Teste de Software para Web

Revisão de Lógica de Programação

MSc. Jonathas Silva dos Santos

Roteiro

Comandos de Repetição

Comando `while`

Listas

Comando `for`

Comandos `break` e `continue`

Laços Aninhados

Exercícios

Comandos de Repetição

Comandos de Repetição

- Até agora, vimos como escrever programas capazes de executar comandos de forma sequencial e, se necessário, tomar decisões com relação a executar ou não um bloco de comandos.
- Entretanto, muitas vezes é necessário executar um bloco de comandos várias vezes para obter o resultado desejado.

Exemplos

- Imprimindo todos os números inteiros de 1 até 5.

```
1 print(1)
2 print(2)
3 print(3)
4 print(4)
5 print(5)
```

Exemplos

- Imprimindo todos os números inteiros de 1 até 100.

```
print(1)
print(2)
print(3)
...
print(100)
```

Exemplos

- Imprimindo todos os números inteiros de 1 até n.

```
n = int(input("Digite um número: "))
if n >= 1:
  print(1)
if n >= 2:
  print(2)
if n >= 3:
  print(3)
if n >= 4:
  print(4)
...
if n >= 100:
  print(100)
```

- Note que só resolvemos o problema para $n \leq 100$.

Comando `while`

Comando `while`

- O primeiro comando de repetição que aprenderemos é o `while`.

```
while <condição>:
# este bloco irá se repetir até a condição ser falsa
  <comando1>
  <comando2>
  ...
  <comandoY>
```

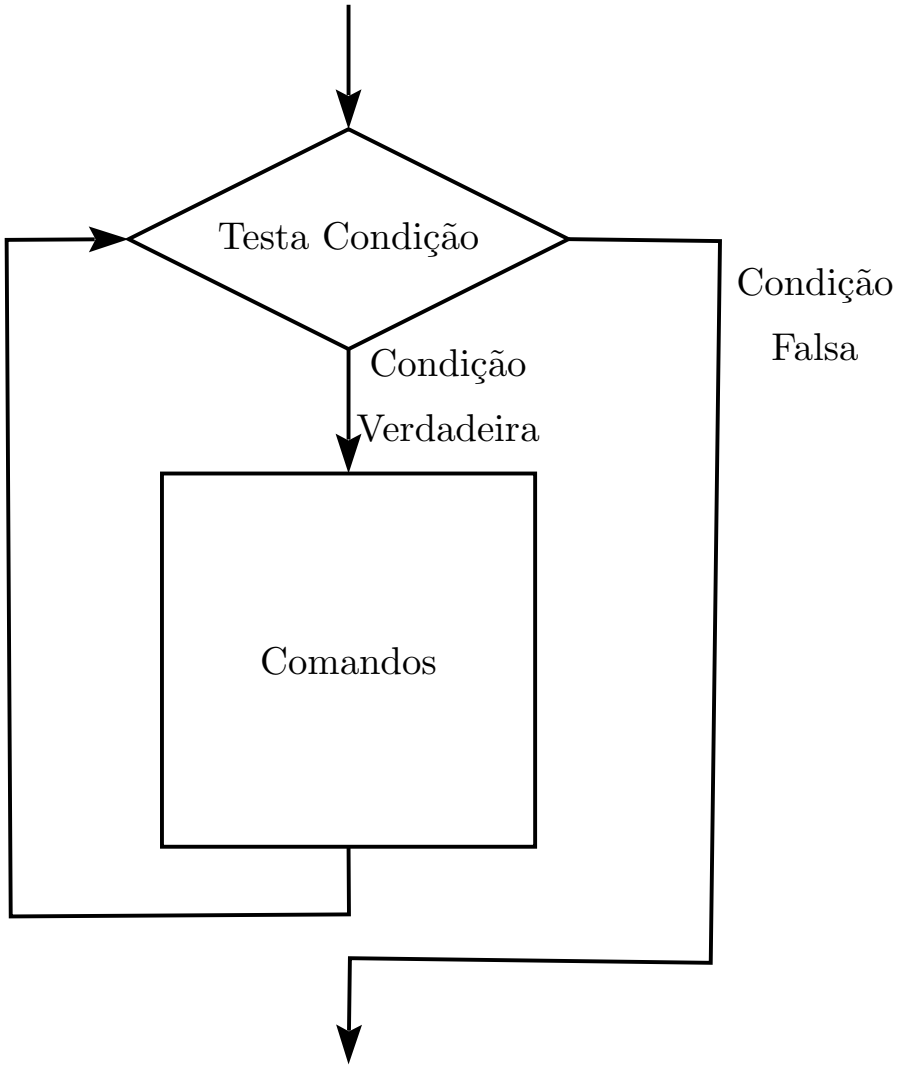
- Funcionamento:

Passo 1: Se a condição for verdadeira, vá para o Passo 2.
Caso contrário, encerre o bloco de repetição (**`while`**) e prossiga para o próximo comando (conforme o diagrama do próximo slide).

Passo 2: Execute o bloco de comandos.

Passo 3: Volte para o Passo 1.

Comando `while`



Exemplo do Comando `while`

- Imprimindo todos os números inteiros de 1 até 100.

```
i = 1
while i <= 100:
  print(i)
  i = i + 1
print("Fim do programa!")
```

- Imprimindo todos os números inteiros de 1 até n.

```
n = int(input("Digite um número inteiro positivo: "))
i = 1
while i <= n:
  print(i)
  i = i + 1
print("Fim do programa!")
```

Exemplo do Comando `while`

- Contagem regressiva:

```
1 n = int(input("Digite um número inteiro positivo: "))
2
3 while n >= 0:
4   print(n)
5   n = n - 1
6
7 print("BOOM!")
```

Exemplo do Comando `while`

- Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

```
1  dividendo = int(input("Entre com o dividendo: "))
2  divisor = int(input("Entre com o divisor: "))
3
4  quociente = 0
5
6  while dividendo >= divisor:
7    dividendo = dividendo - divisor
8    quociente = quociente + 1
9
10 print("Quociente:", quociente)
11 print("Resto:", dividendo)
```

Exponenciação

- Como calcular b^e (para $b \in \mathbb{R}$, $e \in \mathbb{N}$)?
- Para $e \geq 1$, usando $e - 1$ multiplicações:
  - $b^e = b \times b \times \ldots \times b$
- Para $e \geq 0$, usando e multiplicações:
  - $b^e = 1 \times b \times b \times \ldots \times b$

Exemplo do Comando `while`

- Dado um número real (base) e um número inteiro positivo (expoente), calcule $base^{expoente}$, usando apenas somas e multiplicações.

```
base = float(input("Base: "))
exp = int(input("Expoente: "))

result = base
i = 1

while i < exp:
  result = result * base
  i = i + 1

print(result)
```

Exemplo do Comando `while`

- Dado um número real (base) e um número inteiro não negativo (expoente), calcule $base^{expoente}$, usando apenas somas e multiplicações.

```
base = float(input("Base: "))
exp = int(input("Expoente: "))

result = 1
i = 1

while i <= exp:
  result = result * base
  i = i + 1

print(result)
```

Comando `while`

O que acontece se a condição for falsa na primeira vez?

```
1 a = 0
2 while a > 0:
3   a = a + 1
4   print(a)
```

- Resposta: o programa nunca entra no bloco de repetição.

O que acontece se a condição for sempre verdadeira?

```
1 a = 0
2 while a >= 0:
3   a = a + 1
4   print(a)
```

- Resposta: o programa entra no bloco e nunca sai (*loop infinito*).

Exponenciação

- Como calcular b^e (para $b \in \mathbb{R}$, $e \in \mathbb{Z}$)?
- Para $e \geq 0$, usando e multiplicações:
  - $b^e = 1 \times b \times b \times \ldots \times b$
- Para $e < 0$, usando $|e|$ divisões:
  - $b^e = (((1/b)/b)/\ldots)/b$

Exemplo do Comando `while`

- Dado um número real (base) e um número inteiro (expoente), calcule $base^{expoente}$, usando apenas somas e multiplicações.

```
base = float(input("Base: "))
exp = int(input("Expoente: "))

result = 1

while exp > 0:
  result = result * base
  exp = exp - 1

while exp < 0:
  result = result / base
  exp = exp + 1

print(result)
```

Exemplo do Comando `while`

- Dado um número real (base) e um número inteiro (expoente), calcule $base^{expoente}$, usando apenas somas e multiplicações.

```
1  base = float(input("Base: "))
2  exp = int(input("Expoente: "))
3
4  result = 1
5
6  while exp < 0:
7    result = result / base
8    exp = exp + 1
9
10 while exp > 0:
11   result = result * base
12   exp = exp - 1
13
14 print(result)
```

Exemplo do Comando `while`

- Determinando a quantidade de números inteiros positivos fornecidos para um programa (até a leitura de um inteiro não positivo).

```
1  n = 0
2  OK = True
3
4  while OK:
5    x = int(input("Entre com um número inteiro positivo: "))
6    if x > 0:
7      n = n + 1
8    else:
9      OK = False
10
11 print("Quantidade de números positivos fornecidos:", n)
```

Exemplo do Comando `while`

- Dado um número inteiro positivo *n*, calcular o valor de *n*!.

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3 i = 1
4
5 while i <= n:
6   fat = fat * i
7   i = i + 1
8
9 print("Resultado:", fat)
```

Exemplo do Comando `while`

- Dado um número inteiro positivo *n*, calcular o valor de *n*!.

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3 i = 2
4
5 while i <= n:
6   fat = fat * i
7   i = i + 1
8
9 print("Resultado:", fat)
```

Exemplo do Comando `while`

- Dado um número inteiro positivo *n*, calcular o valor de *n*!.

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3
4
5 while n > 0:
6   fat = fat * n
7   n = n - 1
8
9 print("Resultado:", fat)
```

Exemplo do Comando `while`

- Dado um número inteiro positivo *n*, calcular o valor de *n*!.

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3
4
5 while n > 1:
6   fat = fat * n
7   n = n - 1
8
9 print("Resultado:", fat)
```

Listas

Listas

- Uma lista é uma estrutura de Python que armazena múltiplos dados.

```
lista = [<dado_1>, <dado_2>, <dado_3>, ..., <dado_n>]
```

- Podemos ter todos os dados do mesmo tipo.

```
# Uma lista com dados do tipo int
lista_de_int = [40, 3, 61, 7, 3]

# Uma lista com dados do tipo bool
lista_de_bool = [True, False, True]
```

- Podemos ter dados de tipos diferentes misturados.

```
lista_mista = ["Gato", 42, True, 5.4, "Cachorro", 73]
```

Listas

- Podemos acessar um elemento em uma lista indicando a sua posição (o primeiro elemento fica na posição 0 da lista).

```
lista = ["Azul", 51, "Amarelo", 55, True, 7.2]
print(lista[0])  # imprime o primeiro elemento da lista
# Azul
print(lista[1])  # imprime o segundo elemento da lista
# 51
print(lista[2])  # imprime o terceiro elemento da lista
# Amarelo
print(lista[-1]) # imprime o último elemento da lista
# 7.2
```

- A função `len()` retorna o número de elementos de uma lista.

```
print(len(lista))
# 6
```

`while` com Listas

- Podemos usar um **`while`** para percorrer todos os elementos de uma lista.

```
1 letras = ["A", "B", "C", "D", "E", "F", "G"]
2 i = 0
3 while i < len(letras):
4   print(letras[i])
5   i = i + 1
```

Exemplo do Comando `while` com Listas

- Encontrando o máximo de um conjunto de números positivos.

```
numeros = [3, 1, 7, 9, 4]
maximo = 0

i = 0
while i < len(numeros):
  if numeros[i] > maximo:
    maximo = numeros[i]
  i = i + 1

print(maximo) # 9
```

Exemplo do Comando `while` com Listas

- Encontrando o máximo de um conjunto de números quaisquer.

```
numeros = [-3, -1, -7, -9, -4]
maximo = numeros[0]

i = 1
while i < len(numeros):
  if numeros[i] > maximo:
    maximo = numeros[i]
  i = i + 1

print(maximo) # -1
```

Comando `for`

Comando `for`

- Podemos percorrer uma lista de forma mais compacta com o comando `for`.

```
for <variavel> in <lista>:
# este bloco irá repetir para todos os valores da lista
  <comando1>
  <comando2>
  ...
  <comandoY>
```

Comando `for`

- Estes dois códigos são equivalentes.

```
letras = ["A", "B", "C", "D", "E", "F", "G"]
i = 0
while i < len(letras):
  print(letras[i])
  i = i + 1
```

```
letras = ["A", "B", "C", "D", "E", "F", "G"]
for letra in letras:
  print(letra)
```

Exemplo do Comando `for`

- Encontrando o máximo de um conjunto de números positivos.

```
numeros = [3, 1, 7, 9, 4]
maximo = 0

for numero in numeros:
  if numero > maximo:
    maximo = numero

print(maximo) # 9
```

Exemplo do Comando `for`

- Encontrando o máximo de um conjunto de números.

```
numeros = [3, 1, 7, 9, 4]
maximo = numeros[0]

for numero in numeros:
  if numero > maximo:
    maximo = numero

print(maximo) # 9
```

Função `range`

- A função `range` gera uma sequência de inteiros.
- Uma sequência de inteiros pode ser transformada numa lista usando a função `list`.

```
print(list(range(2, 6)))
# [2, 3, 4, 5]
```

- A função `range` recebe como argumentos os limites da sequência a ser gerada: o primeiro número é incluído na sequência, mas o último não.

```
print(list(range(1, 5)))
# [1, 2, 3, 4]
print(list(range(0, 4)))
# [0, 1, 2, 3]
print(list(range(0, -5)))
# []
```

Função `range`

- Se a sequência começa em zero, o primeiro número pode ser omitido.

```
print(list(range(4)))
# [0, 1, 2, 3]
```

- A função `range` pode receber um terceiro argumento (opcional), que define o incremento usado na geração da sequência de inteiros.

```
print(list(range(2, 10, 2)))
# [2, 4, 6, 8]
print(list(range(1, 15, 3)))
# [1, 4, 7, 10, 13]
print(list(range(5, 0, -1)))
# [5, 4, 3, 2, 1]
```

Repetindo n vezes

- Note que podemos utilizar `range(n)` em conjunto com o `for` para repetir uma operação n vezes.

```
for i in list(range(100)):
  print("Esta frase será impressa 100 vezes")
```

- No comando `for`, não precisamos explicitamente converter a sequência de inteiros gerada pelo comando `range()` numa lista.

```
n = int(input("Digite um número inteiro positivo: "))
for i in range(n):
  print("Esta frase será impressa", n, "vezes")
```

Exemplo do Comando `for` com `range`

- Imprimindo todos os números inteiros de 1 até 100.

```
for i in range(1, 101):
  print(i)
print("Fim do programa!")
```

- Imprimindo todos os números inteiros de 1 até n.

```
n = int(input("Digite um número inteiro positivo: "))
for i in range(1, n+1):
  print(i)
print("Fim do programa!")
```

Exemplo do Comando `for` com `range`

- Imprimindo as n primeiras potências de 2, sem usar o operador de exponenciação (`**`).

```
1 n = int(input("Digite um número inteiro positivo: "))
2 potencia = 1
3 for i in range(n):
4   potencia = potencia * 2
5   print(potencia)
```

Variável Acumuladora

- Como no exemplo anterior, em vários problemas precisamos combinar dados em uma variável utilizando alguma operação.
- Esse tipo de variável é chamada de acumuladora.
- Exemplo: somando os números inteiros de 1 até n.

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 soma = 0
3 for i in range(1, n+1):
4   soma = soma + i
5 print(soma)
```

Exemplo de Variável Acumuladora

- Calculando n!

```
n = int(input("Digite um número inteiro positivo: "))
fatorial = 1
for i in range(1, n+1):
  fatorial = fatorial * i
print(fatorial)
```

Exemplo de Variável Acumuladora

- Calculando n!

```
n = int(input("Digite um número inteiro positivo: "))
fatorial = 1
for i in range(2, n+1):
  fatorial = fatorial * i
print(fatorial)
```

Exemplo de Variável Acumuladora

- Calculando n!

```
1 n = int(input("Digite um número inteiro positivo: "))
2 fatorial = 1
3 for i in range(n, 1, -1):
4   fatorial = fatorial * i
5 print(fatorial)
```

Comandos `break` e `continue`

Comando `break`

- Vimos que o comando `for` percorre a lista completa.
- Às vezes queremos interromper a execução do comando `for` antes dele percorrer a lista completa.
- O comando `break` faz com que a execução de um laço de repetição seja finalizada, passando a execução para o próximo comando após o laço.

Comando `break`

- Exemplo: procurando um valor em uma lista sem elementos repetidos.

```
x = int(input("Digite um número: "))
for i in [2, 4, 7, 1, 0, 8, 9, 5]:
  print("Verificando elemento", i)
  if i == x:
    print("Elemento", i, "encontrado!")
```

- Após encontrarmos o valor, podemos encerrar a busca.

```
x = int(input("Digite um número: "))
for i in [2, 4, 7, 1, 0, 8, 9, 5]:
  print("Verificando elemento", i)
  if i == x:
    print("Elemento", i, "encontrado!")
    break
```

Comando **break**

- O que será impresso no seguinte código?

```
for i in range(1, 10):
  if i == 5:
    break
  print(i)
print("Fim do programa")
```

- Resposta:

```
1
2
3
4
Fim do programa
```

Comando break

- Determinando a quantidade de números inteiros positivos fornecidos para um programa (até a leitura de um inteiro não positivo).

```
n = 0

while True:
  x = int(input("Entre com um número inteiro positivo: "))
  if x <= 0:
    break
  n = n + 1

print("Quantidade de números positivos fornecidos: ", n)
```

O Comando `break` e o Comando `else`

- Podemos usar o comando **`else`** para executar um bloco de comandos apenas caso o comando **`for`** ou comando **`while`** tenham sido executados sem interrupção (**`break`**).
- Encontrando um elemento x em uma lista.

```
x = int(input("Digite um número: "))
for i in [2, 4, 7, 1, 0, 8, 9, 5]:
  if i == x:
    print("Elemento", i, "encontrado!")
    break
else:
  print("O elemento", x, "não está na lista.")
```

- Quando o **`break`** é executado, o bloco **`else`** é ignorado.

Comando `continue`

- O comando **`continue`** faz com que a execução atual do bloco de comandos do laço de repetição seja finalizada, passando a execução para a próxima iteração do laço.
- O que será impresso no seguinte código?

```
for i in range(1, 6):
  if i == 3:
    continue
  print(i)
print("Fim do programa")
```

- Resposta:

```
1
2
4
5
Fim do programa
```

Exemplo do Comando `continue`

- Estes dois códigos são equivalentes: imprimem apenas os elementos pares da lista.

```
for i in [2, 4, 7, 1, 0, 8, 9, 5]:
  if i % 2 == 0:
    print(i)
```

```
for i in [2, 4, 7, 1, 0, 8, 9, 5]:
  if i % 2 == 1:
    continue
  print(i)
```

Laços Aninhados

Laços Aninhados

- Em muitas situações é necessário implementar um laço (bloco de repetição) dentro de outro laço.
- Estes blocos de comandos são conhecidos como laços aninhados.

Exemplo de Laços Aninhados

- Imprimindo as tabuadas dos números de 1 a 10.

```
1 for i in range(1, 11):
2   print("Tabuada do ", i, ":", sep = "")
3   for j in range(1, 11):
4     print(i, "x", j, "=", i * j)
```

Exemplo de Laços Aninhados

- Imprimindo um retângulo com n linhas e m colunas (n $\times$ m caracteres #).

```
n = int(input("Entre com o número de linhas: "))
m = int(input("Entre com o número de colunas: "))

for i in range(n):
  for j in range(m):
    print("#", end = "")
  print()
```

Exemplo de Laços Aninhados

- Imprimindo um retângulo com n linhas e m colunas (n $\times$ m caracteres #).

```
1 n = int(input("Entre com o número de linhas: "))
2 m = int(input("Entre com o número de colunas: "))
3
4 for i in range(n):
5   print("#" * m)
6
7
```

Exemplo de Laços Aninhados

- Imprimindo um triângulo retângulo isósceles, com catetos de tamanho dado.

```
n = int(input("Entre com o tamanho dos catetos: "))
c = input("Entre com caractere a ser usado: ")

for i in range(1, n+1):
  for j in range(i):
    print(c, end = "")
  print()
```

Exemplo de Laços Aninhados

- Imprimindo um triângulo retângulo isósceles, com catetos de tamanho dado.

```
1 n = int(input("Entre com o tamanho dos catetos: "))
2 c = input("Entre com caractere a ser usado: ")
3
4 for i in range(1, n+1):
5   print(c * i)
6
7
```

Exemplo de Laços Aninhados

- Imprimindo todos os horários de um dia, no formato HH:MM (horas e minutos).

```
for h in range(24):
  for m in range(60):
    print(h, m, sep = ":")

```

Exemplo de Laços Aninhados

- Imprimindo todos os horários de um dia, no formato HH:MM (horas e minutos).

```
for h in range(24):
  for m in range(60):
    print('{:02d}:{:02d}'.format(h, m))

```

Exemplo de Laços Aninhados

- Imprimindo todos os horários de um dia, no formato HH:MM:SS (horas, minutos e segundos).

```
for h in range(24):
  for m in range(60):
    for s in range(60):
      print('{:02d}:{:02d}:{:02d}'.format(h, m, s))
```