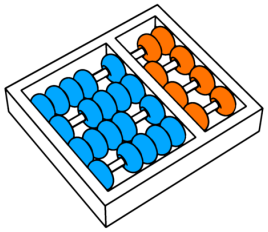


MC102 – Aula 04
Comandos de Repetição

Algoritmos e Programação de Computadores

Zanoni Dias

2019

Instituto de Computação

Roteiro

Comandos de Repetição

Comando `while`

Comando `for`

Comandos `break` e `continue`

Laços Aninhados

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 repetir até a condição ser falsa
  <comando1>
  <comando2>
  ...
  <comandoY>
```

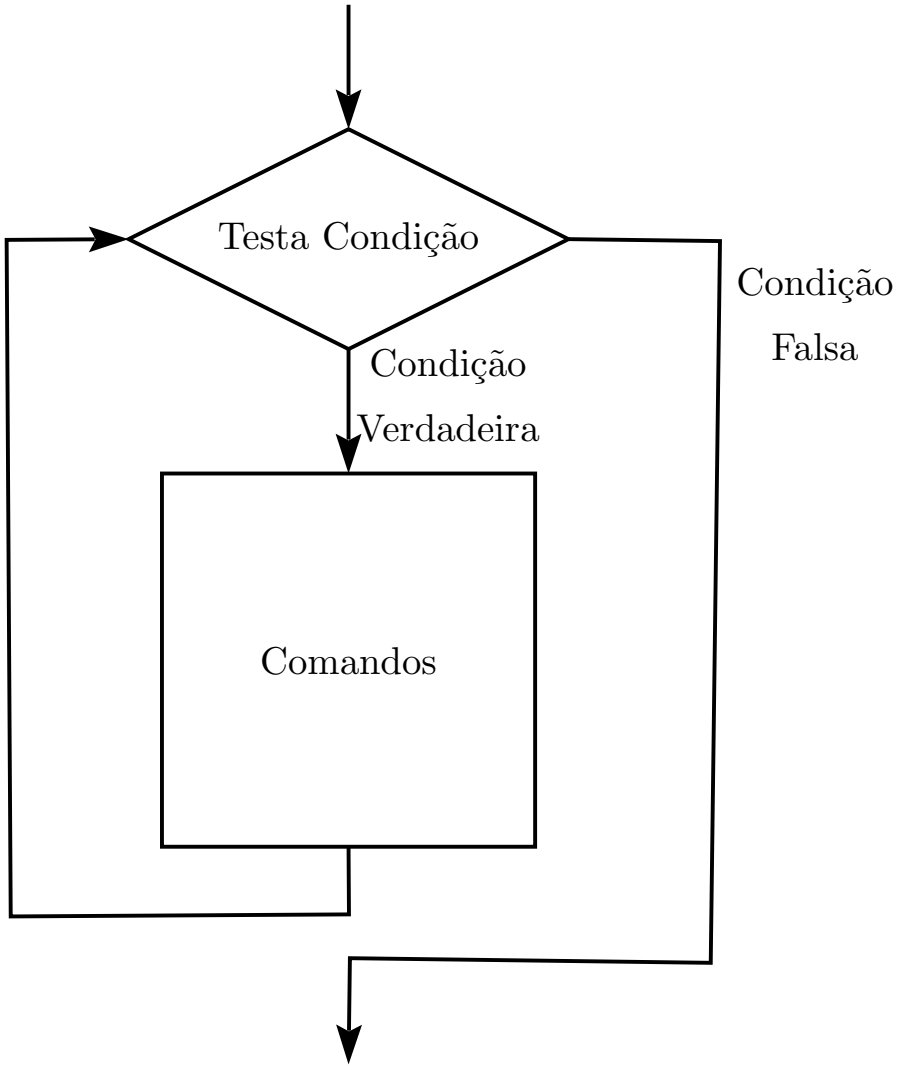
- Funcionamento:

Passo 1: Se a condição for verdadeira, vá para o Passo 2.
Caso contrário, encerre o bloco de repetição (**`while`**).

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
```

- Imprimido todos os números inteiros de 1 até n.

```
n = int(input("Digite um número: "))
i = 1
while i <= n:
  print(i)
  i = i + 1
```

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)
```

Exemplo do Comando `while`

- Dados dois números inteiros positivos (base e expoente), calcule $base^{expoente}$, usando apenas somas e multiplicações.

```
base = int(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*).

Exemplo do Comando `while`

- Dados dois números inteiros (base e expoente), calcule $base^{expoente}$, usando apenas somas, subtrações e multiplicações.

```
1  base = int(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`

- Contando 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())
6    if x > 0:
7      n = n + 1
8    else:
9      OK = False
10
11 print(n)
```

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 a tamanho 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 em 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 em 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(numeros):
  print(letras[i])
  i = i + 1
```

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

Exemplo do Comando `for`

- Encontrando o máximo em 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
```

Função `range`

- Python possui a função **`range`**, que pode ser usada para criar uma lista de inteiros.

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

- Esta função 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 lista começa em 0, o primeiro número pode ser omitido.

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

- A função `range` pode receber um terceiro argumento (opcional), que determina o incremento usado na geração da lista.

```
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 range(5):
  print("Esta frase será impressa 5 vezes")
```

```
n = int(input("Digite um número: "))
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.

```
1 for i in range(1, 101):
2   print(i)
```

- Imprimido todos os números inteiros de 1 até n.

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

Exemplo do Comando `for` com `range`

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

```
n = int(input("Digite um´numero: "))
potencia = 1
for i in range(n):
  potencia = potencia * 2
  print(potencia)
```

Variável Acumuladora

- Como no exemplo anterior, em vários problemas precisamos combinar dados em uma variável utilizando alguma operação.
- Esta variável é chamada de variável acumuladora.
- Exemplo: Somando os números inteiros de 1 até 100.

```
1 soma = 0
2 for i in range(1, 101):
3   soma = soma + i
4 print(soma)
```

Exemplo de Variável Acumuladora

- Calculando n!

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

Comandos `break` e `continue`

Comando `break`

- Vimos que o `for` percorre a lista completa. As vezes queremos percorrer apenas alguns elementos do início da lista.
- 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(i)
```

- Depois de encontrarmos x, não precisamos continuar percorrendo a lista.

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

Comando `break`

- 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.
- 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`

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

```
1 n = 0
2
3 while True:
4   x = int(input())
5   if x <= 0:
6     break
7   n = n + 1
8
9 print(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(i)
    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 da iteração corrente 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, ":")
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())
m = int(input())

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

Exercício

Imprimindo Divisores

Escreva um programa que leia um número inteiro positivo ($n > 1$) e imprima os seus divisores.

- Possível Resposta:

```
1 n = int(input())
2
3 for divisor in range(1, n+1):
4   if n % divisor == 0:
5     print(divisor)
```

Exercício

Contando Divisores

Escreva um programa que leia um número inteiro positivo ($n > 1$) e imprima o número de seus divisores.

- Possível Resposta:

```
n = int(input())
divisores = 0
for divisor in range(1, n+1):
  if n % divisor == 0:
    divisores = divisores + 1
print(divisores)
```

Exercício

Números Primos

Escreva um programa que leia um número inteiro positivo ($n > 1$) e determine se ele é primo.

- Possível Resposta:

```
n = int(input())
divisores = 0
for divisor in range(1, n+1):
  if n % divisor == 0:
    divisores = divisores + 1

if divisores == 2:
  print("Primo")
else:
  print("Composto")
```

Exercício

Números Primos

Escreva um programa que leia um número inteiro positivo ($n > 1$) e determine se ele é primo.

- Possível Resposta:

```
n = int(input())
primo = True
for divisor in range(2, n):
  if n % divisor == 0:
    primo = False

if primo:
  print("Primo")
else:
  print("Composto")
```

Exercício

Números Primos

Escreva um programa que leia um número inteiro positivo ($n > 1$) e determine se ele é primo.

- Possível Resposta:

```
n = int(input())
primo = True
for divisor in range(2, n):
  if n % divisor == 0:
    primo = False
    break
if primo:
  print("Primo")
else:
  print("Composto")
```

Exercício

Números Primos

Escreva um programa que leia um número inteiro positivo ($n > 1$) e determine se ele é primo.

- Possível Resposta:

```
n = int(input())
primo = True
for divisor in range(2, int(n/2)+1):
  if n % divisor == 0:
    primo = False
    break
if primo:
  print("Primo")
else:
  print("Composto")
```

Exercício

Números Primos

Escreva um programa que leia um número inteiro positivo ($n > 1$) e determine se ele é primo.

- Possível Resposta:

```
n = int(input())
primo = True
for divisor in range(2, int(n**0.5)+1):
  if n % divisor == 0:
    primo = False
    break
if primo:
  print("Primo")
else:
  print("Composto")
```

Exercício

Fatoração em Números Primos

Escreva um programa que leia um número inteiro positivo ($n > 1$) e imprima sua fatoração em números primos.

- Possível Resposta:

```
n = int(input())
divisor = 2
while n != 1:
  if n % divisor == 0:
    print(divisor)
    n = n / divisor
  else:
    divisor = divisor + 1
```

Exercício

Fatoração em Números Primos

Escreva um programa que leia um número inteiro positivo ($n > 1$) e imprima sua fatoração em números primos.

- Possível Resposta:

```
n = int(input())
divisor = 2
while n != 1:
  while n % divisor == 0:
    print(divisor)
    n = n / divisor
  divisor = divisor + 1

```

Exercício

Fatores Primos de um Números

Escreva um programa que leia um número inteiro positivo ($n > 1$) e imprima uma única vez cada um dos seus fatores primos.

- Possível Resposta:

```
n = int(input())
divisor = 2
ultimo = 1
while n != 1:
  if n % divisor == 0:
    if ultimo != divisor:
      print(divisor)
      ultimo = divisor
    n = n / divisor
  else:
    divisor = divisor + 1
```

Exercício

Somas de Progressões Aritméticas

Escreva um programa que leia um número inteiro ($n \geq 1$) e imprima os valores de

$$\sum_{i=1}^{j} i$$

para todo inteiro j de 1 até n, um valor por linha.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 for j in range(1, n+1):
3   soma = 0
4   for i in range(1, j+1):
5     soma = soma + i
6   print("Soma de 1 até", j, ":", soma)
```

Exercício

Somas de Progressões Aritméticas

Escreva um programa que leia um número inteiro ($n \geq 1$) e imprima os valores de

$$\sum_{i=1}^{j} i$$

para todo inteiro j de 1 até n, um valor por linha.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 soma = 0
3
4 for j in range(1, n+1):
5   soma = soma + j
6   print("Soma de 1 até", j, ":", soma)
```

Exercício

Somas de Somas

Escreva um programa que leia um número inteiro ($n \geq 1$) e imprima o valor de

$$\sum_{j=1}^{n} \sum_{i=1}^{j} i.$$

- Possível Resposta:

```
n = int(input("Digite um número inteiro positivo: "))
soma = 0
for j in range(1, n+1):
  parcial = 0
  for i in range(1, j+1):
    parcial = parcial + i
  soma = soma + parcial
print(soma)
```

Exercício

Somas de Somas

Escreva um programa que leia um número inteiro ($n \geq 1$) e imprima o valor de

$$\sum_{j=1}^{n} \sum_{i=1}^{j} i.$$

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 soma = 0
3
4 for j in range(1, n+1):
5   for i in range(1, j+1):
6     soma = soma + i
7
8 print(soma)
```

Exercício

Somas de Somas

Escreva um programa que leia um número inteiro ($n \geq 1$) e imprima o valor de

$$\sum_{j=1}^{n} \sum_{i=1}^{j} i.$$

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 soma = 0
3 parcial = 0
4 for j in range(1, n+1):
5   parcial = parcial + j
6   soma = soma + parcial
7
8 print(soma)
```

Agradecimentos

- Parte desta aula foi baseada em materiais dos seguintes professores:
  - Eduardo C. Xavier: `https://www.ic.unicamp.br/~eduardo`
  - Marcio M. Pereira: `https://iviarcio.wordpress.com`