

**“AZƏRBAYCAN HAVA YOLLARI” CJSC NATIONAL AVIATION ACADEMY**

**Individual Work № 3:**

**Topic:** Lists. Sets. Stacks. Queues. Time complexity of different operations.

**Subject: Obyektyönümlü proqramlaşdırma**

**Teacher: Mamed Shahmaliyev**

**Group: 1459i Student: Azadaliyev Rustam**

**Date: 11.10.2021 Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**1) Using list**  
**Stack** works on the principle of “Last-in, first-out”. Also, the inbuilt functions in Python make the code short and simple. To add an item to the top of the list, i.e., to push an item, we use **append()** function and to pop out an element we use **pop()** function. These functions work quiet efficiently and fast in end operations.

Let’s look at an example and try to understand the working of push() and pop() function:  
Example:

|  |
| --- |
| # Python code to demonstrate Implementing  # stack using list  stack = ["Amar", "Akbar", "Anthony"]  stack.append("Ram")  stack.append("Iqbal")  print(stack)    # Removes the last item  print(stack.pop())    print(stack)    # Removes the last item  print(stack.pop())    print(stack) |

**Output:**

['Amar', 'Akbar', 'Anthony', 'Ram', 'Iqbal']

Iqbal

['Amar', 'Akbar', 'Anthony', 'Ram']

Ram

['Amar', 'Akbar', 'Anthony']

**Queue** works on the principle of “First-in, first-out”. Below is list implementation of queue. We use pop(0) to remove the first item from a list.

|  |
| --- |
| # Python code to demonstrate Implementing  # Queue using list  queue = ["Amar", "Akbar", "Anthony"]  queue.append("Ram")  queue.append("Iqbal")  print(queue)    # Removes the first item  print(queue.pop(0))    print(queue)    # Removes the first item  print(queue.pop(0))    print(queue) |

**Output:**

['Amar', 'Akbar', 'Anthony', 'Ram', 'Iqbal']

Amar

['Akbar', 'Anthony', 'Ram', 'Iqbal']

Akbar

['Anthony', 'Ram', 'Iqbal']

**2) Using Deque**  
In case of stack, list implementation works fine and provides both append() and pop() in O(1) time. When we use deque implementation, we get same time complexity.

|  |
| --- |
| # Python code to demonstrate Implementing  # Stack using deque  from collections import deque  queue = deque(["Ram", "Tarun", "Asif", "John"])  print(queue)  queue.append("Akbar")  print(queue)  queue.append("Birbal")  print(queue)  print(queue.pop())  print(queue.pop())  print(queue) |

**Output:**

deque(['Ram', 'Tarun', 'Asif', 'John'])

deque(['Ram', 'Tarun', 'Asif', 'John', 'Akbar'])

deque(['Ram', 'Tarun', 'Asif', 'John', 'Akbar', 'Birbal'])

Birbal

Akbar

deque(['Ram', 'Tarun', 'Asif', 'John'])

But when it comes to queue, the above list implementation is not efficient. In queue when pop() is made from the beginning of the list which is slow. This occurs due to the properties of list, which is fast at the end operations but slow at the beginning operations, as all other elements have to be shifted one by one.  
So, we prefer the use of dequeue over list, which was specially designed to have fast appends and pops from both the front and back end.

Let’s look at an example and try to understand queue using collections.deque:

|  |
| --- |
| # Python code to demonstrate Implementing  # Queue using deque  from collections import deque  queue = deque(["Ram", "Tarun", "Asif", "John"])  print(queue)  queue.append("Akbar")  print(queue)  queue.append("Birbal")  print(queue)  print(queue.popleft())  print(queue.popleft())  print(queue) |

**Output:**

deque(['Ram', 'Tarun', 'Asif', 'John'])

deque(['Ram', 'Tarun', 'Asif', 'John', 'Akbar'])

deque(['Ram', 'Tarun', 'Asif', 'John', 'Akbar', 'Birbal'])

Ram

Tarun

deque(['Asif', 'John', 'Akbar', 'Birbal'])