



# Современные технологии ООП численных методов

Карпаев Алексей,  
аспирант ФАКИ

# Обзор курса

Лекция 1

# Проблема №1

Недостаток времени на рассмотрение технологий программирования в курсе вычислительной математики

# 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

**ООП:** 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**

# Цепочка обобщений

Переменная – для хранения 1 величины



Структура – для хранения нескольких величин



Класс – ?

## Структура: 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: 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 реального мира: вектор, человек, транспортное средство, записная книжка, 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**



# Инкапсуляция: пример

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

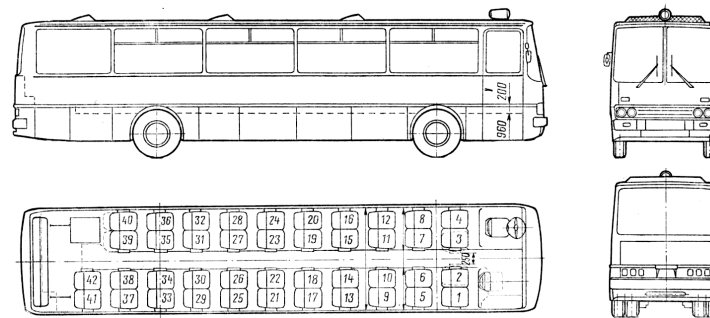


# Наследование

- 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

# Полиморфизм: пример

(Из языка C++) функции с одним именем:

- `int add(int a, int b)`
- `double add(double a, double b)`

Закончили с ООП.  
Осталось перечислить оставшиеся проблемы...

## Проблема №2: нежелательное оформление кода

```
procedure ALFA (KX, KY, Q, eps, T, P, dx, dy, J, K, M, nmax);
procedure KX, KY, Q, P; label M;
real eps, dx, dy;
integer J, K, nmax; array T;
Blok1:
begin
real alfa, kx, ky, almax, xy, yx, BO, DO, qO, C, G, R, max;
integer i, j, k, n, Kn, Kk;
array al [0:8], b, B [1:K, 0:J], f, H [0:K-1, 0:J],
c, D [0:K, 1:J], e, F [0:K, 0:J-1], E, q, d, V, dT [0:K, 0:J];
procedure min;
begin real a, a1;
a := 2/((1 + kyxxyxxy/kx)xJxK);
al := 2/((1 + kxxyxxyx/ky)xJxK);
min := if a < al then a else al end max;
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 - j ÷ 3 - 3 x i) / 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) x 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) x xy;
if j = J then F [k, j] := - KX ((j + 1/2)xdx, kxdy) x yx;
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] = 2 x B [K, j];
```

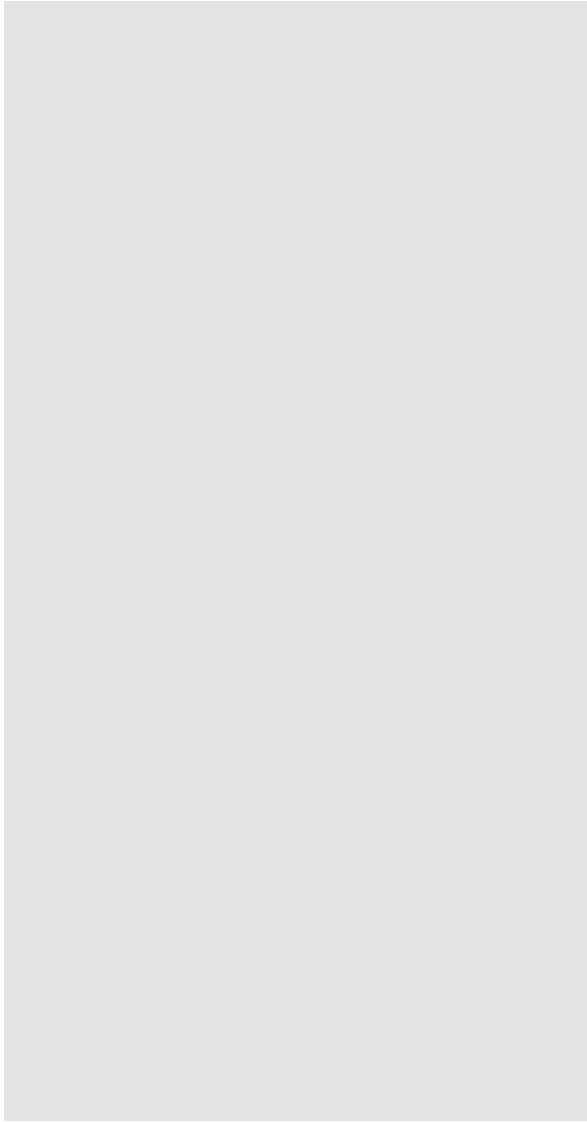

## Проблема №3

Отсутствие примеров реализации ОО-подхода для решения вычислительных задач в большинстве объемных книг по программированию



Проблема №1  
Проблема №2  
Проблема №3

- Данные проблемы попробуем решить в этом курсе...



# Выбор языка программирования для курса

# Почему Python, а не C++?



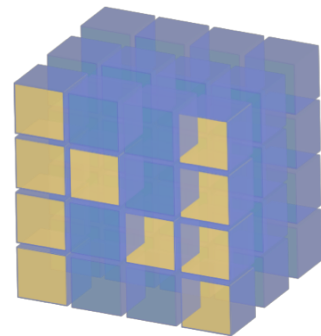
- Более лаконичный синтаксис позволяет сконцентрироваться только на изучении аспектов ООП
- Программы на Python исполняются «медленней» программ на C++, но эта проблема решается (в некоторых случаях) с помощью использования библиотек

# Почему Python, а не Matlab?



- Более широкий спектр применения: научные вычисления, web-разработка, GUI, Unix-shell скрипты, ...
- Open Source

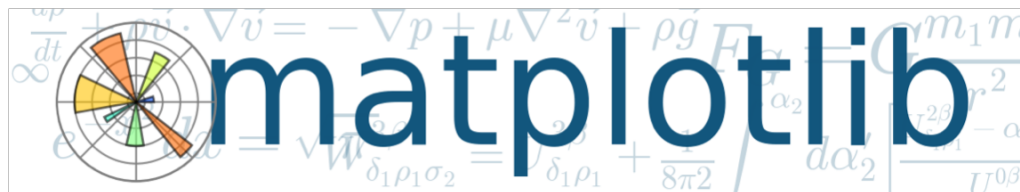
# Язык Python: библиотеки из курса



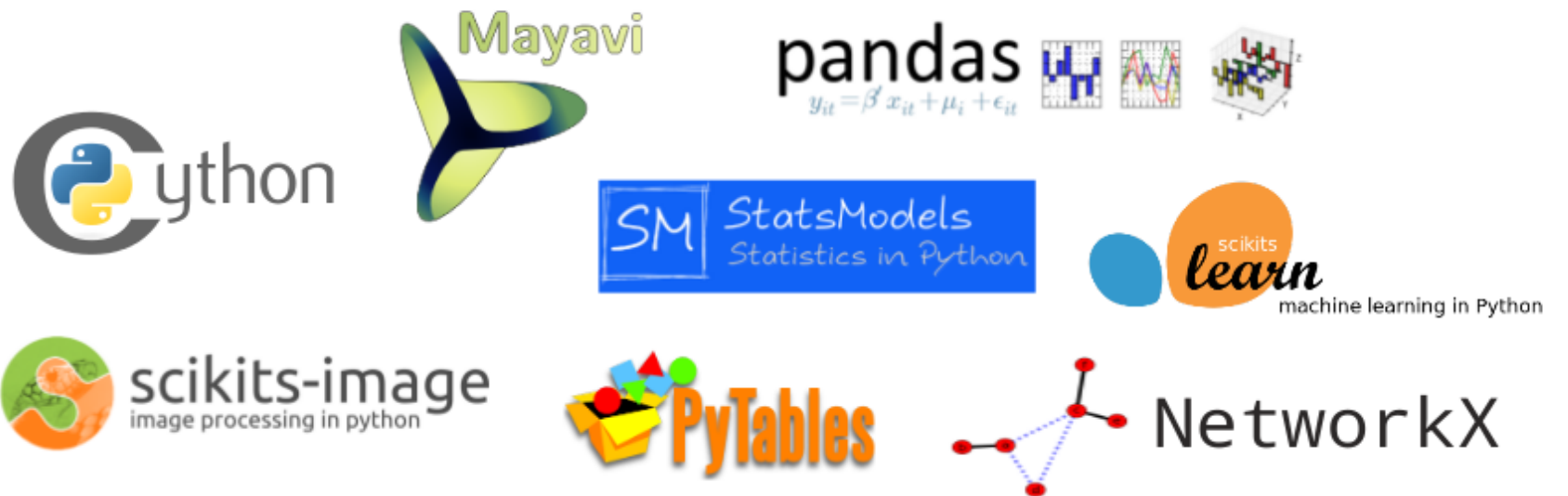
NumPy



SymPy



# Язык Python: еще библиотеки...



# Темы из курса

Введение в синтаксис  
Python

Введение в библиотеки и  
синтаксис классов

Решение стандартных  
задач вычислительной  
математики с помощью  
ООП

Обзор симулятора Chaste  
для моделирования в  
электрофизиологии

Сравнение средств  
ускорения Python

Численное  
дифференцирование

Численное  
интегрирование

Реализация  
функциональных  
пространств


■ ■  
МКР для ОДУ

МКР для УрЧП

Аппроксимация  
функций

МКЭ для ОДУ

# Итоги

- ООП используется при написании крупных программных комплексов
- Основные принципы ООП: **абстракция, инкапсуляция, наследование и полиморфизм**
- Язык  python : с лаконичным синтаксисом, «медленный», но с большим набором библиотек



# Цели курса

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

Спасибо за  
внимание.  
Готов ответить на  
ваши вопросы.

[karpaev@phystech.edu](mailto:karpaev@phystech.edu)

Спасибо за  
внимание.  
Готов ответить на  
ваши вопросы.

[karpaev@phystech.edu](mailto:karpaev@phystech.edu)