Практика по дисциплине

Теория принятия решений

Вводное занятие 1

Пономарёв Андрей Васильевич

Организационное

- Много информации, имеющей отношение к задачам:
  - Moodle
  - **https://a-v-ponomarev.github.io/** (далее просто Сайт)
- Три задачи – как правило, две ЛП, одна ДП (есть исключения):
  - Разработать математическую модель
  - Найти решение задачи оптимизации
  - Ответить на вопросы по заданию (исследование полученного решения)
- Варианты:
  - До следующей среды (18.02):
    - Старостам групп будет направлено распределение вариантов (**ответным письмом**)
    - Сами тексты заданий появятся на Сайте

Организационное. Отчетность

- Диф. зачет –> Мустафин Николай Алексеевич
- Практические задания:
  - Определяющее влияние на диф. зачет
  - Отчет:
    - Содержание: формализация, решение, интерпретация, ответы на доп. вопросы (см. Сайт)
    - Оформление: приличное, с достаточным количеством пояснений и визуализаций
    - Формат: блокнот Jupyter или PDF в соответствии с шаблоном на Сайте
  - Индивидуальная защита (очно):
    - Могу попросить изменить какое-нибудь условие, что-то добавить и т.п.
    - Перед защитой прислать электронную версию отчета (**тема письма: «ТПР: XXX», где XXX – номер варианта**)
    - Можно сдавать задачи по мере готовности

Компонент оценки за практику	Максимальный балл
Контрольная точка 1 (27 марта)	5
Контрольная точка 2 (24 апреля)	5
Защита задачи (*3)	30
Небрежное оформление	-5

Рекомендуемая оценка	Баллы
Отлично	85 – 100
Хорошо	50 – 84
Удовлетворительно	30 – 49

Организационное. Отчетность. Контрольные точки

- Цель – убедиться, что вы на правильном пути и в разумные сроки сможете получить результат приемлемого качества
- Контрольная точка может проходится неформально (а можно показывать и готовые куски отчета):
  - показать формальную постановку
  - показать код, решающий основную задачу
  - показать/рассказать как планируется отвечать на ***все*** дополнительные вопросы, указанные в тексте задания (какие нужны модификации кода и пр.)
- Контрольная точка 1: 1 задача (**любая!**)
- Контрольная точка 2: 2 задачи (**любые! В том числе, показанная на КТ 1**)
- Физический механизм прохождения КТ: очно/Moodle/почта

Организационное. Фазы

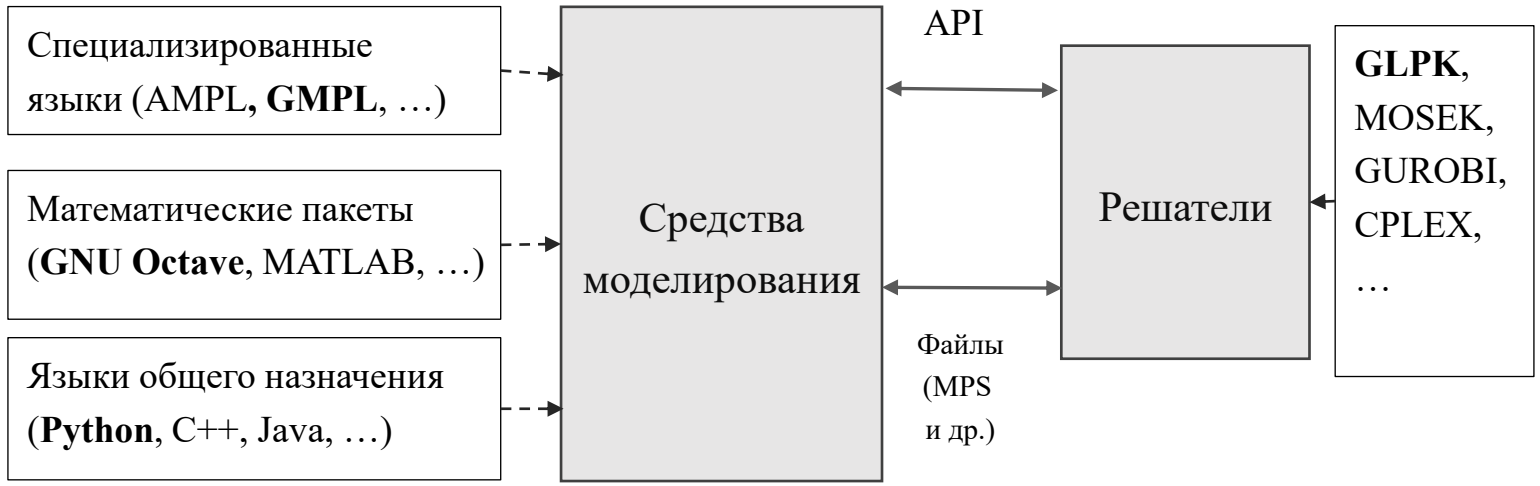
- Фазы нашего взаимодействия:
  1. Введение [*примерно до середины апреля*]
     - разбор у доски отдельных задач и «сюжетов», в них используемых (ЛП, ДП, марковские процессы принятия решений, диаграммы влияния)
     - индивидуальные консультации по задачам в оставшееся время
  2. Консультации + прием КТ и готовых работ [*до даты КТ 2*]
  3. Прием работ [*после даты КТ 2*]

Программное обеспечение

- Инструменты:
  - Рекомендуемый инструмент: Python (Jupyter, CVXOPT/PuLP, seaborn, pandas etc.)
  - Возможно, но не рекомендуется (legacy mode):
    - Для задачи динамического программирования - Octave/C++/Python/Java
    - Для линейного программирования – Octave (glpk) или GLPK (через GUSEK)
- Почему это так?
  - Вы решали задачи линейного программирования, транспортные задачи, "руками" в соответствующих курсах
  - В рамках этого курса будет полезно познакомиться с тем, что представляет собой мир **ПО для оптимизации** и "пощупать" его руками
  - Python. Популярная инфраструктура для обработки и *анализа данных*
    - + язык достаточно широкого назначения (чем выгодно отличается от Octave/R и пр.)

Программное обеспечение. Пояснения

- В мире оптимизации существует два класса приложений:
  - 1) **программные средства для моделирования** - с их помощью вы на понятном, удобном синтаксисе записываете математическую модель вашей задачи.
  - 2) **солверы**. Они либо встраиваемые (линкуются как библиотеки), либо считывают файлы какого-либо из проприетарных или распространенных форматов (mps).



План

- Постановка задачи линейного программирования (ЗЛП) (почти как в индивидуальных заданиях)
- Решение ЗЛП в Python
- Решение ЗЛП в GNU Octave (*очень бегло*)
- Решение ЗЛП в GLPK/MathProg (*очень бегло*)

Постановка задачи линейного программирования (пример)

Условие

- Рацион стада крупного рогатого скота включает **пищевые продукты A, B, C, D, E** (витамины/микроэлементы/питательные вещества).
- В сутки одно животное должно съедать **не менее A1 [кг] продукта A, B1 [кг] продукта B, C1 [кг] продукта C, D1 [кг] продукта D, E1 [кг] продукта E (суточные нормы).**
- В чистом виде указанные продукты не производятся, они содержатся в кормовых культурах **К-1, К-2, К-3, К-4**. Известно **долевое содержание** каждого из продуктов в каждой из культур.
- Известен **запас каждой из кормовых культур (в т)**.
- Определить **максимальное поголовье скота**, которое можно содержать **в течение года**, используя имеющийся запас культур.

Решение

- *Параметры*
  - Будем использовать *i* для индексирования продуктов, *j* – для индексирования культур.
  - Долевое содержание (r_{ij}), $i \in \{1..5\}$, $j \in \{1..4\}$.
  - Суточные нормы продуктов d_i, $i \in \{1..5\}$ [кг].
  - Запас культур s_j [т].
- Переменные
  - *n* – количество животных;
  - $x_j \in \{1..4\}$ – количество культуры *j*-того типа, которую мы используем в кормовом плане [т].
- Тогда

  (ЦФ) *n* -> max

  $x_1 r_{11} + x_2 r_{12} + x_3 r_{13} + x_4 r_{14} >= 0{,}365\ d_1\ n$ - слева количество первого продукта

  …

  $x_1 <= s_1$ (согласовать единицы измерения!)

Линейные задачи. Таблица

	X1	**X2**	**X3**	**X4**	**n**	**Неравенство**	**Правая часть**
f	0	0	0	0	1	-	max
y1	r_{11}	r_{12}	r_{13}	r_{14}	-0,365*d_1	>=	0
…	…	…	…	…	…	…	…
y6	1	0	0	0	0	<=	s_1
…	…	…	…	…	…	<=	…

Решение задачи ЛП в Python

Конфигурирование среды

1. Установить менеджер пакетов, зависимостей и окружений – Conda (Miniconda)
   - Особенно важно для Windows
   - Вариант: весь дистрибутив Anaconda (но это overkill)
2. Сконфигурировать окружение:

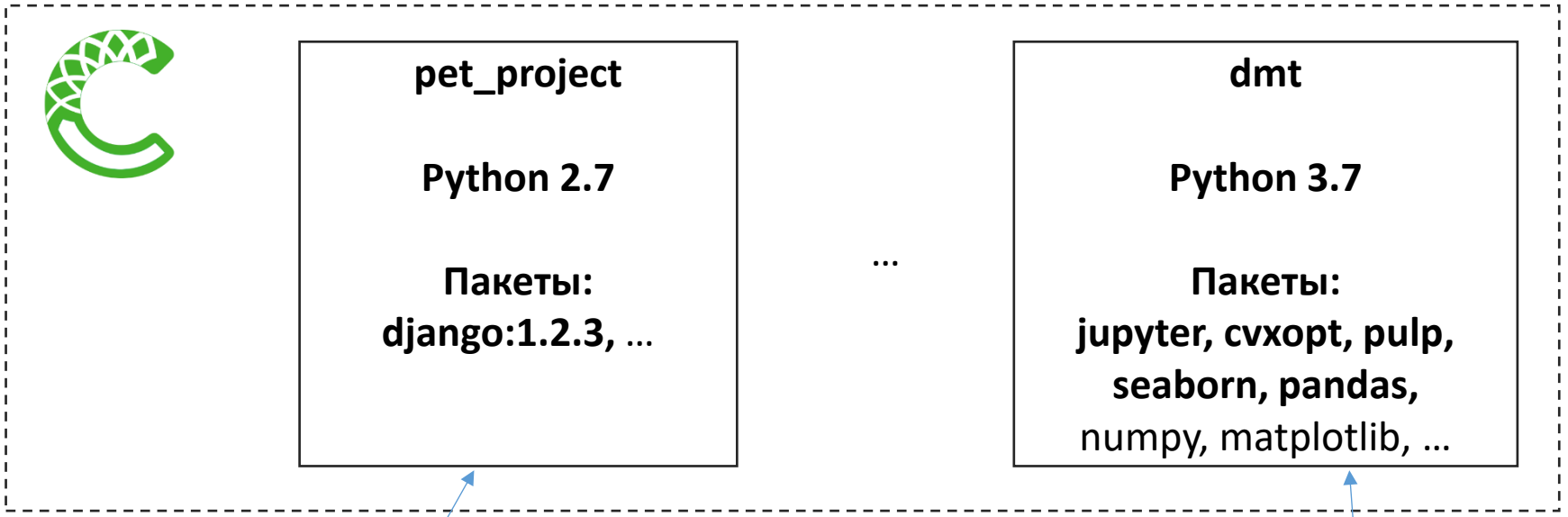
```
conda create --name dmt python=3.11
conda activate dmt
conda install jupyter cvxopt pulp seaborn pandas
jupyter notebook
```

или

создать проект в своей любимой IDE для Python (Pycharm?) указав, что он должен выполняться в соответствующем окружении

Больше информации на https://conda.io/

Менеджер пакетов, зависимостей и окружений Conda



Anaconda Prompt

```
(base) C:\> activate dmt
(dmt) C:\> jupyter notebook
```

Больше информации на https://conda.io/

Docker

```
$ docker pull jupyter/minimal-notebook
$ docker run -it --rm -p 8888:8888 jupyter/minimal-notebook
```

Внутри контейнера:

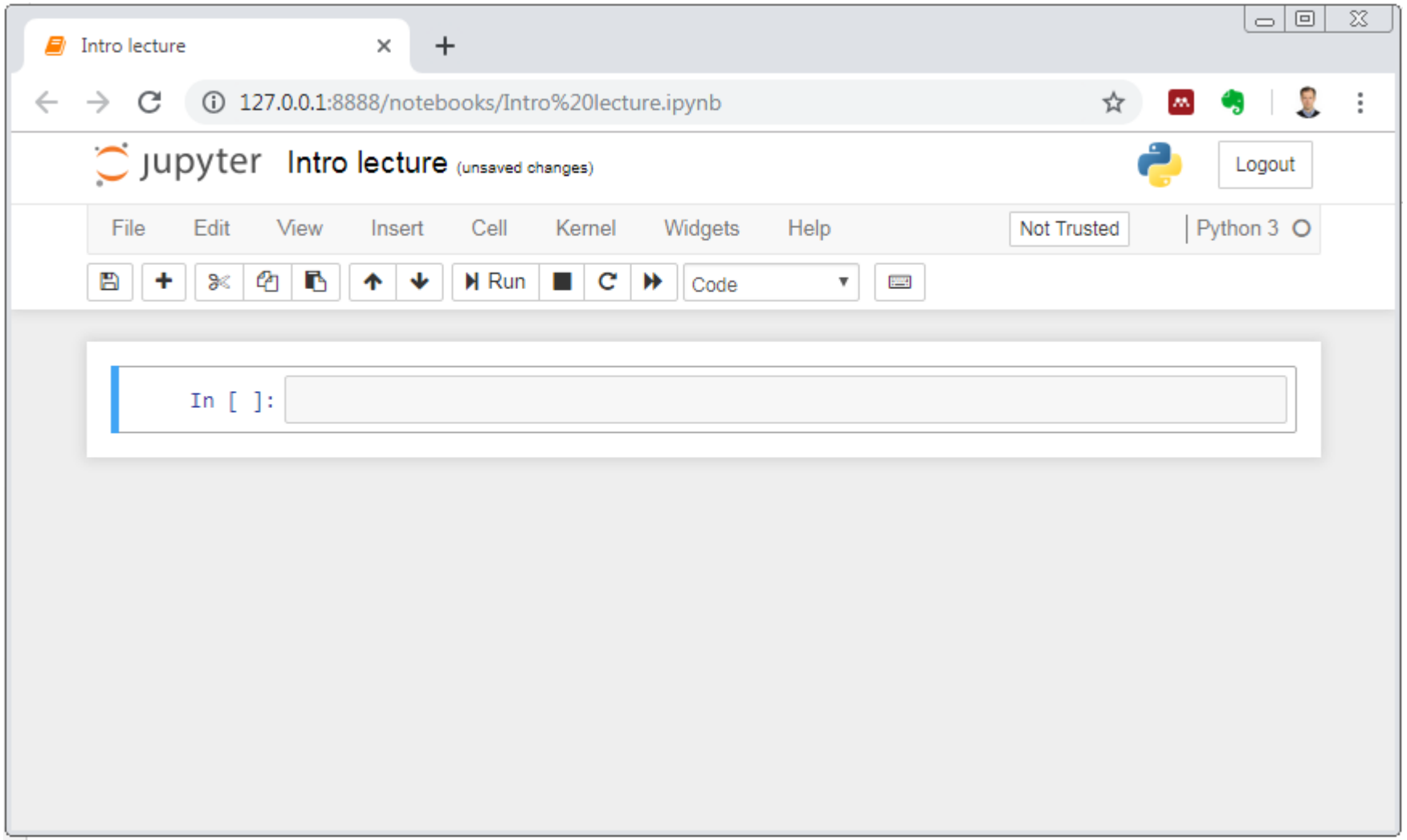
```
$ conda install cvxopt pulp seaborn pandas
```

Google Colaboratory

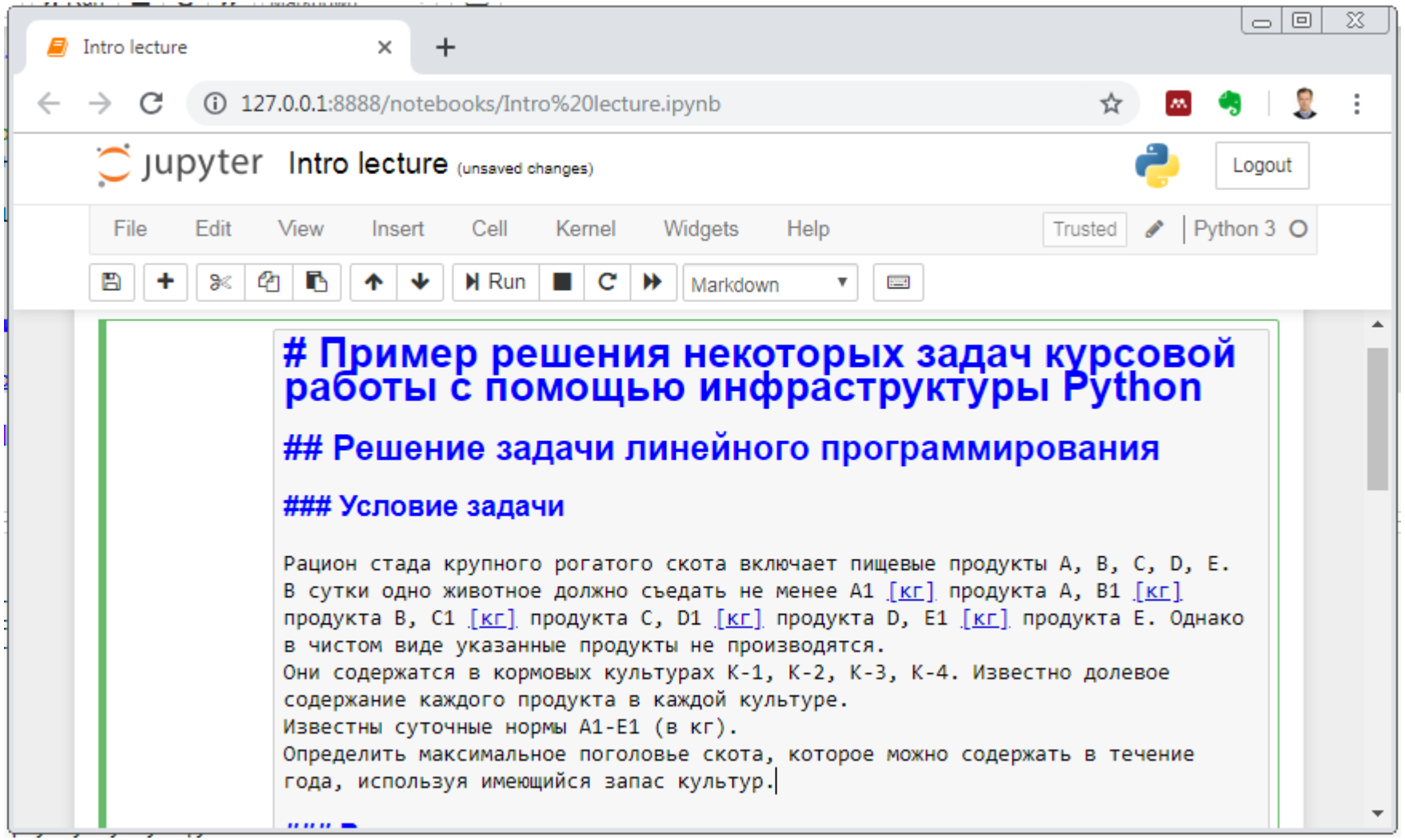
Некоторые библиотеки Python для решения задач ЛП

Библиотека	Матричный интерфейс	Моделирование	MILP	Анализ чувствительности	Пакет conda	Внешние солверы
CVXOPT	**+**	**+/-**	**+**	**+**	**+**	**+**
PuLP	**-**	**+**	**+**	**-**	**+**	**+**
scipy.optimize.linprog	+	-	-	-	+	-
Pyomo	-	+	+	?	+	?

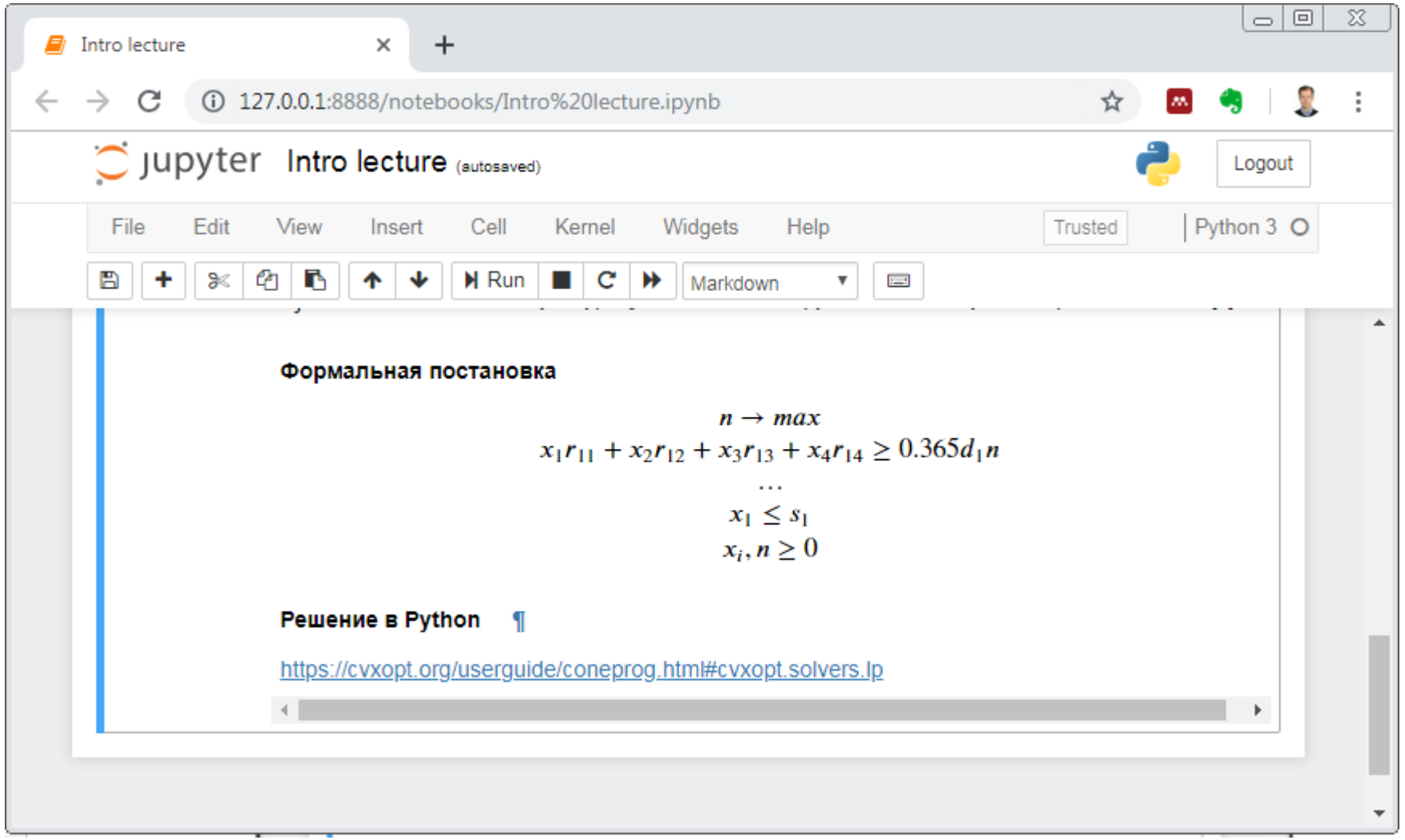
Использование Jupyter



Использование Jupyter



Использование Jupyter



Использование CVXOPT

cvxopt.solvers.lp(*c*, *G*, *h*[, *A*, *b*[, *solver*[, *primalstart*[, *dualstart*]]]])

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Gx + s = h \\ & Ax = b \\ & s \succeq 0 \end{array}$$

$$\begin{array}{ll} \text{maximize} & -h^T z - b^T y \\ \text{subject to} & G^T z + A^T y + c = 0 \\ & z \succeq 0. \end{array}$$

Таким образом:

c – коэффициенты переменных в целевой функции;
G – коэффициенты переменных в ограничениях-неравенствах (<=);
h – правые части ограничений-неравенств (<=);
A – коэффициенты переменных в ограничениях-равенствах;
b – правые части ограничений-равенств.

Использование CVXOPT: заполнение матриц

	x1	**x2**	**x3**	**x4**	**n**	**Неравенство**	**Пр. часть**
c	0	0	0	0	1	-	max (-1)
y1	r_{11}	r_{12}	r_{13}	R_{14}	-0,365*d_1	>= (-1)	0
…	…	…	…	…	…	…	…
y6	1	0	0	0	0	<=	s_1
…	…	…	…	…	…	…	…
y11	1	0	0	0	0	>= (-1)	0
…	…	…	…	…	…	…	…

Использование CVXOPT: заполнение матриц

c – коэффициенты переменных в целевой функции

	x1	x2	x3	x4	n	Неравенство	Пр. часть
c	0	0	0	0	1	-	max (-1)
y1	r_{11}	r_{12}	r_{13}	r_{14}	-0,365*d_1	>= (-1)	0
…	…	…	…	…	…	…	…
y6	1	0	0	0	0	<=	s_1
…	…	…	…	…	…	…	…
y11	1	0	0	0	0	>= (-1)	0
…	…	…	…	…	…	…	…

```
from cvxopt import matrix, solvers

c = matrix([0, 0, 0, -1], tc='d')
```

Использование CVXOPT: заполнение матриц

G – коэффициенты переменных в ограничениях-неравенствах

	x1	x2	x3	x4	n	Неравенство	Пр. часть
c	0	0	0	0	1	-	max (-1)
y1	r_{11}	r_{12}	r_{13}	r_{14}	$-0{,}365*d_1$	>= (-1)	0
…	…	…	…	…	…	…	…
y6	1	0	0	0	0	<=	s_1
…	…	…	…	…	…	…	…
y11	1	0	0	0	0	>= (-1)	0
…	…	…	…	…	…	…	…

```
from cvxopt import matrix, solvers

c = matrix([0, 0, 0, -1], tc='d')

G = matrix([[-r11,-r12,-r13,-r14,0.365*d1],
            [-r21,-r22,-r23,-r24,0.365*d2],
            …
            [  1,    0,    0,    0,        0],
            [  0,    1,    0,    0,        0],
            [  0,    0,    1,    0,        0],
            [  0,    0,    0,    1,        0],
            [ -1,    0,    0,    0,        0],
            [  0,   -1,    0,    0,        0],
            [  0,    0,   -1,    0,        0],
            [  0,    0,    0,   -1,        0],
            [  0,    0,    0,    0,       -1]
           ], tc='d')
```

Использование CVXOPT: заполнение матриц

h – значения правых частей ограничений-неравенств

	x1	x2	x3	x4	n	Неравенство	Пр. часть
c	0	0	0	0	1	-	max (-1)
y1	r_{11}	r_{12}	r_{13}	r_{14}	$-0{,}365*d_1$	>= (-1)	0
…	…	…	…	…	…	…	…
y6	1	0	0	0	0	<=	s_1
…	…	…	…	…	…	…	…
y11	1	0	0	0	0	>= (-1)	0
…	…	…	…	…	…	…	…

```
from cvxopt import matrix, solvers

c = matrix([0, 0, 0, -1], tc='d')

G = matrix([[-r11,-r12,-r13,-r14,0.365*d1],
            [-r21,-r22,-r23,-r24,0.365*d2],
            …
            [  1,    0,    0,    0,        0],
            [  0,    1,    0,    0,        0],
            [  0,    0,    1,    0,        0],
            [  0,    0,    0,    1,        0],
            [ -1,    0,    0,    0,        0],
            [  0,   -1,    0,    0,        0],
            [  0,    0,   -1,    0,        0],
            [  0,    0,    0,   -1,        0]
            [  0,    0,    0,    0,       -1]
           ], tc='d')

h = matrix([0, 0, 0, 0, 0, 0,
            s1, s2, s3, s4
            0, 0, 0, 0, 0], tc='d')
```

Пример решения ЗЛП с помощью CVXOPT

```
from cvxopt import matrix, solvers

c = matrix([0, 0, 0, 0, -1], tc='d')
G = matrix([[-0.05,-0.13,    0,    0,  0.365 * 1.5],
            [-0.10,-0.20,-0.10,    0,  0.365 * 1.4],
            [-0.04,-0.06,-0.15,    0,  0.365 * 1.0],
            [   0, -0.10,-0.08,-0.15,  0.365 * 0.8],
            [-0.15,    0,-0.14,-0.10,  0.365 * 0.6],
            [1, 0, 0, 0, 0],
            [0, 1, 0, 0, 0],
            [0, 0, 1, 0, 0],
            [0, 0, 0, 1, 0],
            [-1, 0, 0, 0, 0],
            [ 0,-1, 0, 0, 0],
            [ 0, 0,-1, 0, 0],
            [ 0, 0, 0,-1, 0],
            [ 0, 0, 0, 0,-1],
           ], tc='d')
```

```
h = matrix([0, 0, 0, 0, 0, 5, 6, 7, 8,
                                    0, 0, 0, 0, 0], tc='d')

solution = solvers.lp(c, G.T, h, solver='glpk')
print('Status: ', solution['status'])
print('x = \n', solution['x'])
print('Objective: ', solution['primal objective'])
```

```
Status: optimal
x =
[ 5.00e+00]
[ 6.00e+00]
[ 8.44e-01]
[ 0.00e+00]
[ 1.88e+00]
Objective: -1.881278538812786
```

Моделирование с помощью CVXOPT (1)

```
from cvxopt.modeling import variable, op


# Определение переменных
x = variable(4, 'x')
n = variable(1, 'n')


# Определение задачи оптимизации
# (всегда минимизация)
problem = op(-x[0] - x[1],    # Ц.Ф.
             [x[0] + x[1] + x[2] + x[3] <= 4,
              x >= 0,
              n >= 0])


# Решение задачи
problem.solve(solver='glpk')
```

```
print('Status: ', problem.status)
print('x = \n', x.value)
print('Objective: ', problem.objective.value())
```

```
Status: optimal
x = [ 4.00e+00]
    [ 0.00e+00]
    [ 0.00e+00]
    [ 0.00e+00]
Objective: [-4.00e+00]
```

Моделирование с помощью CVXOPT (2)

```
from cvxopt.modeling import variable, op

# Определение переменных
x = variable(4, 'x')

# Определение задачи оптимизации
# (всегда минимизация)
problem = op(-sum(x),    # Ц.Ф.
             [x[i] <= x[i+1] - 1 for i in range(3)] + \
             [x <= 10,
              x >= 0])

# Решение задачи
problem.solve(solver='glpk')
```

```
tmp = []
for i in range(3):
    tmp.append(x[i] <= x[i+1] - 1)
```

См. также **PuLP**

Решение задачи ЛП в GNU Octave

Пример решения ЗЛП в Octave (1)

- Параметры функции **glpk**:
  - c – вектор-столбец коэффициентов целевой функции;
  - A – матрица с коэффициентами ограничений;
  - b – вектор-столбец со значениями свободных членов;
  - lb – вектор с нижними границами для каждой переменной (0);
  - ub – вектор с верхними границами для каждой переменной, если нет, то бесконечность;
  - ctype – массив символов с типами ограничений:
    - F – свободное ограничение (не ограничение вовсе);
    - U – <= (upper bound)
    - S - =
    - L - >= (lower bound)
    - D - -b <= Ax <= b
  - vartype – строка с типами переменных:
    - C – непрерывная переменная;
    - I – дискретная переменная.
  - sense – 1 – minimization (default), -1 – maximization.
  - param – структура с дополнительными параметрами.

Пример решения ЗЛП в Octave (2)

- Возвращает:
  - xopt – значения переменных;
  - fopt – значение целевой функции;
  - errnum – код ошибки:
    - 0 – нет ошибки;
    - 2 – вырожденная матрица коэффициентов;
    - 10 – задача не имеет допустимых решений;
    - 11 – двойственная задача не имеет допустимых решений;
  - extra – дополнительная информация о решении:
    - lambda – теневые цены;
    - costs- приведенные цены;
    - status (5 – оптимальное решение).

Нужно совместно анализировать errnum и status!

Пример решения ЗЛП в Octave (3)

	X1 [C]	**X2 [C]**	**X3 [C]**	**X4 [C]**	**n [C]**	**Неравенство**	**b**
c	0	0	0	0	1	-	max (-1)
y1	r_{11}	r_{12}	r_{13}	r_{14}	$-0{,}365*d_1$	>= (L)	0
…	…	…	…	…	…	…	…
y6	1					<= (U)	s_1

Пример решения ЗЛП в Octave (4)

```
1;

c = [0 0 0 0 1]';

A = [0.05 0.13 0    0    -0.365 * 1.5;
     0.10 0.20 0.10 0    -0.365 * 1.4;
     0.04 0.06 0.15 0    -0.365 * 1;
     0    0.10 0.08 0.15 -0.365 * 0.8;
     0.15 0    0.14 0.10 -0.365 * 0.6;
     1  0  0  0 0;
     0  1  0  0 0;
     0  0  1  0 0;
     0  0  0  1 0];
```

```
b = [0 0 0 0 0 5 6 7 8]';

[x_max, z_max, en] = glpk(c, A, b,
   zeros(5, 1),    # lb
   [],             # ub
   "LLLLLUUUU",    # ctype
   "CCCCC",        # vartype
   -1);            # sense

x_max

z_max

en
```

Решение задачи ЛП в GLPK/MathProg

GNU Linear Programming Kit

- GLPK – свободно распространяемый пакет для решения задач ЛП (в том числе, целочисленного ЛП и смешанного ЛП)
- Структура:
  - Библиотека
  - «Обертки» для разных языков
- Язык математического моделирования GMPL (GNU MathProg)
  - Подмножество AMPL

Элементы GMPL

- Модель описывается с помощью объектов следующих видов:
  - Множества
  - Параметры
  - Переменные
  - Ограничения
  - Цели
- Объекты задаются с помощью языковых конструкций (предложений)
  - Декларативные и функциональные
- Разделение модели и данных (разные секции или даже разные файлы)

Элементы GMPL. Множества

- `set` *`name alias domain`*`,` *`attrib`*`, …,` *`attrib`* `;`
  - *`name`* – символическое имя определяемого множества;
  - *`alias`* – необязательная строка, задающая псевдоним множества;
  - *`domain`* – выражение, задающее пространство индексов объекта;
  - *`attrib`*, …, *`attrib`* – необязательные атрибуты множества:
    - `dimen` *`n`* – размерность элементов множества;
    - `within` *`expression`* – «надмножество»;
    - `:=` *`expression`* – состав;
    - `default` *`expression`* – определяет состав множества, если его состав не был определен в секции описания данных.
- Примеры:
  - `set nodes;`
  - `set arcs within nodes cross nodes;`
  - `set DaysOfWeek := {Mon, Tue, Wed, Thu, Fri, Sat, Sun};`

Элементы GMPL. Параметры

- `param` *`name alias domain`*`,` *`attrib`*`, …,` *`attrib`* `;`
  - *`name`* – имя;
  - *`alias`* – псевдоним;
  - *`domain`* – выражение, задающее пространство индексов объекта;
  - *`attrib`*, …, *`attrib`* – необязательные атрибуты:
    - `integer, binary, symbolic` – тип параметра; если этот атрибут не указан, то параметр может принимать любые числовые значения;
    - *`relation expression`* – (где relation может быть одним из <, <=, =, ==, >=, >, <>, !=) задает условие, которому должны удовлетворять значения параметра;
    - `in` *`expression`* – значения параметра должны принадлежать указанному множеству;
    - `within` *`expression`* – «надмножество»;
    - `:=` *`expression`* – состав;
    - `default` *`expression`* – определяет состав множества, если его состав не был определен в секции описания данных.
- Примеры:
  - `param N;`
  - `param N integer;`
  - `param price{e in arcs};`
  - `param demand{i in 1..N};`

Элементы GMPL. Переменные

- `var` *`name alias domain`*`,` *`attrib`*`, …,` *`attrib`* `;`
  - *`name`* – имя;
  - *`alias`* – псевдоним;
  - *`domain`* – выражение, задающее пространство индексов объекта;
  - *`attrib`*, …, *`attrib`* – необязательные атрибуты:
    - `integer, binary` – тип переменной; если не указан, то вещественный;
    - *`relation expression`* – (где *`relation`* может быть одним из <=, =, >=) ограничение на множество значнений.
- Примеры:
  - `var x >= 0;`
  - `var flow{a in arcs} >= 0;`
  - `var take{o in objects} binary;`

Элементы GMPL. Ограничения

- `subject to` *`name alias domain : expr`*`, = expr;`
- `subject to` *`name alias domain : expr`*`, <= expr;`
- `subject to` *`name alias domain : expr`*`, >= expr;`
  - *`name`* – имя;
  - *`alias`* – псевдоним;
  - *`domain`* – выражение, задающее пространство индексов объекта;
  - *`expr`* – линейное выражение, используемое для вычисления компонента ограничения;
- subject to = subj to = s.t.
- Примеры:
  - `s.t. r: x[1] + x[2] <= 17;`
  - `subj to weight: sum{o in objects} w[o] * take[o] <= W_max;`
  - `subject to ration{p in products}: sum{c in crops} x[c] * r[p, c] <= R[p];`

Элементы GMPL. Целевая функция

- `maximize` *`name alias`* `:` *`expr`*`;`
- `minimize` *`name alias`* `:` *`expr`*`;`
  - *`name`* – имя;
  - *`alias`* – псевдоним;
  - *`expr`* – линейное выражение, используемое для вычисления функции;
- Примеры:
  - `maximize total_cost: sum{o in objects} price[o] * take[o];`

Элементы GMPL. Функциональные предложения

- `solve`
- `display`
  - `display : ‘x=‘, x, ‘y=‘, y;`
  - `display{o in objects} : o, take[o];`
- `printf`
  - `printf{i in 1..4} : “x[%d] = %.3f\n“, i, x[i] > “result.txt”;`

Элементы GMPL. Описание данных. Множества (1)

- `set` *`name`*`, ` *`record`*`, …, ` *`record`* `;`
- `set` *`name`* `[`*`symbol`*`, …, `*`symbol`*`] , `*`record`*`, …, record ;`
  - *`name`* – имя множества;
  - *`symbol`*`, …, `*`symbol`* – индексы, задающие конкретный элемент, если у множества *`name`*, определено пространство индексов;
  - *`record`*`, …, `*`record`* – записи с данными:
    - `:=` - необязательная запись, которая может использоваться для повышения читаемости кода;
    - *`simple_data`* – данные в простой форме;
    - `:` *`matrix_data`* – данные в матричной форме;

Элементы GMPL. Описание данных. Множества (2)

- `set nodes := LED VKO KGD TJM ;`
- `set arcs := (LED, VKO) (VKO, LED) (VKO, KGD) (VKO, TJM);`
- `set arcs : LED VKO KGD TJM :=`

```
        LED  -   +   -   -
        VKO  +   -   +   +
        KGD  -   -   -   -
        TJM  -   -   -   - ;
```

Элементы GMPL. Описание данных. Параметры (1)

- `param` *`name`*`, ` *`record`*`, …, ` *`record`* `;`
- `param` *`name`* `default` *`value`*`, ` *`record`*`, …, ` *`record`* `;`
  - *`name`* – имя параметра;
  - *`value`* – значение по умолчанию;
  - *`record`*`, …, ` *`record`* – записи с данными:
    - `:=` - необязательная запись, которая может использоваться для повышения читаемости кода;
    - *`plain-data`* – данные в простой форме;
    - `:` *`tabular-data`* – данные в матричной форме`;`

Элементы GMPL. Описание данных. Параметры (2)

- `param N := 4;`
- `param demand := 1 8 2 14 3 3 4 9;`
- `param r :  1   2    3 :=`
  `1   0.5 0.3 0.12`
  `2   0.2 0.1 0.22 ;`
- `param trans_cost :=`
  `[*,*,wool]:    1   2    3 :=`
  `1   0.3 0.2 0.11`
  `2   0.1 0.1 0.32`
  `[*,*,steel]:   1   2    3 :=`
  `1   0.7 0.4 0.45`
  `2   0.2 0.2 0.64 ;`

Пример решения ЗЛП в GLPK (1)

ration.mod

```
set Products;
set Crops;

param r{i in Products, j in Crops} >= 0, <= 1, default 0;
param d{i in Products} >= 0;
param s{j in Crops} >= 0; # Запасы культур

var n >= 0;  # Искомое количество голов скота
var x {j in Crops} >= 0; # Расход культуры [т]

maximize z: n;

# Ограничения
s.t. Ration{i in Products}: sum{j in Crops} r[i, j] * x[j] >= 0.365 * d[i] * n;  # Рацион по i-тому продукту
s.t. Stock{j in Crops}: x[j] <= s[j]; # Ограничение по запасам j-той культуры

solve;
end;
```

Пример решения ЗЛП в GLPK (2)

ration.dat

```
data;

set Products := A B C D E;
set Crops := K1 K2 K3 K4;

param r : K1    K2    K3    K4 :=
      A   0.05  0.13  0     0
      B   0.1   0.2   0.1   0
      C   0.04  0.06  0.15  0
      D   0     0.1   0.08  0.15
      E   0.15  0     0.14  0.1;

param d A 1.5 B 1.4 C 1 D 0.8 E 0.6;
param s := [K1] 5 [K2] 6 [K3] 7 [K4] 8;

end;
```

Пример решения ЗЛП в GLPK (3)

- Из консоли:
  - glpsol –m ration.mod –d ration.dat –o ration.solution
- Из GUSEK:
  - GUSEK | File | New (сохранить файл .mod)
  - File | New (сохранить в той же директории файл с данными – имя должно совпадать с модельным, расширение - .dat)
  - Tools | Generate Output File on Go (должно быть отмечено)
  - Tools | Go (справа появится вывод консольной GLPK, и откроется новый файл out, содержащий ответ в форме, которую еще нужно научиться читать – но об этом в следующий раз)