



Современные технологии ООП численных методов

Карпаев Алексей,
аспирант ФАКИ

Обзор курса

Лекция 1

Вступительные вопросы

- Кто имел дело с языком Python?
- Какие online-курсы проходили?

Мотивация создания курса

Ситуация: недостаток времени на рассмотрение технологий программирования в институтском курсе «Вычислительная математика» \Rightarrow проблемы:

- неоптимальный выбор языка программирования
- написание программ в неоптимальной парадигме/подходе
- нежелательное оформление кода программ:

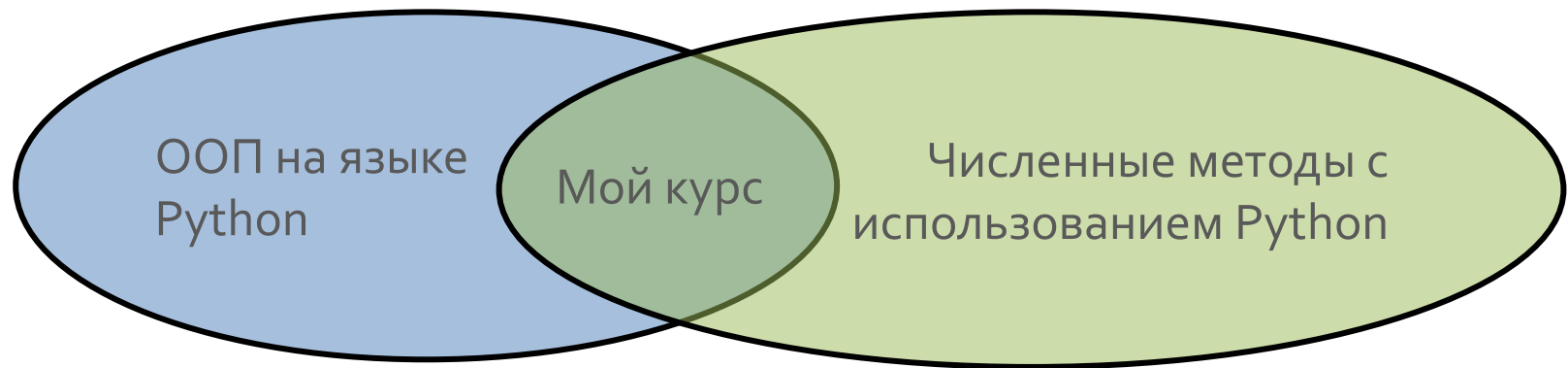
```
Bibop alfa:
xy: = dx/dy; yx: = dy/dx;
begin almax: 0;
for k: = 0 step 1 until K do
for j: = 0 step 1 until J do
begin kx: = KX (jxdx, kxdy);
ky: = KY(jxdx, kxdy);
almax: = almax + min end j, k;
almax: = almax /((J + 1) x (K + 1)) end NE 5;
for j: = 0, 3, 6 do for i: = 0, 1, 2 do al [ j + i ] := - almax ↑((8
for k: = 0 step 1 until K do
for j: = 0 step 1 until J do begin if k = 0 then
B [k, j] := - KY (ixdx, (k - 1/2)xdy)xy;
if j = 0 then D [k, j] := - KX((j - 1/2)xdx, kxdy);
xyx; if k = K then H [k, j] := - KY (jxdx, (k + 1/2)xdy)xyx;
if j = J then F [k, j] := - KX((j + 1/2)xdx, kxdy)xyx;
E [k, j] := - ((if k = K then H [k, j] else B [k, j] ) + (if k = 0
then B [k, j] else H [k, j] ) + (if j = 0 then D [k, j] else
F [k, j] ) + (if j = J then F [k, j] else D [k, j] ))
end kj;
for j: = 0 step 1 until J do
begin B [K, j] = 2xB [K, j] ;
```

Цели курса

- Обучение основам языка Python
- Обучение использованию принципов парадигмы ООП на **простых примерах вычислительных задач**
- Демонстрация приближенного стандарта оформления кода вычислительных программ.

Мотивация посещений курса

- Ситуация: множество курсов по Python в сети Интернет



- В чем отличие моего курса?
 - Совмещение этих подходов.

Введение в ООП

Некоторые парадигмы программирования

Парадигма программирования - это совокупность принципов, методов и понятий, определяющих способ конструирования программ.

Процедурное программирование: разбиение программы на процедуры (функции), использование ранее написанных процедур

Структурное программирование - более развитая версия процедурного

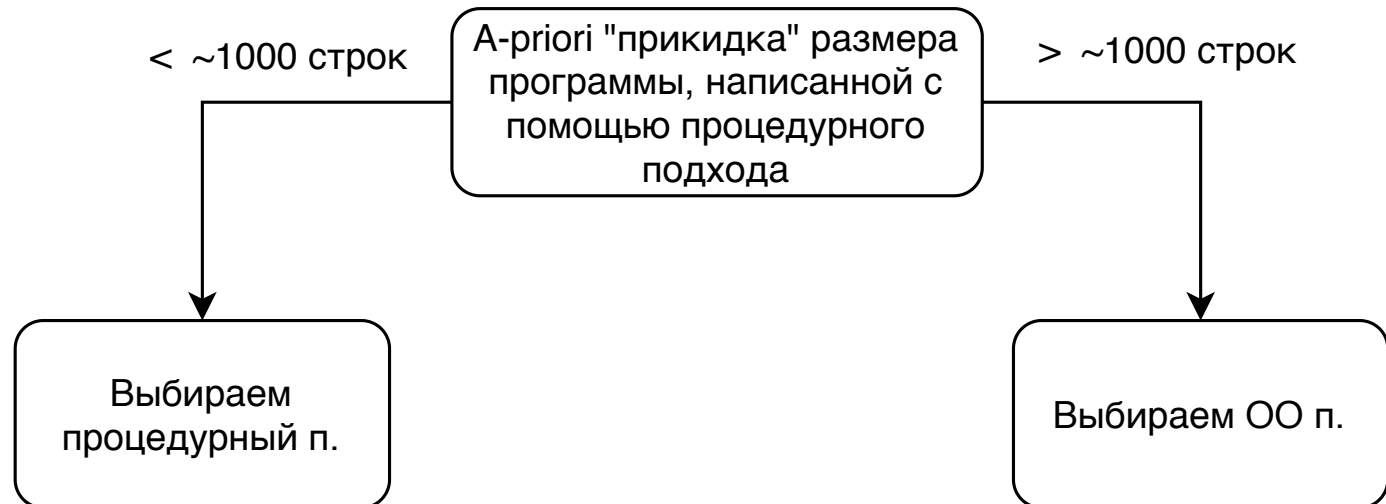
ОО программирование: объединение переменных и процедур, соответствующих определенным сущностям, в единое целое

Когда выгодно использовать ООП?

ООП используют **при создании крупных программных комплексов**. По сравнению с процедурным программированием:

- Лучшая организация кода
 - Более легкое расширение функциональности
- ООП основано на понятиях **класса** и **объекта**.

Мой алгоритм выбора подхода:



Цепочка обобщений

Переменная – для хранения 1 величины



Структура – для хранения нескольких величин



Класс – ?

Структура: простейший пример

```
struct Vector {  
    double x, y;  
};  
  
// функция для работы со структурой  
double calculatewidth(struct Vector* v) {  
    return sqrt((v->x)*(v->x) + (v->y)*(v->y));  
}
```

Класс: простейший пример

Если поместим функцию внутрь структуры – «получим» класс

```
class Vector {  
    public:  
        double x, y;  
        double calculatewidth() {  
            return sqrt(x*x + y*y);  
        }  
};
```

Функция внутри класса «знает» переменные этого класса – передавать аргументы не требуется.

Классы – основа ООП

- **Класс** – обобщение типа данных **структура**
- **Класс** = **переменные** + **функции** для работы с этими переменными
- **Классы** создаются для представления сущностей: вектор, человек, транспортное средство, записная книжка, ... ; интегратор, солвер ОДУ, ...

Классы — основа ООП

- Переменные класса — **поля**
- Функции класса для работы с полями — **методы**
- Экземпляр класса — **объект.**

Основные принципы ООП

Абстракция

Инкапсуляция

Наследование

Полиморфизм

Абстракция

- Принцип позволяет определить, какие параметры необходимо поместить в класс в качестве полей, а какими параметрами можно пренебречь в контексте конкретной программы.

Абстракция: пример

Человек, параметры:

- ФИО
- пол
- возраст
- вес
- цвет волос
- национальность
- прописка
- результаты ЕГЭ
- факультет в ВУЗе
- оценки в ВУЗе
- зарплата
- должность
- ...

Человек

Абстракция: пример

КОНТЕКСТ	НЕОБХОДИМЫЕ ПАРАМЕТРЫ
Работник в компании	ФИО, должность, зарплата
Студент в университете	ФИО, результаты ЕГЭ, факультет, оценки
Пациент в поликлинике	ФИО, пол, возраст, вес
Гражданин в государстве	ФИО, пол, возраст, национальность, прописка

Инкапсуляция

- Принцип сокрытия данных от внешнего воздействия.
- Обеспечивает безопасность пользования программой сторонними пользователями
- Осуществляется разделением прав доступа к полям класса на **private** и **public** (или **protected**).

Инкапсуляция: пример

- Пользователю доступны только кнопки переключения каналов (аналоги методов)
- Устройство «шестеренок» внутри (аналоги полей) сокрыто.

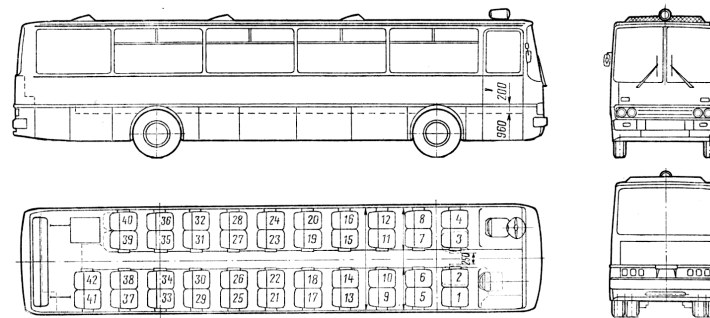


Наследование

- Принцип позволяет избежать повторения кода при расширении функциональности программ, используя ранее написанный код.

Наследование: пример

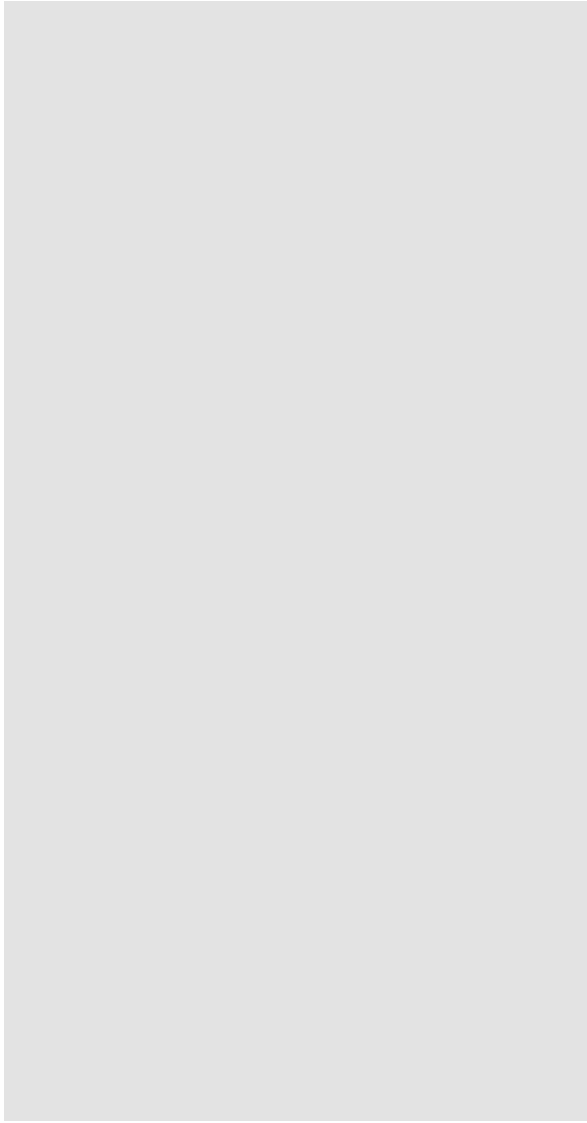

- У трамваев, автобусов и троллейбусов имеются общие параметры: число мест, дверей, колес, ...
- Чтобы избежать повторения кода эти параметры 1 раз помещаются в отдельный класс, от которого наследуются классы трамвай, автобус и троллейбус.



Полиморфизм

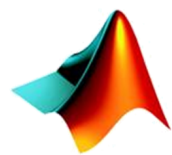
- Это способность объектов с одинаковым названием иметь различные реализации
- Примеры полиморфизма:
 - полиморфизм функций: работа с различными типами входных параметров
 - одинаковый набор методов, но различны их реализации (в классах, входящих в иерархию наследования)
 - одинаковая реализация, но различны типы аргументов (шаблоны из C++).

Закончили с ООП.



Выбор языка программирования для курса

Языки
программиро
вания:
широкий
выбор



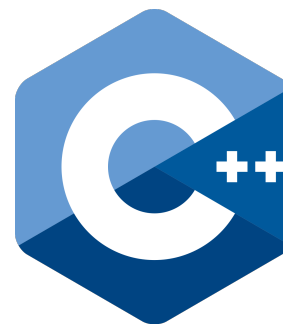
MATLAB®



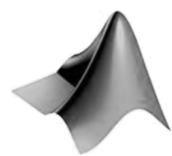
GNU Octave



python



Языки
программиро
вания:
широкий
выбор



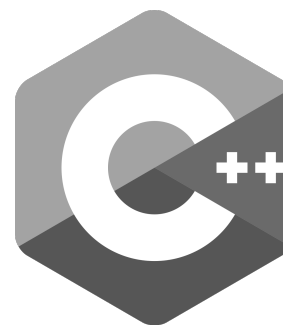
MATLAB®



GNU Octave



python



Почему Python, а не Matlab?



- Более широкий спектр применения:
 - вычислительная математика (как и Matlab)
 - web-разработка
 - GUI
 - Unix-shell скрипты
- Open Source
- Высокая популярность.

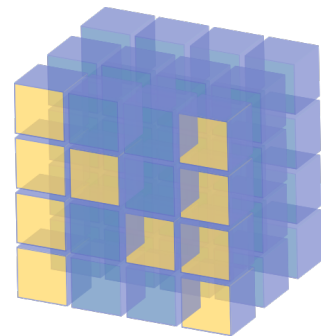
Почему Python, а не C++?



- Более лаконичный синтаксис позволяет сконцентрироваться только на изучении аспектов ООП

Программы на Python исполняются «медленней» программ на C++, но эта проблема отчасти решается с помощью библиотек.

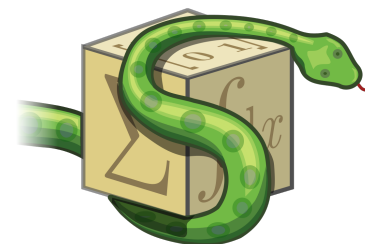
Язык Python: библиотеки из курса



NumPy



matplotlib

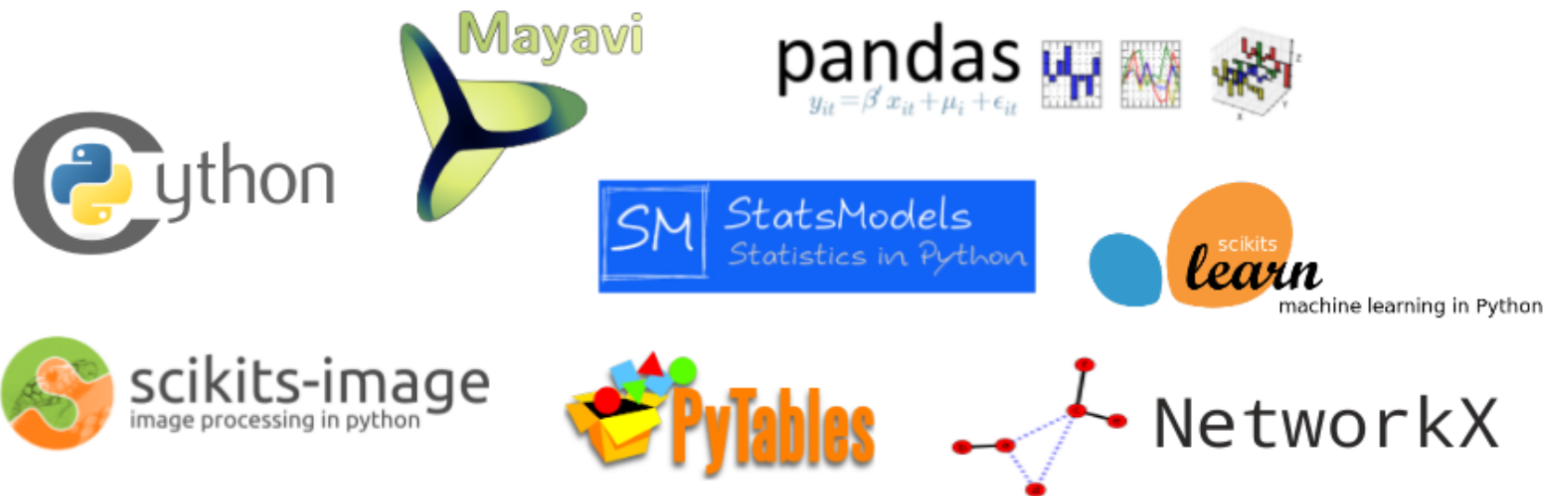


SymPy



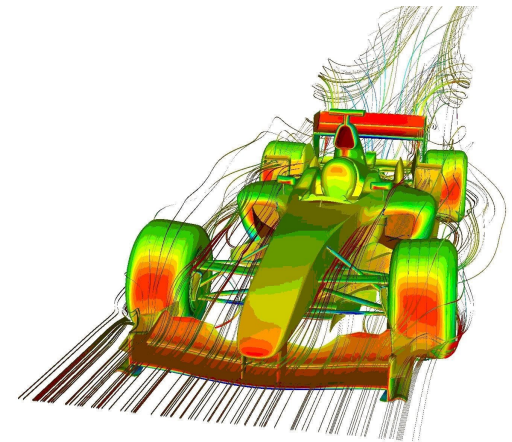
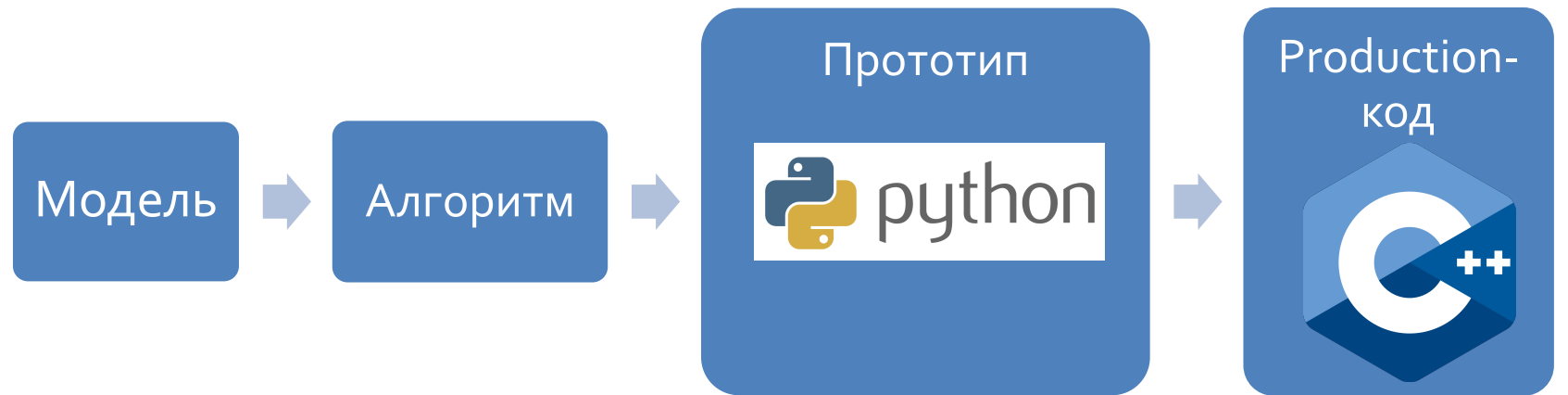
Numba

Язык Python: другие библиотеки



На заметку

Последовательность этапов разработки вычислительных программ:



Темы из курса

Введение в синтаксис
Python

Введение в библиотеки и
синтаксис классов

Решение стандартных
задач вычислительной
математики с помощью
ОО-программ

Обзор симулятора **Chaste** для
численного моделирования в
электрофизиологии

Сравнение средств
ускорения Python

Численное
дифференцирование

Численное
интегрирование

Моделирование
функциональных
пространств

■ ■ МКР для ОДУ

МКР для УрЧП

МКЭ для аппроксимации
функций (раздел дисциплины «Анализ
данных»)

МКЭ для ОДУ

Итоги

1. ООП используется при написании крупных программных комплексов

2. Основные принципы ООП:

- абстракция
- инкапсуляция
- наследование
- полиморфизм

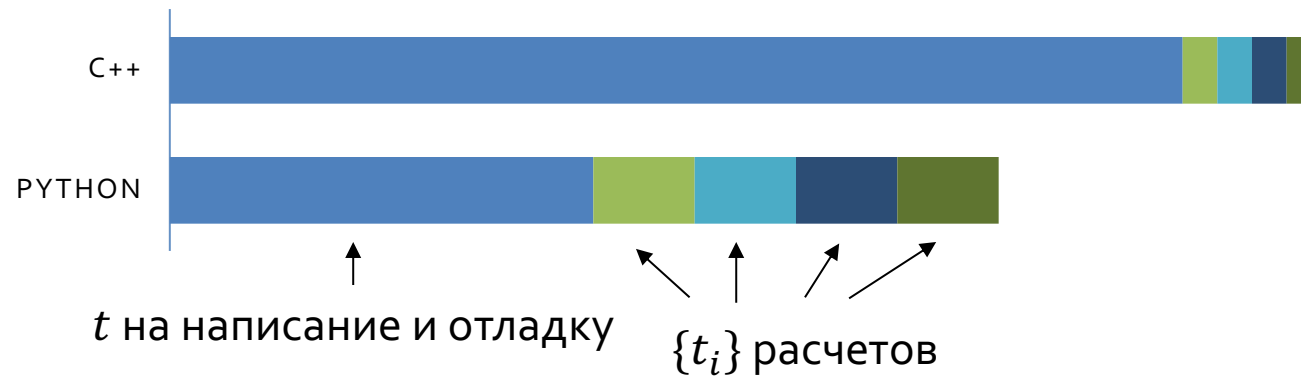
3. Язык  python :

- с лаконичным синтаксисом
- медленный
- большой набор библиотек
- подходит для прототипирования

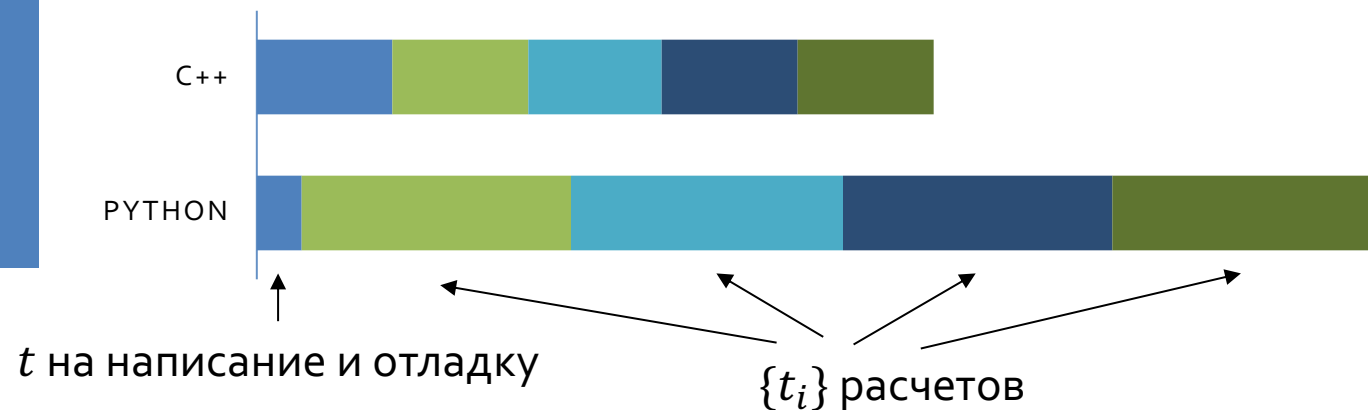
P.S. «Ускоренный» прототип – может подойти на роль программы для защиты диплома (Вы должны уточнить это у своего научного руководителя; возможно, Вам скажут заменить Python → Matlab)

Касательно дипломной программы

Случай численного решения системы уравнений нефтяной гидродинамики (Є классу УрЧП) на мелкой пространственной сетке (соотношения примерные):



Случай численного решения системы уравнений молекулярной динамики с «огромным» (требуемым на практике) числом частиц (соотношения примерные):





Заключение

По окончании курса

- Знание языка Python и его «основных» библиотек на 70%
- Знание основных принципов ООП
- Умение использовать последние для написания крупных вычислительных программ

Полученные в курсе знания можно применить для написания программы, выносящейся на защиту диплома.

Спасибо за
внимание.
Готов ответить на
ваши вопросы.

karpaev@phystech.edu

Спасибо за
внимание.
Готов ответить на
ваши вопросы.

karpaev@phystech.edu