Tecnologias da Informação

Alexandre Xavier Falcão

Instituto de Computação - UNICAMP

afalcao@ic.unicamp.br

Introdução à programação em Python

Este módulo aborda os seguintes conceitos básicos em Python.

- Variáveis simples e conversões entre tipos de variáveis,
- entrada (via teclado) e saída (no console) de dados,
- operações matemáticas e precedência entre operadores,
- formatação de cadeias de caracteres,
- comandos condicionais e repetitivos,
- funções, tuplas, listas, e dicionários.

Variáveis e tipos

Voltando a analogia da memória do computador como um armário de gavetas, cada gaveta é uma **variável** (objeto em Python), com uma localização, um nome, e um **tipo** de dado armazenado. Exemplos são:

- **int** - inteiros: $\ldots -3, -2, -1, 0, 1, 2, 3, \ldots$.
- **float** - reais: e.g., $-3.2, -2.0, 1e5, 2.5e7$, etc.
- **bool** - boleano: True (verdadeiro) e False (falso).
- **complex** - complexos: e.g., $-2.0 + 3.4j$.
- **str** - *string* (cadeias de caracteres entre aspas dupla ou simples): e.g., 'A', "B", '23', '3.5', "MC001", "Eu quero programar", "#$%&?;:", etc.

Entrada e saída de dados

- Os comandos (instruções) **input** e **print** são usados para a entrada de dados via teclado e a saída de dados no console.

- O **input** aceita uma cadeia de caracteres como argumento, para informar ao usuário o que deve ser digitado, e retorna uma cadeia de caracteres, que deve ser convertida para o tipo de dado desejado.

- O **print** aceita qualquer tipo de dado como argumento e mais de um argumento, separados por vírgula.

Para entrar um valor inteiro e armazená-lo na variável A.

```
A = input("Entre com o valor de A: ")
print(A, type(A))
A = int(A) # Conversao para tipo inteiro
print(A, type(A))
```

Entrada e saída de dados

Executando o código e entrando com o valor 20, veremos no console:

Entre com o valor de A: 20

20 ⟨ class 'str'⟩

20 ⟨ class 'int'⟩

Outro exemplo de conversões entre tipos é

```
A = 20          # valor inteiro
B = float(A)    # converte para real
C = str(A)      # converte para cadeia de caracter
D = bool(A)     # converte para True (so 0 eh False)
E = complex("2-3j")
print(A,type(A),B,type(B),C,type(C),\
      D,type(D),E,type(E))
```

com saída 20 ⟨class 'int'⟩ 20.0 ⟨class 'float'⟩ 20 ⟨class 'str'⟩ True ⟨class 'bool'⟩ 2-3j ⟨class 'complex'⟩

Operações matemáticas e precedência entre operadores

As operações aritméticas possuem a seguinte ordem de precedência:

1. potência **,
2. multiplicação *, divisão real /, divisão inteira //,
3. resto da divisão inteira %, e
4. soma + e subtração -, na ordem que aparecem na expressão.

Verifique, por exemplo, a saída para o código abaixo.

```
print(8 / 2 + 5 * 3)
print(8 / 2 % 3)
print(5 * 10 + 4 - 9 / 3)
print(20 // 3 + 4 ** 2 - 5)
```

Operações matemáticas e precedência entre operadores

O uso de parênteses na expressão aritmética força a precedência, fazendo com que a expressão mais interna seja executada antes das demais. Verifique, por exemplo, a saída para o código abaixo.

```
print(2 ** (3 + 1) + (2 - 3j))
print(2 * ((3 + 4) - 1))
print((20 // 3) ** (2 + 2))
print((20 / 3) ** (2 + 2))
A = 2 ** 3 + 3j
B = 4 * 3 - 4j
C = 20 - 5 + 9
D = (A + B)*C
print(D)
```

Operações lógicas

Operações lógicas são realizadas sobre valores boleanos (True e False). Comparações entre variáveis numéricas e/ou expressões aritméticas resultam valores boleanos. A precedência é da operação aritmética e uma expressão lógica pode conter várias operações.

```
A = True
B = False
C = A or B
D = A and B
E = not A
print("A=",A,", B=",B,", C=",C,", D=",D,", E=",E)
A = (4 < 2)
B = (3 <= 5) and (2 > 20)
C = (3 - 1 <= 2) and (4 == 8 / 2 ) or (2 < 0)
D = A or B and (2 != 2)
E = not (A and B)
print("A=",A,", B=",B,", C=",C,", D=",D,", E=",E)
```

Formatação de cadeias de caracteres

Cadeias de caracteres podem ser formatadas e concatenadas.

```
nome   = "Maria"
frase = "Oi {}".format(nome)
print(frase)
idade   = 10
mesada = 53.6677
frase = "{} tem {} anos e ganha {:.2f} reais". \
        format(nome, idade, mesada)
print(frase)
frase = "{2} tem {1} anos e ganha {0:.2f} reais". \
        format(mesada, idade, nome)
print(frase)
```

Formatação de cadeias de caracteres

Podemos entrar com múltiplos números separados por um caracter, separar a cadeia resultante, e converter as subcadeias em números.

```
A, B, C = input("Entre com A, B, e C \
            separados por ; ").split(";")
A = int(A)
B = int(B)
C = float(C)
print("{:03d} {:05d} {:.2f}".format(A,B,C), \
        end=" ") # o end=" " troca o caracter
    # "\n" de pular linha
print("123",end=" ")
print("456",end=" ")
print("789")
```

Comando condicional

A formatação mais geral de um comando condicional (de desvio de fluxo) pode conter várias expressões lógicas com um bloco de código para cada condição até encontrar um *else*.

```
if (expressão lógica 1):
    bloco 1 de código
elif (expressão lógica 2):
    bloco 2 de código
elif (expressão lógica 3):
    bloco 3 de código
else:
    último bloco de código
```

onde as instruções em cada bloco (exceto o último) são executadas quando a expressão lógica é verdadeira.

Comando condicional

```
a, b, o = input('Entre com a, b, e o: ').split(',')
a = float(a); b = float(b)
if (o == '+'):
    print('a+b= {:.2f}'.format(a+b))
elif (o == '-'):
    print('a-b= {:.2f}'.format(a-b))
elif (o == '*'):
    print('a*b= {:.2f}'.format(a*b))
elif (o == '/'):
    if (b != 0):
        print('a/b= {:.2f}'.format(a/b))
    else:
        print('Divisao invalida')
elif (o == '**'):
    print('a**b= {:.2f}'.format(a**b))
else:
    print('Operacao invalida')
```

Comando condicional

Lembrando o problema de determinar se um dado triângulo é retângulo.

```
A, B, C = input("Entre com A, B, e C: ").split(";")
A = float(A); B = float(B); C = float(C)
# descubra o maior lado guardando os outros dois
maior  = A; menor1 = B; menor2 = C
if (B > maior):
    maior  = B; menor1 = A; menor2 = C
if (C > maior):
    maior  = C; menor1 = B; menor2 = A
print("maior ",maior,"menor1 ",\
      menor1,"menor2 ",menor2)
if (maior*maior == menor1*menor1 + menor2*menor2):
    print("O triangulo eh retangulo")
else:
    print("O triangulo nao eh retangulo")
```

Comando condicional

Outro exemplo é a determinação das raízes de uma equação de segundo grau,

$$y = ax^2 + bx + c,$$

onde a, b, e c são números reais. As raízes são os valores de x que fazem com que $y = 0$. Elas são obtidas por

$$\begin{aligned} x_1 &= \frac{-b + \sqrt{\Delta}}{2a}, \\ x_2 &= \frac{-b - \sqrt{\Delta}}{2a}, \\ \Delta &= b^2 - 4ac. \end{aligned}$$

No entanto, $\Delta < 0$ gera raízes complexas, $a = 0$ degenera a equação em uma reta $y = bx + c$, com raiz única $x_1 = -c/b$, e se a e b forem zero, não existe raiz para $y = c$.

Comando condicional

Essas situações deveriam ser tratadas, conforme exemplo abaixo.

```
a,b,c = input("Entre com a,b,c: ").split(',')
a = float(a); b = float(b); c = float(c)
if (a == 0):
    if (b == 0):
        print('Nao existe raiz')
    else:
        print('raiz unica: ',-c/b)
else:
    delta = b**2 - 4*a*c
    x1    = (-b + delta**(1/2))/(2*a)
    x2    = (-b - delta**(1/2))/(2*a)
    if (type(x1) is complex):
        print('x1= ({:.2f},{:.2f}j),\
              x2 = ({:.2f},{:.2f}j)'. \
                  format(x1.real,x1.imag,x2.real,x2.imag))
    else:
        print('x1= {:.2f}, x2 = {:.2f}'.format(x1,x2))
```

Exercícios com o comando condicional

1. Imprima em ordem crescente três números dados.
2. Para dois números inteiros de $[1, 6]$, imprima "vencedor" se a soma dos números for 7 ou 11, "perdedor" se for 2, 3, ou 12, ou simplesmente imprima a soma dos números.
3. Para dois números inteiros, d_1 e d_2, de $[1, 6]$, imprima "vencedor" se $d_1 = d_2$ e a soma dos números for 4, 6, 8, ou 10 , "perdedor" se $d_1 \neq d_2$ e a soma dos números for 4, 6, 8, ou 10, e ' tente novamente' em qualquer outro caso.
4. Repita o exercício anterior verificando se os números dados estão no intervalo de $[1, 6]$. Caso não estejam, imprima uma mensagem.
5. Para quatro números dados, x_1, x_2, x_3 e x_4, imprima "vencedor" se $x_1 > x_2$ e $x_3 > x_4$ ou $x_1 > x_2$ e $x_2 > x_4$, "perdedor" se $x_4 > x_2 > x_1$, e "tente novamente" em qualquer outro caso.

Comandos repetitivos

- Vamos focar inicialmente no comando *while*, o qual repete um dado bloco de código enquanto uma dada expressão lógica for verdadeira.

- Sua construção requer pelo menos uma variável de controle da expressão lógica, a qual deve ser inicializada antes do comando e atualizada no seu bloco de código.

```
while expressão lógica:
    bloco de código
```

Comandos repetitivos

Lembrando o problema de calcular o valor médio de uma sequência com n números $x_1, x_2, \ldots, x_n$ dados.

```
n = int(input('Entre com o tamanho da sequencia: '))
i = 1 # inicializando a variavel de controle
media = 0.0
while (i <= n):
    x = float(input('Entre com x{}: '.format(i)))
    media = media + x
    i = i + 1 # atualizando a variavel de controle
if (n > 0):
    media = media / n
    print('A media eh {:.2f}'.format(media))
```

Note que o código pode ser facilmente modificado para que a média considere apenas os valores pares.

Comandos repetitivos

A variável de controle pode ser inclusive a que recebe os valores da sequência. Por exemplo, calcule a média enquanto esses valores são positivos.

```
x = float(input('Entre com x1: '))

if (x > 0):
    media = 0.0
    n = 1
    while (x > 0):
        media = media + x
        n     = n + 1
        x = float(input('Entre com x{}: '.format(n)))

    media = media / (n-1)
    print('A media eh {:.2f}'.format(media))
```

Exercícios

1. Dada uma cadeia de caracteres s, os elementos em s podem ser acessados um a um como $s[i]$, $i = 0, 1, \ldots, len(s) - 1$. Por exemplo, para s="Falcao", s[2] é 'l'. Leia uma cadeia de caracteres e imprima seus caracteres separados por '-'.

2. Leia uma cadeia de caracteres s e imprima seus caracteres de traz para a frente.

3. Palíndromos são palavras que podem ser lidas de traz para a frente. Leia uma cadeia de caracteres e verifique se ela é palíndromo. Note que $s_1 = ''$ define uma cadeia vazia e $s_1 = s_1 + s_2$ concatena as cadeias s_1 e s_2 colocando o resultado em s_1. Use essas informações para resolver o problema.

Exercícios

1. Dada uma sequência de bits como uma cadeia de caracteres, converta-a para o número inteiro correspondente. Assuma que não estamos trabalhando com o bit de sinal. Ou seja, o número decimal é $b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_0 2^0$ para uma sequência $b_{n-1}b_{n-2}\ldots b_0$ com n bits.

2. Implemente o conversor inverso de inteiro positivo para a sequência de bits correspondente.

3. Repita ambos exercícios com o bit de sinal mais a esquerda e a representação em complemento de 2.

Laços encaixados

- Vários problemas requerem um comando repetitivo dentro de outro e isso pode se estender a vários laços encaixados.

Laços encaixados

- Vários problemas requerem um comando repetitivo dentro de outro e isso pode se estender a vários laços encaixados.

- Por exemplo, considere o problema de imprimir todos os pares de números de 0 a 99 cuja soma dos números é 150.

Laços encaixados

- Vários problemas requerem um comando repetitivo dentro de outro e isso pode se estender a vários laços encaixados.

- Por exemplo, considere o problema de imprimir todos os pares de números de 0 a 99 cuja soma dos números é 150.

```
n1 = 0;
while (n1 < 100):
    n2 = 0
    while (n2 < 100):
        if (n1 + n2 == 150):
            print(n1,n2)
        n2 = n2 + 1
    n1 = n1 + 1
```

Laços encaixados

- Vários problemas requerem um comando repetitivo dentro de outro e isso pode se estender a vários laços encaixados.

- Por exemplo, considere o problema de imprimir todos os pares de números de 0 a 99 cuja soma dos números é 150.

```
n1 = 0;
while (n1 < 100):
    n2 = 0
    while (n2 < 100):
        if (n1 + n2 == 150):
            print(n1,n2)
        n2 = n2 + 1
    n1 = n1 + 1
```

Note que existem soluções repetidas. Como modificar o algoritmo para evitar essas repetições?

Laços encaixados

Outro exemplo similar seria encontrar as três dimensões possíveis para caixas com volume igual a 100000 cm^3, tais que nenhuma dimensão seja menor do que $20cm$.

Laços encaixados

Outro exemplo similar seria encontrar as três dimensões possíveis para caixas com volume igual a 100000 cm^3, tais que nenhuma dimensão seja menor do que $20cm$.

```
d1 = 20;
while (d1 <= 50):
    d2 = 20
    while (d2 <= 50):
        d3 = 20
        while (d3 <= 50):
            if (d1*d2*d3 == 100000):
                print(d1,d2,d3)
            d3 = d3 + 1
        d2 = d2 + 1
    d1 = d1 + 1
```

Laços encaixados

Podemos ainda imaginar que cada laço corresponde a um eixo em um sistema de coordenadas. Por exemplo, em um sistema de coordenadas (x, y), podemos imprimir todas as coordenadas de $(0, 0)$ a $(1, 1)$ com deslocamentos $(0.2, 0.2)$.

Laços encaixados

Podemos ainda imaginar que cada laço corresponde a um eixo em um sistema de coordenadas. Por exemplo, em um sistema de coordenadas (x, y), podemos imprimir todas as coordenadas de $(0, 0)$ a $(1, 1)$ com deslocamentos $(0.2, 0.2)$.

```
y = 0;
while (y <= 1.0):
    x = 0
    while (x <= 1.0):
        print("({:.1f},{:.1f})".format(x,y), end=' ')
        x = x + 0.2
    print()
    y = y + 0.2
```

Laços encaixados

Podemos ainda imaginar que cada laço corresponde a um eixo em um sistema de coordenadas. Por exemplo, em um sistema de coordenadas (x, y), podemos imprimir todas as coordenadas de $(0, 0)$ a $(1, 1)$ com deslocamentos $(0.2, 0.2)$.

```
y = 0;
while (y <= 1.0):
    x = 0
    while (x <= 1.0):
        print("({:.1f},{:.1f})".format(x,y), end=' ')
        x = x + 0.2
    print()
    y = y + 0.2
```

Para uma função $f(x, y) = x^2 + xy - y^2$, modifique o algoritmo para imprimir o valor desta função nos pontos amostrados.

Laços encaixados

Podemos ainda imaginar que cada laço corresponde a um eixo em um sistema de coordenadas. Por exemplo, em um sistema de coordenadas (x, y), podemos imprimir todas as coordenadas de $(0, 0)$ a $(1, 1)$ com deslocamentos $(0.2, 0.2)$.

```
y = 0;
while (y <= 1.0):
    x = 0
    while (x <= 1.0):
        print("({:.1f},{:.1f})".format(x,y), end=' ')
        x = x + 0.2
    print()
    y = y + 0.2
```

Para uma função $f(x, y) = x^2 + xy - y^2$, modifique o algoritmo para imprimir o valor desta função nos pontos amostrados. Depois modifique o algoritmo para encontrar o valor mínimo de $f(x, y)$ no intervalo dado com deslocamentos de $(0.01, 0.01)$.

Listas

Em muitas situações precisamos armazenar uma sequência de itens em uma lista.

Listas

Em muitas situações precisamos armazenar uma sequência de itens em uma lista.

```
frutas   = ["banana", "abacate", "laranja"]
numeros  = [1, 20, 25, 87, 0.56, 3.78]
vazia    = []
mista    = ["banana", 70, 9.98, "vermelho"]
vetor    = [1.0, 0.5, 3.0]
matriz   = [[1, 3, 4], [0, 2, -8], [1, 2, 5]]
print(frutas[1],len(frutas),numeros[2],len(vazia),\
      mista[-2],matriz[1][2])
```

Um exemplo típico é o cômputo da variância σ^2 de uma sequência de números dados $x_1, x_2, \ldots, x_n$.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Listas

Podemos armazenar os números em uma lista e depois calcular a média e a variância da sequência dada.

```
n = int(input('Entre com n: '))

# cria a lista

i = 1;   L = []
while (i <= n):
    x = float(input('Entre com x{:d}: '.format(i)))
    L.append(x) # insere x in L
    i = i + 1
```

Listas

```
# calcula a media
media = 0.0; i = 0
while (i < n):
    media = media + L[i]; i = i + 1
media = media / n

# calcula a variancia
variancia = 0.0; i = 0
while (i < n):
    variancia = variancia + (L[i]-media)**2; i = i + 1
variancia = variancia / n
print("{:.2f}, {:.2f}".format(media, variancia))
```

Listas

```
# calcula a media
media = 0.0; i = 0
while (i < n):
    media = media + L[i]; i = i + 1
media = media / n

# calcula a variancia
variancia = 0.0; i = 0
while (i < n):
    variancia = variancia + (L[i]-media)**2; i = i + 1
variancia = variancia / n
print("{:.2f}, {:.2f}".format(media, variancia))
```

A lista no exemplo acima foi gerada elemento por elemento, mas também podemos gerar uma lista das seguintes formas.

Listas

```
L1 = input("Entre com a lista 1: ").split(';')
L2 = list(range(20,101,10)) # inicio ,fim+1, incremento
L3 = list("Abacate")
L4 = "3;4;9".split(';')
print(L1)
print(L2)
print(L3)
print(L4)
# saida
# ['30', 'abacate', 'verde']
# [20, 30, 40, 50, 60, 70, 80, 90, 100]
# ['A', 'b', 'a', 'c', 'a', 't', 'e']
# ['3', '4', '9']
```

Listas

Uma lista de números de entrada deve ser convertida para números reais se formos fazer contas com esses números.

```
L = input("Entre com os numeros: ").split(';')
i = 0
while (i < len(L)):
    L[i] = float(L[i])
    i = i + 1
print(L)

# Outra alternativa
# L = list(map(float,L))
# print(L)
```

Listas e Cadeias

As listas aceitam algumas formas de manipulação comuns com as cadeias. Por exemplo:

```
L1 = [2,3,4]
L2 = [3,2,1]
L3 = L1 + L2
print(L3)
print(L3[2:5:1])
if (2 in L2):
    print("sim")

# saida
# [2, 3, 4, 3, 2, 1]
# [4, 3, 2]
# sim
```

Exercícios

1. Sabendo que o produto interno entre dois vetores $\vec{v} = v_x\vec{i} + v_y\vec{j} + v_z\vec{k}$ e $\vec{u} = u_x\vec{i} + u_y\vec{j} + u_z\vec{k}$ é $v_x u_x + v_y u_y + v_z u_z$, faça um programa que lê e armazena os vetores em duas listas, e depois percorre as listas para calcular o produto interno entre eles.

2. Dada uma sequência de números de 0 a 9, gere uma lista com o número de ocorrências de cada número da sequência. Dica: inicialize a lista de ocorrências com zeros e 10 posições.

3. Dadas duas sequências crescentes de números, gere uma lista com os números intercalados sem quebrar a ordem crescente entre eles. Dica: percorra as duas sequências simultaneamente, copiando os números em ordem para a lista até o final da mais curta. Depois copie o resto dos números da mais longa.

Objetos que possibilitam iteração

O comando for permite iterar sobre alguns tipos de objetos (e.g., listas e cadeias), substituindo o comando while.

```
L = [20, "abacate", "azul"]
for item in L:
    print(item, end=' ')
# saida
# 20 abacate azul
print()
for ind in range(2,-1,-1): # inicio, fim-1, incremento
    print(ind, end=' ')
print()
# saida
# 2 1 0
for ind, item in enumerate(L):
    print(ind,item)
# saida
# 0 20
# 1 abacate
# 2 azul
```

Mais sobre o comando for

O comando for pode então ser utilizado para construir laços encaixados.

```
M = []; i = 0;
for lin in range(3): # gera matriz 3 x 4 de 0 a 11
    M.append([])
    for col in range(4):
        M[lin].append(i)
        i = i + 1
print(M) # imprime a lista de listas
# saida
# [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]
for lin in range(3): # imprime a matriz 3 x 4
    for col in range(4):
        print("{:02d}".format(M[lin][col]), end=' ')
    print()
# saida
# 00 01 02 03
# 04 05 06 07
# 08 09 10 11
```

Arquivos em Python

- Os dados que informamos via teclado podem ser armazenados em arquivos usando um padrão texto ou binário. Vamos estudar o primeiro caso apenas.

Arquivos em Python

- Os dados que informamos via teclado podem ser armazenados em arquivos usando um padrão texto ou binário. Vamos estudar o primeiro caso apenas.

- O código abaixo abre um arquivo de notas de alunos para leitura, lê e imprime as suas linhas, e depois fecha o arquivo.

```
arq_notas = open("./notas.txt", "r") # abre arquivo
for linha in arq_notas:
    print(linha) # Imprime linha
arq_notas.close() # fecha arquivo
# saida
# Paulo 8.0 5.0 4.0

# Maria 7.0 9.0 6.0

# Tereza 9.0 10.0 8.5
```

Arquivos em Python

- Os dados que informamos via teclado podem ser armazenados em arquivos usando um padrão texto ou binário. Vamos estudar o primeiro caso apenas.

- O código abaixo abre um arquivo de notas de alunos para leitura, lê e imprime as suas linhas, e depois fecha o arquivo.

```
arq_notas = open("./notas.txt", "r") # abre arquivo
for linha in arq_notas:
    print(linha) # Imprime linha
arq_notas.close() # fecha arquivo
# saida
# Paulo 8.0 5.0 4.0

# Maria 7.0 9.0 6.0

# Tereza 9.0 10.0 8.5
```

- Cada linha tem um caracter de pular linha, \n, escondido.

Arquivos em Python

O rstrip() elimina espaços em branco, caracteres especiais, e quaisquer outros caracteres especificados ao final da cadeia.

```
cadeia = "Dialogo: Oi     ,.???ÃSÃS"
cadeia = cadeia.rstrip(" ,.?ÃS")
print(cadeia)
# saida
# Dialogo:  Oi
```

Arquivos em Python

O rstrip() elimina espaços em branco, caracteres especiais, e quaisquer outros caracteres especificados ao final da cadeia.

```
cadeia = "Dialogo: Oi     ,.???ÃSÃS"
cadeia = cadeia.rstrip(" ,.?ÃS")
print(cadeia)
# saida
# Dialogo: Oi
```

O default é eliminar espaços em branco e o \n. Se o arquivo não existir, uma mensagem de erro também será gerada. Então a forma correta de implementar o exemplo anterior é dada a seguir.

Arquivos em Python

```
try:
    arq_notas = open("./notas.txt", "r")
    for linha in arq_notas:
        print(linha.rstrip())
    arq_notas.close()
except:
    print("Arquivo nao encontrado")
# saida
# Paulo 8.0 5.0 4.0
# Maria 7.0 9.0 6.0
# Tereza 9.0 10.0 8.5
```

Arquivos em Python

```
try:
    arq_notas = open("./notas.txt", "r")
    for linha in arq_notas:
        print(linha.rstrip())
    arq_notas.close()
except:
    print("Arquivo nao encontrado")
# saida
# Paulo 8.0 5.0 4.0
# Maria 7.0 9.0 6.0
# Tereza 9.0 10.0 8.5
```

Para manipular o conteúdo do arquivo, podemos ainda criar uma lista de listas com o conteúdo de cada linha.

Arquivos em Python

```
try:
    arq_notas = open("./notas.txt", "r")
    Alunos = []
    for linha in arq_notas:
        Alunos.append(linha.rstrip().split())
    arq_notas.close()
    print(Alunos) # uma lista de listas
    for i in range(len(Alunos)):
        for item in Alunos[i]:
            print(item, end=' ')
        print()
except:
    print("Arquivo nao encontrado")
# ultima saida
# Paulo 8.0 5.0 4.0
# Maria 7.0 9.0 6.0
# Tereza 9.0 10.0 8.5
```

Arquivos em Python

Vamos então calcular a média de cada aluno e gravar um novo arquivo que inclui essas médias.

```
try:
    arq_notas = open("./notas.txt", "r")
    Alunos = []
    for linha in arq_notas:
        Alunos.append(linha.rstrip().split())
    arq_notas.close()
    arq_medias = open("./medias.txt", "w")
```

Arquivos em Python

```
        for i in range(len(Alunos)):
            nome  = Alunos[i][0];
            nota1 = float(Alunos[i][1])
            nota2 = float(Alunos[i][2]);
            nota3 = float(Alunos[i][3])
            media = (nota1 + nota2 + nota3)/3.0
            Alunos[i].append("{:.2f}".format(media))
            linha = ""
            for item in Alunos[i]:
                linha = linha + item + " "
            linha = linha + "\n"
            arq_medias.write(linha)
        arq_medias.close()
    except:
        print("Arquivo nao encontrado")
```