Введение в Python и Eric

Иван Хахаев, 2009

Линейные алгоритмы. Работа с числами.

Пусть требуется решить «классическую» (в смысле, часто встречающуюся в начальных курсах по алгоритмизации) задачу перестановки двух чисел . Реализация этого алгоритма опять-таки «классическим» способом (т.е. в духе языков BASIC или Pascal, традиционно используемых для обучения реализации алгоритмов) приведена ниже. Один из вариантов решения такой задачи, который будет работать при использовании любых языков программирования — использование промежуточной переменной.

```
# -*- coding: utf-8 -*-
#Перестановка местами двух чисел
#с использованием промежуточной  переменной
#
a=input("Первое число (a): ")
b=input("Второе число (b): ")
c=a
a=b
b=c
print "Новое значение a: ", a
print "Новое значение b: ", b
```

Важно понять, что для ввода чисел необходимо использовать функцию `input()`, а не `raw_input()` (`raw_input()` означает «ввод необработанной последовательности байтов»).

Инструкция (оператор) `print` может выводить произвольные значения, если их разделить запятой. Количество выводимых значений (строковых констант, числовых или строковых переменных) не ограничено.

В рассмотренной нами программе решения такой простой задачи получилось 7 (семь) строк кода (не считая комментариев). Возможно ли уменьшить количество строк (объём программы)?

Оказывается, что средства языка Python действительно позволяют уменьшить число строк в решениях подобных задач.

Рассмотрим следующий вариант кода (`script-02-b.py`):

```
# -*- coding: utf-8 -*-
#Перестановка местами двух чисел с использованием кортежа
#
(a,b)=input("Введите исходные значения (a, b) через запятую: ")
(a,b) = (b,a)
print "Новое значение a: ",a,"\n","Новое значение b: ",b
```

Текст программы сократился до 3 (трёх) строк.

Здесь мы сталкиваемся сразу с несколькими особенностями языка Python:

1. Функция может иметь своим результатом несколько значений

2. Такая функция в качестве результата выдаёт особую ***структуру данных — кортеж*** (набор значений, разделённых запятой). В кортеж могут входить элементы разных типов, в том числе и кортежи. Элементы кортежа нумеруются с нуля.

3. В операторе `print` могут использоваться ***управляющие последовательности*** символов.

Управляющая последовательность `"\n"` в операторе `print` означает переход на новую строку (new line). Кроме этого, существуют управляющие последовательности `"\t"` (tabulation) и `"\c"` (carriage return). Символ `"\t"` полезно использовать для выравнивания результатов по столбцам, а `"\c"` - для возврата назад на одну позицию (если это зачем-то нужно).

Рассмотрим ещё один пример программы (`script-02-c.py`), а потом обсудим свойства и особенности кортежей.

```
# -*- coding: utf-8 -*-
#Упражнение с кортежами
#
k1=(a,b,c)=input("Введите исходные значения (a, b, c) через
запятую: ")
(c,b,a)=(a,b,c)
b=5
print "Новое значение a: ",a,"\n","Новое значение b: ",b
print "Новое значение c: ",c
print "Третий элемент кортежа:", k1[2]
```

В первой строке программы формируется кортеж с именем `k1`, состоящий из трёх значений, присваиваемых соответственно, переменным `a`, `b` и `c`. Обратите внимание, что Python позволяет всё это описать в одном операторе присваивания. Затем формируется новый кортеж с изменённым порядком элементов исходного кортежа. Потом элементу нового кортежа с именем `b` присваивается числовое значение. После этого выводятся значения элементов нового кортежа и значение третьего (последнего) элемента исходного кортежа.

Запустите эту программу на выполнение и посмотрите, что она выводит при значениях `a`, `b` и `c`, равных, соответственно, 1, 2 и 3.

Кортеж (в англоязычной документации по Python — «tuple») может содержать ноль элементов (`k=()` – пустой кортеж), один элемент (`k=(a,)`) и любое другое количество элементов. Однако количество элементов в кортеже не может изменяться в ходе выполнения программы. К элементам кортежа, как мы уже видели, можно обращаться как по именам, так и номерам (с учётом того, что первый элемент в кортеже имеет номер 0).

Теперь посмотрим, что получится, если ввести не три исходных значения, а больше (ведь в окне выполнения не написано. сколько значений нужно ввести)? В этом случае мы как раз и получим ошибку времени выполнения («исключение» в терминологии Python, рис. 1).

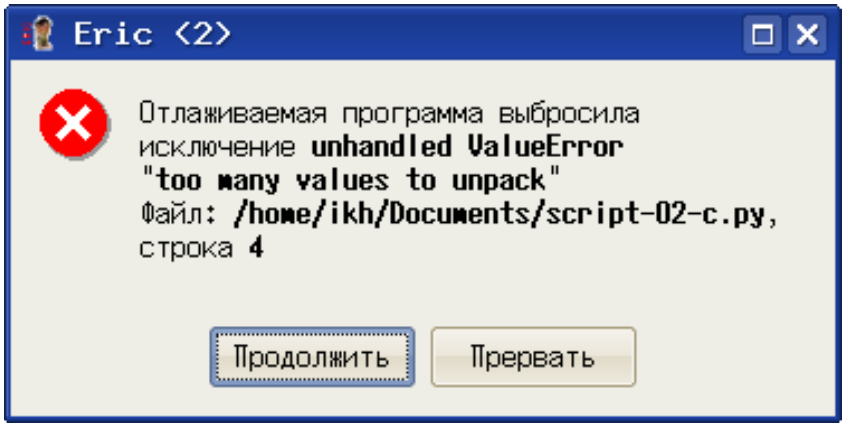

Рисунок 1. Сообщение IDE Eric об ошибке времени выполнения

Проанализировав это сообщение, можно предположить, что мы пытаемся записать в кортеж больше значений, чем он может содержать («too many values» - «слишком много значений...»). В этом случае программу нужно прервать и запустить на выполнение снова.

В заключение этой главы приведём ещё один пример, касающийся работы с числами (`script-03.py`).

```
#  -*- coding: utf-8 -*-
# Задача: Известны оклад и ставка процента подоходного налога.
# Определить размер подоходного налога и сумму к выдаче.
# ДАНО:
# oklad - размер оклада.
# proc - процент подоходного налога.
# НАЙТИ:
# nalog - размер налога.
# summa - сумма к выдаче.
#
oklad=input("Оклад: ")
proc=input("% налога: ")
nalog=oklad*proc/100
summa=oklad-nalog
print "Сумма на руки: ",summa
print "Уплачено налога: ",nalog
```

На рис. 2 показан пример выполнения программы.

```
>>> Оклад: 6543
% налога: 13
Сумма на руки:  5693
Уплачено налога:  850

>>>
```

Рисунок 2. Решение задачи о налоге

Однако решение неправильное! Ведь `6543*13/100 = 850,59` ! Куда подевались копейки? А дело в том, что Python воспринимает числа по их написанию. Раз в исходных данных отсутствует дробная часть, Python считает, что все числа должны быть целыми и отбрасывает дробную часть результата деления.

Чтобы всё считалось правильно, нужно какое-то из исходных данных ввести как вещественное число (с дробной частью в виде десятичной дроби, рис. 3):

```
>>> Оклад: 6543.0
% налога: 13
Сумма на руки:  5692.41
Уплачено налога:  850.59
```

Рисунок 3. Правильное решение задачи о налоге

Обратите внимание, что десятичный разделитель — точка!