при поддержке
{а}вижн

Лекция 09

Перевод и адаптация материалов Джона ДеНиро (John DeNero). Используется с разрешения автора.

Данные

Типы данных

У каждого значения есть тип.

(Пример)

Свойства встроенных типов данных:

1. Значения этих типов получаются при выполнении простых выражений.
2. Для работы с этими значениями существуют встроенные функции, операторы и методы.

Числовые типы в Python’e:

```
>>> type(2)
<class 'int'>
```

Представляют целые числа точно

```
>>> type(1.5)
<class 'float'>
```

Представляют действительные числа приблизительно

```
>>> type(1+1j)
<class 'complex'>
```

Объекты

(Пример)

- Объект — это представление информации.
- Он состоит из данных и алгоритмов, объединенных для создания абстракций.
- Объекты могут представлять вещи, свойства, взаимодействия и процессы.
- Тип объекта называют классом; в Python’е они рассматриваются как значения.
- Объектно-ориентированное программирование:
  - Метафора для организации больших программ
  - Особый синтаксис может улучшить структуру программы
- В Python’е каждое значение — это объект.
  - У всех объектов есть атрибуты.
  - Большая часть обработки данных происходит в методах объекта.
  - Функции делают одну вещь; объекты делают множество связанных вещей.

Абстракция данных

Абстракция данных

- Составные объекты состоят из объединения других объектов
  - Дата – год, месяц и день
  - Географическая координата – широта и долгота
- Абстракция данных позволяет работать с составным объектом как с единой сущностью.
- Любую использующую данные программу можно разделить на две части:
  - как представлены данные (объединение частей)
  - как обрабатываются данные (единой сущностью)

Абстракция данных – это методология, в соответствии с которой функции создают границы абстракции (барьеры) между ***представлением*** и ***использованием***.

Все программисты

Лучшие программисты

Рациональные числа

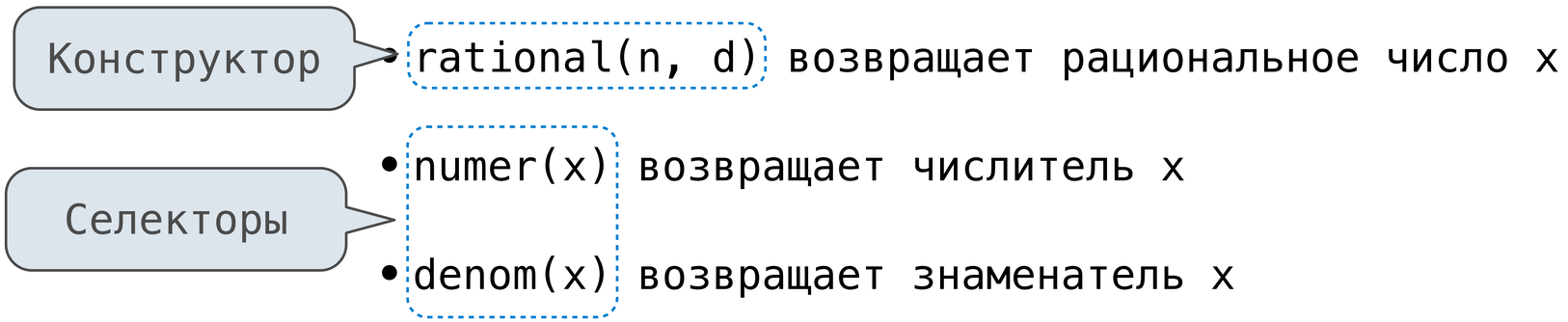
$$\frac{\text{числитель}}{\text{знаменатель}}$$

Точное представление дробей

Пара целых чисел

Как только возникает деление, точное представление может быть утрачено!

Представь, что можно составлять и разбивать рациональные числа:



Арифметика рациональных чисел

$$\frac{3}{2} * \frac{3}{5} = \frac{9}{10}$$

$$\frac{3}{2} + \frac{3}{5} = \frac{21}{10}$$

Пример

$$\frac{nx}{dx} * \frac{ny}{dy} = \frac{nx*ny}{dx*dy}$$

$$\frac{nx}{dx} + \frac{ny}{dy} = \frac{nx*dy + ny*dx}{dx*dy}$$

В общем виде

Реализация арифметики рациональных чисел

```
def mul_rational(x, y):
    return rational(numer(x) * numer(y),
                    denom(x) * denom(y))

def add_rational(x, y):
    nx, dx = numer(x), denom(x)
    ny, dy = numer(y), denom(y)
    return rational(nx * dy + ny * dx, dx * dy)

def print_rational(x):
    print(numer(x), '/', denom(x))

def rationals_are_equal(x, y):
    return numer(x) * denom(y) == numer(y) * denom(x)
```

Конструктор

Селекторы

$$\frac{nx}{dx} * \frac{ny}{dy} = \frac{nx*ny}{dx*dy}$$

$$\frac{nx}{dx} + \frac{ny}{dy} = \frac{nx*dy + ny*dx}{dx*dy}$$

- rational(n, d) возвращает рациональное число x
- numer(x) возвращает числитель x
- denom(x) возвращает знаменатель x

Эти функции реализуют абстрактный тип данных для рациональных чисел

Пары

Представление пар с помощью списков

```
>>> pair = [1, 2]
>>> pair
[1, 2]
```

Задание списка:
Выражения через запятую в квадратных скобках

```
>>> x, y = pair
>>> x
1
>>> y
2
```

«Распаковка» списка

```
>>> pair[0]
1
>>> pair[1]
2
```

Выбор элемента с использованием оператора выбора

```
>>> from operator import getitem
>>> getitem(pair, 0)
1
>>> getitem(pair, 1)
2
```

Функция выбора элемента

Продолжение в следующей лекции

Представление рациональных чисел

```
def rational(n, d):
    """Создает рациональное число, представляющее n/d."""
    return [n, d]
```

Создает список

```
def numer(x):
    """Возвращает числитель рационального числа X."""
    return x[0]

def denom(x):
    """Возвращает знаменатель рационального числа X."""
    return x[1]
```

Выбирает элемент из списка

(Пример)

Сокращение дробей

Пример:

$$\frac{3}{2} * \frac{5}{3} = \frac{5}{2} \qquad \frac{2}{5} + \frac{1}{10} = \frac{1}{2}$$

$$\frac{15}{6} * \frac{1/3}{1/3} = \frac{5}{2} \qquad \frac{25}{50} * \frac{1/25}{1/25} = \frac{1}{2}$$

```
from fractions import gcd

def rational(n, d):
    """Создает рациональное число, представляющее n/d."""
    g = gcd(n, d)
    return [n//g, d//g]
```

Наибольший общий делитель

(Пример)

Границы абстракции

Границы абстракции

Части программы, которые...	Рациональные числа — это...	Используются...
используют рациональные числа для вычислений	отдельные значения данных	`add_rational`, `mul_rational` `rationals_are_equal`, `print_rational`
создают рациональные числа или реализуют операции над ними	числители и знаменатели	`rational`, `numer`, `denom`
реализуют селекторы и конструктор для рациональных числе	двухэлементные списки	списки и выбор элементов

Реализация списков

Нарушение границ абстракции

Не используется конструктор

Дважды!

```
add_rational( [1, 2], [1, 4] )
```

```
def divide_rational(x, y):
    return [ x[0] * y[1], x[1] * y[0] ]
```

Нет селекторов!

И нет конструктора!

Представления данных

Что такое данные?

- Нужно гарантировать, что конструктор и селекторы совместно работают для достижения правильного поведения.
- Условие поведения: Если создать рациональное число x из числителя n и знаменателя d, то numer(x)/denom(x) должно равняться n/d.
- Абстрактный тип данных — это набор некоторых селекторов и конструкторов с учетом условия (условий) поведения.
- Если условия поведения выполняются, тогда представление является правильным.

Абстрактный тип данных определяется поведением, а не классом!

Условия поведения пары

При реализации абстрактного типа данных рациональных чисел используются двухэлементный списки.

Единственный ли это способ создания пар? Нет!

Конструкторы, селекторы и условия поведения:

Если пара p была создана из элементов x и y, тогда

- select(p, 0) возвращает x и
- select(p, 1) возвращает y.

Все селекторы типа вместе противоположны конструктору.

Это справедливо для большинства контейнерных типов.

(Пример)

Не справедливо для рациональных чисел из-за НОД

Функциональная реализация

```
def rational(n, d):
    def select(name):
        if name == 'n':
            return n
        elif name == 'd':
            return d
    return select
```

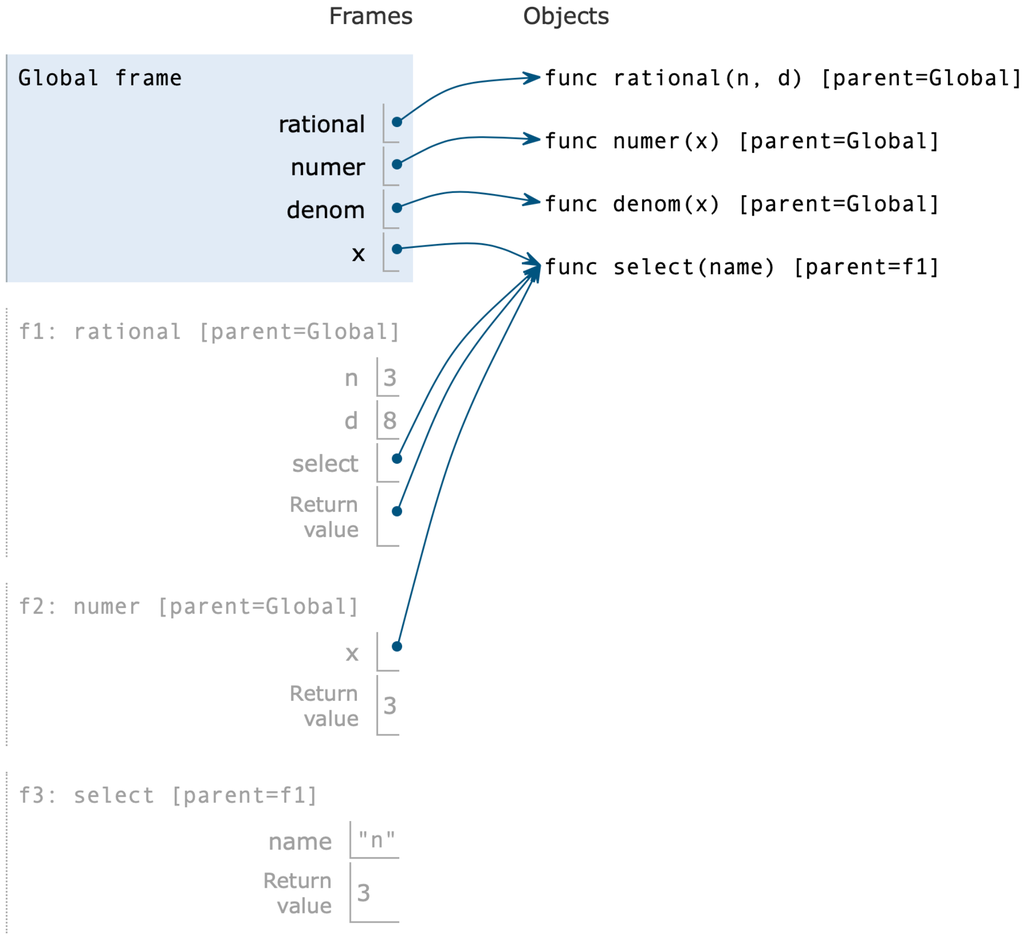
Эта функция отражает рациональное число

Конструктор – функция высшего порядка

```
def numer(x):
    return x('n')

def denom(x):
    return x('d')
```

Селектор вызывает функцию x




```
x = rational(3,8)
numer(x)
```