MAC2166 - Introdução à Computação

Prof. Dr. Helder Oliveira

Agenda

- **Listas**
  - **len()**
  - **append()**
  - **percorrendo listas com o comando for**

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.

```
1  numeros = [-3, -1, -7, -9, -4]
2  maximo = numeros[0]
3  i = 1
4  while i < len(numeros):
5      if numeros[i] > maximo:
6          maximo = numeros[i]
7      i = i + 1
8  print(maximo)
9  # -1
```

Adicionar elementos na lista

- O método append() é usado para adicionar um elemento ao final de uma lista em Python.
- Ele modifica a lista original, não retorna nada.
- Criando uma lista

```
numbers = [1, 2, 3]
```

- Usando append para adicionar um número ao final da lista

```
numbers.append(4)
```

- Imprimindo a lista
  - print(numbers)

```
#[1, 2, 3, 4]
```

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`

- Encontre o máximo de um conjunto de números positivos, usando **for**.

Exemplo do Comando `for`

- Encontrando o máximo de um conjunto de números positivos.

```
1 numeros = [3, 1, 7, 9, 4]
2 maximo = 0
3
4 for numero in numeros:
5   if numero > maximo:
6     maximo = numero
7
8 print(maximo) # 9
```

Exemplo do Comando `for`

- Encontrando o máximo de um conjunto de números.

```
1 numeros = [3, 1, 7, 9, 4]
2 maximo = numeros[0]
3
4 for numero in numeros:
5   if numero > maximo:
6     maximo = numero
7
8 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.

```
1 print(list(range(2, 6)))
2 # [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.

```
1 print(list(range(1, 5)))
2 # [1, 2, 3, 4]
3 print(list(range(0, 4)))
4 # [0, 1, 2, 3]
5 print(list(range(0, -5)))
6 # []
```

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.

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

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

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

Função reverse

- O tipo list possui o método reverse(), que inverte os elementos da lista.

```
1 seq = [1, 2, 3, 4, 5]
2 seq.reverse()
3 print(seq)
```

- Solução:
  $[5, 4, 3, 2, 1]$

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!")
```

Exercicio do Comando `for` com `range`

- Utilizando o comando **for** imprima as n primeiras potências de 2, sem usar o operador de exponenciação (**).

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)
```

Exercício

- Calcule o fatorial de um número n, utilizando o comando **for**

Exemplo de Variável Acumuladora

- Calculando n!

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

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?

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

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).

```
1 n = 0
2
3 while True:
4   x = int(input("Entre com um número inteiro positivo: "))
5   if x <= 0:
6     break
7   n = n + 1
8
9 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.

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.

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

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)
```

Exercício

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

Exercício

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

```
1 n = int(input("Digite um número inteiro positivo: "))
2
3 for divisor in range(1, n+1):
4   if n % divisor == 0:
5     print(divisor)
```

Exercício

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

Exercício

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

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisores = 0
3 for divisor in range(1, n+1):
4   if n % divisor == 0:
5     divisores = divisores + 1
6 print(divisores)
```

Exercício

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

Exercício

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

```
1  n = int(input("Digite um número inteiro positivo: "))
2  divisores = 0
3  for divisor in range(1, n+1):
4    if n % divisor == 0:
5      divisores = divisores + 1
6
7  if divisores == 2:
8    print ("Primo")
9  else:
10   print("Composto")
```

Exercício

- Crie uma lista com 5 nomes de frutas.
- Use a função len() para imprimir o número de elementos da lista.

Exercício

- Crie uma lista com 5 nomes de frutas.
- Use a função len() para imprimir o número de elementos da lista.

```python
# Lista de frutas
frutas = ["maçã", "banana", "laranja", "uva", "morango"]

# Usando len() para contar o número de elementos na lista
numero_de_frutas = len(frutas)

# Imprimir o resultado
print("Número de frutas:", numero_de_frutas)
```

Exercício

- Crie uma lista vazia. Adicione os números de 1 a 5 na lista usando o método append(). Imprima a lista resultante.

Exercício

- Crie uma lista vazia. Adicione os números de 1 a 5 na lista usando o método append(). Imprima a lista resultante.

```
1  # Lista vazia
2  numeros = []
3
4  # Usando append() para adicionar os números de 1 a 5
5  for i in range(1, 6):
6      numeros.append(i)
7
8  # Imprimir a lista
9  print("Lista com números de 1 a 5:", numeros)
```

Exercício

- Dada a lista idades = [15, 23, 30, 18, 42], use um laço for para percorrer a lista e imprimir a idade de cada pessoa.

Exercício

- Dada a lista idades = [15, 23, 30, 18, 42], use um laço for para percorrer a lista e imprimir a idade de cada pessoa.

```
1  # Lista de idades
2  idades = [15, 23, 30, 18, 42]
3
4  # Usando um laço for para percorrer a lista
5  for idade in idades:
6      print("Idade:", idade)
```

Exercício

- Dada a lista de temperaturas temperaturas = [25, 30, 35, 40, 45], use um laço for para percorrer e imprimir uma mensagem dizendo "Muito quente" se a temperatura for maior que 30 e "Agradável" caso contrário.

Exercício

- Dada a lista de temperaturas temperaturas = [25, 30, 35, 40, 45], use um laço for para percorrer e imprimir uma mensagem dizendo "Muito quente" se a temperatura for maior que 30 e "Agradável" caso contrário.

```python
# Lista de temperaturas
temperaturas = [25, 30, 35, 40, 45]

# Percorrendo a lista e imprimindo a mensagem de acordo com a temperatura
for temp in temperaturas:
    if temp > 30:
        print(f"{temp}°C: Muito quente")
    else:
        print(f"{temp}°C: Agradável")
```

Exercício

- Crie uma lista de strings palavras = ['python', 'java', 'c', 'ruby', 'javascript'].
- Use len() para contar o número de palavras na lista e depois use um laço for para imprimir a quantidade de letras de cada palavra.

Exercício

- Crie uma lista de strings palavras = ['python', 'java', 'c', 'ruby', 'javascript'].
- Use len() para contar o número de palavras na lista e depois use um laço for para imprimir a quantidade de letras de cada palavra.

```python
# Lista de palavras
palavras = ['python', 'java', 'c', 'ruby', 'javascript']

# Usando len() para contar o número de palavras
numero_de_palavras = len(palavras)

# Imprimir a quantidade de letras de cada palavra
print("Número de palavras:", numero_de_palavras)
for palavra in palavras:
    print(f"A palavra '{palavra}' tem {len(palavra)} letras.")
```

Exercício

- Crie uma lista vazia chamada quadrados. Use um laço for para adicionar os quadrados dos números de 1 a 10 na lista, utilizando o método append(). Imprima a lista resultante.

Exercício

- Crie uma lista vazia chamada quadrados. Use um laço for para adicionar os quadrados dos números de 1 a 10 na lista, utilizando o método append(). Imprima a lista resultante.

```python
# Lista vazia para armazenar os quadrados
quadrados = []

# Adicionar os quadrados de 1 a 10 na lista
for i in range(1, 11):
    quadrados.append(i ** 2)

# Imprimir a lista com os quadrados
print("Lista de quadrados:", quadrados)
```

Referencias

- Slides baseados em:
  - Zanoni Dias, MC102- Algoritmos e Programação de Computadores, 2020.

Dúvidas