Введение в Python и Eric

Иван Хахаев, 2009

Ассоциативные массивы.

Ассоциативный массив лучше всего описывается табличным представлением данных, когда каждая строка таблицы описывает характеристики какого-то объекта из множества однородных объектов (типичный пример — список учеников, их домашних телефонов и адресов). Таким образом, по значению из первого столбца такой таблицы (ключу) можно однозначно определить значения из остальных столбцов, т.е. значение ключа ассоциируется с остальными характеристиками объекта (в случае ученика — по фамилии можно найти другую информацию).

Если в ассоциативном массиве только два столбца («ключ» и «значение»), то такой массив называется «хэш». Такие ассоциативные массивы очень часто используются в современных информационных системах (например, пары «логин-пароль»).

Для хэш-массивов в Python определена специальная структура данных — словарь, с которой можно обращаться аналогично списку, но словарь содержит пары «ключ-значение».

Однако довольно часто встречаются задачи (особенно в области моделирования процессов и явлений), в которых значению «ключа» соответствует несколько параметров (например, номеру химического элемента однозначно соответствует название, атомный вес, валентность, количество протонов и пр.). В таких задачах простые хэш-массивы использовать уже неудобно. Конечно, можно сделать несколько пар «ключ-значение», а потом их связывать по значению ключа.

Рассмотрим пример задачи по обработке ассоциативного массива и вариант её решения с использованием списков и элементов функционального программирования в Python (`script-10.py`).

```
#  -*- coding: utf-8 -*-
# Решить задачу, связанную с оценкой экономической
# деятельности группы предприятий на основе известных данных:
# Название предприятия
# Плановый объем розничного товарооборота
# Фактический объем розничного товарооборота
# Требуется определить:
#   1. процент выполнения плана каждым предприятием
#   2. количество предприятий, недовыполнивших план
#   3. наибольший плановый товарооборот
#   4. упорядочить предприятия по возрастанию планового
товарооборота.
#
# k - количество предприятий
# name - список названий предприятий
# plan - список значений планового товарооборота
# fact - список значений фактического товарооборота
# proc - список значений % выполнения плана
#
k=input("Количество предприятий: ")
```

```
name=[]
plan=[]
fact=[]
proc=[]
#
for i in range(k):
    n=raw_input("Название: ")
    name.append(n)
    p1=input("План: ")
    plan.append(p1)
    p2=input("Факт: ")
    fact.append(p2)
    proc.append(p2*100/p1)
#
fakty=map(None,name,proc)
plany=map(None,plan,name)
plany.sort()

print 16*"="
print "Процент выполнения плана каждым предприятием:"
#
nedo=0
for i in range(k):
    s1=fakty[i][0]
    s2=fakty[i][1]
    if s2 < 100:
        nedo=nedo+1
    #
    print s1, ": ",s2

print "Количество предприятий, недовыполнивших план: ", nedo
print "Наибольший плановый товарооборот: ", max(plan)
#
print "Предприятия по возрастанию плана:"

for i in range(k):
    s1=plany[i][1]
    s2=plany[i][0]
    print s1, ": ",s2
```

Пример выполнения программы в IDE Eric показан на рис. 1.

```
>>> Количество предприятий: 4
Название: Фанта
План: 56
Факт: 58
Название: Прима
План: 49
Факт: 47
Название: Люкос
План: 112
Факт: 121
Название: Мона
План: 94
Факт: 88
================
Процент выполнения плана каждым предприятием:
Фанта : 103
Прима : 95
Люкос : 108
Мона : 93
Количество предприятий, недовыполнивших план: 2
Наибольший плановый товарооборот: 112
Предприятия по возрастанию плана:
Прима : 49
Фанта : 56
Мона : 94
Люкос : 112

>>>
```

Рисунок 1. Решение задачи по анализу работы предприятий

Описание основных переменных приведено в комментариях в тексте программы.

Здесь формируется четыре списка — три на основе исходных данных (название, план и факт), а четвёртый — как результат вычислений.

С помощью функции `map()` формируются списки кортежей (первый аргумент функции `map()` позволяет задать операцию, которую. нужно выполнить со списками, имена которых являются остальными аргументами, а если первым аргументом является ключевое слово `None`, то никаких операций не производится), в каждый из которых попадает пара элементов из соответствующих списков. Таким образом на каждом этапе выполнения программы получаем нужные пары «ключ-значение».

Для списка `plany` порядок аргументов в функции `map()` изменён, поскольку сортировка производится по «первому столбцу».

При формировании вывода по процентам выполнения плана попутно очевидным способом считается количество предприятий, не выполнивших план (переменная `nedo`).

Упражнение: Определите необходимость использования функции `map()` при выводе процентов выполнения плана и напишите свой вариант этого вывода.

Аналогично функции `map()` с первым аргументом `None` действует функция `zip()`, также формирую список кортежей.