Algoritmos e Programação I: Aula 34*

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
79070-900 Campo Grande, MS
http://ava.ufms.br

Sumário

1 Lista de Listas

Na solução do problema das médias de um conjunto de alunos, vimos que o armazenamento das notas dava um certo trabalho, pois tínhamos 5 notas para cada estudante, e ou usávamos 5 listas diferentes, uma para cada avaliação ou líamos as notas e só armazenávamos a média de cada aluno. Uma forma mais eficiente de resolver esse problema é usar uma lista de listas, onde cada elemento da lista também é uma lista com as 5 notas de cada estudante.

Exemplo

```
# lista das listas das notas
notas = [[7.5, 9.0, 9.0, 8.0, 9.0], [6.0, 6.0, 7.5, 8.0, 5.0],
         [5.0, 4.0, 6.0, 8.5, 9.5], [8.3, 7.9, 9.3, 9.1, 9.0],
         [7.8, 9.6, 8.8, 9.1, 8.9]]
```

nesse caso temos que:

```
# lista das listas das notas
notas[0] = [7.5, 9.0, 9.0, 8.0, 9.0]
notas[1] = [6.0, 6.0, 7.5, 8.0, 5.0]
notas[2] = [5.0, 4.0, 6.0, 8.5, 9.5]
notas[3] = [8.3, 7.9, 9.3, 9.1, 9.0]
notas[4] = [7.8, 9.6, 8.8, 9.1, 8.9]
```

e além disso, possemos acessar os elementos individuais de cada uma das listas:

*Este material é para o uso exclusivo da disciplina de Algoritmos e Programação I da FACOM/UFMS e utiliza as referências bibliográficas da disciplina.

```
# lista das notas do estudante 2
notas[1][0] = [6.0]
notas[1][1] = [6.0]
notas[1][2] = [7.5]
notas[1][3] = [8.0]
notas[1][4] = [5.0]
```

Vamos agora descrever como definir uma lista de listas e outras operações que podem ser feitas nesse caso.

A declaração de uma lista de lista segue a ideia de uma lista simples. Para atribuir os valores das notas de cada um dos estudantes, podemos usar a seguinte declaração:

```
# lista das listas das notas
tam = int(input()) # número de estudantes

notas = [[0.0]*5 for i in range(tam)]
```

1.1 Média das Notas 3

Vamos retornar ao problema da Média das Notas, onde agora queremos imprimir além da média, as notas de cada estudante. A média da disciplina é dada por 75% da média aritmética das três provas mais 25% da média aritmética dos dois trabalhos. O programa deve calcular a média da disciplina e ordenar o conjunto pela ordem não crescente das médias. O programa deve imprimir os nomes dos estudantes, as três notas das provas, as duas notas dos trabalhos, as médias, sendo que as médias devem estra ordenadas. Para a ordenação vamos utilizar o método de ordenação por inserção.

A entrada é dada por um arquivo onde a primeira linha contém um inteiro n representando o número de alunos da disciplina, seguido de n linhas, onde cada linha contém o número, nome e notas de cada um dos alunos. Para armazenar essas informações, usaremos três listas com n elementos, `numeros` para os números, `nomes` para os nomes e `medias` para as médias. No caso das notas, vamos utilizar uma lista de listas `notas`, onde cada elemento de `notas` é uma lista com as 5 notas do estudante.

```
notas[i] = [notas[i][0],notas[i][1],notas[i][2],notas[i][3],notas[i][4]]
```

Da mesma forma que no exemplo anterior, vamos ler cada cada linha do arquivo como uma string e usando os separadores, e depois, usando o `split()`, vamos atribuir os dados a cada uma das listas.

Exemplo

```
# formato da entrada
5
10122, Karin Tuiuiu, 7.5 9.0 9.0 8.0 9.0
11201, Carlos Bigua, 6.0 6.0 7.5 8.0 5.0
12304, Turibio Garca, 5.0 4.0 6.0 8.5 9.5
10101, Maria Dourado, 8.3 7.9 9.3 9.1 9.0
12305, Bartolomeu Pintado, 7.8 9.6 8.8 9.1 8.9
```

Para usar a atribuição direta, temos que inicializar as listas:

```
# Inicialize as abstrações
numeros = ['']*tam
nomes = ['']*tam
notas = [[0.0]*5 for i in range(tam)]
medias = [0.0]*tam
```

além disso, vamos usar a string `dados` para ler a linha toda e a string `avalia` para armazenar o conjunto de notas da string `dados`.

```
1 # Leia os dados dos alunos e calcule as médias
2 for i in range(tam):
3    dados = input() # lê a linha inteira
4    numeros[i],nomes[i],avalia = dados.split(', ') # separa em 3 strings
5    notas[i] = list(map(float,avalia.split()))
```

onde a linha 3 lê a linha inteira do arquivo. Na linha 4, a linha é separada usando as vírgulas (“,”s), onde a string `numeros[i]` recebe a primeira parte da string `dados`, a string `nomes[i]` recebe a segunda parte da string `dados` e a string `avalia` recebe a terceira parte da string `dados`. Após a atribuição da string `avalia`, na linha 5 usamos novamente o `split()` (com espaços) para separar as notas e atribuir aos elementos da lista `notas[i]`, `notas[i][0]`, `notas[i][1]`, `notas[i][2]`, `notas[i][3]` e `notas[i][4]`.

O programa abaixo ilustra uma solução do problema:

```
1  # -*- coding: utf-8 -*-
2  '''
3  # Programa: nomes01.py
4  # Programador:
5  # Data: 20/11/2019
6  # Este programa lê os dados referentes ao número, nome, as 3 notas das
7  # provas e as 2 notas de trabalho dos alunos da disciplina de Algoritmos e
8  # Programação I. A média das provas é dada por MP = (P1+P2+P3)/3.0, a
9  # média dos trabalhos por MT = (T1 + T2)/2.0 e a média final por
10 # MF = 0.75*MP + 0.25*MT. O programa computa a média final, ordena os
11 # dados dos alunos usando a média e imprime o número, nome e a média final
12 # de cada aluno, onde as notas estão ordenadas em ordem não crescente.
13 # Os dados de cada aluno são dados em uma linha, onde as informações são
14 # separadas por vírgulas.
15 # descrição das variáveis utilizadas
16 # list    numeros, nomes, notas, medias
17 # string  dados, avalia
18 # int     tam
19 '''
20
21 # pré: tam para i em 0,.,tam-1: numero[i], nome[i], notas[0] notas[1] notas[2]
22 #      notas[3] notas[4]
23
24 # Passo 1. Leia a entrada e inicialize as variáveis
25 # Passo 1.1. Leia o número de alunos
26 tam = int(input())
```

```
# Passo 1.2. Inicialize as abstrações
numeros = ['']*tam
nomes = ['']*tam
notas = [[0.0]*5 for i in range(tam)]
medias = [0.0]*tam
# Passo 1.3. Leia os dados dos alunos e calcule as médias
for i in range(tam):
   dados = input()
   numeros[i],nomes[i],avalia = dados.split(', ')
   notas[i] = list(map(float,avalia.split()))
# Passo 2. Calcule a média e ordene os dados pela média
# Passo 2.1. Calcule a média dos trabalhos
for i in range(0,tam):
   mprovas = (notas[i][0] + notas[i][1]+notas[i][2])/3.0
   mtrabs = (notas[i][3]+notas[i][4])/2.0
   medias[i] = 0.75*mprovas + 0.25*mtrabs
# Passo 2.2. Ordene a lista medias[0:i]
for i in range (1,tam):
   temp1 = medias[i]
   temp2 = numeros[i] # temos que ordenar junto numeros[i]
   temp3 = nomes[i] # temos que ordenar junto nomes[i]
# Passo 2.2.1. Insira temp1 em medias[0:i] | medias[i-1]<temp1<medias[i+1]
   j = i - 1 # inicie no final da lista medias[0:i-1]
   while j >=0 and temp1 > medias[j]:
# Passo 2.2.1.1. Mova os elementos para a direita em cada lista
      medias[j+1] = medias[j]
      numeros[j+1] = numeros[j]
      nomes[j+1] = nomes[j]
      j = j - 1 # mova a comparação para esquerda
# Passo 2.2.3. Insira os elementos na posição j (j+1) de cada lista
   medias[j+1] = temp1
   numeros[j+1] = temp2
   nomes[j+1] = temp3
# Passo 3. Imprime os resultados
for i in range(0,tam):
   print('0:s 1:s 2:.1f'.format(numeros[i],nomes[i],medias[i]))

# pós: para i em 0 .. tam-1:numeros[i] nomes[i] medias[i] and
#      medias[i] == 0.75*(notas[0]+notas[1]+notas[2])/3.0 +
#      0.25*(notas[3]+notas[4])/2.0 and medias[i] >= medias[i+1]
# fim do módulo principal
```

A saída consiste em escrever a lista ordenada pelas médias dos estudantes usando o seguinte formato:

Exemplo

```
# formato da entrada
5
10122, Karin Tuiuiu, 7.5 9.0 9.0 8.0 9.0
11201, Carlos Bigua, 6.0 6.0 7.5 8.0 5.0
```

```
12304, Turibio Garca, 5.0 4.0 6.0 8.5 9.5
10101, Maria Dourado, 8.3 7.9 9.3 9.1 9.0
12305, Bartolomeu Pintado, 7.8 9.6 8.8 9.1 8.9

# após o Passo 1.3
['10122', '11201', '12304', '10101', '12305']
['Karin Tuiuiu', 'Carlos Bigua', 'Turibio Garca', 'Maria Dourado',
                                                    'Bartolomeu Pintado']
[8.5, 6.5, 6.0, 8.6, 8.8]

# ordenação
[8.5, 6.5] # não altera
['10122', '11201']
['Karin Tuiuiu', 'Carlos Bigua']
[8.5, 6.5, 6.0]  # não altera
['10122', '11201', '12304']
['Karin Tuiuiu', 'Carlos Bigua', 'Turibio Garca']
[8.5, 6.5, 6.0, 8.6] # move para a posição 0
['10122', '11201', '12304', '10101']
['Karin Tuiuiu', 'Carlos Bigua', 'Turibio Garca', 'Maria Dourado']
[ 8.6, 8.5, 6.5, 6.0,] # move para a posição 0
[ '10101', '10122', '11201', '12304']
['Maria Dourado', 'Karin Tuiuiu', 'Carlos Bigua', 'Turibio Garca']
[8.6, 8.5, 6.5, 6.0, 8,8] # move para a posição 0
['10101', '10122', '11201', '12304', '12305']
['Maria Dourado', 'Karin Tuiuiu', 'Carlos Bigua', 'Turibio Garca',
                                                    'Bartolomeu Pintado']
[8,8, 8.6, 8.5, 6.5, 6.0] # move para a posição 0
['12305', '10101', '10122', '11201', '12304']
['Bartolomeu Pintado', 'Maria Dourado', 'Karin Tuiuiu', 'Carlos Bigua',
                                                        'Turibio Garca']

# formato da saída
12305 Bartolomeu Pintado 7.8 9.6 8.8 9.1 8.9 8.8
10101 Maria Dourado 8.3 7.9 9.3 9.1 9.0 8.6
10122 Karin Tuiuiu 7.5 9.0 9.0 8.0 9.0 8.5
11201 Carlos Bigua  6.0 6.0 7.5 8.0 5.0 6.5
12304 Turibio Garca 5.0 4.0 6.0 8.5 9.5 6.0
```

1.2 Lista Telefônica 2

Na solução do problema da lista telefônica, usamos três listas para armazenar os dados dos assinantes. A primeira lista armazenava a string com o primeiro nome do assinante, a segunda lista armazenava a string com sobrenome e a terceira armazenava a string com o respectivo número do telefone do assinante. O problema da lista telefônica consiste em ordenar os assinantes de Big Field da Companhia Telefônica Pantaneira pelo último nome do assinante, e a base de assinantes só contém o nome, sobrenome e o número de telefone de cada assinante. Além disso, não existem dois clientes com o mesmo nome e sobrenome. Vamos agora usar apenas uma lista `catalogo`, onde `catalogo[i]` é também uma lista com

Nome	Sobrenome	Telefone

Figura 1: Lista Telefônica

três strings, para armazenar os dados dos assinantes.

```
catalogo[0] = ['Owl', 'Tuiuiu', '1234-5786']
catalogo[0] = ['Hippopotamus', 'Sucuri', '1234-6758']
catalogo[0] = ['Puma', 'Caiman', '1234-8765']
catalogo[0] = ['Zebra', 'Quati', '1234-8756']
...
```

Após a execução do programa, a lista estará ordenada pelos sobrenomes, e como é uma única lista, na ordenação a abordagem é um pouco diferente.

Exemplo

```
# formato da entrada
25
Owl Tuiuiu 1234-5786
Hippopotamus Sucuri 1234-6758
Puma Caiman 1234-8765
Zebra Quati 1234-8756
Tiger Capivara 1234-5687
Skank Morcego 1234-7658
Codfish Tucano 1234-4678
Aligator Arara 1234-5678
Tortuga Jaboti 1234-8576
Condor Urutu 1234-5876
Parrot Dourado 1234-6587
Eagle Raposa 1234-8657
Python Dourado 1234-8675
Gnu Sucuri 1234-6798
Lizard Tucano 1234-5768
Bear Bigua 1234-6789
Elephant Dourado 1234 -6578
Douphin Onca 1234-7685
Bison Tamandua 1234-6785
Rabbit Tuiuiu 1234-5867
Cheeta Pintado 1234-7856
Flamingo Garca 1234-8567
Shark Pintado 1234-7865
Wolf Jacare 1234-7568
Rino Lobo 1234-7586
```

```
# formato da saída
Arara Aligator 1234-5678
Bigua Bear 1234-6789
Caiman Puma 1234-8765
Capivara Tiger 1234-5687
Dourado Elephant 1234 -6578
Dourado Parrot 1234-6587
Dourado Python 1234-8675
Garca Flamingo 1234-8567
Jaboti Tortuga 1234-8576
Jacare Wolf 1234-7568
Lobo Rino 1234-7586
Morcego Skank 1234-7658
Onca Douphin 1234-7685
Pintado Cheeta 1234-7856
Pintado Shark 1234-7865
Quati Zebra 1234-8756
Raposa Eagle 1234-8657
Sucuri Gnu 1234-6798
Sucuri Hippopotamus 1234-6758
Tamandua Bison 1234-6785
Tucano Codfish 1234-5678
Tucano Lizard 1234-5768
Tuiuiu Owl 1234-5786
Tuiuiu Rabbit 1234-5867
Urutu Condor 1234-5876
```

O programa abaixo ilustra a montagem da lista telefônica usando ordenação por bolha, ordenando pelo sobrenome e modificando as listas associadas.

```
1  # -*- coding: utf-8 -*-
2  '''
3  # Programa: listatelefonica.py
4  # Programador:
5  # Data: 25/05/2020
6  # Este programa lê um conjunto de tam nomes e telefones de uma
7  # empresa telefônica. Os nomes são dados pelo nome, sobrenome
8  # e número de telefone. O programa elabora uma lista telefônica
9  # ordenando o conjunto pelo sobrenome. Os dados são armazenados
10 # em uma lista, onde cada elemento é uma lista, com três strings,
11 # nomes, sobrenomes e numeros.
12 # início do módulo principal.
13 # descrição das variáveis utilizadas
14 # list    catalogo
15 # int     tam
16 '''
17
18 # pré: tam catalogo[0] catalogo[1] .. catalogo[tam-1]
19
```

```
# Passo 1. Leia a entrada e inicialize as variáveis
# Passo 1.1. Leia o tamanho do catálogo
tam = int(input())
# Passo 1.2. Inicialize as listas
catalogo = [['']*3 for i in range(tam)]
# Passo 1.3. Leia os dados dos assinantes
for i in range(tam):
   catalogo[i] = list(map(str,input().split()))
# Passo 2. Ordene pelo sobrenome dos assinantes
# Passo 2.1. Assuma que houve troca
trocou = True
# Passo 2.2. Compare os termos dois a dois, trocando quando necessário
while (trocou):
# Passo 2.2.1. No início da iteração não houve troca
   trocou = False
   for i in range(0,tam-1):
      if catalogo[i][1] > catalogo[i+1][1]:
         catalogo[i],catalogo[i+1]=catalogo[i+1],catalogo[i]
         trocou = True
# Passo 3. Imprima o catálogo
for i in range(0,tam):
   print('{0:s} {1:s} {2:s}'.format(catalogo[i][1],catalogo[i][0],
                                                        catalogo[i][2]))

# pós: para i em {0..tam-1}:catalogo[i] and
#      catalogo[i][1] <= catalogo[i+1][1]
# fim do módulo principal
```

1.3 Média de Temperaturas

Matrizes $m \times n$ surgem naturalmente na modelagem de uma uma grande variedade de aplicações. O conjunto das matrizes $m \times n$, junto com as operações de adição, subtração e multiplicação tem propriedades matemáticas importantes. Podemos usar listas de listas para representar essas matrizes. Uma matriz $m \times n$ também pode ser vista como um arranjo bidimensional. Posteriormente, usaremos a biblioteca `numpy` para efetuar operações com arranjos bidimensionais.

Para ilustrar, considere a matriz de zeros $A_{3\times4}$:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Podemos representá-la com uma lista de listas, onde cada elemento da lista também é uma lista, ou seja cada elemento da lista pode ser interpretado como uma linha da matriz. Essa matriz pode ser representada pela lista `L` definida como `L = [[0]*4 for i in range(3)]`:

```
L = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

Vamos usar uma matriz para modelar o problema de computar a média das temperaturas em vários horários na cidade de Cachorro Sentado, onde o Serviço de Meteorologia Capivara

registra a temperatura em três locais distintos e quatro vezes ao dia. Suponha que os dados foram coletados e registrados numa tabela com 4 linhas e 3 colunas, onde em cada linha estão registradas as temperaturas observadas nos três locais, e as temperaturas foram registradas 4 vezes ao dia, às 6:00, 12:00, 18:00 e 0:00 horas. Queremos projetar um programa para computar as temperaturas médias em cada um dos horários que a temperatura foi medida.

	Reserva	Centro	Bairro
06:00	25.6	26.2	24.7
12:00	27.3	27.9	26.6
18:00	26.4	27.2	25.9
00:00	25.6	26.4	24.8

A tabela pode ser vista como uma matriz, e para representar esses dados num computador, ele usou uma lista `tabtemp`, onde cada um dos 4 elementos da lista é também uma lista, uma para cada horário, contendo cada uma 3 elementos, representando os três locais onde a temperatura foi registrada.

```
tabtemp = [[25.6, 26.2, 24.7], [27.3, 27.9, 26.6], [26.4, 27.2, 25.9],
           [25.6, 26.4, 24.8]]
```

Para implementação dessa solução temos que descrever mais algumas funcionalidades das listas no Python. Antes de mais nada temos que saber como os dados da entrada serão fornecidos. Nesse exemplo vamos supor que a entrada é dada por 4 linhas, cada um com três temperaturas.

```
# formato da entrada
25.6 26.2 24.7
27.3 27.9 26.6
26.4 27.2 25.9
25.6 26.4 24.8
```

Para armazenar esses dados utilizaremos a lista `tabtemp`. Para a leitura dos dados da entrada, temos que definir a forma de atribuição desses dados para a lista `tabtemp`. Vamos assumir que queremos possibilitar a atribuição direta à lista. Em função disso, temos que inicializar a lista com o espaço necessário para armazenar as temperaturas. Lembramos que `tabtemp` é uma lista com 4 listas, `tabtemp[0]`, `tabtemp[1]`, `tabtem[2]` e `tabtemp[3]`, onde cada uma delas possui 3 elementos `tabtemp[i][0]`, `tabtemp[i][1]` e `tabtem[i][2]`, para $0 \leq \texttt{i} \leq 3$.

```
tabtemp = [[0.0]*3 for i in range(4)]
```

nessa definição, a atribuição `[0.0]*3`

```
lista = [0.0]*3
print(lista)
[0.0, 0.0, 0.0]
```

cria e inicializa com 0.0 uma lista de 3 elementos. Se repetimos essa atribuição por 4 vezes, temos uma lista de 4 listas, onde cada lista tem três elementos 0.0.

```
tabtemp = [[0.0]*3 for _ in range(4)]
print(tabtemp)
[[0.0, 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 0.0]]
```

onde o _ funciona como uma variável invisível.

Vimos anteriormente que inicializando as listas maneira, podemos atribuir (modificar) valores individuais a cada um dos elementos da lista. Vimos também que é possível ler e atribuir um conjunto de dados diretamente a uma lista.

```
lista = list(map(float, input().split())) # lê uma linha
```

para o exemplo abaixo:

```
# formato da entrada
25.6 26.2 24.7

# formato da saída
[25.6, 26.2, 24.7]
```

No nosso caso, temos que ler quatro linhas. Lembrando que `tabtemp[i]` é uma lista, podemos fazer as atribuições do seguinte modo:

```
tabtemp[0] = list(map(float, input().split())) # lê a linha 0
tabtemp[1] = list(map(float, input().split())) # lê a linha 1
tabtemp[2] = list(map(float, input().split())) # lê a linha 2
tabtemp[3] = list(map(float, input().split())) # lê a linha 3
```

Usando essas ideias, vamos projetar e implementar um programa que leia as 3 temperaturas nos respectivos locais, 4 vezes ao dia (seis horas, meio dia, dezoito horas e meia noite), calcule a temperatura média em cada um dos horários em que a temperatura foi registrada, e imprima os resultados. Utilize uma lista `media` para armazenar a temperatura média em cada uma dos horários em que ela foi coletada. Os valores das temperaturas são medidos com pelo menos uma casa decimal.

Abaixo uma sugestão de estrutura de dados em Python para resolver o problema.

```
# descrição das variáveis utilizadas
# list  tabelatemp - lista das listas das temperaturas
# list  media - lista com as medias
# float soma
```

A entrada é dada pelo arquivo `temperatura1.in`. O arquivo é composto por quatro linhas, cada uma com três números de ponto flutuante (`float`), representando a tabela com as temperaturas. Cada bloco de linhas representa as temperaturas da cidade em um dado dia e cada linha do bloco representa as temperaturas de três locais distintos da cidade onde as medidas foram efetuadas.

Para computar as médias das temperaturas em cada um dos horários, basta somarmos as temperaturas medidas nos três locais e calcular a média. No caso da tabela, temos que somar cada uma das linhas de `tabtemp` (locais em que a temperatura foi registrada em cada um dos horários) e dividir pelo número de locais (3). Por exemplo, a média da temperatura às 12:00 horas fica:

```
soma = 27.3 + 27.9 + 26.6
media = soma/3.0
```

Inicialmente temos que inicializar e ler a entrada. Considerando que a entrada é dada por:

```
# formato da entrada
25.6 26.2 24.7
27.3 27.9 26.6
26.4 27.2 25.9
25.6 26.4 24.8
```

isso pode ser feito da seguinte forma:

```
# Passo 1. Leia a entrada
# Passo 1.1. Inicialize as estruturas de dados
media = [0.0]*4 # média de 4 linhas
# Passo 1.2. Leia a tabela de temperaturas (linha a linha)
for i in range(0,4):
   tabtemp[i] = list(map(float, input().split())) # leia a linha i
```

Para calcular a média, devemos somar cada linha da tabela e dividir pelo número de locais (3). Como a tabela está armazenada numa lista de listas, temos que somar os elementos e calcular a média:

```
soma = tabtemp[0][0]+tabtemp[0][1]+tabtemp[0][2]
media[0] = soma/3.0 # média da temperatura no horário 0 (6:00)
soma = tabtemp[1][0]+tabtemp[1][1]+tabtemp[1][2]
media[1] = soma/3.0 # média da temperatura no horário 1 (12:00)
soma = tabtemp[2][0]+tabtemp[2][1]+tabtemp[2][2]
media[2] = soma/3.0 # média da temperatura no horário 2 (18:00)
soma = tabtemp[3][0]+tabtemp[3][1]+tabtemp[3][2]
media[3] = soma/3.0 # média da temperatura no horário 3 (0:00)
```

podemos reescrever as linhas acima como:

```
# Passo 2. Calcule as médias para cada horário
for i in range(0,4): # fixa a linha
   soma = 0.0
   for j in range(0,3): # varia a coluna
      soma = soma + tabtemp[i][j] # soma dos elementos da linha
   media[i] = soma/3.0 # media da linha i
```

Queremos que a impressão da lista resultante seja feita com apenas uma casa decimal. Nesse caso, ou usamos a função `print` com o `format` para imprimir cada um dos elementos, ou fazemos o ajuste diretamente na lista e depois imprimimos a lista. Isso pode ser feito da seguinte maneira:

```
# Passo 3. Imprima a lista da temperatura média em cada horário
saida = [round(item,1) for item in media] # ajusta para uma casa decimal
print(saida)
```

A implementação final do programa fica:

```
# -*- coding: utf-8 -*-
# Programa: temperatura01.py
# Programador:
# Data: 21/11/2015
# Este programa lê 4 medidas diárias de temperatura em três locais
# distintos, computa a média da temperatura para cada um dos
# horários de medição e imprime o resultado
# início do módulo principal
# descrição das variáveis utilizadas
# list  tabelatemp - lista das listas das temperaturas
# list  media - lista com as medias
# float soma

# pré: tabtemp[0] == tabtemp[0][0]..tabtemp[0][2]
#      tabtemp[1] == tabtemp[1][0]..tabtemp[1][2]
#      tabtemp[2] == tabtemp[2][0]..tabtemp[2][2]
#      tabtemp[3] == tabtemp[3][0]..tabtemp[3][2]

# Passo 1. Leia a entrada e crie as estruturas de dados
# Passo 1.1. Leia a tabela de temperaturas
for i in range(0,4):
   tabtemp[i] = list(map(float,input().split())) # leia a linha i
# Passo 1.2. Inicialize a lista media
media = [0.0]*4 # média de 4 linhas
# Passo 2. Compute a média das temperaturas para cada horário
for i in range(0,4): # fixa a linha
   soma = 0.0
   for j in range(0,3): # varia a coluna
      soma = soma + tabtemp[i][j] # soma dos elementos da linha
   media[i] = soma/3.0 # media da linha i
# Passo 3. Imprima a lista da temperatura média em cada horário
# Passo 3.1. Formate todos elementos da lista com uma casa decimal
saida = [round(item,1) for item in media] # ajusta para uma casa decimal
# Passo 3.2. Imprima a saída
print(saida)

# pós: para i em {0,...,3}:media[i] and media[i] == (soma)
#      j em {0,...,2}:tabtemp[i][j])/3.0
# fim do módulo principal
```

Você pode criar dois arquivos para testar o seu programa, uma para a matriz da entrada e outro para a lista das médias da saída. No nosso caso, vamos denominar o arquivo de entrada como `temperatura01.in` e o arquivo de saída como `temperatura01.out`. O programa pode ser executado com:

```
$ python temperatura01.py < temperatura01.in > temperatura01.out
```

Exemplo

```
# formato da entrada
25.6 26.2 24.7
27.3 27.9 26.6
26.4 27.2 25.9
25.6 26.4 24.8

# formato da saída
[25.5, 27.4, 28.7, 27.8]
```