Universidade Federal de Minas Gerais
Instituto de Ciências Agrárias

Professor: Renato Dourado Maia
Disciplina: Programação de Computadores
Curso: Engenharia de Alimentos
Turma: Terceiro Período

Condições[1]

1 Introdução

Este documento deverá ser utilizado como um tutorial, ou seja, você deve lê-lo e executar as atividades que são propostas. Procure tentar desenvolver as atividades inicialmente sozinho, pedindo ajuda a colegas ou ao professor apenas depois de ter tentado várias vezes.

2 Entrada de Dados

Até agora, os programas que você escreveu trabalharam apenas com valores escritos diretamente no programa, por meio de atribuições a variáveis.

Pergunta: Não seria *mais interessante* que o programa pudesse *solicitar ao usuário* informações pelo teclado?

Isso pode ser feito por meio das funções `raw_input` e `input` da linguagem Python, considerando as versões[2] 2 e 3, respectivamente. Essas funções recebem como parâmetro uma *mensagem* que será apresentada na tela, retornando uma `string` contendo o que foi digitado. O Programa 1 demonstra

[1]Este material é baseado nos capítulos 3, 4 e 5 do livro "MENEZES, N. N. C. Introdução à Programação com Python. São Paulo: Novatec Editora, 2010."
[2]Este documento considerará a versão 2 da linguagem Python.

a utilização dessas funções.

```
1 x = raw_input('Digite um número: ') # input em Python 3
2 print x # print(x) em Python 3
```

Programa 1: Exemplo de Entrada de Dados.

A linha 1 solicita a *entrada de dados*. Essa linha faz com que seja exibida a mensagem `"Digite um número:  "` e o programa aguarda até que o usuário pressione `ENTER`. Apenas após ser pressionada a tecla `ENTER` o programa continua a sua execução.

Escreva o Programa 1 utilizando o editor de código do `IDLE` e o execute. Altere o programa de modo que ele mostre na tela o valor digitado multiplicado por cinco. Execute o seu programa. O resultado obtido está de acordo com o que você esperava? Se não, explique o porquê.

2.1 Conversão da Entrada de Dados

As funções `raw_input` e `input`, tal como já foi mencionado, retornam valores do tipo `string`. No Programa 2, mesmo quando você digita um número, ele é armazenado na variável `x` como uma `string`. Para resolver esse problema, podem ser utilizadas as funções `int` e `float` para *converter* o valor retornado em inteiro e ponto flutuante, respectivamente.

O Programa 2 demonstra a utilização dessas funções. Teste-o para diferentes valores de entrada, *inclusive letras*. O que acontece quando você digita `a`?

```
1 anos = int(raw_input("Anos de serviço: "))
2 valor_por_ano = float(raw_input("Valor por ano:" )
3 bonus = anos * valor_por_ano
4 print "Bonus de R$ %5.2f" % bonus
```

Programa 2: Cálculo de Bônus por Tempo de Serviço.

2.1.1 Exercícios – Entrada de Dados

1. Faça um programa que peça dois números inteiros e imprima a sua soma na tela.

2. Escreva um programa que leia um valor em metros e o exiba convertido em milímetros.

3. Escreva um programa que leia a quantidade de dias, horas, minutos e segundos e imprima o total equivalente em segundos.

4. Escreva um programa que calcule o aumento de um salário. O programa deve solicitar o valor do salário e a porcentagem do aumento.

5. Escreva um programa que solicite o preço de uma mercadoria e o porcentual de desconto e exiba o valor do desconto e o preço a pagar.

6. Escreva um programa que calcule o tempo de uma viagem de carro. Pergunte a distância a percorrer e a velocidade média esperada para a viagem.

7. Escreva um programa que converta uma temperatura digitada em $^{\circ}C$ em $^{\circ}F$. A fórmula que realiza a conversão é

$$F = \frac{9 \times C}{5} + 32.$$

8. Escreva um programa que pergunte a quantidade de quilômetros percorridos por um carro alugado pelo usuário, assim como a quantidade de dias pelos quais o carro foi alugado. Calcule o preço a pagar, sabendo que o aluguel custa R$ 60,00 por dia e R$ 0,15 por quilômetro rodado.

9. Escreva um programa para calcular a redução do tempo de vida de um fumante. Pergunte a quantidade de cigarros fumados por dia e há quantos anos ele é fumante. Considere que um fumante perde 10 minutos de vida a cada cigarro fumado. Calcule quantos dias de vida o fumante perderá e exiba na tela.

3 Condições

Nem sempre todas as linhas dos programas serão executadas. Em muitas situações será mais interessante *decidir* que partes do programa devem ser executadas com base em alguma(s) *condição*(*ões*).

3.1 `if`

As condições servem para *selecionar* quando uma parte do programa deve ser *ativada* e quando deve ser simplesmente *ignorada*. Em Python, a *estrutura de decisão* é o `if`. O seu formato é apresentado no Programa 3.

```
1 if Condição:
2     primeiro comando dentro do if
3     segundo comando dentro do if
4 primeiro comando fora do if
```

Programa 3: Formato do `if`.

Note que o *bloco de código* que contém os comandos que ficam dentro do `if` é delimitado pela edentação, assim como vimos para o `while`. O `if` corresponde ao `se` em português e pode, portanto, ser entendido da seguinte forma: **se** a condição for verdadeira, **faça** alguma coisa (os comandos que estão dentro do `if`).

O Programa 4 apresenta um exemplo de utilização do `if`.

```
1 # -*- coding: cp1252 -*-
2 a = int(raw_input('Digite o primeiro número: '))
3 b = int(raw_input('Digite o segundo número: '))
4 if a > b:
5     print('O primeiro é maior!')
6 if b > a:
7     print('O segundo é maior')
```

Programa 4: Exemplo de de utilização do `if`.

Na linha 4 há a condição `a > b`. Essa expressão será avaliada e, caso seja verdadeira, a linha 5 será executada. No caso de a condição ser falsa, a linha 5 será ignorada. O mesmo ocorre para a linha 6, que contém a condição `b > a`: se o resultado for verdadeiro, a linha 7 será executada e, se for falso, será ignorada. **Resumindo**: se a condição `a > b` for verdadeira, serão executadas as linhas 2, 3, 4, 5 e 6; se a condição `b > a` for verdadeira, serão executadas as linhas 2, 3, 4, 6 e 7. Note que as linhas com as condições são executadas mesmo se o resultado for falso, pois o teste precisa ser realizado.

Pergunta: o que acontece se `a` = `b`? Teste o programa para esse caso. O resultado foi coerente? O Programa 5 é mais *apropriado* do que o Programa 4?

```
1 # -*- coding: cp1252 -*-
2 a = int(raw_input('Digite o primeiro número: '))
3 b = int(raw_input('Digite o segundo número: '))

5 if a > b:
6     print 'O primeiro é maior!'
7 if b > a:
```

```
8       print 'O segundo é maior'
9   if b = a:
10      print 'Os numeros são iguais'
```

Programa 5: Repensando o Programa 4.

Exercício: escreva um programa que pergunte a velocidade do carro de um usuário. Caso a velocidade ultrapasse 80 km/h, exiba uma mensagem dizendo que o usuário foi multado. Nesse caso, exiba o valor da multa, cobrando R$ 5,00 por km acima de 80.

Um problema comum é o de pagamento de imposto de renda. Normalmente, paga-se o imposto por *faixa de rendimento* (salário). Imagine que para salários menores do que R$ 1.000,00, a alíquota é de 0%, para salários entre R$ 1.000,00 e R$ 3.000,00 a alíquota é de 20% e, para salários superiores a R$ 3.000,00, a alíquota é de 35%. Note que o imposto é cobrado de forma diferente para cada faixa: quem ganha R$ 4.000,00 tem os primeiros R$ 1.000,00 isentos de imposto, com o montante entre R$ 1.000,00 e R$ 3.000,00 pagando 20% e o restante pagando 35%. O Programa 6 apresenta a solução para esse problema.

```
1  # -*- coding: cp1252 -*-
2  salario = float(raw_input("Digite o salário para cálculo do
       imposto: "))
3  base = salario
4  imposto = 0
5  if base > 3000:
6       imposto = imposto + ((base - 3000) * 0.35)
7       base = 3000
8  if base > 1000:
9       imposto = imposto + ((base - 1000) * 0.20)
10 print "Salário: R$%6.2f Imposto a pagar: R$%6.2f" % (salario,
       imposto)
```

Programa 6: Cálculo de Imposto de Renda.

Utilizando o `Online Python Tutor`, disponível no endereço

http://pythontutor.com/visualize.html,

analise o funcionamento do Programa 6 para salários iguais a R$ 500,00, R$ 1.500,00, R$ 3.000,00 e R$ 5.000,00. Verifique *cuidadosamente* os valores das variáveis `salario`, `base` e `imposto`. Essa análise pode ser feita com o *`Debbuger`* do `IDLE`, mas o `Online Python Tutor` é mais didático.

3.1.1 Exercícios – `if`

1. Escreva um programa que leia três números e imprima o maior e o menor.

2. Escreva um programa que pergunte o salário do funcionário e calcule o valor do aumento. Para salários superiores R$ 1.250,00, calcule um aumento de 10%. Para os inferiores ou iguais, de 15%. O programa deve imprimir na tela os valores do salário e do aumento.

4 `else`

Analise *cuidadosamente* o Programa 7, que diz se um carro é novo ou velho em função da sua idade.

```
1 # -*- coding: cp1252 -*-
2 idade = int(raw_input("Digite a idade do carro: "))
3 if idade <= 3:
4       print "O seu carro é novo!"
5 if idade > 3:
6       print "O seu carro é velho!"
```

Programa 7: Carro Novo ou Velho, Dependendo da Idade.

Note que as condições das linhas 3 (`idade <= 3`) e 5 (`idade > 3`) são *mutuamente excludentes*, ou seja, quando uma é verdadeira, a outra é falsa. Isso faz com que a segunda condição seja *desnecessariamente* testada no caso em que a primeira é verdadeira. Em casos como esse, pode ser aplicada a *cláusula* `else`, cuja utilização está demonstrada no Programa 8.

```
1 # -*- coding: cp1252 -*-
2 idade = int(raw_input("Digite a idade do carro: "))
3 if idade <= 3:
4       print "O seu carro é novo!"
5 else:
6       print "O seu carro é velho!"
```

Programa 8: Carro Novo ou Velho, Dependendo da Idade (Utilizando `else`).

Note que após o `else` há `":"`, pois o `else`, assim como o `if` e o `while`, inicia um *bloco de código*, cujo fim é delimitado pela edentação. Cuide para que o `else` seja iniciado na mesma coluna do `if`, pois, caso contrário, o interpretador indicará um *erro*. A utilização do `else` deixa os programas *mais claros* e evita que sejam realizados *testes desnecessários*.

4.1 Exercícios – `else`

1. Escreva um programa que pergunte a distância que um passageiro deseja percorrer em km. Calcule o preço da passagem, cobrando R$ 0,50 por km para viagens de até 200 km e R$ 0,45 para viagens mais longas.

5 Estruturas Aninhadas

Aninhar quer dizer utilizar uma estrutura – um `if` –, por exemplo, dentro de outra, que pode ser um `if`, um `else` ou *estruturas de repetição*, como `for` e `while`.

O Programa 9 mostra como calcular o valor a ser pago por um cliente da empresa de telefonia celular *Tchau*. Essa empresa oferece os seguintes planos, que variam o preço por minuto em função da quantidade de minutos utilizados:

- Abaixo de 200 minutos – R$ 0,20 por minuto;
- Entre 200 e 400 minutos – R$ 0,18 por minuto; e
- Acima de 400 minutos – R$ 0,15 por minuto.

```
1  # -*- coding: cp1252 -*-
2  minutos = int(raw_input("Quantos minutos você utilizou neste
       mês: "))
3  if minutos < 200:
4       preco = 0.20
5  else:
6       if minutos < 400:
7            preco = 0.18
8       else:
9            preco = 0.15
10 print "Você pagará neste mês %6.2f" % (minutos*preco)
```

Programa 9: Conta de Telefone com Três Faixas de Preço.

Tal como você fez para o Programa 6, utilize o `Online Python Tutor` para verificar o funcionamento do Programa 9 para diferentes valores de minutos utilizados.

A utilização de estruturas `if\else` pode deixar o código pouco legível, como no Programa 10, que lê a categoria de um produto e determina o preço de acordo com a Tabela 1.

```
# -*- coding: cp1252 -*-
Categoria = int(raw_input('Digite a categoria: '))
if Categoria == 1:
    Preco = 10
else:
    if Categoria == 2:
        Preco = 18
    else:
        if Categoria == 3:
            Preco = 23
        else:
            if Categoria == 4:
                Preco = 26
            else:
                if Categoria == 5:
                    Preco = 31
                else:
                    print('Categoria inválida! Digite um
                       valor entre 1 e 5!')
                    Preco = 0
print('O preco do produto é: R$%6.2f' % Preco)
```

Programa 10: Categoria × Preço.

Tabela 1: Categorias de Produto e Preço

Categoria	Preço (R$)
1	10,00
2	18,00
3	23,00
4	26,00
5	31,00

Note que o alinhamento se tornou um problema, em função da endentação necessária para cada `else`. Um outro aspecto importante do do Programa 10 é a introdução da *validação da entrada*. Essa estratégia *não é muito prática e tampouco elegante*, mas faz com que o usuário receba uma mensagem caso digite um valor inválido.

A Tabela 2 apresenta quais linhas são executadas para algumas categorias. **Complete** essa tabela, inserindo as linhas que serão executadas para as demais categorias.

Tabela 2: Linhas do Programa 10 que Serão Executadas para Cada Categoria

Categoria	Linhas Executadas
1	2, 3, 4, 20
2	
3	
4	
5	
outras	2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17, 18, 19, 20

6 `elif`

No Programa 10, a utilização de múltiplos `ifs` aninhados fez com que o código fique pouco legível. Em Python, a *cláusula* `elif` substitui um par `else\if`, mas *sem a necessidade de ser criado outro nível na estrutura* (*de edentação*), tornando o código mais legível. O Programa 11 corresponde ao Programa 10 reescrito, utilizando `elif` para substituir os pares `else\if`.

```
1  Categoria = int(raw_input('Digite a categoria: '))
2
3  if Categoria == 1:
4      Preco = 10
5  elif Categoria == 2:
6      Preco = 18
7  elif Categoria == 3:
8      Preco = 23
9  elif Categoria == 4:
10     Preco = 26
11 elif Categoria == 5:
12     Preco = 31
13 else:
14     print('Categoria invalida! Digite um valor entre 1 e 5!')
15 print('O preco do produto e: R$%6.2f' % Preco)
```

Programa 11: Categoria × Preço Utilizando `elif`.

7 Revisitando os Laços: `break` e `continue`

Imagine que você deseja fazer um programa para somar números digitados pelo usuário. A entrada de dados será realizada até que o número zero seja

digitado. Esse problema envolve uma repetição que não pode ser implementada por meio de um `for`, pois não se sabe *a priori* quantas iterações serão realizadas. Esse é um exemplo de problemas em que a habilidade de encerrar uma repetição dentro do bloco a ser repetido é interessante.

A instrução `break` é utilizada para interromper a execução de uma repetição. No caso de um `while`, a repetição é encerrada de forma independente do teste que é feito no início da estrutura. No caso de um `for`, a repetição é interrompida independente de todos os elementos da lista utilizada na estrutura terem sido visitados.

O Programa 12 apresenta a solução para o problema apresentado no início desta seção, utilizando a instrução `break`.

```
1 # -*- coding: cp1252 -*-
2 Soma = 0
3 while True: # A condição é sempre verdadeira
4     Num = float(raw_input("Digite um número a ser somado ou 0
          para sair: "))
5     if Num == 0:
6         break
7     Soma += Num
8 print "O valor da soma é %6.2f" % Soma
```

Programa 12: Interrompendo um `while` com um `break`.

Agora considere que o problema que foi resolvido por meio do Programa 12 seja modificado, de modo que o desejado seja ler 10 números, mas encerrando a leitura caso seja digitado o número 0. Isso pode ser feito por meio de um `for` que será interrompido por um `break`, no caso de ser digitado o número 0. Essa abordagem está apresentada no Programa 13.

```
1 # -*- coding: cp1252 -*-
2 Soma = 0
3 for i in range(10): # 10 iterações...
4     Num = float(raw_input("Digite um número a ser somado ou 0
          para sair: "))
5     if Num == 0:
6         break
7     Soma += Num
8 print "O valor da soma é %6.2f" % Soma
```

Programa 13: Interrompendo um `for` com um `break`.

A instrução `continue` é similar ao `break`, mas, em lugar de interromper a repetição, indica que todas as linhas, a partir da instrução, até o fim da

repetição, devem ser ignoradas, ou seja, a repetição deve ser levada para a próxima iteração.

8 Exercícios

1. Escreva um programa que leia dois números e que pergunte qual operação o usuário deseja realizar. O programa deve permitir que seja calculada a soma, subtração, multiplicação e divisão entre o primeiro e o segundo número lidos. Teste o seu programa para caso em que o segundo número digitado é 0.

2. Escreva um programa para aprovar o empréstimo bancário para a compra de uma casa. O programa deve perguntar o valor da casa a ser comprada, o salário do comprador e a quantidade de anos do financiamento. O valor da prestação mensal não pode ser superior a 30% do salário. Calcule o valor da prestação como sendo o valor da casa a ser comprada dividido pelo número de meses do financiamento.

3. Escreva um programa que calcule o preço a pagar pelo fornecimento de energia elétrica. Pergune a quantidade de kWh consumida e o tipo de instalação, que pode ser `R` para residência, `I` para indústria e `C` para comércio. Calcule o preço de acordo com a Tabela 3.

Tabela 3: Preço por Tipo e Faixa de Consumo

Tipo	Faixa (kWh)	Preço (R$)
R	Até 500	0,40
	Acima de 500	0,65
C	Até 1.000	0,55
	Acima de 1.000	0,60
I	Até 5.000	0,55
	Acima de 5.000	0,60

4. Modifique o Progama 12 de modo que ele apresente também quantidade de números digitados e a média aritmética.

5. Escreva um programa para controlar uma pequena máquina registradora. Você deve solicitar ao usuário que digite o código do produto e a

quantidade comprada. A Tabela 4 apresenta os códigos e preços. Seu programa deve exibir o total das compras depois que o usuário digitar 0. Quando o usuário fornecer um código inválido, deve ser apresentada a mensagem de erro "Código Inválido".

Tabela 4: Códigos e Preços

Código	Preço (R$)
1	0,50
2	1,00
3	4,00
5	7,00
9	8,00