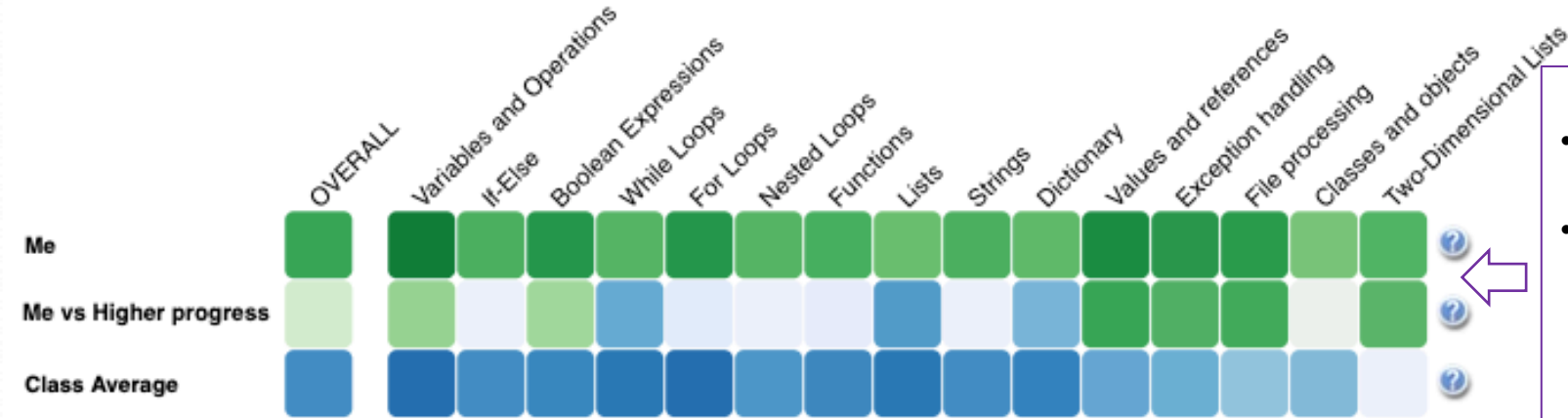


### Compared Group Explanation:

Your progress is compared to the average progress of students in the **higher half of the class** in terms of their **progress** in this system.

## Me and group (Higher progress students)



### Progress Visualization

- First row (Me) shows **your progress** (Darker green means more progress on that topic)
- Second row (Me vs group) **compares your progress with only higher progress students** (Darker green means you have more progress than the group; darker blue means they have more progress than you; grey means equal progress).
- Third row (Group) shows the **average progress of 'higher progress students'** (Darker blue means more progress on that topic)

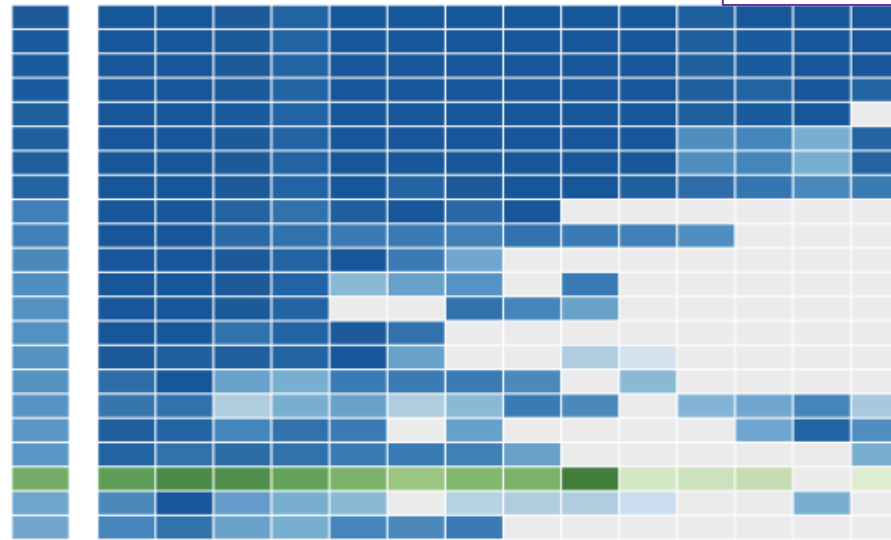
Load the rest of learners

Higher progress students you are 20th out of 69)

Click the button above to load the list of other students (anonymized) and shows in which position you are in terms of progress.

**Note:** If you are not among the higher progress students (higher half of the class), your ranking will not be shown here. You need to increase your progress in the system to enter this ranking.

20. Me ->



To have more greener cells on ***Me*** row, you need to interact with the learning activities inside each topic.

Click on a topic cell as shown below and access the contents. Viewing animation steps, clicking on example lines or solving challenges, questions and parsons problems to increase your progress.

Play animation steps to how the program executed

The screenshot shows a Jupyter Notebook interface. At the top, a code cell contains the following Python code:

```

1 account1 = 2540
2 account2 = 13250
3 price = 3400
4
5 can_afford = account1 >= price or account2 >= price
6 can_afford = account2 >= price or account1 >= price
7
8 account2 = account2 - price
9
10 money_left = account1 > 0 and account2 > 0
11 limit_exceeded = account1 < 0 and account2 < 0

```

Below the code cell, a "Stack" frame is visible, showing the current state of the program:

```

Stack frame
account1
2540
13250

```

To the right of the stack frame, a "Text console" is visible, displaying the output of the code cell:

```

Text console
>= or - + and
<

```

At the bottom of the interface, a status bar indicates "Fetching value 13250 - ready." and a set of navigation icons (back, forward, search, etc.) is visible.

Check how a program is constructed line by line in examples and challenge yourself with challenges and complete the missing lines.

### Example: Determining When to Buy a New Phone (Case 1)

Construct a program that determines whether it is time to buy a new phone based on the inputs that it receives from the user. A new phone should be bought if the phone breaks or the phone is at least 3 years old.

### Challenge: Determine When a Student Fails a Course (Case 2)

Construct a program that determines whether a student fails the course based on the inputs that it receives from the instructor. The student fails the course if the exam score is less than 55 or when the student has cheated.

```

1 #Step 1: Read the user inputs
2 text = input("Enter the phone age in years:")
3 phone_age = int(text)
4 text = input("Enter 1 if the phone is broken, otherwise enter 0:")
5 input_num = int(text)
6 #Step 2: Determine whether the phone is broken
7 if input_num == 1:
8     is_broken = True
9 else:
10    is_broken = False
11 #Step 3: Write the boolean expression to determine whether the student
12 #Step 4: Print the result
13 if need_phone == True:
14    print("Yes! It is time to buy a new phone")
15 else:
16    print("No! It is not yet the time to buy a new phone")

```

To determine whether the phone is broken, we need to compare the integer

Back

Drag a tile from here

CHECK

```

is_failing = exam_score < 55 or not has_cheated
is_failing = exam_score < 55 or has_cheated
is_failing = exam_score < 55 and not has_cheated

```

```

1 #Step 1: Read the instructor inputs
2 text = input("Enter the exam score:")
3 exam_score = int(text)
4 text = input("Enter 1 if the student has cheated, otherwise enter 0:")
5 input_num = int(text)
6 #Step 2: Determine whether the student has cheated
7 if input_num == 1:
8     has_cheated = True
9 else:
10    has_cheated = False
11 #Step 3: Write the boolean expression to determine whether the student fails the course
12 is_failing = not ( exam_score <= 55 ) or not has_cheated
13 #Step 4: Print the result
14 if is_failing == True:
15    print("Yes! The student fails the course.")

```

Drag a tile to each missing field to construct this program.

is\_failing = not ( exam\_score <= 55 ) or not has\_cheated

Predict the output of the program. It is either the console output or the value of **result** variable.

```
Tester.py

account1 = 186
account2 = 186 + 50
price = 250
can_afford = account1 >= price or account2 >= price
account2 = account2 - price
money_left = account1+account2 > 0
result = money_left
```

What is the final value of **result**?

**CORRECT!**

**Your Answer is:**  
True

**Correct Answer is:**  
True

[Try Again](#)

Reorder the program lines to solve the given task at the bottom of the screen. Pay attention to indentation.

Drag from here

else:

elif input\_a == 0 and input\_b == 0:

print(??)

print(??)

Construct your solution here

```
if input_a == 1 and input_b == 1:  
    print(??)
```