## Лабораторная работа №10. Очереди

Очередь (англ. queue) • абстрактный тип данных с возможностью доступа к элементам «первый пришёл — первый вышел» (FIFO, First In — First Out). • Добавление элемента (enqueue — поставить в очередь) возможно лишь в конец очереди. • Выборка — только из начала очереди (dequeue — убрать из очереди), при этом выбранный элемент из очереди удаляется.

**Создание очереди Queue в Python**

```
from queue import Queue
q = Queue()
q.put('eat')
q.put('sleep')
q.put('code')
print(q) # <queue.Queue object at 0x1070f5b38> print(q.get()) # 'eat'
print(q.get()) # 'sleep'
print(q.get()) # 'code'
print(q.get_nowait()) # queue.Empty q.get()
```

**Создание multiprocessing.Queue в Python**

```
from multiprocessing import Queue
q = Queue()
q.put('eat')
q.put('sleep')
q.put('code')
print(q)
# <multiprocessing.queues.Queue object at 0x1081c12b0>
print(q.get())  # 'eat'
print(q.get())  # 'sleep'
print(q.get())  # 'code'
```

**Объявление и заполнение очереди**

```
from collections import deque
q = deque()
append
q.('eat')
q.append('sleep')
q.append('code')
print(q)
print(q.popleft()) # 'eat'
print(q.popleft()) # 'sleep'
print(q.popleft()) # 'code'
# deque(['eat', 'sleep', 'code'])
```

**Объявление очереди**

```
from collections import deque
fruit_list = deque(['Apple', 'Mango', 'Peaches', 'Banana', 'Papaya'])
import collections
my_deque = collections.deque([10, 20, 30, 40, 50])
```

**Наличие значения в очереди**

```
if 30 in my_deque:
    print('30 is in queue')
```

**Перебор всех значений**

```
for x in my_deque:
    print(x)
```

**Цикл по очереди**

```
import collections
my_deque = collections.deque([10, 20, 30, 40, 50])
mine_deque = collections.deque()
for x in my_deque:
    mine_deque.append(x)
print(mine_deque)
```

**Методы Deque**

**Добавление элемента в конец**

```
d = deque()
d.append(1)
d.append(2)
print(d)  # deque([1, 2])
```

**Добавление элемента в начало**

```
d.appendleft(0)
print(d)  # deque([0, 1, 2])
```

**Удаление и возвращение последнего элемента**

```
last_element = d.pop()
print(last_element)  # 2
print(d)  # deque([0, 1])
```

**Удаление и возвращение первого элемента**

```
first_element = d.popleft()
print(first_element)  # 0
print(d)  # deque([1])
```

**Добавление нескольких элементов в конец**

```
d.extend([2, 3])
print(d)  # deque([1, 2, 3])
```

**Удаление первого вхождения**

```
d.remove(2)
print(d)  # deque([0, -1, 1, 3])
```

**Сдвиг элементов на указанное количество позиций. Положительное число сдвигает вправо, отрицательное — влево.**

```
d.rotate(1)
print(d)  # deque([3, 0, -1, 1])
d.rotate(-1)
print(d)  # deque([0, -1, 1, 3])
```

**Удаление всех элементов**

```
d.clear()
print(d)  # deque([])
```

**Задания для самостоятельной работы**

1. С клавиатуры вводится последовательность натуральных чисел, ограниченная нулем. Значение чисел не превышают 999. Вывести значения поступивших чисел по группам: сначала одноразрядные, затем двухразрядные и в конце трехразрядные. Порядок следования чисел в каждой группе не должен изменяться по сравнению с порядком ввода чисел.
2. Очереди q1 и q2 содержат данные типа integer. Выполнить слияние, получив очередь q, содержащую все элементы очередей q1 и q2, расположенные в неубывающем порядке.
3. Реализуйте функцию, которая принимает список чисел и создает очередь (deque), добавляя элементы в порядке их приоритета (меньшее значение имеет больший приоритет).
4. Напишите функцию, которая принимает список и возвращает его обратный вариант, используя deque.
5. Реализуйте функцию, которая проверяет, является ли строка палиндромом, используя deque.

**Домашняя работа**

1. Реализуйте конвейер, который обрабатывает задачи. Задачи поступают в deque. При обработке каждая задача может добавлять в очередь новую задачу. После обработки задача должна быть удалена из очереди.
2. Создайте функцию, которая принимает список чисел и целое число k, и возвращает k наибольших чисел, используя deque. Элементы должны сохранять порядок появления в списке.
3. Напишите функцию, которая возвращает последние n элементов списка, используя deque.
4. Постройте последовательность Фибоначчи, используя deque для хранения последних двух значений.