К переменным типа список можно применять методы, перечислим некоторые из них:

Методы, не изменяющие список и возвращающие значение:

count(x) - подсчитывает число вхождений значения x в список. Работает за время O(N)

index(x) - находит позицию первого вхождения значения x в список. Работает за время O(N)

index(x, from) - находит позицию первого вхождения значения x в список, начиная с позиции from. Работает за время O(N)

Методы, не возвращающие значение, но изменяющие список:

append(x) - добавляет значение x в конец списка

extend(otherList) - добавляет все содержимое списка otherList в конец списка. В отличие от операции + изменяет объект к которому применен, а не создает новый

remove(x) - удаляет первое вхождение числа x в список. Работает за время O(N)

insert(index, x) - вставляет число x в список так, что оно оказывается на позиции index. Число, стоявшее на позиции index и все числа правее его сдвигаются на один вправо. Работает за время O(N)

reverse() - Разворачивает список (меняет значение по ссылке, а не создает новый список как myList[::-1]). Работает за время O(N)

Методы, возвращающие значение и изменяющие список:

pop() - возвращает последний элемент списка и удаляет его

pop(index) - возвращает элемент списка на позиции index и удаляет его. Работает за время O(N)

Рассмотрим такую задачу: необходимо выбрать все нечетные элементы списка myList и удалить их из него.

Попробуем решить задачу в лоб - просто будем перебирать все позиции в строке и, если на этой позиции стоит нечетное число, будем удалять его.

```
numbers = list(map(int, input().split()))
for i in range(len(numbers)):
    if numbers[i] % 2 != 0:
        numbers.pop(i)
print(' '.join(map(str, numbers)))
```

Такое решение будет работать неправильно в ситуации, когда в списке есть хоть одно нечетное число. Это связано с тем, что объект без названия с типом iterable и значением range(len(numbers)) сгенерируется один раз, когда интерпретатор впервые дойдет до этого места и уже никогда не изменится. Если в процессе мы выкинем из списка numbers хоть одно значение, то в процессе перебора всех индексов выйдем за пределы нашего списка. range, используемый в for, не будет менять свое значение если в процессе работы изменились параметры функции range.

Решение можно переписать с помощью while:

```
numbers = list(map(int, input().split()))
i = 0
while i < len(numbers):
```

```
    if numbers[i] % 2 != 0:
        numbers.pop(i)
    else:
        i += 1
print(' '.join(map(str, numbers)))
```

Такое решение будет работать, но оно не очень эффективно. Каждый раз при удалении элемента нам придется совершать количество операций, пропорциональное длине списка. Итоговое количество операций в худшем случае будет пропорционально квадрату количества элементов в списке.

В случае, если нет очень строгого ограничения в памяти, в задачах, где нужно удалить часть элементов списка гораздо проще создать новый список, в который нужно добавлять только подходящие элементы.

```
numbers = list(map(int, input().split()))
newList = []
for i in range(len(numbers)):
    if numbers[i] % 2 == 0:
        newList.append(numbers[i])
print(' '.join(map(str, newList)))
```

Сложность такого решения пропорциональна длине исходного списка, что намного лучше.