# Абстрактные типы данных

Хайрулин Сергей Сергеевич
s.khayrulin@gmail.com

# Overview

1. Абстрактные типы данных(определение)
2. Массив(основные операции)
3. Динамический массив(основные операции)
4. Список(основные операции)
    - Односвязный список
    - Двусвязный список
    - Циклический список
    - Стек
    - Очередь
    - Дек

# Литература и др. источники


1. Дональд Эрвин Кнут. Искусство программирования (Том 1, 2, 3) // Вильямс 2015.
2. Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман. Структуры данных и алгоритмы // Вильямс 2000.
3. Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И. Лекции по теории графов // М.: Наука, 1990.
4. Харари Ф. Теория графов // М.: Мир, 1973.
5. Косточка А. В. Дискретная математика. Часть 2 //Новосибирск: НГУ, 2001.
6. Котов В. Е., Сабельфельд В. К. Теория схем программ // Наука 1991.
7. http://algolist.manual.ru
8. ….

# Абстрактные типы данных

**Абстракция** – это выделение и придание совокупности объектов общих свойств, которые определяют их концепутальные границы и отличают от всех других видов объектов. Иными словами, абстракция позволяет “пролить свет” на нужные нам данные объектов и, при этом, “затенить” те данные, которые нам не важны.

# Абстрактные типы данных

**АТД** – это такой тип данных, который скрывает свою внутреннюю реализацию от клиентов. Удивительно то, что путем применения абстракции АТД позволяет нам не задумываться над низкоуровневыми деталями реализации, а работать с высокоуровневой сущностью реального мира (*Стив Макконнелл*)

# Абстрактные типы данных

- Инкапсуляция деталей реализации.
- Снижение сложности.
- Ограничение области использования данных.
- Высокая информативность интерфейса.

# Массив

**Массив** (в некоторых языках программирования также таблица, ряд, матрица) — тип или структура данных в виде набора компонентов (элементов массива), расположенных в памяти непосредственно друг за другом. При этом доступ к отдельным элементам массива осуществляется с помощью индексации, то есть через ссылку на массив с указанием номера (индекса) нужного элемента. За счёт этого, в отличие от списка, массив является структурой данных, пригодной для осуществления произвольного доступа к её ячейкам.

WIKI

# Операции над массивами

- Индексация
- Перебор элементов
- Поиск

# Динамические массивы

**Динамический массив** – массив, который может менять свой размер в течении времени

# Операции над динамическими массивами

- Вставка элемента
- Удаление элемента
- Поиск
- Индексация

# Список

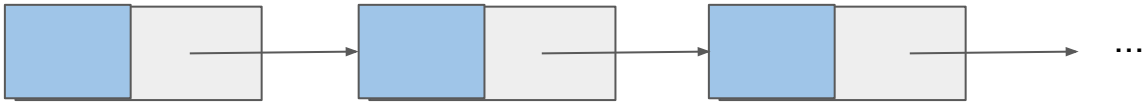
**Списки** являются чрезвычайно гибкой структурой, так как их легко сделать большими или меньшими, и их элементы доступны для вставки или удаления в любой позиции списка. Списки также можно объединять или разбивать на меньшие списки.

# Список

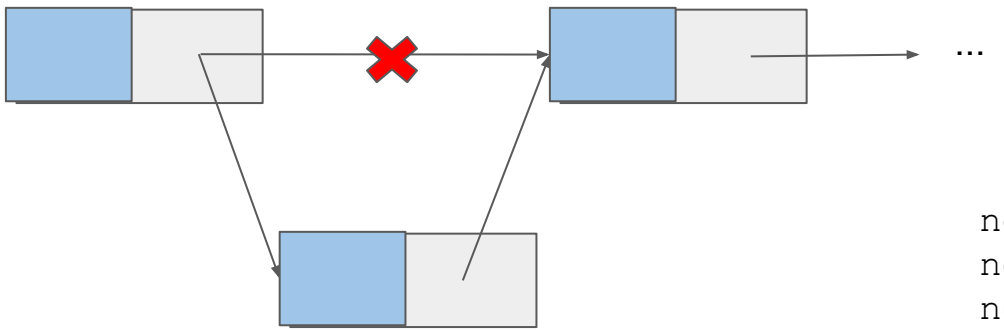
```
class List<T>:
    T data
    List * next
head = List<T>
```

```
class Node<T>:
    T data
    Node * next
class List<T>:
    Node<T> * head
```

# Список (однонаправленный)

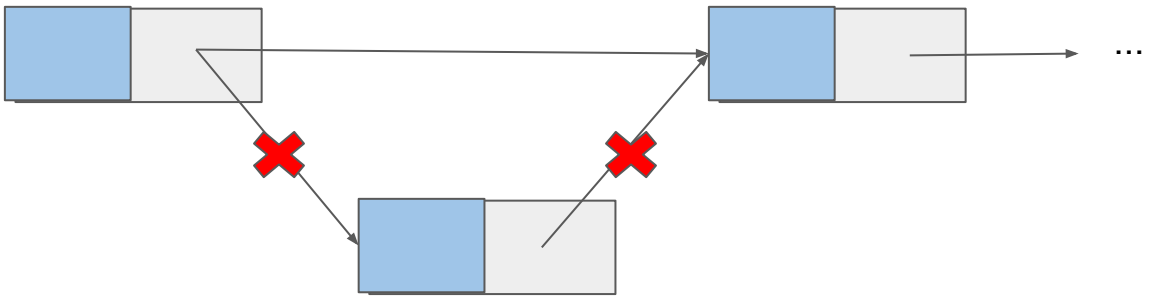


# Список (вставка)

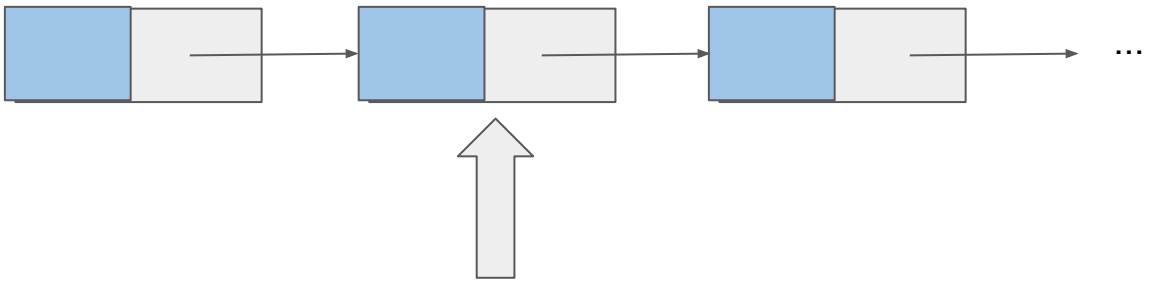



```
new_node = Node(value)
new_node.next = node_place.next
node_place.next = new_node
```

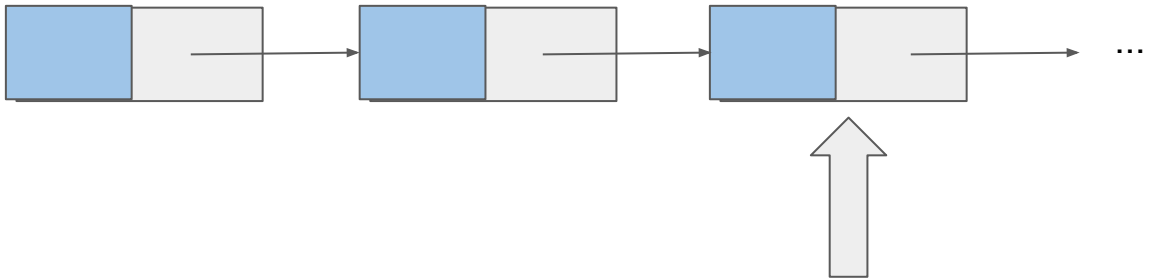
# Список(удаление)



# Список(поиск)

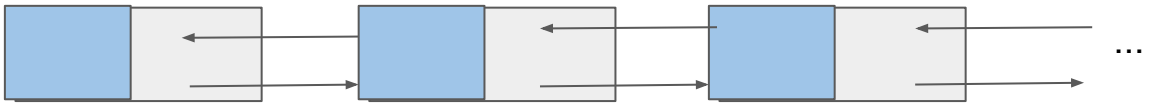


# Список(поиск)

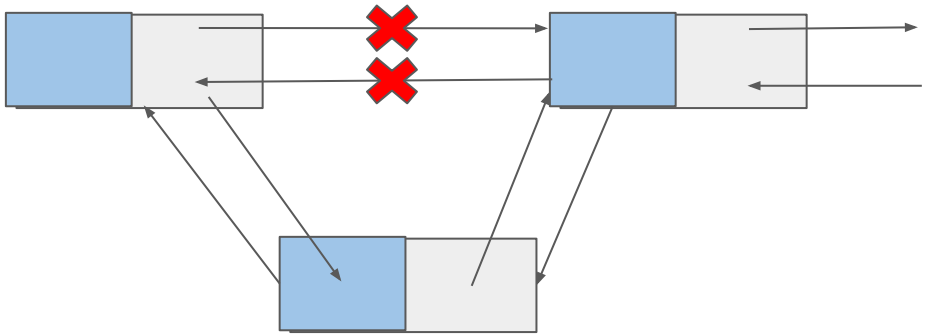



```
def search(Node * head):
    Node *node = head
    while node != None:
        if node.data == target:
            return node
        node = node.next
    return None
```

# Список (двунаправленный)

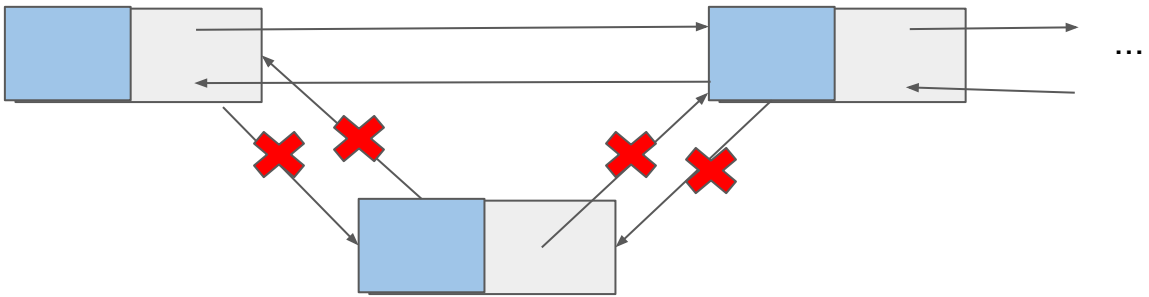


# Список (вставка)

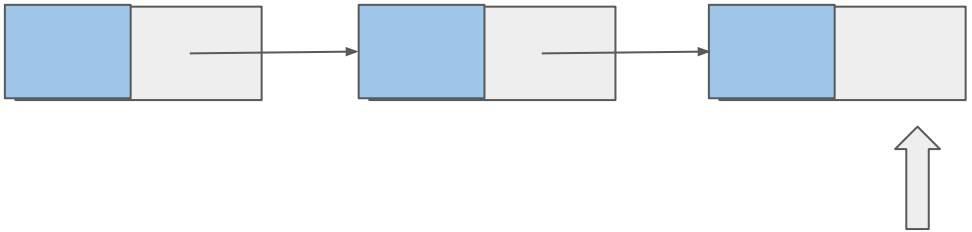



```
new_node = Node(value)
new_node.next = node_place.next
new_node.prev = node_place
node_place.next = new_node
new_node.next.prev = new_node
```

# Список(удаление)



# Стек (LIFO)

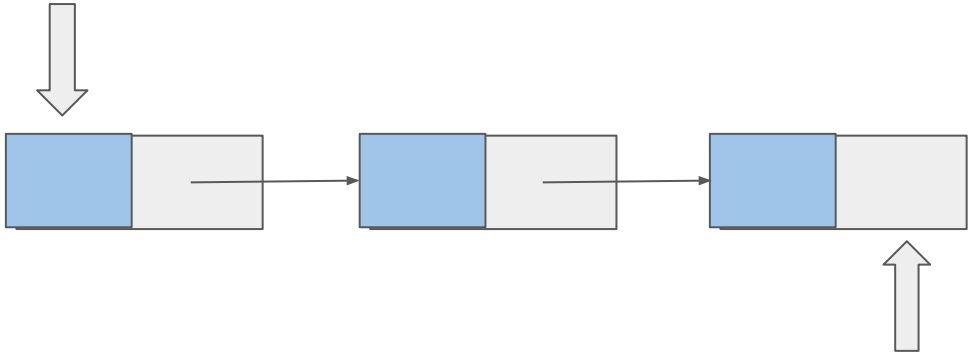


# Стек

## Операции

- push(data) → на доске
- pop() → на доске
- pop(index) → на доске

# Очередь (FIFO)

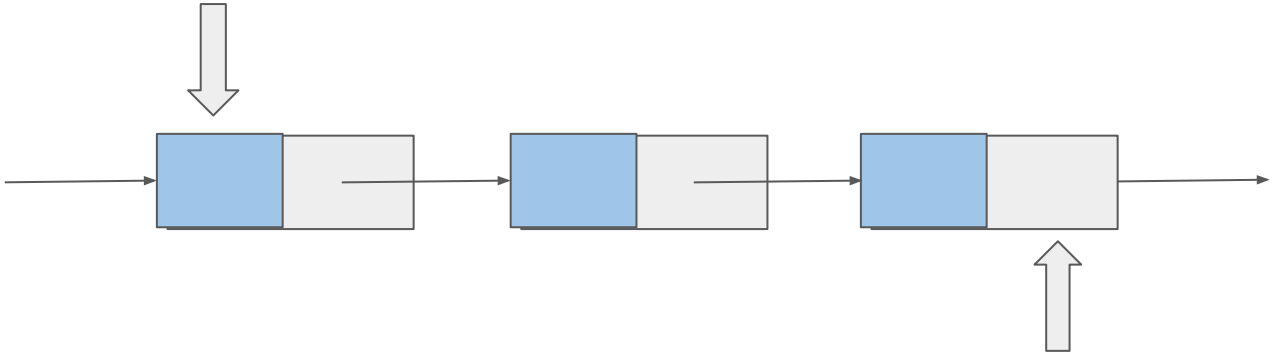


# Очередь

## Операции

- push(data) → на доске
- pop() → на доске
- top() → на доске

# Дек

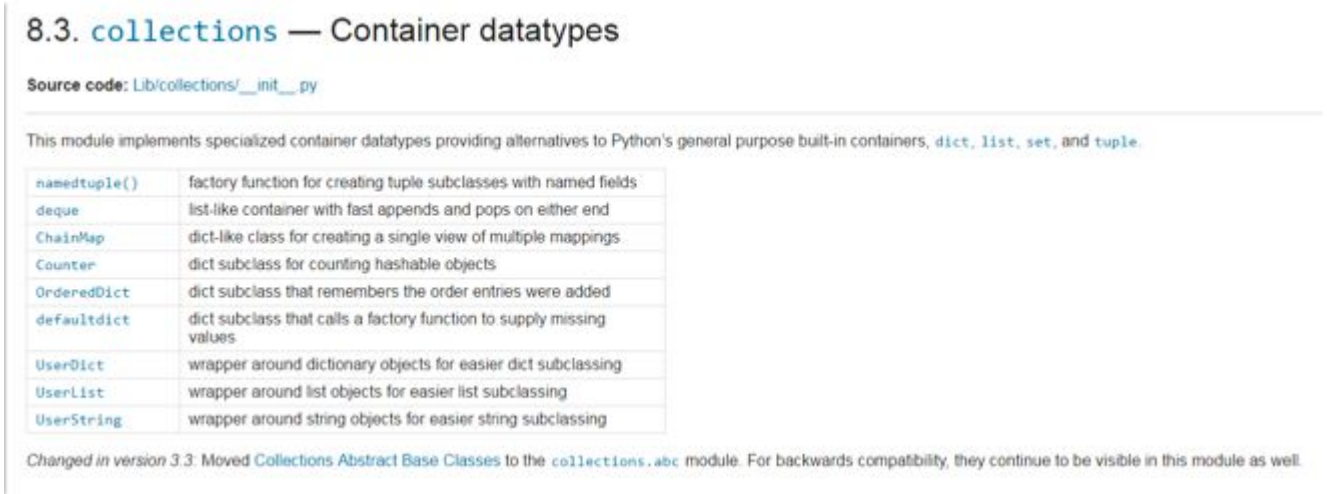


# Очередь

## Операции

- push_back(data) → на доске
- pop_back() → на доске
- top() → на доске
- push_head() → на доске
- pop_head() → на доске

# Python & ADT



## 8.3. `collections` — Container datatypes

**Source code:** Lib/collections/__init__.py

This module implements specialized container datatypes providing alternatives to Python's general purpose built-in containers, `dict`, `list`, `set`, and `tuple`.

| | |
|---|---|
| `namedtuple()` | factory function for creating tuple subclasses with named fields |
| `deque` | list-like container with fast appends and pops on either end |
| `ChainMap` | dict-like class for creating a single view of multiple mappings |
| `Counter` | dict subclass for counting hashable objects |
| `OrderedDict` | dict subclass that remembers the order entries were added |
| `defaultdict` | dict subclass that calls a factory function to supply missing values |
| `UserDict` | wrapper around dictionary objects for easier dict subclassing |
| `UserList` | wrapper around list objects for easier list subclassing |
| `UserString` | wrapper around string objects for easier string subclassing |

*Changed in version 3.3:* Moved Collections Abstract Base Classes to the `collections.abc` module. For backwards compatibility, they continue to be visible in this module as well.

https://docs.python.org/3.6/library/collections.html

# Указания для задач

Для замера работы функции нужно использовать метод now() класса datetime модуля datetime

# Указания для задач

```
array = [0] * N
array.insert(N,0)
```

```
import datetime


def main():
    t1 = datetime.datetime.now()
    #You'r code here
    ...
    print(datetime.datetime.now() - t1)

if __name__ == '__main__':
    main()
```

# Указания для задач

```
import numpy as np


...
# Generate numpy Array with N random numbers
array = np.random.rand(N)
#Sort Array by quick sort
sorted(array)
...
```

# Указания для задач

jupyter MyTests Last Checkpoint: a minute ago (unsaved changes)

File Edit View Insert Cell Kernel Help Python 2

Code CellToolbar

In [6]:
```
import datetime
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
```

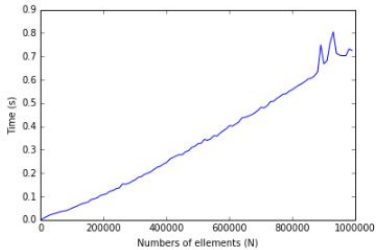
In [7]:
```
times = []
for N in xrange(1000, 1000000, 10000):
    array = np.random.rand(N)
    t1 = datetime.datetime.now()
    sorted(array)
    t_res = datetime.datetime.now() - t1
    times.append([N,t_res])
y = [t[1].total_seconds() for t in times]
x = [t[0] for t in times]
```

In [8]:
```
%matplotlib inline
```

In [9]:
```
plt.plot(x, y)

plt.xlabel('Numbers of ellements (N)')
plt.ylabel('Time (s)')

plt.savefig("test.png")
plt.show()
```



# Задачи

- Реализовать алгоритм перемножения квадратных матриц. Матрицы могут задаваться как список списков. Считывать можно из файла потока ввода, или задавать случайным образом (используя функцию np.random.rand(N)). Оценить временную и ассимптотическую сложность алгоритма, построить график.
- Найти все пифагоровы тройки ($c^2 = a^2 + b^2$) для заданного интервала. Интервал задается парой чисел через пробел считанных из входного потока (например: 10 100) помните, что верхняя грань отрезка должна быть больше нижней. Если задано одно число, то считаем, что ограничение снизу равно по умолчанию 1. Оценить временную и ассимптотическую сложность алгоритма, построить график.
- Реализовать алгоритм факторизации числа (разложение числа как произведение двух других чисел). Оценить временную и ассимптотическую сложность алгоритма, построить график.
- Реализовать алгоритм рассчитывающий сочетания и размещения.
- Факториал довольно емкостная функция, при расчете которого для больших значений может случится переполнение (т.е. полученное число будет больше чем максимально возможное число в вашей системе). Подумайте как преодолеть эту проблему.

# Задачи

Написать оболочку для работы с графами:

- создавать графы
- Выводить граф (в виде таблицы смежности)
- Удалять ребра
- Ищет путь в графе для заданных вершин
  - Флойда-Уоршела
  - Форда-Беллмана
  - Дейкстра

# Задачи

1. Скачать файл https://goo.gl/z7H7DU
2. Файл содержит карту препятствия обозначены символом ‘%’ клетки, по которым можно передвигаться обозначены ‘-’, при этом каждая клетка по которой можно двигаться имеет вес 1.
3. Робот начинает движение в клетке обозначенной буквой ‘P’ и движется в клетку обозначенной буквой ‘T’.
4. Нужно рассчитать оптимальную траекторию пути робота с помощью  алгоритма A*.
5. Выведите траекторию в отдельный файл.

# Задачи

1. Реализуйте функцию DFS
2. С помощью вашей функции реализуйте алгоритм разбиение графа на компоненты связности.
3. Реализуйте алгоритм проверки орграфа на цикличность
4. Реализуйте алгоритм Крускала/Прима для поиска минимального остовного дерева взвешенного графа.

# Задачи

1. Как можно объединить два списка. Напишите программу делающую это
2. Реализуйте очередь через два стека.

Спасибо за внимание!