DATA STRUCTURES AND ALGORTHM

Assignment 04

University Of Rwanda (UR)
College of Business and Economics (CBE)
Business and Information Technology (BIT)
Year 2

Elisa Dushimimana
Elisadushimimana311@gmail.com
2240015036

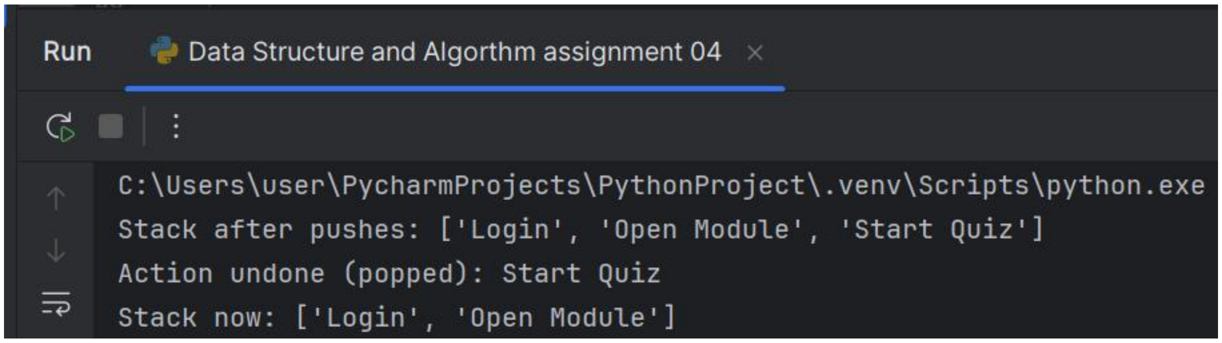
Project 19

Stack Questions:

✓ Practical (Rwanda): In Canvas, push ["Login", "Open Module", "Start Quiz"]. Pop one. Which is undone?

```
# Stack implementation using Python list
canvas_stack = []

# Push actions onto the stack (like stacking books)
canvas_stack.append("Login")          # First action
canvas_stack.append("Open Module")    # Second action
canvas_stack.append("Start Quiz")     # Third action

# Pop one action (undo the last action)
undone_action = canvas_stack.pop()

# Screenshot-style output
print("Stack after pushes: ['Login', 'Open Module', 'Start Quiz']")
print(f"Action undone (popped): {undone_action}")
print(f"Stack now: {canvas_stack}")
```

Output



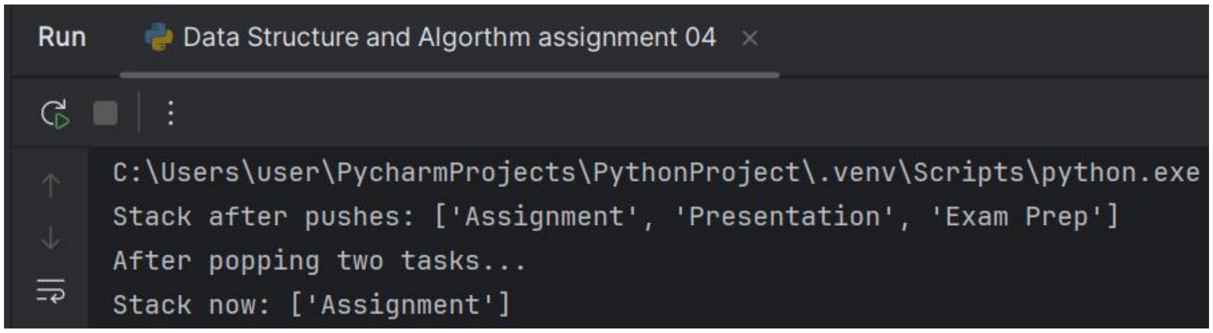
```
C:\Users\user\PycharmProjects\PythonProject\.venv\Scripts\python.exe
Stack after pushes: ['Login', 'Open Module', 'Start Quiz']
Action undone (popped): Start Quiz
Stack now: ['Login', 'Open Module']
```

✓ Practical (Rwanda): UR student pushes ["Assignment", "Presentation", "Exam Prep"]. Pop two. What remains?

```
✓ # Stack implementation using Python list
  student_stack = []

  # Push tasks onto the stack
```

```
    student_stack.append("Assignment")
    student_stack.append("Presentation")
    student_stack.append("Exam Prep")

    # Pop two tasks (remove the last two added)
    student_stack.pop()  # Removes "Exam Prep"
    student_stack.pop()  # Removes "Presentation"

    # Screenshot-style output
    print("Stack after pushes: ['Assignment', 'Presentation', 'Exam Prep']")
    print("After popping two tasks...")
    print(f"Stack now: {student_stack}")
```

Output



```
C:\Users\user\PycharmProjects\PythonProject\.venv\Scripts\python.exe
Stack after pushes: ['Assignment', 'Presentation', 'Exam Prep']
After popping two tasks...
Stack now: ['Assignment']
```

✓ *Challenge: Reverse list ["One", "Two", "Three", "Four"] using stack.*

```
# Challenge: Reverse a list using a stack

original_list = ["One", "Two", "Three", "Four"]
stack = []

# Step 1: Push all items onto the stack
for item in original_list:
    stack.append(item)  # Each append puts the item on top of the stack

# Step 2: Pop all items from the stack to get them in reverse order
reversed_list = []
while stack:
    reversed_list.append(stack.pop())  # Pop removes the top item

# Screenshot-style output
print(f"Original list: {original_list}")
print(f"Reversed list: {reversed_list}")
```

- ✓ Reflection: Why can't stack manage first-come-first-serve processes?

A stack operates on a Last-In-First-Out (LIFO) principle, meaning the last item added is the first to be removed. In real life, this is like a stack of plates: you always take the top plate first.

First-Come-First-Serve (FCFS) processes, like people waiting in line for a bus, require the first person to arrive to be the first served. This is the opposite of how a stack works. If you used a stack for FCFS, the last person to arrive would be served first, which is unfair and not how queues work.