



Engenharia de Software

Computacional thinking with Python

Prof. Dr. Francisco Elânio

Objetivo

- **Condições: estruturas condicionais (if, elif, else)**
- **Verificação de números ou caracteres em uma lista in, not in.**
- **Estrutura de controle aninhada (condição dentro de uma condição).**
- **Estrutura de controle de fluxo (for, break, continue).**
- **If dentro de um for**
- **Exercícios**

Condições - 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.

```
if <condição>:  
    bloco verdadeiro
```

```
a = int(input( "Primeiro valor: "))
```

```
b = int(input( "Segundo valor: "))
```

```
if a > b:
```

```
    print( "O primeiro valor é o Maior!")
```

```
if b > a:
```

```
    print("O segundo valor é o Maior!")
```

Algoritmo em Python

Verificar

Lista (variável)

```
cobertura_requisitada = ['frango', 'queijo_extra', 'pepperoni']
```

Primeira condição

```
if 'frango' in cobertura_requisitada:
```

```
    print("Adicione frango.")
```

Primeira condição

```
if 'pepperoni' in cobertura_requisitada:
```

```
    print("Adicione pepperoni.")
```

Primeira condição

```
if 'queijo_extra' in cobertura_requisitada:
```

```
    print("Adicione queijo_extra.")
```

```
    print("\nSua pizza esta pronta!")
```

Três condições if

São executados independentemente do teste anterior ter sido aprovado ou não.

Comparações múltiplas com operadores lógicos

Para fazer múltiplas comparações, devemos criar múltiplas condições, usando operadores de comparação.

```
valor = int(input("Insira valores entre 1 e 10: "))
```

```
if valor > 0 and valor <=10:  
    print("Você digitou o número", valor)
```

Condições – if/elif

Caso a condição if não seja atendida, é possível criar uma nova condição usando elif.

if <condição>:

bloco verdadeiro

elif <condição>:

bloco falso

```
a = int(input( "Primeiro valor: "))
```

```
b = int(input( "Segundo valor: "))
```

```
c = int(input( "Terceiro valor: "))
```

```
if a > b:
```

```
    print( "A é maior que B." )
```

```
elif c > a:
```

```
    print("C é maior que A.")
```

Algoritmo em Python

Verificar se carro é novo ou velho

Inserir idade do carro

```
idade = float(input("Insira a idade do seu carro"))
```

Condição para verificar idade do carro

```
if idade <= 3:
```

```
    print("Seu carro é novo")
```

```
elif idade > 3:
```

```
    print("Seu carro é velho")
```

If e elif

- Caso if seja atendido, retorna um resultado.
- Caso if não seja atendido, uma nova condição pode ser criada, neste caso, elif.

Condições – if/elif/else

Caso a condição if e elif não sejam atendidas, usamos else.

if <condição>:

condição principal

elif <condição>:

condição secundário

else:

caso primeira e segunda

condição não sejam atendidas

```
a = int(input( "Primeiro valor: "))
```

```
b = int(input( "Segundo valor: "))
```

```
c = int(input( "Terceiro valor: "))
```

```
if a > b:
```

```
    print( "A é maior que B.")
```

```
elif c > a:
```

```
    print("C é maior que A.")
```

```
else:
```

```
    print("B é maior que A ou B é  
maior que C.")
```


Descrição do problema

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

Algoritmo em Python

Calcular valor de multa em função da velocidade

```
# Pergunta a velocidade do carro
velocidade = float(input("Qual é a velocidade do carro em km/h? "))

# Verifica se ultrapassou 80 km/h
if velocidade > 80:
    # Calcula a multa
    multa = (velocidade - 80) * 5
    print(f"Você foi multado! A multa é de R$ {multa:.2f}.")
else:
    print("Velocidade dentro do limite permitido.")
```

Variável

Cálculo da multa em
função do aumento
da velocidade

Comparações múltiplas com operadores lógicos

```
dia_semana = input("Digite o dia da semana: ")  
dia_mes = int(input("Digite o dia do mês: "))
```

```
if 5 <= dia_mes <= 10 and (dia_semana.lower() == "sabado" or dia_semana.lower() == "domingo):  
    print("Espere um dia útil para receber o pagamento.")  
else:  
    print("Receba o salário.")
```

Função lower() e upper()

upper() - converte um texto para letras maiúsculas
lower() - converte um texto para letras minúsculas

Exercícios

Comparações múltiplas com operadores lógicos

Desenvolva um algoritmo para calcular área de um retângulo, quadrado e círculo.

- O usuário deve escolher qual objeto será calculado a área.
- Para cada escolha do usuário deverá ser apresentado as informações para calcular a área.

Cálculo da área em função do tipo de área a ser calculada

```
import math
pi = math.pi
area_desejada = ['retangulo', 'quadrado', 'circulo']
escolha = (input("Escolha uma forma (retangulo, quadrado, circulo): ")).lower()

if escolha == 'retangulo':
    lado_1 = float(input("Insira o primeiro lado: "))
    lado_2 = float(input("Insira o segundo lado: "))
    print(f"A área deste retângulo é: {lado_1 * lado_2}", "m2")
elif escolha == 'quadrado':
    lado_1 = float(input("Insira o primeiro lado: "))
    lado_2 = float(input("Insira o segundo lado: "))
    print(f"A área deste quadrado é {lado_1 * lado_2}", "m2")
elif escolha == 'circulo':
    raio = float(input("Insira valor do raio: "))
    area = math.pi * (raio ** 2)
    print(f"A área deste círculo é {area:.2f} m2")
else:
    print("Escolha inválida.")
```

Observação

Foi necessário inserir quatro condições utilizando if, dois elif e else.

Operador in

Verifica se uma determinada letra está na palavra

Solicite ao usuário uma letra e uma palavra

```
letra = input("Digite uma letra: ")
```

```
palavra = input("Digite uma palavra: ")
```

Verifica se a letra está na palavra

```
if letra in palavra:
```

```
    print(f"A letra '{letra}' está na palavra.")
```

```
else:
```

```
    print(f"A letra '{letra}' não está na palavra.")
```

in

Verifica se um valor
está presente em
uma sequência

Operador in

Verifica se a marca de um carro está na lista

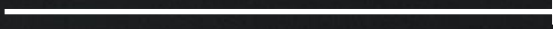
Cria uma lista de carros

```
carros = ['audi', 'bmw', 'subaru', 'toyota']
```

Usuário digita uma marca de carro de sua escolha

```
carro_digitado = (str(input("Digite o nome da marca do carro"))).lower()
```

Condição para verificar se carro digitado está na lista

```
if carro_digitado in carros: 
```

```
    print("O carro escolhido é:", carro_digitado)
```

```
else:
```

```
    print("Carro não está na lista!")
```

in

Verifica se um valor
está presente em
uma sequência

Operador not in

Verifica se uma letra não está na palavra

Solicite ao usuário uma letra e uma palavra

```
letra = input("Digite uma letra: ").lower()
```

```
palavra = input("Digite uma palavra: ").lower()
```

Verifique se a letra não está na palavra

```
if letra not in palavra:
```

```
    print(f"A letra '{letra}' não está na palavra.")
```

```
else:
```

```
    print(f"A letra '{letra}' está na palavra.")
```

not in

Verifica se um valor
está presente em
uma sequência

Operador not in

Verifica se um número não está na lista

Cria uma lista de números

```
numeros = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Usuário digita uma marca de carro de sua escolha

```
num_digitado = int(input("Digite um numero entre 0 e 10"))
```

Condição para verificar se carro digitado está na lista

```
if num_digitado not in numeros: →
```

```
    print("Número digitado:", num_digitado)
```

```
    print("Digite um número da lista:", numeros)
```

```
else:
```

```
    print("Número digitado é", num_digitado)
```

not in

Verifica se um valor
está presente em
uma sequência

Estrutura de controle aninhada ou condicional aninhada

São utilizadas em casos em que é necessário estabelecer verificações de condições sucessivas (condições dentro de condições). Este tipo de estrutura pode possuir diversos níveis de condições aninhadas.

```
if condição_1 == True:
```

```
    if condição_2 == True:
```

```
        print("Resultado para condições verdadeiras")
```

Estrutura de controle aninhada

Número secreto por meio de 2 dígitos

```
primeiro_num = int(input("Digite o primeiro número"))
segundo_num = int(input("Digite o segundo número"))
```

```
if primeiro_num >= 1 and primeiro_num <= 10:
    if segundo_num >= 1 and segundo_num <= 10:
        print("Seu número secreto é:", primeiro_num * segundo_num)
    else:
        print("Segundo número incorreto!")
else:
    print("Primeiro número incorreto!")
```

Diagram illustrating the nested control structure:

- `if primeiro_num >= 1 and primeiro_num <= 10:` → Primeira condição
- `if segundo_num >= 1 and segundo_num <= 10:` → Segunda condição
- `print("Seu número secreto é:", primeiro_num * segundo_num)` (inside the second condition)
- `else:` (inside the first condition) → Caso segunda condição não seja atendida
- `else:` (outside the first condition) → Caso primeira condição não seja atendida

Exercícios

Comparações múltiplas com operadores lógicos

Desenvolva um algoritmo na qual o usuário possa inserir o dia da semana (segunda-feira a domingo).

- Se dia for sexta-feira, imprima dia de festa!.
- Caso seja sábado, usuário deve inserir sua condição de saúde. Caso tenha dor de cabeça, print você deve se recuperar.
- Caso seja sábado e condição da saúde seja saudável, print (Podemos ir para o show do Foo Fighters).
- Caso contrário, descanse.
- Caso não atenda nenhuma condição, print você deve trabalhar.

Estrutura de controle aninhada

Condição em função do dia

```
dia_semana = str(input("Digite um dia da semana"))
```

```
if dia_semana.lower() == 'sexta-feira':  
    print("Dia de festa", "uhuuuuuuuu !!!")
```

→ Condição caso seja sábado

```
elif dia_semana.lower() == 'sabado':  
    cond = str(input("Digite sua condição"))
```

→ Condição caso seja domingo

```
    if cond == 'dor_de_cabeca':  
        print("Você deve se recuperar")
```

→ Caso seja domingo, digite sua condição

```
    else:  
        print("Descanse")
```

→ Caso seja domingo e não tenha dor de cabeça, será printado Descanse

```
else:  
    print("Você precisa trabalhar, trabalhar e trabalhar!")
```

→ Caso não seja sábado ou domingo será printado o segundo else.

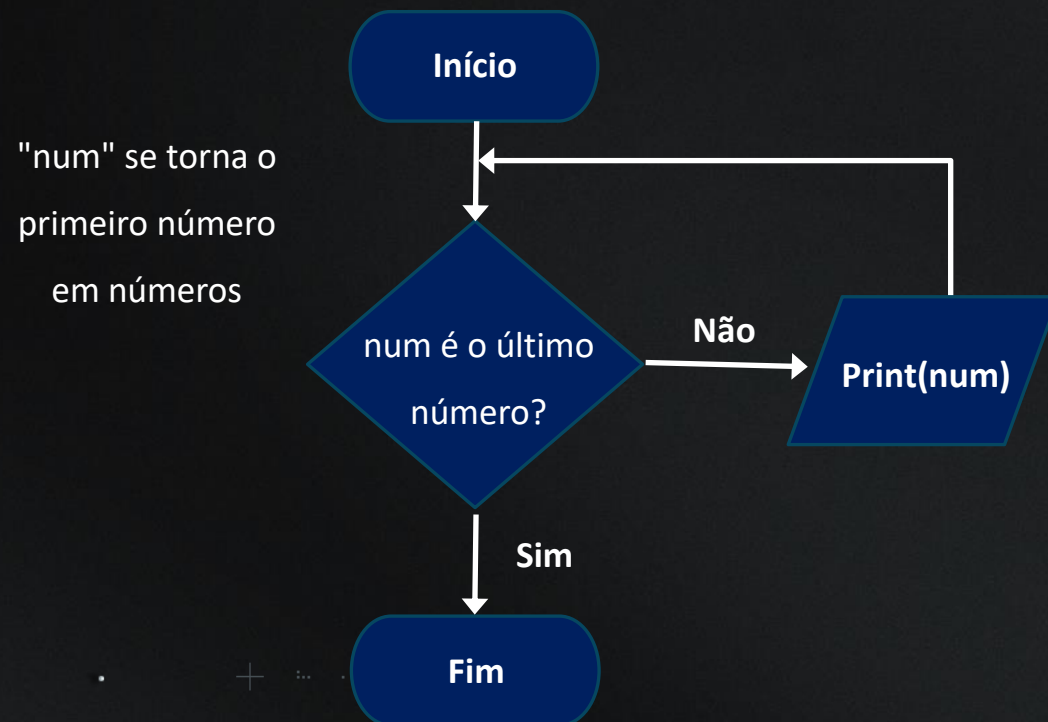
Exercício

Estrutura de controle aninhada

Crie um algoritmo para verificar a elegibilidade para votação com base na idade e na posse de um documento de identificação válido (RG).

Instrução for

É uma estrutura de controle de fluxo que permite percorrer elementos em uma sequência (como uma lista, string, etc.) e realizar operações em cada um deles.



O `print(numero)` é executado enquanto ainda houver números na lista

```
numeros = [1, 2, 3, 4, 5]
```

```
for numero in numeros:  
    print(numero)
```

Instrução for

É uma estrutura de controle de fluxo que permite percorrer elementos em uma sequência (como uma lista, string, etc.) e realizar operações em cada um deles.

```
coberturas_requisitadas = ['pepperoni', 'tomate', 'queijo_extra', 'cebola']
```

```
for cobertura_requisitada in coberturas_requisitadas:
```

```
    print(f"Adicione {cobertura_requisitada}.")
```

```
print("\nSua pizza esta pronta!")
```



for

Percorre a lista coberturas_requisitadas e realiza uma ação para cada elemento na lista.

Instrução for e continue

```
frutas = ["maça", "banana", "melância"]
```

```
for x in frutas:
```

```
    if x == "maça":
```

```
        continue
```

```
    print(x)
```



continue

Com a instrução continue podemos parar a iteração atual do loop e continuar com a próxima

```
frutas = ["maça", "banana", "melância"]
```

```
for x in frutas:
```

```
    print(x)
```

```
    if x == "banana":
```

```
        break
```



break

Com a instrução break podemos parar o loop antes que ele percorra todos os itens

Instrução if dentro de um for

E se a pizzeria ficar sem pepperoni? Uma declaração if dentro do loop for pode lidar com esta situação:

```
coberturas_requisitadas = ['pepperoni', 'tomate', 'queijo_extra', 'cebola']  
for cobertura_requisitada in coberturas_requisitadas:  
    if cobertura_requisitada == 'pepperoni':  
        print("Descule, mas estamos sem pepperoni.")  
    else:  
        print(f"Adicione {cobertura_requisitada}.")  
print("\nSua pizza esta pronta!")
```



for e if

Neste caso, uma condição não pode ser atendida. Então, podemos usar um if dentro do for.

Instrução for dentro de um for

Será feito uma combinação de x (elemento da lista ordem) com y (elemento da lista titulo).

```
ordem = ["first", "second", "third", "fourth"]
```

```
titulo = ["king", "queen", "princess", "Duke"]
```

```
for x in ordem:
```

```
    for y in titulo:
```

```
        print(x, y)
```



For dentro for

A cada iteração dos loops aninhados, a função print(x, y) é chamada.

Instrução if dentro de um for

Exemplo numérico com um loop for e estrutura if

```
numeros = [1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
```

Loop for para percorrer a lista de números

```
for numero in numeros:
```

```
    if numero % 2 == 0:
```

```
        print(f"{numero} é um número par.")
```

```
    else:
```

```
        print(f"{numero} é um número ímpar.")
```



Número Par ou Ímpar

"if" é utilizado para verificar se o número atual (numero) é divisível por 2 (se é par).

Exercício

Encontrando um Carro no Estacionamento

Descrição: Um estacionamento possui vários carros de diferentes marcas estacionados. O objetivo é criar um programa Python que ajude um cliente a encontrar seu carro desejado dentro do estacionamento.

Lista de carros: Toyota, Honda, Ford, Nissan, Chevrolet, Hyundai, Volkswagen

Utilize as funções for, if, break, in, is not, entre outras apresentadas na aula para resolver este problema.

*O importante é não parar
de questionar (Einstein)*

Referências

ASCENCIO, A. F. G, CAMPOS, E. A. V. Fundamentos da Programação de Computadores: algoritmos, Pascal, C/C++ e Java, 2ª Edição, São Paulo: Pearson 2007.

SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madson. Algoritmos. São Paulo: Pearson, 2004.