Aula 3: Operações e Expressões Relacionais

Luís Felipe

UFF

25 de Agosto de 2025

Luís Felipe

25/08/25

Revisão de algumas utilizações

- Exemplo de uso de variáveis, expressões e formatação de saída:

Exemplo: Cálculo da Área de um Círculo (sem precisão e com precisão de duas casas decimais)

```
1  pi = 3.1415
2  r = 7
3  area = pi * r * r
4  print("A área de um circulo de raio ", str(r), " é: ", str(area))
5  print("A área de um circulo de raio %.2f" % r, " é: %.2f" % area)
```

ou

```
1  pi = 3.1415
2  r = 7
3  area = pi * r * r
4  print(f"A área de um círculo de raio {str(r):.2s} é: {str(area):.8s}")
5  print(f"A área de um círculo de raio {r:.2f} é: {area:.2f}")
```

Saída:

```
A área de um circulo de raio 7 é: 153.9335
A área de um circulo de raio 7.00 é: 153.93
```

Luís Felipe

25/08/25

Uso do parâmetro end na função print

- A função print normalmente pula uma linha ao final da impressão.
- Para evitar a quebra de linha automática, use o parâmetro end.

```
1 print("3, ", end="")
2 print("4, ", end="")
3 print("5 ", end="")
```

Saída:

```
3, 4, 5
```

Luís Felipe

25/08/25

A função `input` e conversão com `int()`

- A função `input()` lê uma **string** digitada pelo usuário.
- Podemos converter essa string em número usando a função `int()`.

```
1  print("Digite um numero:")
2  a = int(input())
3  a = a * 10
4  print("O número digitado vezes 10 é:", a)
```

- Podemos fazer o mesmo para números ponto flutuante usando a função float().

```
1  print("Digite um numero:")
2  a = float(input())
3  a = a*10
4  print("O número digitado vezes 10 é %.2f: " %a)
```

- Nos dois exemplos acima é esperado que o usuário digite um número.
- Se o usuário digitar um texto não numérico o programa encerrará com um erro de execução.

Luís Felipe

25/08/25

Leitura com `input()` e soma de dois números

- O programa abaixo lê dois números reais e imprime a soma deles (precisão de 2 casas decimais).
- Podemos incluir um texto explicativo diretamente dentro do `input()`.

```
1  a = float(input("Digite um número: "))
2  b = float(input("Digite um número: "))
3  print("A soma dos números é: %.2f" % (a + b))
```

Saída esperada:

```
Digite um número: 10.5
Digite um número: 2.3
A soma dos números é: 12.80
```

Luís Felipe

25/08/25

Expressões Aritméticas — Parte 1

- **Operadores**: +, -, *, /, //, **, %

+ Soma:

```
>>> 56 + 9
65
```

- Subtração:

```
>>> 56 - 9
47
```

***** Multiplicação:

```
>>> 56 * 9
504
```

/ Divisão comum (retorna `float`):

```
>>> 27 / 9
3.0
>>> 27 / 2
13.5
```

Luís Felipe

25/08/25

Expressões Aritméticas — Parte 2

// Divisão inteira:

```
>>> 5 // 2
2
>>> 5 // 2.0
2.0
```

****** Potência: Expressão da esquerda elevado a expressão da direita

```
>>> 2 ** 4
16
>>> 2.2 ** 4
23.425600000000006
>>> (2+4)**(5-3)
36
```

% Módulo (resto da divisão inteira): Obs.: Ao dividir a por b, temos: $a = p * b + r$, onde $r < b$. Assim: $a\%b = r$

```
>>> 5 % 2
1
>>> 9 % 7
2
>>> 2 % 5
2
```

Luís Felipe

25/08/25

Precedência de Operadores

- Precedência determina a ordem em que os operadores são avaliados na expressão.
- Em Python, a ordem de precedência é:
  - `**` (potência)
  - `*`, `/`, `//`, `%` (na ordem em que aparecem)
  - `+`, `-` (na ordem em que aparecem)
- **Exemplo:**

```
1 >>> 8 + 10 * 6
2 68
```

Quais as ordens de precedência das seguintes contas:

$1 + 2 * *4/2\%5$

$1 + 2 * *4\%2/5$

$2 * *4\%2 + 1/5$

Luís Felipe

25/08/25

Alterando a precedência

- O uso de parênteses ( ) altera a ordem de avaliação da expressão.
- O que está entre parênteses é avaliado primeiro.
- Exemplo:

```
>>> 5 + 10 % 3
6
>>> (5 + 10) % 3
0
```

- Use parênteses sempre que quiser deixar clara a ordem de avaliação!

Luís Felipe

25/08/25

Expressões e operadores relacionais

- Como visto na Aula 2, o tipo bool especifica os valores booleanos falso e verdadeiro.
- O uso mais comum é na verificação de resultados de expressões relacionais e lógicas.

Expressões Relacionais são aquelas que realizam uma comparação entre duas expressões e retornam False, se o resultado é falso, ou True, se o resultado é verdadeiro.

Operadores Relacionais em Python são:

== : Igualdade

!= : Diferente

> : Maior que

< : Menor que

>= : Maior ou igual que

<= : Menor ou igual que

Luís Felipe

25/08/25

Operadores Relacionais em Python

- expressão == expressão: Retorna True se forem iguais.

```
>>> 9 == 9
True
>>> 9 == 10
False
```

- expressão != expressão: Retorna True se forem diferentes.

```
>>> 9 != 9
False
>>> 9 != 10
True
```

- expressão > expressão: Retorna True se a esquerda for maior.

```
>>> 9 > 5
True
```

- expressão < expressão: Retorna True se a esquerda for menor.

```
>>> 9 < 5
False
```

- expressão >= expressão: Retorna True se a esquerda for maior ou igual.

```
>>> 9 >= 5
True
```

- expressão <= expressão: Retorna True se a esquerda for menor ou igual.

```
>>> 9 <= 5
False
```

O que será impresso pelo programa?

```
print(9 > 3)
print((3*4)/2 != (2*3) )
a = 1
b = -1
print(a!=b)
```

Luís Felipe

25/08/25

Expressões Lógicas e Operadores Lógicos

- **Expressões lógicas:**
  - Realizam operações como **E**, **OU**, **NÃO** (and, or, not).
  - Retornam True ou False, como as expressões relacionais.

- **Operadores Lógicos em Python:**
  - and (operador **E**): retorna `True` se ambas as expressões forem verdadeiras.
  - or (operador **OU**): retorna `True` se pelo menos uma das expressões for verdadeira.
  - not (operador **NÃO**): inverte o valor lógico da expressão.

Luís Felipe

25/08/25

Tabela verdade

Tabela que mostra o resultado lógico de uma operação para todas as combinações de valores dos operandos.

A	not A
True	False
False	True

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

```
>>> a = 0
>>> b = 0
>>> a == 0 and b == 0
True
```

```
>>> a = 0
>>> b = 1
>>> a == 0 or b == 0
True
```

```
>>> a = 0
>>> b = 1
>>> a == 0 and b == 0
False
```

O que será impresso pelo programa?

```
print( (8 > 9) and (10 != 2))
print( (14 > 100) or (2 > 1))
print( (not (14 > 100)) and (not (1 > 2)) )
```