PYTHON - СВОБОДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ В ОБРАЗОВАТЕЛЬНОЙ ДЕЯТЕЛЬНОСТИ

PYTHON-FREE SOFTWARE IN EDUCATIONAL ACTIVITIES

УДК 378.14.015.62

Лейко Н. Н., кандидат технических наук, доцент, кафедра автоматики и вычислительной техники, Мурманский государственный технический университет, г. Мурманск

Бучкова З. А., старший преподаватель, кафедра автоматики и вычислительной техники, Мурманский государственный технический университет, г. Мурманск

Leyko N.N., leykonn@yandex.ru

Buchkova Z. A., leykonn@yandex.ru

Аннотация

В данной работе авторы исследуют возможности расширенного применения свободного программного обеспечения в учебном процессе ВУЗа.

Предметом исследования является язык программирования Python и онлайн-сервисы с применением языка программирования Python. Обосновывается идея о том, что применение свободного программного обеспечения для решения задач оптимизации по затратам времени и эффективности лучше, чем проприетарное MS Excel.

На рассматриваемом примере авторы демонстрируют, что для задач возможно создавать образцы - памятки для использования в дальнейшей работе, что значительно облегчит и ускорит решение задач в последующем.

Annotation

In this article, the authors explore the possibilities of expanding the use of free software in the educational process of the University.

The subject of the study is the Python programming language and online services using the Python programming language. The article substantiates the idea that the use of free software for solving optimization problems in terms of time and efficiency is better than proprietary MS Excel.

Using this example, the authors come to the conclusion that it is possible to create sample memos for tasks for use in future work, which will greatly facilitate and speed up the solution of tasks in the future.

Ключевые слова: образование; свободное программное обеспечение (СПО); Python; онлайн-сервисы Python; линейное программирование; оптимизация.

Keywords: education; free software (open source software); Python; Python online services; linear programming; optimization.

Свободно программное обеспечение (СПО) все, в том числе и ВУЗ и студенты могут установить на компьютеры. Обучение студентов использованию СПО, способствует тому, что студенты будут использовать его и после завершения учебы.

Python – это универсальный современный ЯП высокого уровня относится к СПО. Особенно стоит отметить, что Python не обязательно устанавливать на свой компьютер, так как имеются ресурсы в интернете, прежде всего Google Colaboratory, ReplIt, Online GDB и ряд других.

Для повышения заинтересованности изучения языка, рассмотрим пример применения для решения задачи линейной оптимизации, имеющей практическое значение после обучения.

Общей задачей линейного программирования называется задача, которая состоит в определении максимального (минимального) значения линейной функции нескольких переменных при наличии линейных ограничений, т.е. линейных равенств или неравенств, связывающих эти переменные.[1]

Наиболее часто встречающаяся - это задача определения оптимального ассортимента продукции по аналогии с приведенным примером.[2]

Рассмотрим задачу о производстве продукции $P_1 \, и \, P_2$, связанной с планированием производства или оптимальным выпуском продукции.

Математическая постановка задачи. Предприятие выпускает два вида продукции $P_1 \, и \, P_2 .. (n = 2)$. Для производства этих видов продукции $P_1 \, и \, P_2$ используется три типа исходного сырья $(m = 3) - \text{типа} \, \text{M}_1, M_2, M_3$ Необходимо определить такой объем выпускаемой продукции, который максимизирует общую стоимость продукции, а использованные ресурсы не превосходят имеющихся на предприятии запасов (ресурсов).

На производство одной весовой единицы продукции $P_1 \, и \, P_2$ i-го вида ($i \in \{1, 2\}$) требуется $a_{i,j}$ единиц исходного сырья j-го вида ($j \in \{1, 2, 3\}$). Расход сырья для получения каждого вида продукции приводится в таблице (1).

На складе предприятия имеются в наличии запасы (ресурсы) исходного сырья $m_1, m_2 \, и \, m_3$. Стоимость каждого вида продукции составляет $c_1 \, и \, c_2$.

Предположим, что стоимость каждого вида продукции измеряется в *руб/кг,* запасы исходных компонентов (ресурсы) - в *кг,* расход этих компонентов для получения каждого вида продукции - в *кг.*

---

[1] Шадрина Н.И., Берман Н.Д. Решение задач оптимизации в Microsoft Excel 2010 // Хабаровск : Изд-во Тихоокеан. гос. ун-та, 2016 – 101 с.

[2] Там же.

Таблица 1- Расход материалов

Исходное сырье	Продукция	
	P_1	P_2
M_1	$a_{1,1}$	$a_{1,2}$
M_2	$a_{2,1}$	$a_{2,2}$
M_3	$a_{3,1}$	$a_{3,2}$

Исходными переменными математической модели задачи о производстве продукции являются: x_1, а объем выпуска продукции P_1 , x_2 объем выпуска продукции P_2. Тогда математическая постановка рассматриваемой индивидуальной задачи о производстве продукции может быть записана в следующем виде:

$$c_1 * x_1 + c_2 * x_2 \to \max$$

где: max выбирается из множества допустимых альтернатив, сформированных следующей системой ограничений типа неравенств (1):

$$\begin{cases} a_{11} * x_1 + a_{12} * x_2 \le m_1 \\ a_{21} * x_1 + a_{22} * x_2 \le m_2 \\ a_{31} * x_1 + a_{22} * x_2 \le m_3 \\ x_1, x_2 \ge 0 \end{cases} \qquad (1)$$

Для решения предположим, что в нашем варианте

$a_{11} = 0{,}12;\ a_{21} = 0{,}22;\ a_{31} = 0{,}14;\ a_{12} = 0{,}24;\ a_{22} = 0{,}09;\ a_{32} = 0{,}06$

$c_1 = 320;\ c_2 = 290;\ m_1 = 120;\ m_2 = 80;\ m_3 = 60;$

Тогда математическая модель примет вид (2):

$$\begin{cases} F_{ц} = 320 * x_1 + 290 * x_2 \to \max \\ F_1 = 0{,}12 * x_1 + 0{,}24 * x_2 \le 120 \\ F_2 = 0{,}22 * x_1 + 0{,}09 * x_2 \le 80 \\ F_3 = 0{,}14 * x_1 + 0{,}06 * x_2 \le 60 \\ x_1, x_2 \ge 0 \end{cases} \quad (2)$$

Для решения задачи используем пакет Scipy - универсальный пакет для научных вычислений с Python. При необходимости его требуется установить. Команда для установки: `pip install Scipy`

Отметим, что в интернете в свободном доступе есть необходимая информация от разработчиков на английском языке[3], есть и на русском языке[4]. Незначительные различия в источниках для разных версий Scipy несущественны. Используя эти источники, применим из пакета Scipy функцию linprog (входящую в компонент optimize: scipy.optimize.linprog(c, A_ub=None, b_ub=None, A_eq=None, b_eq=None, bounds=None, method='simplex', callback=None, options={'maxiter': 5000, 'disp': False, 'presolve': True, 'tol': 1e-12, 'autoscale': False, 'rr': True, 'bland': False}, x0=None) .

Функция решает следующую задачу оптимизации:

$c(x) \to min$

$A\_ub(x) \le b\_ub$

$A\_eq(x) = b\_eq$

$x \ge 0$

То есть, она поддерживает только два вида ограничений. Не поддерживаемые явно ограничения «≥» преобразуются в «≤».

---

[3] Scipy.org linprog(method='simplex'), электронный ресурс, режим доступа: https://docs.scipy.org/doc/scipy/reference/optimize.linprog-simplex.html

[4] Пономарев А.В. Решение задач линейного программирования с использованием GNU Octave, GLPK и Python, электронный ресурс, режим доступа: https://cais.iias.spb.su/ponomarev/LP_tutorial.pdf

В соответствии с синтаксисом языка Python, те параметры, в определении которых есть оператор присваивания (например, =None), имеют значение по умолчанию, которое будет использовано в том случае, если при вызове функции данный параметр не будет задан. Таким образом, функцию linprog можно вызвать с единственным параметром c.

Для решения нашей задачи применим linprog со следующими параметрами:

linprog(c=k_F, A_ub=k_M, b_ub=res_M, method="revised simplex")

Описание параметров:

c – коэффициенты целевой функции;

A_ub – двумерный массив ndarray (или список списков) с коэффициентами ограничений – верхних границ («≤»);

b_ub – правая часть ограничений – верхних границ;

bounds – ограничения на значения переменных. При значении этого параметра None – все переменные неотрицательны. Пропускаем параметр bounds.

В задаче значение целевой функции необходимо максимизировать, а linprog решает задачу минимизации, следовательно, коэффициенты целевой функции нужно задать с обратным знаком. Все ограничения этой задачи являются ограничениями сверху, которые напрямую поддерживаются функцией, а значит, никаких преобразований производить не нужно. Ограничений типа равенства нет, значит, соответствующие параметры можно не задавать, поскольку для них и так есть значения по умолчанию.

Обе части целевой функции умножаем на -1, и в соответствии с этим вводим значения -320 и -290, то есть, исходное условие:

$F_{\text{ц}} = 320 * x_1 + 290 * x_2 \to \max$ принимает вид

$$F_{ц} = -320 * x_1 - 290 * x_2 \rightarrow \min$$

Скрипт можно набрать в Блокноте Notepad++ , при этом установить кодировку utf-8, выбрать синтаксис Python или в Python создать новый файл набрать скрипт (рис.1).

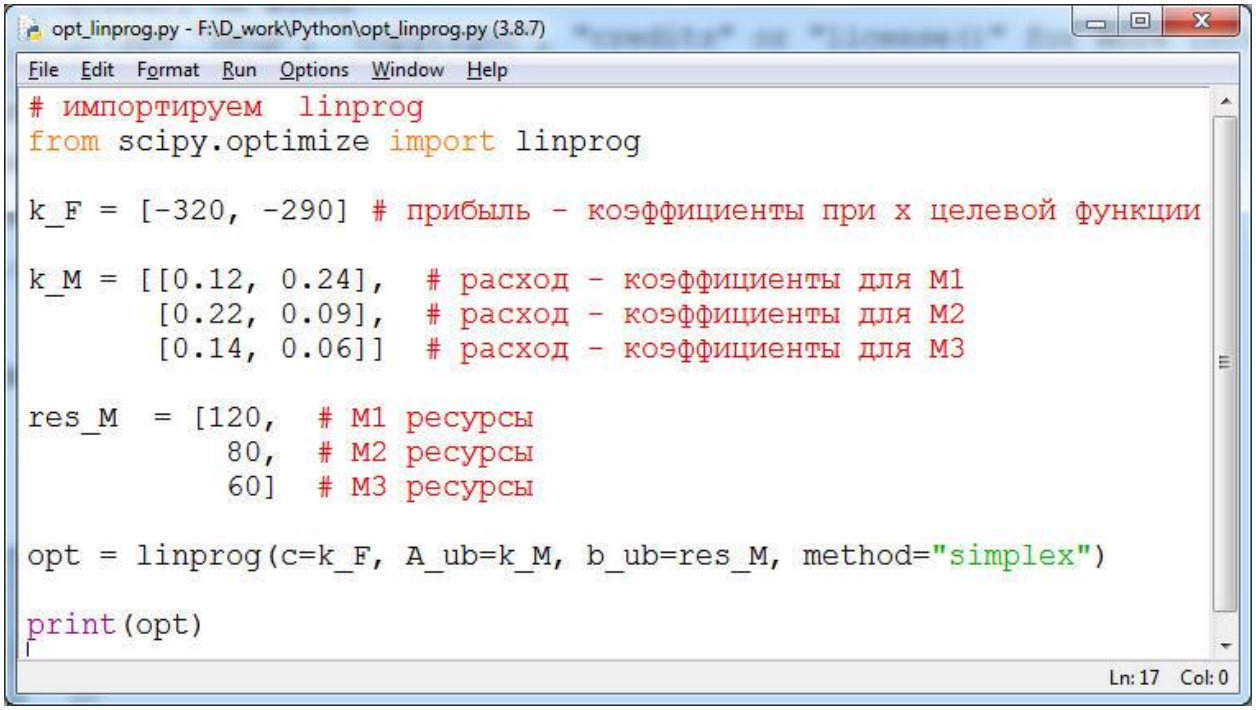

Рис. 1. Скрипт открыт в IDLE Python.

Результат работы скрипта в командном окне Python (рис. 2).

```
==================== RESTART: F:\D_work\Python\opt_linprog.py ========
     con: array([], dtype=float64)
     fun: -180000.0
 message: 'Optimization terminated successfully.'
     nit: 5
   slack: array([-1.42108547e-14,  0.00000000e+00,  8.00000000e+00])
  status: 0
 success: True
       x: array([200., 400.])
>>>
```

Рис. 2. Результат работы скрипта в командном окне Python

Значения результатов работы скрипта.

Возвращаемым значением является объект scipy.optimize.OptimizeResult, содержащий следующие поля:

con: остатки ограничений по равенству. В примере: нет

fun: оптимальное значение целевой функции (если оно найдено). В примере: -180 000

message: это статус решения. В примере: « Optimization terminated successfully» (Оптимизация успешно завершена)

nit: количество итераций, необходимых для завершения расчета. В примере: 5

slack: это значения переменных резерва или разницы между значениями левой и правой сторон ограничений. В примере: для М1 – 0; М2 – 0; М3 – 8 ед.

status: представляет собой целое число между 0 и, 4 которое показывает состояние решения, (0: Оптимизация закончилась успешно; 1: Достигнут итеративный предел; 2: Проблема, кажется, неосуществима; 3: Проблема, кажется, неограниченна; 4: Сталкиваются с числовыми трудностями). В примере: 0, т.е. поиск завершился успешно

success: логическое значение, показывающее, найдено ли оптимальное решение. В примере: true, т.е. найдено оптимальное решение

x: представляет собой массив NumPy, содержащий оптимальные значения переменных решения. В примере: 200 и 400 ед

При вводе скрипта этого примера в онлайн-сервисе repl.it[5] получен этот же результат и (рис. 3).

[5] Pyton – online.Repl.it, электронный ресурс, режим доступа: https://repl.it/languages/python3

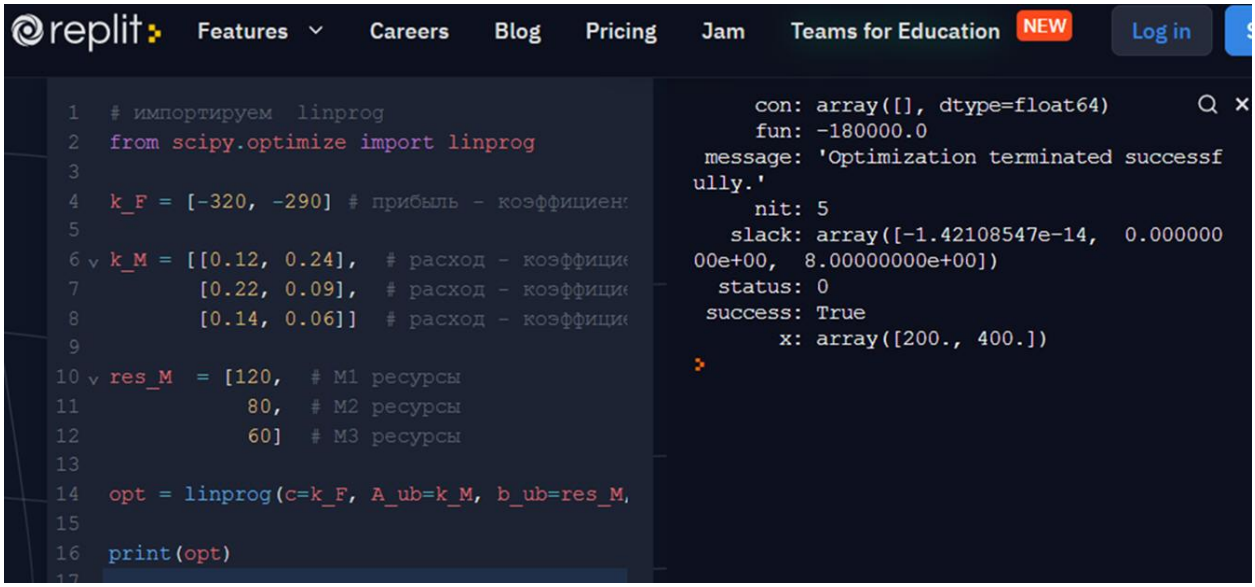

Рис. 3. Результат работы скрипта в командном окне Replit.

Для ввода скрипта в Google Colaboratory необходимо иметь аккаунт Google, зайти на Google-Диск, нажать на кнопку Создать, выбрать Google Colaboratory, и ввести скрипт. При этом делаем незначительное изменение в скрипте: в последней строке вместо print(opt) записываем opt.

В современных условиях рядовой инженер тратит 60% времени на поиск аналога того решения, которое ему необходимо. То есть сидит в интернете и ищет там какую-то готовую модель, которую потом ему надо будет подправить[6].

Продемонстрированный типовой пример, по мнению авторов, может послужить своеобразной памяткой для решения задач данного вида.

Литература

[6] «Мы снова оказались в 1929 году…», электронный ресурс, режим доступа: https://yablor.ru/blogs/mi-snova-okazalis-v-1929-godu/5659828

1. Пономарев А.В. Решение задач линейного программирования с использованием GNU Octave, GLPK и Python, электронный ресурс, режим доступа: https://cais.iias.spb.su/ponomarev/LP_tutorial.pdf

2. Шадрина Н. И., Берман Н. Д., Решение задач оптимизации в Microsoft Excel 2010 : учеб. Пособие //; [науч. ред. Э. М. Вихтенко]. – Хабаровск : Изд-во Тихоокеан. гос. ун-та, 2016. – 101 с.

3. «Мы снова оказались в 1929 году…», электронный ресурс, режим доступа: https://yablor.ru/blogs/mi-snova-okazalis-v-1929-godu/5659828

4. Pyton – online.Repl.it, электронный ресурс, режим доступа: https://repl.it/languages/python3

5. Scipy.org linprog(method='simplex'), электронный ресурс, режим доступа: https://docs.scipy.org/doc/scipy/reference/optimize.linprog-simplex.html

Literature

1. Ponomarev A.V. Solving linear programming problems using GNU Octave, GLPK, and Python, electronic resource, access mode: https://cais.iias.spb.su/ponomarev/LP_tutorial.pdf

2. Shadrina N. I., Berman N. D., Solving optimization problems in Microsoft Excel 2010: textbook. Manual//; [scientific ed. by E. M. Vikhtenko]. - Khabarovsk: Publishing House of the Pacific State University, 2016. – 101 c.

3. " We were back in 1929...", electronic resource, access mode: https://yablor.ru/blogs/mi-snova-okazalis-v-1929-godu/5659828

4. Pyton – online.Repl.it, electronic resource, access mode: https://repl.it/languages/python3

5. Scipy.org linprog(method='simplex'), electronic resource, access mode: https://docs.scipy.org/doc/scipy/reference/optimize.linprog-simplex.html