Algoritmos e Programação I: Aula 09*

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

Sumário

1 Solucionando de Problemas Numéricos com Estrutura de Seleção

Nesta aula, apresentaremos mais exemplos de problemas que utilizam a estrutura de controle condicional em suas soluções. Nesses exemplos usamos a estrutura de seleção de duas vias (`if-else`).

1.1 Quociente e Resto do Maior pelo Menor

Projete e implemente um programa que leia dois números inteiros, compute o maior e o menor entre eles e calcule o quociente e o resto da divisão do maior pelo menor. A entrada é dada por dois números inteiros e a saída consiste em imprimir para cada par de inteiros lidos o resultado da divisão do maior número pelo menor número e o resto da divisão do maior número pelo menor (com mensagens apropriadas). Imprima o quociente e o resto da divisão usando o formato abaixo:

```
maior / menor = quociente
maior % menor = resto
```

Exemplo 1

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

```
# formato da entrada
4
5

# formato da saída
5 / 4 = 1
5 % 4 = 1
```

Exemplo 2

```
# formato da entrada
30345
30345

# formato da saída
30345 / 30345 = 1
30345 % 30345 = 0
```

Exemplo 3

```
# formato da entrada
10
9

# formato da saída
10 / 9 = 1
10 % 9 = 1
```

Na solução deste problema, após a leitura dos dois número inteiros, antes de computar o quociente e o resto, temos que compará-los e computar o maior e o menor. Como vimos anteriormente, um programa pode ser visto como uma função ou uma composição de funções. Neste caso temos uma função

$$f(a, b) = (\max(a, b), \min(a, b))$$

que lê dois números inteiros e computa o maior e o menor, composta com a função

$$g(dividendo, divisor) = (dividendo//divisor, dividendo\%divisor)$$

que computa a quociente e o resto da divisão de dois números inteiros, onde:

$$dividendo == \max(a, b)$$

$$divisor == \min(a, b)$$

$$g(f(a, b)) = (\max(a, b)// \min(a, b), \max(a, b)\% \min(a, b))$$

Em aulas anteriores, já resolvemos o problema de computar a divisão e o resto entre dois números inteiros. Usando o programa `divisaoresto.py` discutido anteriormente, e atribuindo o maior número ao dividendo e o menor número ao divisor, o problema estará resolvido. Posteriormente abordaremos como usar funções/métodos do Python para compor a solução de problemas. Agora faremos apenas a "reutilização" de código.

Em função disso, vamos descrever uma solução para o problema de ler dois números inteiros e computar qual é o maior e qual é o menor dos números lidos. Após isso, usaremos o programa `divisaoresto.py` e finalizamos a solução do problema de computar a divisão e o resto do maior pelo menor número lido.

O nosso problema consiste na leitura de dois números inteiros, computar qual é o maior e qual é o menor e atribuí-los a dividendo e divisor, respectivamente.

Exemplo 1

```
# formato da entrada
4
5

# estado das variáveis na saída
dividendo == 5
divisor == 4
```

Exemplo 2

```
# formato da entrada
30345
30345

# estado das variáveis na saída
dividendo == 30345
divisor == 30345
```

Exemplo 3

```
10
9

# estado das variáveis na saída
dividendo == 10
divisor == 9
```

Dados dois números inteiros, para computar o maior e o menor número, temos que comparar o primeiro com o segundo. Se o primeiro for maior ou igual ao segundo, atribuímos o primeiro como o maior número e o segundo como o menor número. Caso contrário, atribuímos como maior o segundo e o primeiro como o menor.

Com essas observações, a descrição detalhada para esse problema fica:

```
# Este programa lê dois números inteiros, calcula o maior e o menor entre
# os números e atribui maior a dividendo e menor a divisor, tal que
# menor != 0.
```

Agora faremos as especificações da entrada e saída do programa. A entrada para este problema consiste de dois números inteiros, sendo que o menor deve ser diferente de 0. A saída é dada pela atribuição do maior a dividendo e o menor a divisor.

Entrada	**Saída**
2 números inteiros, sendo que menor != 0	maior atribuído a dividendo menor atribuído a divisor.

Os identificadores abaixo indicam as variáveis e seus respectivos tipos para armazenar as informações de entrada e saída.

```
# descrição das variáveis utilizadas
# int num1, num2
# int dividendo, divisor
```

e as pré e pós-condições ficam:

```
# pré: num1 num2 and min(num1,num2) != 0

# pós: dividendo == max(num1,num2) and divisor == min(num1,num2)
```

Exemplo 1:

```
# formato da entrada
4 # num1
5 # num2

# estado das variáveis após a leitura
num1 == 4
num2 == 5

# cálculo do maior e menor
dividendo = max(num1,num2)
divisor = min(num1,num2)

# estado das variáveis na saída
dividendo == 5
divisor == 4
```

Exemplo 2:

```
# formato da entrada
30345 # num1
30345 # num2

# estado das variáveis após a leitura
num1 == 30345
num2 == 30345
```

```
# cálculo do maior e menor
dividendo = max(num1,num2)
divisor = min(num1,num2)

# estado das variáveis na saída
dividendo == 30345
divisor == 30345
```

Exemplo 3:

```
# formato da entrada
10 # num1
9 # num2

# estado das variáveis após a leitura
num1 == 10
num2 == 9

# cálculo do maior e menor
dividendo = max(num1,num2)
divisor = min(num1,num2)

# estado das variáveis na saída
dividendo == 10
divisor == 9
```

Agora vamos descrever os passos do algoritmo para solucionar o problema:

```
# Algoritmo: MaiorMenor
# Passo 1. Leia dois números inteiros
# Passo 2. Calcule o maior e o menor entre num1 e num2
# Passo 2.1. Calcule o maior e atribua a dividendo
# Passo 2.2. Calcule o menor e atribua a divisor
# fim do algoritmo
```

Temos que detalhar o Passo 2, pois no cálculo do maior e do menor vamos usar a instrução `if else` (não vamos usar as funções `max()` e `min()`). No cálculo do maior e menor, temos que comparar os dois valores da entrada (`num1 >= num2`), se o resultado da comparação for True, o maior valor será `num1` e o menor valor será `num2`, caso contrário o maior valor será `num2` e o menor valor será `num1`. O maior valor será atribuído a `dividendo` e o menor valor será atribuído a `divisor`. Isso pode ser implementado em *Python* usando a instrução `if else`:

```
if num1 >= num2:
   dividendo = num1
   divisor = num2
else:
   dividendo = num2
   divisor = num1
```

Exemplo 1

```
# formato da entrada
4 # num1
5 # num2

# cálculo do dividendo e divisor
if num1 >= num2:
   dividendo = num1
   divisor = num2
else:
   dividendo = num2
   divisor = num1

# como num1 == 4 e num2 == 5, a expressão num1 >= num2 é False
# e serão executadas as instruções depois do else

if 4 >= 5: # expressão falsa
   dividendo = num1
   divisor = num2
else:
   dividendo = num2
   divisor = num1

# estado das variáveis após o if-else
dividendo == 5
divisor == 4
```

Exemplo 2

```
# formato da entrada
10 # num1
9 # num2

# cálculo do dividendo e do divisor
if num1 >= num2:
   dividendo = num1
   divisor = num2
else:
   dividendo = num2
   divisor = num1

# como num1 == 10 e num2 == 9, a expressão num1 >= num2 é True
# e serão executadas as instruções depois de '':'' e antes do else

if 10 >= 9: # expressão verdadeira
   dividendo = num1
   divisor = num2
else:
   dividendo = num2
```

```
   divisor = num1
```

```
# -*- coding: utf-8 -*-
# Programa: maiormenor.py
# Programador:
# Data: 05/04/2016
# Este programa lê dois números inteiros, calcula o maior e o menor entre
# os números e atribui maior a dividendo e menor a divisor, tal que
# menor != 0.
# início do módulo principal
# descrição das variáveis utilizadas
# int num1, num2
# int dividendo, divisor

# pré: num1 num2 and min(num1,num2) != 0

# Passo 1. Leia dois números inteiros
print('Leia um inteiro num1: ')
num1 = int(input())
print('Leia um inteiro num2: ')
num2 = int(input())
# Passo 2. Calcule o maior e o menor entre num1 e num2
if num1 >= num2:
   dividendo = num1 # atribua maior a dividendo
   divisor = num2 # atribua menor a divisor
else:
   dividendo = num2 # atribua maior a dividendo
   divisor = num1 # atribua menor a divisor

# pós: dividendo == max(num1,num2) and divisor == min(num1,num2)
# fim do módulo principal
```

Agora vamos compor os dois programas `divisaoresto.py` e `maiormenor.py`. O programa resultante lê dois números inteiros e calcula a divisão e o resto entre o maior e o menor.

```
# -*- coding: utf-8 -*-
# Programa: divresmaiormenor.py
# Programador:
# Data: 05/04/2016
# Este programa lê dois números inteiros, calcula o maior e o menor entre
# os números e atribui maior a dividendo e menor a divisor, tal que
# menor != 0. O programa calcula o quociente e o resto da divisão entre
# o dividendo e o divisor e imprime o resultado.
# início do módulo principal
# descrição das variáveis locais utilizadas
# int num1, num2, dividendo, divisor, quociente, resto

# pré: num1 num2 and min(num1,num2) != 0
```

```
14
15  # Passo 1. Leia dois números inteiros
16  print('Leia um inteiro num1: ')
17  num1 = int(input())
18  print('Leia um inteiro num2: ')
19  num2 = int(input())
20  # Passo 2. Calcule o maior e o menor entre num1 e num2
21  if num1 >= num2:
22     dividendo = num1 # atribua maior a dividendo
23     divisor = num2 # atribua menor a divisor
24  else:
25     dividendo = num2 # atribua maior a dividendo
26     divisor = num1 # atribua menor a divisor
27
28  # pós: dividendo == max(num1,num2) and divisor == min(num1,num2)
29  # pré: dividendo divisor and divisor != 0
30
31  # Passo 3. Calcule o quociente e o resto da divisão entre os dois números
32  # Passo 3.1. Calcule o quociente da divisão inteira entre os dois números
33  quociente =  dividendo // divisor
34  # Passo 3.2. Calcule o resto da divisão inteira entre os dois números
35  resto = dividendo % divisor
36  # Passo 4. Imprima o quociente e resto
37  print('{0:d} / {1:d} = {2:d}').format(dividendo, divisor, quociente)
38  print('{0:d} % {1:d} = {2:d}').format(dividendo, divisor, resto)
39
40  # pós: quociente, resto and quociente == dividendo // divisor and
41  #      resto == dividendo % divisor
42  # fim do módulo principal
```

Exemplo 1

```
# execução do programa
# vamos assumir que os valores lidos são 4 e 5
# o valor em azul será digitado pelo usuário, seguido de um enter
# as mensagens em magenta indicam o que está sendo impresso pelo programa
# formato da entrada
Leia um inteiro num1: # linha 16
4 # linha 17
Leia um inteiro num2: # linha 18
5 # linha 19

# estado das variáveis após a leitura dos dados (após a linha 19)
num1 == 4
num2 == 5

# as linhas 21 a 23 computam o dividendo (max(num1,num2)) e
# divisor (min(num1,num2))
# na execução da linha 21 temos
if num1 >= num2:
```

```
if 4 >= 5:
# a instrução if avalia a expressão 4 >= 5, cujo resultado é False, e com
# isso desvia a execução do programa para as instruções das linha 24 a 26
   dividendo = num2
   divisor = num1
# e passa a executar o programa a partir da linha 33, as linhas
# 22 e 23 são ignoradas

# estado das variáveis após a linha 26
num1 == 4
num2 == 5
dividendo == 5
divisor == 4

# na execução das linhas 33 e 35 temos
quociente = 5 // 4
resto = 5 % 4

# estado das variáveis após a linha 35
num1 == 4
num2 == 5
dividendo == 5
divisor == 4
quociente == 1
resto == 1

# na execução das linhas 37 e 38 será gerada a saída
5 / 4 = 1
5 % 4 = 1

# e o programa é encerrado
```

Exemplo 2

```
# execução do programa
# vamos assumir que os valores lidos são 10 e 9
# o valor em azul será digitado pelo usuário, seguido de um enter
# as mensagens em magenta indicam o que está sendo impresso pelo programa
# formato da entrada
Leia um inteiro num1: # linha 16
10 # linha 17
Leia um inteiro num2: # linha 18
9 # linha 19

# estado das variáveis após a leitura dos dados (após a linha 19)
num1 == 10
num2 == 9

# as linhas 21 a 23 computam o dividendo (max(num1,num2)) e
# divisor (min(num1,num2))
```

```
# na execução da linha 21 temos
if num1 >= num2:
if 10 >= 9:
# a instrução if avalia a expressão 10 >= 9, cujo resultado é True, e com
# isso leva a execução das instruções das linha 22 a 23
   dividendo = num1
   divisor = num2
# e desvia a execução do programa a partir da linha 33, as linhas
# 24, 25 e 26 são ignoradas

# estado das variáveis após a linha 26
num1 == 10
num2 == 9
dividendo == 10
divisor == 9

# na execução das linhas 33 e 35 temos
quociente = 10 // 9
resto = 10 % 9

# estado das variáveis após a linha 35
num1 == 10
num2 == 9
dividendo == 10
divisor == 9
quociente == 1
resto == 1

# na execução das linhas 37 e 38 será gerada a saída
10 / 9 = 1
10 % 9 = 1

# e o programa é encerrado
```

Usando um ambiente para programas Python (p. ex. IDLE). edite e execute o programa para vários valores de entrada. Se você editou o programa corretamente a execução não gerará nenhuma advertência ou erro. Caso isso ocorra, veja o número da linha e verifique o que você digitou de forma errada. Após ter corrigido as advertências e erros, execute o programa o programa novamente até não ter nenhuma advertência ou erros.

1.2 Média Notas II

A avaliação da disciplina de Algoritmos e Estrutura de Dados I é feita por 3 notas de provas, (`P1`, `P2` e `P3` e 2 notas de trabalhos `T1` e `T2`. As notas das provas e trabalhos são dadas por números de ponto flutuante com uma casa decimal variando de 0 e 10.0. A média final é dada pela soma de 75% da média das provas (`MP`) com 25% da média dos trabalhos (`MP`), `0.75*MP + 0.25*MT`. A média das provas `MP` é dada por `(P1 + P2 + P3)/3.0` e a média dos trabalhos `MT` é dada por `(T1 + T2)/2.0`. A média final da disciplina é calculada com uma casa decimal, onde valores maiores ou iguais a cinco na segunda casa decimal são

aproximados para um décimo a mais na primeira casa decimal. Valores menores que cinco na segunda casa decimal não são considerados. A aproximação só considera duas casas decimais a esquerda do ponto decimal, nesse caso, o valor 8.445 é aproximado para 8.4. O(A)s estudantes com médias maiores ou iguais a 6.0 são considerados aprovado(a)s (`AP`) e o(a)s estudantes com médias inferiores a 6.0 são considerado reprovados (`RP`). Projete um programa para ler as notas das três provas e dos dois trabalhos, calcular a média final de um(a) estudante e verificar se ele foi aprovado ou reprovado. Para esse problema, a entrada é dada pelo RGA do estudante, as notas das três provas, e a nota dos dois trabalhos, e a saída é dada pela média final do estudante e a impressão da mensagem correspondente, `AP` ou `RN`.

Exemplo

```
# formato da entrada
20181910045-6 # entrada do número do estudante
6.0 7.0 6.0 # entrada das notas das provas
9.0 9.0 # entrada das notas dos trabalhos

# formato da saída
RGA: 20181910045-6: Média =  7.0 - AP
```

onde o cálculo das médias pode ser feito da seguinte forma:

```
Média das provas: (6.0 + 7.0 + 6.0)/3.0 = 6.333333
Média dos trabalhos: (9.0 + 9.0)/2.0 = 9.0
Média Final: (0.75*6.333333 + 0.25*9.0)=7.0
```

Anteriormente, já resolvemos o problema de computar a média da disciplina de Algoritmos e Programação I (`medianotas01.py`). O problema acima, além da média, necessita computar uma mensagem indicando se o(a) estudante foi aprovado `AP` ou reprovado `RP`. Um outro ponto que deveremos abordar, é o da aproximação, pois teremos que fazer uma comparação com a média de aprovação. No programa `medianotas01.py`, a aproximação era feita na impressão com uma casa decimal ({.1f}), usando a função `print()`. No caso em que a média final fosse `5.95`, a impressão da média usando o programa `medianotas01.py` seria `6.0`, mas se compararmos `5.95 <= 6.0`, a resposta será `False`, e nesse caso teríamos uma média `6.0` e uma mensagem `RP`, o que não seria consistente.

Vamos detalhar a aproximação e como é computada a impressão da mensagem. No mais, vamos ajustar a descrição, as especificações, as variáveis e as pré e pós-condições do problema Média Notas I. Na descrição para esse problema temos que descrever alguns detalhes do formato das notas e das aproximações que serão utilizadas nas médias.

```
# A média final de um aluno para um dado curso é computada pela soma de
# 75% da média das provas com 25% da média dos trabalhos. A média das
# provas é computada pela média aritmética das notas das três provas, e a
# média dos trabalhos é computada pela média aritmética das notas dos dois
# trabalhos. O programa lê as notas das três provas e as notas dos dois
# trabalhos, computa as médias das provas, dos trabalhos e final e imprime
# o resultado. As notas são dadas por números de ponto flutuante com uma
# casa decimal variando de 0 e 10.0. As médias são calculadas também com
```

```
# uma casa decimal e a aproximação é feita da direita para a esquerda e
# dígitos maiores ou iguais a cinco, aumenta em uma unidade o dígito da
# esquerda (7.675 == 7.7). Alunos com médias maiores ou iguais a 6.0
# são considerados aprovados, e reprovados caso a média seja menor que 6.0
```

As especificações de entrada e saída ficam:

Entrada	**Saída**
5 valores de avaliações Prova1 , Prova2, Prova3 Trabalho1, Trabalho2	Média final do aluno MSG de aprovado (AP) ou reprovado (RP)

Os identificadores abaixo representam as variáveis do tipo `float` para armazenar as informações das notas e das médias e do tipo `str` para armazenar o número do(a) estudante e a mensagem de aprovação ou reprovação:

```
# descrição das variáveis utilizadas
# float prova1, prova2, prova3, trabalho1, trabalho2
# float mediaprovas, mediatrabalhos, mediafinal
# str rga, msg
```

e as pré e pós-condições ficam:

```
# pré: prova1, prova2, prova3, trabalho1, trabalhos2

# pós: mediafinal and (AP and mediafinal >= 6.0 or RP) and
#      mediafinal = 0.75*mediaprovas + 0.25*mediatrabalhos and
#      mediaprovas == (prova1 + prova2 + prova3) / 3.0 and
#      mediatrabalhos == (trabalho1 + trabalho2) / 2.0
```

Exemplo

```
# formato da entrada
20181910048-6 # entrada do número do estudante
7.5 8.2 6.5 # entrada das notas das provas
9.0 8.0 # entrada das notas dos trabalhos


# estado das variáveis após a leitura
rga == '20181910048-6'
nota1 == 7.5
nota2 == 8.2
nota3 == 6.5
trabalho1 == 9.0
trabalho2 == 8.0

# cálculo da média das provas
mediaprovas = (nota1 + nota2 + nota3)/3.0
mediaprovas = (7.5 + 8.2 + 6.5)/3.0
# cálculo da média dos trabalhos
```

```
mediatrabalhos = (trabalho1 + trabalho2)/2.0
mediatrabalhos = (9.0 + 8.0)/2.0
# cálculo da média final
mediafinal = 0.75*mediaprovas + 0.25*mediatrabalhos
mediafinal = 0.75*7.4 + 0.25*8.5

# estado das variáveis após o cálculo das médias
mediaprovas == 7.4
mediatrabalhos == 8.5
mediafinal == 7.675

# aproximação da média - a ser detalhado a seguir
mediafinal == 7.7

# calcule a msg de aprovação ou reprovação - a ser detalhado a seguir
# como mediafinal == 7.7 e é maior ou igual a 6.0, a msg será 'AP'

# estado das variáveis após o cálculo da mensagem
msg == 'AP'

# formato desejado da saída
RGA: 20181910048-6: Média =  7.7 - AP
```

Vamos agora descrever os passos do Algoritmo:

```
# Algoritmo: Média Notas 02
# Passo 1. Leia o RGA e as notas das provas e trabalhos
# Passo 1.1. Leia o RGA do aluno
# Passo 1.2. Leia as notas das provas
# Passo 1.3. Leia a nota dos trabalhos.
# Passo 2. Calcule as médias das avaliações
# Passo 2.1. Calcule a média das provas
# Passo 2.2. Calcule a média dos trabalhos
# Passo 2.3. Calcule a média das avaliações
# Passo 2.4. Compute a aproximação na media final
# Passo 2.5. Compute a msg de aprovação ou reprovação
# Passo 3. Imprima os resultados
# fim algoritmo
```

Temos que computar a aproximação da media final e a mensagem correspondente, `AP` ou `RP`. Para a aproximação, vamos usar a função `round(x,d)`, onde `x` é um `float` e `d` é o número de casas decimais desejadas. Em da representação binária de números de ponto flutuante, algumas vezes a aproximação dada por `round()` pode não ser a esperada. Dependendo da situação, podemos usar uma aproximação explícita como fizemos no programa `telefone.py`. No cálculo da mensagem de aprovação ou reprovação, temos que comparar a média final com 6.0:

```
# Passo 2.5. Compute a msg de aprovação ou reprovação
# Passo 2.5.1. Se mediafinal >= 6.0
```

```
# Passo 2.5.1.1. Aprovado
# Passo 2.5.2. Caso contrário
# Passo 2.5.2.1. Reprovado
```

os passos acima do algoritmo podem ser implementados em Python como:

```
if mediafinal >= 6.0:
   msg = 'AP'
else:
   msg = 'RP'
```

Exemplo - cont.

```
# estado da variável mediafinal sem a aproximação
mediafinal == 7.675

# cálculo da aproximação da média final
mediafinal = round(mediafinal,1) # 1 casa decimal
mediafinal = round(7.675,1)
mediafinal = 7.7

# estado da variável mediafinal com a aproximação
mediafinal == 7.7

# cálculo da mensagem de aprovação ou reprovação
if mediafinal >= 6.0:
# como mediafinal == 7.7, 7.7 >=6.0 é True
# e será executada a instrução do if depois de ':' e antes do else
if 7.7 >= 6.0:
   msg = 'AP'
else:
   msg = 'RP'
```

A codificação em Python fica:

```
# Programa: medianotas02.py
# Programador:
# Data: 14/09/2010
# A média final de um aluno para um dado curso é computada pela soma de
# 75% da média das provas com 25% da média dos trabalhos. A média das
# provas é computada pela média aritmética das notas das três provas, e a
# média dos trabalhos é computada pela média aritmética das notas dos dois
# trabalhos. O programa lê as notas das três provas e as notas dos dois
# trabalhos, computa as médias das provas, dos trabalhos e final e imprime
# o resultado. As notas são dadas por números de ponto flutuante com uma
# casa decimal variando de 0 e 10.0. As médias são calculadas também com
# uma casa decimal e a aproximação é feita da direita para a esquerda e
# dígitos maiores ou iguais a cinco, aumenta em uma unidade o dígito da
# esquerda (7.675 == 7.7). Alunos com médias maiores ou iguais a 6.0
```

```
# são considerados aprovados, e reprovados caso a média seja menor que 6.0
# início do módulo principal

# float prova1, prova2, prova3, trabalho1, trabalho2
# float mediaprovas, mediatrabalhos, mediafinal
# str rga, msg

# pré: prova1, prova2, prova3, trabalho1, trabalhos2

# Passo 1. Leia o RGA e as notas das provas e trabalhos
# Passo 1.1. Leia o RGA do aluno
rga = input('Leia o número do estudante: ')
# Passo 1.2. Leia as notas das provas
print('Entre com as notas das provas: ')
prova1, prova2, prova3 = map(float,input().split())
# Passo 1.3. Leia a nota dos trabalhos.
print('Entre com as notas dos trabalhos: ')
trab1, trab2 = map(float,input().split())
# Passo 2. Calcule as médias das avaliações
# Passo 2.1. Calcule a média das provas
mediaprovas = (prova1 + prova2 + prova3)/3.0
# Passo 2.2. Calcule a média dos trabalhos
mediatrab = (trab1 + trab2)/2.0
# Passo 2.3. Calcule a média das avaliações
mediafinal = 0.75 * mediaprovas + 0.25 * mediatrab
# Passo 2.4. Compute a aproximação na media final
mediafinal = round(mediafinal,1)
# Passo 2.5. Compute a msg de aprovação ou reprovação
if mediafinal >= 6.0:
   msg = "AP"
else:
   msg = "RP"
# Passo 3. Imprima os resultados
print('RGA: {0:s}: Média = {1:5.1f} - {2:s}'.format(rga,mediaFinal,msg))

# pós: mediafinal and (AP and mediafinal >= 6.0 or RP) and
#      mediafinal = 0.75*mediaprovas + 0.25*mediatrabalhos and
#      mediaprovas == (prova1 + prova2 + prova3) / 3.0 and
#      mediatrabalhos == (trabalho1 + trabalho2) / 2.0
# fim do módulo principal
```

Exemplo

```
# execução do programa
# vamos assumir que os valores lidos são
# rga do aluno 20181910045-6
# notas das provas 6.0 7.0 6.0
# notas dos trabalhos 9.0 9.0
# o valor em azul será digitado pelo usuário, seguido de um enter
# as mensagens em magenta indicam o que está sendo impresso pelo programa
```

```
# formato da entrada
Leia o número do estudante:  20181910045-6 # linha 26
Entre com as notas das provas:  # linha 28
6.0 7.0 6.0 # linha 19
Entre com as notas dos trabalhos:  # linha 31
9.0 9.0 # linha 32

# estado das variáveis após a leitura dos dados (após a linha 19)
rga == '20181910045-6'
nota1 == 6.0
nota2 == 7.0
nota3 == 6.0
trabalho1 == 9.0
trabalho2 == 9.0

# a linha 35 computa a média das provas
mediaprovas = (prova1 + prova2 + prova3)/3.0
mediaprovas = (6.0 + 7.0 + 6.0)/3.0
mediaprovas = 6.333333
# a linha 37 computa a média dos trabalhos
mediatrab = (trab1 + trab2)/2.0
mediatrab = (9.0 + 9.0)/2.0
mediatrab = 9.0
# a linha 39 computa a média final
mediafinal = 0.75*mediaprovas + 0.25*mediatrab
mediafinal = 0.75*6.333333 + 0.25*9.0
mediafinal = 4.75 + 2.25
mediafinal = 7.0
# a linha 41 computa o arredondamento
mediafinal = round(mediafinal,1)
mediafinal = round(7.0,1)
mediafinal = 7.0
# na execução da linha 44 temos
if mediafinal >= 6.0:
if 7.0 >= 6.0:
# a instrução if avalia a expressão 7.0 >= 6.0, cujo resultado é True,
# e com isso executa a instrução do programa da linha 44
   msg = 'AP'
# e passa a executar o programa a partir da linha 48, as linhas
# 45 e 46 são ignoradas

# na execução das linhas 48 será gerada a saída
RGA: 20181910045-6: Média =   7.0 - AP

# e o programa é encerrado
```

Edite o programa e execute para vários valores de entrada.

1.3 Calculado as raízes de uma equação do 2° Grau

Dado os coeficientes de uma equação do 2º grau $ax^2 + bx + c = 0$ projetar um algoritmo para ler os coeficientes e calcular e imprimir (se existirem) as raízes da equação.

Ex: a = 1, b = 3 e c = -4 Usando a fórmula de Báscara, as raízes da equação são $r_1 = -4$ e $r_2 = 1$.

A solução desse problema depende de uma decisão, pois as raízes só podem ser calculadas de o valor de $\Delta = b^2 - 4ac$ for maior ou igual a zero.

Descreveremos uma solução para esse problema seguindo a metodologia MAPS.

A descrição do problema é imediata:

```
# Este algoritmo lê três coeficientes, a, b e c de uma equação do segundo
# grau (ax^2 + bx + c = 0), e utilizando a fórmula de Báscara calcula (se
# existirem) as raízes da equação ou imprime uma mensagem caso contrário
```

e as especificações de entrada e saída ficam:

Entrada	**Saída**
Três números reais, a, b e c.	As raízes reais r_1 e r_2 (se existirem), ou uma mensagem.

Os identificadores abaixo representam as variáveis do tipo `float` para armazenar os valores dos coeficientes de `a`, `b` e `c`, do valor de `delta=b**2-4*c`, e das raízes `r1` e `r2`. As pré e pós-condições ficam:

```
# pré: a b c and a != 0

# pós: (para i em 1,2:r[i] ==(-b +- sqrt(Delta))/(2*a) or MSG,
#      Delta == b*b-4*a*c
```

Os passos do Algoritmo são:

```
Algoritmo: EquaçãoSegundoGrau
# Passo 1. Leia os coeficientes da equação
# Passo 2. Calcule as raízes reais ou uma msg
# Passo 3. Imprima o resultado
# fim do algoritmo
```

O passos 2 necessita um refinamento, pois não está descrito de uma forma que possa ser implementado num computador. Refinando o passo 2 temos:

```
# Passo 2. Calcule as raízes reais
# Passo 2.1. Calcule delta
# Passo 2.2. Se delta >= 0 calcule as raízes
# Passo 2.3. Caso contrário, gere uma mensagem
```

Temos que descrever agora como o Δ é calculado e como as raízes de uma equação do segundo grau são calculadas. A existência dessas raízes depende do valor de $\Delta = b^2 - 4ac$. Se $\Delta \geq 0$ as raízes podem ser calculadas como $r_1 = (-b - \sqrt(\Delta))/2a$ e $r_2 = (-b + \sqrt(\Delta))/2a$. Se $\Delta < 0$ nenhum cálculo das raízes é efetuado e uma mensagem é gerada (`'A equação não possui raízes reais'`).

A descrição completa do programa em Python fica:

```
# -*- coding: utf-8 -*-
# Programa: equacao2grau.py
# Programador:
# Data: 01/09/2016
# Este algoritmo lê três coeficientes, a, b e c de uma equação do segundo
# grau (ax^2 + bx + c = 0), e utilizando a fórmula de Báscara calcula (se
# existirem) as raízes da equação ou imprime uma mensagem caso contrário
# declaração das bibliotecas utilizadas
import math
# início do módulo principal
# descrição das variáveis utilizadas
# float a, b, c, delta, r1, r2

# pré: a b c && a != 0

# Passo 1. Leia os coeficientes da equação
print('Entre com os três coeficientes: ')
a, b, c = map(float,input().split())
# Passo 2. Calcule as raízes reais
# Passo 2.1. Calcule o valor de delta
delta = b*b - 4*a*c
# Passo 2.2. Compute a msg sobre as raízes
if delta >= 0: # possui raízes
   r1 = (-b - math.sqrt(delta))/(2*a)
   r2 = (-b + math.sqrt(delta))/(2*a)
   msg = '{0:f}, {1:f}'.format(r1, r2)
else: # não possui raízes
   msg = 'A equação não possui raízes reais'
# Passo 3. Imprima as raízes ou a mensagem adequada
print(msg)

# pós: (para i em {1,2}, r[i] ==(-b +- sqrt(delta))/(2*a) and
#      delta >=0 and delta == b*b-4*a*c or
#      msg == 'A equação não possui raízes reais'
# fim do módulo principal
```

1.4 Calculando a idade

Projete e implemente um programa que tenha como entrada a data de nascimento de uma pessoa e a data atual. O programa então calcula a idade em anos completos da pessoa. As datas são dadas na seguinte forma `dd/mm/aaaa`, onde `dd == dia`, `mm == mês` e `aa == ano`. O programa imprime a idade da pessoa.

Exemplo 1

```
# formato da entrada
Entre com a data de referência:06/04/2011
Entre com a data do nascimento:14/11/1957

# formato da saída
```

```
A pessoa tem 53 anos completos
```

Exemplo 2

```
# formato da entrada
Entre com a data de referência:06/04/2011
Entre com a data do nascimento:05/04/2000

# formato da saída
A pessoa tem 11 anos completos
```

A descrição desse problema é bem simples:

```
# Este programa lê a data de nascimento de uma pessoa e uma data dada
# e calcula a idade em anos completos da pessoa na data dada. As datas
# dadas no formato dd/mm/aaa. O programa imprime a idade da pessoa
# na data dada.
```

As especificações de entrada e saída são:

Entrada	**Saída**
6 números inteiros representado a data de referência e a data de nascimento	um inteiro representado a idade

Nesse exemplo as varáveis são são compostas de tipos básicos. Necessitamos do tipo `int` para armazenar os valores de `adia`, `ames` e `aano`, `ndia`, `nmes` e `nano`. Quando da implementação do programa pode surgir a necessidade da utilizarmos variáveis auxiliares para armazenar valores intermediários.

```
# descrição das variáveis utilizadas
# int hdia, hmes, hano
# int ndia, nmes, nano
```

e as pré e pós-condições ficam:

```
# pré: adia ames aano ndia nmes nano

# pós: idade
```

Exemplo:

```
# formato da entrada
Entre com a data de referência:06/04/2011
Entre com a data do nascimento:05/10/1989

# estado das variáveis após a leitura dos dados
adia == 6
```

```
ames == 4
ano == 2011
ndia == 5
nmes == 10
nano == 1989

# após o cálculo da idade
idade == 21

# formato da saída
A pessoa tem 21 anos completos
```

Os passos dos algoritmos são:

```
Algoritmo: Idade
# Passo 1. Leia a data atual e de nascimento da pessoa
# Passo 2. Calcule a idade da pessoa
# Passo 2.1. Calcule a diferença dos anos
# Passo 2.2. Verifique se o mês do aniversário ainda não passou
# Passo 2.3. Se o mês atual é o do aniversário verifique o dia
# Passo 3. Imprima a Idade
# fim Algoritmo
```

No Passo 2 temos que comparar a data mês e do dia do nascimento com a data do mês e do dia para ver se a pessoa completou anos na data de hoje. Caso ela ainda não tenha comemorado aniversário até a data atual, sua idade deve ser diminuída de 1.

O Passo 2 pode ser descrito em Python como:

```
# Passo 2. Calcule a idade da pessoa
# Passo 2.1. Calcule a diferença dos anos
idade = aano - nano
# Passo 2.2. Verifique se o mês do aniversário ainda não passou
if nmes > ames:
   idade = idade - 1
# Passo 2.3. Se o mês atual é o do aniversário verifique o dia
else:
   if nmes == ames and ndia > adia:
      idade = idade - 1
```

A Codificação completa em `Python` fica:

```
1 # -*- coding: utf-8 -*-
2 # Programa: idade.py
3 # Programador:
4 # Data: 03/04/2016
5 # Este programa lê a data de nascimento de uma pessoa e uma data dada
6 # e calcula a idade em anos completos da pessoa na data dada. As datas
7 # dadas no formato dd/mm/aaa. O programa imprime a idade da pessoa
8 # na data dada.
```

```
# início do módulo principal

# pré: hdia hmes hano ndia nmes nano

# Passo 1. Leia as datas da entrada
# Passo 1.1. Leia a data de referência
hdia,hmes,hano=map(int,input('Data de referência:').split('/'))
# Passo 1.2. Leia a data do aniversário
ndia,nmes,nano=map(int,input('Data de nascimento:').split('/'))
# Passo 2. Calcule a idade da pessoa
# Passo 2.1. Calcule a diferença dos anos
idade = aano - nano
# Passo 2.2. Verifique se o mês do aniversário ainda não passou
if nmes > ames:
   idade = idade - 1
# Passo 2.3. Se o mês atual é o do aniversário verifique o dia
else:
   if nmes == ames and ndia > adia:
      idade = idade - 1
# Passo 3. Imprima a idade da pessoa
print('A pessoa tem {0:d} anos completos'.format(idade))

# pós: idade
# fim do módulo principal
```

Verificação e Teste:
Exemplo 1

```
# formato da entrada
Entre com a data de referência:06/04/2011

# formato da saída
Entre com a data do nascimento:14/11/1957
A pessoa tem 53 anos completos
```

Exemplo 2

```
# formato da entrada
Entre com a data de referência:06/04/2011

# formato da saída
Entre com a data do nascimento:10/05/1989
A pessoa tem 21 anos completos
```

Exemplo 3

```
# formato da entrada
Entre com a data de referência:06/04/2011
```

```
# formato da entrada
Entre com a data do nascimento:05/01/2000
A pessoa tem 11 anos completos
```

2 Solucionando de Problemas com Texto

2.1 Compara

Da mesma maneira que comparamos e ordenamos números, muitas vezes necessitamos comparar ou ordenar um conjunto de palavras ou nomes (ex. Lista Telefônica). Como veremos posteriormente, a comparação e a ordenação de palavras é um pouco diferente da comparação e ordenação de números. No caso da comparação de palavras, usamos o conceito de lexicográfico onde para verificarmos se `palavraA < palavraB`, fazemos a comparação caractere a caractere.

```
'porta' < 'porto' # verdadeiro
```

pois temos `'p'== 'p'`, `'o'== 'o'`, `'r'== 'r'` e `'a'< 'o'`.

Além disso, quando usamos um computador para fazer a comparação, todas os caracteres são representados por algum tipo de codificação. Considere a codificação ASCII. Nessa codificação o caractere `'A'` tem o valor decimal 65 e o caractere `'a'` tem o valor decimal 97. Se comparamos num computador que utiliza a codificação ASCII, `'A'<'a'`, a resposta será True.

Considere o problema de computar a menor (maior) entre duas palavras lidas. O programa abaixo descreve uma solução para esse problema:

```
1  # Programa: compara00.py
2  # Programador:
3  # Data: 07/11/2016
4  # Este programa lê duas palavras, compara as palavras, computa a
5  # maior, lexicograficamente, e imprime a maior.
6  # início do módulo principal
7  # descrição das variáveis utilizadas
8  # string palavra1, palavra2, maior
9
10 # pré: palavra1 palavra2
11
12 # Passo 1. Leia duas palavras
13 palavra1 = input()
14 palavra2 = input();
15 # Passo 2. Calcule a maior palavra (lexicograficamente)
16 if palavra1 >= palavra2:
17 # Passo 2.1. Se Palavra1 for maior, atribua palavra1 a maior
18    maior = palavra1
19 # Passo 2.2. Se Palavra1 for maior, atribua palavra2 a maior
20 else:
21    maior = palavra2
```

```
22 # Passo 3. Imprima a maior palavra
23 print('A maior palavra é {0:s}'.format(maior))
24
25 # pós: maior and maior == max{palavra1, palavra2}
26 # fim do módulo principal
```

Usando uma IDE, implemente a solução acima. Se você editou o programa corretamente a execução do comando de execução não gerará nenhuma advertência ou erro. Caso isso ocorra, veja o número da linha e verifique o que você digitou de forma errada. Após ter corrigido as advertências e erros, execute o programa novamente. Teste o programa para os valores:

Exemplo 1

```
# formato da entrada
ana
paula

# formato da saída
paula
```

Exemplo 2

```
# formato da entrada
facom
ufms

# formato da saída
ufms
```

Usando o programa `compara00.py` como modelo, projete e implemente um programa que leia três palavras e compute e imprima a maior.