ООП

- Повторное использование кода
- Инкапсуляция
- Наследование
- Полиморфизм

Подсчет среднеарифметического

```
1       def mean_int(numbers):
2           res = 0
3           for cnum in numbers:
4               res += cnum
5           return res / len(numbers)
```

Нужно добавить поддержку рациональных чисел

Рациональное число - пара (числитель, знаменатель). Будем передавать рациональные числа в виде кортежа.

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd} \; ; \quad \frac{a}{b}/c = \frac{a}{cd}$$

Среднее рациональные

```
# rational = (num, denom)
def mean_rational(numbers):
    num, denom = (0, 1)
    for cnum, cdenom in numbers:
        num = num * cdenom + denom * cnum
        denom *= cdenom
    return (num, denom * len(numbers))
```

Среднее

```
1       def mean(numbers):
2           assert len(numbers) != 0
3           if isinstance(numbers[0], (int, long, float)):
4               return mean_int(numbers)
5           else:
6               return mean_rational(numbers)
```

Добавляем матрицы

```
1       def mean(numbers):
2           assert len(numbers) != 0
3           if isinstance(numbers[0], (int, long, float)):
4               return mean_int(numbers)
5           elif ismatrix(numbers[0]):
6               return mean_matrix(numbers)
7           else:
8               return mean_rational(numbers)
```

"Идея" СА

```
1       def mean_int(numbers):
2           res = 0
3           for cnum in numbers:
4               res += cnum
5           return res / len(numbers)
```

CA

```
add_num = lambda x, y: x + y
add_rational = lambda (x1, y1), (x2, y2): \
                    (x1 * y2 + y1 * x2, y1 * y2)
div_num = lambda x, y: x / y
div_rational = lambda (x, y), z: (x, y * z)

def mean(numbers, add_func, div_func):
    res = numbers[0]
    for cnum in numbers[1:]:
        res = add_func(res, cnum)
    return div_func(res, len(numbers))

mean([1, 2, 3] , add_num, div_num)
```

```
1       def mean(numbers, add_func, div_func):
2           it = iter(numbers)
3           res = next(it)
4           for cnum in it:
5               #...
```

```
1       val = {'num':1,
2              'denom':2,
3              '__add__':add_rational,
4              '__div__':div_rational}
5
6       def add_rational(x, y):
7           return  (x['num'] * y['denom'] + \
8                    x['denom'] * y['num'],
9                    x['denom'] * y['denom'])
10
11      def div_rational(x, y):
12          return (x['num'], x['denom'] * y)
```

```
1       def mk_rational(num, denom):
2           return {'num':num,
3                   'denom':denom,
4                   'add':add_rational,
5                   'div':div_rational}
```

Нужно автоматически упрощать после операции

```
1       def add_rational_auto_simpl(x, y):
2           res = add_rational(x, y)
3           nod = get_NOD(res['num'], res['denom'])
4           res['num'] /= nod
5           res['denom'] /= nod
6           return res
7
8       def mk_rational_auto_simpl(num, denom):
9           return dict(num=num,
10                      denom=denom,
11                      __add__=add_rational_auto_simpl,
12                      __div__=div_rational_auto_simpl)
```

На предыдущем слайде ошибка

```
    def mk_rational(num, denom):
        return {'num':num,
                'denom':denom,
                '__class__':Rational}

    Rational = dict(__div__=div_rational,
                    __add__=add_rational,
                    __init__=mk_rational)
```

Когда ООП

- Если есть участок кода, требующий определенного ограниченного набора операций над входными данными
- Одновременно в программе могут быть несколько видов подходящих данных, с различной функциональностью для реализации этого интерфейса
- Или группировки функций с общим глобальным состоянием

Классы не предназначен для

- Группировки функций
- Группировки одной функции
- Если вы, ессно, используете нормальный ЯП

Проблема

Проблема

a + b для разных типов

Среднее

```
add = {(int, int):add_int_int,
       (int, tuple):add_int_tuple,
       (tuple, int):add_tuple_int}

def register_add(tp1, tp2, func):
    add[(tp1, tp2)] = func

def mean(numbers):
    res = numbers[0]
    for cnum in numbers[1:]:
        res = add[(type(res), type(cnum))](res, cnum)
    return div[(type(res), int)](res, len(numbers)
```