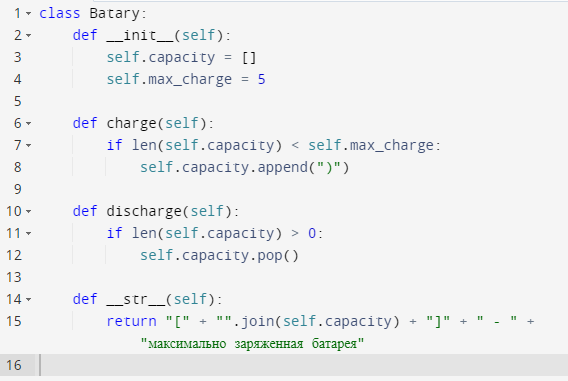
**1.**

**Задание**: Создайте класс Batary, у которой будет определен атрибут capacity = [ ] (емкость), max\_charge = 5 (максимальный заряд) по умолчанию, и методы:

* charge - заряжает батарею
* discharge - разряжает батарею.
* Переопределите метод \_\_str\_\_, чтобы при вызове экземпляра он представлялся в виде: [)))))] - максимально заряженная батарея.

Подсказка: можете использовать методы очень похожего стандартного типа данных. Догадались какого?



2. **Задание**:

* Реализовать класс Queue
* Определить атрибут inside, который будет хранить в себе имена людей в очереди.
* Переопределить метод \_\_str\_\_, чтобы преобразовать его к виду: Name1=>Name2=>...=>Name3
* Реализовать методы:
  + add - который добавляет имя в очередь
  + take\_out убирает первого человека из списка
* Переопределить методы \_\_add\_\_ , \_\_sub\_\_, \_\_iadd\_\_, \_\_isub\_\_ чтобы они соответствовали методам add и take\_out

class Queue:

def \_\_init\_\_(self):

self.inside = []

def add(self, name: str):

self.inside.append(name)

def take\_out(self):

if self.inside:

self.inside.pop(0)

def \_\_str\_\_(self):

return '=>'.join(self.inside)

def \_\_add\_\_(self, name: str):

self.add(name)

return self

def \_\_sub\_\_(self, count: int):

for i in range(count):

self.take\_out()

return self

def \_\_iadd\_\_(self, name: str):

self.add(name)

return self

def \_\_isub\_\_(self, count: int):

for i in range(count):

self.take\_out()

return self

q = Queue()

q.add('Artem')

q.add('Ilya')

q.add('Ruslan')

q.add('Vladislave')

print(q)

q -= 2

print(q)

q += 'Mark'

print(q)

(код не поместился полностью скриншотом)

3. a)Создайте класс Matrix который должен принимать данные (список списков) для формирования матрицы.

b) Следующий реализовать перегрузку метода \_str\_() для вывода матрицы в привычном виде.

C) реализовать перегрузку метода \_add\_() для сложения двух объектов класса Matrix (двух матриц). Результатом должна быть новая матрица

class Matrix:

from typing import List

def \_\_init\_\_(self, data: List[List[int]]):

self.\_data = data

def \_\_str\_\_(self) -> str:

return '\n'.join([' '.join(map(str, row)) for row in self.\_data])

def \_\_add\_\_(self, other: 'Matrix') -> 'Matrix':

if len(self.\_data) != len(other.\_data) or len(self.\_data[0]) != len(other.\_data[0]):

raise ValueError('Matrices should have the same size')

result = []

for i in range(len(self.\_data)):

row = []

for j in range(len(self.\_data[0])):

row.append(self.\_data[i][j] + other.\_data[i][j])

result.append(row)

return Matrix(result)

data = [[3, 9, 6], [7, 5, 1], [4, 2, 8]]

matrix = Matrix(data)

print(matrix)

matrix\_2 = Matrix([[4, 1, 9], [7, 4, 2], [1, 6, 3]])

matrix\_sum = matrix + matrix\_2

print(matrix\_sum)