Curso: ADMINISTRAÇÃO / ECONOMIA / BIOLOGIA
Disciplina: Introdução à Computação
Aula 4
Ministrada em: 1S - 2024

Python

Escrita, Leitura, Expressões Aritméticas, Expressões Relacionais, Expressões Lógicas e Comandos Condicionais

Prof. Maurício Gagliardi Palma

mauricio.palma@ufscar.br

Parcialmente baseado no material da Prof. Letícia Mara Berto

Escrita

Leitura

Expressões Aritméticas

Expressões Relacionais

Expressões Lógicas

Comandos Condicionais

comando: `print()`

- Para imprimir um texto, utilizamos o comando `print()`
- O texto pode ser um literal do tipo **string**

```
>>> print(“Hello World”)
Hello World
```

comando: `print()`

- No meio da string pode-se incluir caracteres especiais de formatação.
- O símbolo especial **\n** é responsável por pular uma linha na saída.

```
>>> print(“Hello \n World”)
Hello
World
```

Escrevendo o Conteúdo de uma Variável na tela

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando **print()**

- Separamos múltiplos argumentos a serem impressos com uma vírgula

```
>>> a = 10
>>> b = 3.14
>>> print("A variável contém o valor", a)
A variável contém o valor 10
>>> print("a contém o valor", a, "e b contém o valor", b)
a contém o valor 10 e b contém o valor 3.14
```

Imprimindo sem Pula Linha

- A função `print()` sempre pula uma linha ao final da impressão
- Se você não quiser que pule uma linha, inclua o argumento **`end=“ ”`** no `print()`

```
>>> print(“3, ”, end=“”)
>>> print(“4, ”, end=“”)
>>> print(“5”, end=“”)
3, 4, 5
```

Escrita

Leitura

Expressões Aritméticas

Expressões Relacionais

Expressões Lógicas

Comandos Condicionais

Lendo do teclado

função: `input()`

- Realiza a leitura de dados a partir do teclado
- Aguarda que o usuário digite um valor e atribui o valor digitado a uma
- Todos os dados lidos são do tipo **string**

```
>>> print(“Digite um número: ”)
>>> numero = input()
>>> print(“O número digitado é: ” + numero)
```

função: input()

- Podemos converter uma string lida do teclado em um número inteiro usando a função **int()**

```
>>> print("Digite um número: ")
>>> numero = int(input())
>>> numero = numero * 10
>>> print("O número digitado vezes 10 é: ", numero)
```

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

```
>>> print("Digite um número: ")
>>> numero = float(input())
>>> numero = numero * 10
>>> print("O número digitado vezes 10 é: ", numero)
```

função: `input()`

- Nos dois exemplos anteriores é esperado que o usuário digite um número

- Se o usuário digitar um texto não numérico o programa encerra com um erro de execução

função: `input()`

- O programa abaixo lê dois números e imprime a soma destes
- Perceba que podemos incluir um texto a ser impresso diretamente no comando `input()`

```
>>> numero1 = float(input("Digite um número: "))
>>> numero2 = float(input("Digite um número: "))
>>> print("A soma dos números é: %.2f" %(numero1 + numero2))
```

Escrita
Leitura
Expressões Aritméticas
Expressões Relacionais
Expressões Lógicas
Comandos Condicionais

Expressões

- Pode ser uma constante, variável, expressão matemática, função, expressão lógica

- As expressões são avaliadas pelo computador quando o programa é executado, fornecendo um valor único que será armazenado na variável.

- Pode envolver outras variáveis
  - Todas as variáveis usadas em uma expressão devem ter um valor associado no momento em que a expressão for avaliada.

Expressões Aritméticas

- Os operadores aritméticos são: **+**, **-**, *, /, //, **%**, **

- **Adição**: expressão **+** expressão
- **Subtração**: expressão **-** expressão
- **Multiplicação**: expressão ***** expressão
- **Divisão**: expressão **/** expressão
  - O resultado é sempre um número ponto flutuante
- **Divisão**: expressão **//** expressão
  - Se os operandos forem inteiros, a divisão é inteira. Se um deles for ponto flutuante, faz uma divisão truncada (inteira).
- **Exponencial** (**potência**): expressão ****** expressão
  - Calcula o valor da expressão à esquerda elevado ao valor da expressão a direita.
- **Resto da divisão**: expressão **%** expressão
  - Calcula o resto da divisão inteira de duas expressões

Expressões Aritméticas - Exemplos

```
>>> 30 + 5
35
>>> 30 - 5
25
>>> 30 * 5
150
>>> 5 / 2
2.5
>>> 5 // 2
2
>>> 2 ** 4
16
>>> 5 % 2
1
```

Expressões Aritméticas - Precedência

- Precedência é a **ordem** na qual os operadores serão avaliados quando o programa for executado
- Em Python, os operadores são avaliados na seguinte ordem:

1. ******
2. *, /, //*, na ordem em que aparecem na expressão
3. **%**
4. **+**, **-**, na ordem em que aparecem na expressão

Expressões Aritméticas - Precedência

- Operadores com a mesma precedência são executados da esquerda para a direita
- **Atenção**: Uma exceção é o operador exponenciação **

Alterando a precedência

- **(expressão)** também é uma expressão, que calcula o resultado da expressão dentro dos parênteses, para só então calcular o resultado das outras expressões.
  - 5 + 10 % 3 é igual a 6
  - (5 + 10) % 3 é igual a 0

- Você pode usar quantos parênteses desejar dentro de uma expressão

- Use sempre parênteses em expressões para deixar claro em qual ordem a expressão é avaliada!

Escrita
Leitura
Expressões Aritméticas
Expressões Relacionais
Expressões Lógicas
Comandos Condicionais

Expressões Relacionais

- Expressões relacionais são aquelas que realizam uma **comparação** entre duas expressões e retornam
  - `False`, se o resultado é falso
  - `True`, se o resultado é verdadeiro

- Os operadores relacionais da linguagem Python são:
  - `==` : igualdade
  - `!=` : diferente
  - `>` : maior que
  - `<` : menor que
  - `>=` : maior ou igual que
  - `<=` : menor ou igual que

Expressões Relacionais

- **expressão == expressão** : Retorna verdadeiro quando as expressões forem iguais.

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

- **expressão != expressão** : Retorna verdadeiro quando as expressões forem diferentes.

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

Expressões Relacionais

- **expressão > expressão** : Retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita

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

- **expressão < expressão** : Retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita

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

Expressões Relacionais

- **expressão >= expressão** : Retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita

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

- **expressão <= expressão** : Retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita

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

Escrita
Leitura
Expressões Aritméticas
Expressões Relacionais
Expressões Lógicas
Comandos Condicionais

Expressões Lógicas

- Expressões lógicas são aquelas que realizam uma operação lógica (**e**, **ou**, **não**, etc…) e retornam **`True`** e **`False`** (como as expressões relacionais).

- Na linguagem Python temos os seguintes operadores lógicos:
  - **`and`** : operador E
  - **`or`** : operador OU
  - **`not`** : operador NÃO

Expressões Lógicas

- **expressão and expressão**: Retorna verdadeiro quando **ambas** as expressões são verdadeiras. Sua tabela verdade é:

Op1	Op2	Op1 and Op2
V	V	V
V	F	F
F	V	F
F	F	F

Expressões Lógicas

- **expressão or expressão**: Retorna verdadeiro quando **pelo menos uma** das expressões é verdadeira. Sua tabela verdade é:

Op1	Op2	Op1 or Op2
V	V	V
V	F	V
F	V	V
F	F	F

Expressões Lógicas

- **not expressão**: Retorna verdadeiro quando a expressão é false e vice-versa. Sua tabela verdade é:

Op1	**not Op1**
V	F
F	V

Precedência de Operadores

Nível	Categoria	Operadores
7 (alto)	exponenciação	**
6	multiplicação	*, /, //, %
5	adição	+, -
4	relacional	==, !=, <=, >=, >, <
3	lógico	not
2	lógico	and
1 (baixo)	lógico	or

Escrita
Leitura
Expressões Aritméticas
Expressões Relacionais
Expressões Lógicas
Comandos Condicionais

Comandos Condicionais

- Um comando condicional é aquele que permite decidir se um determinado **bloco de comandos** deve ou não ser executado, a partir do resultado de uma expressão relacional ou lógica.

- Blocos de Comandos:
  - É um conjunto de instruções agrupadas
  - Os comandos agrupados do bloco devem estar **indentados** dentro de um comando anterior **seguido de dois pontos**.
  - A indentação é feita utilizando **tab**

Comandos Condicionais

- O principal comando condicional é o if, cuja sintaxe é:

```
if expressão relacional ou lógica:
    bloco de comandos
```

- Os comandos são executados somente se a expressão relacional / lógica for verdadeira

- Exemplo:

```
# Informa se o número é par.
numero = int(intput())
if numero % 2 == 0:
    print("O número digitado é par.")
```

Comandos Condicionais

- Uma variação do comando **`if`** é o **`if/else`**, cuja sintaxe é:

```
if expressão relacional ou lógica:
    bloco de comandos
else:
    bloco de comandos
```

- Exemplo:

```
# Informa se o número é par.
numero = int(intput())
if numero % 2 == 0:
     print("O número digitado é par.")
else:
     print("O número digitado é ímpar.")
```

Comandos Condicionais

- Exemplo:

```
# Determina o menor de dois números.
numero1 = int(intput())
numero2 = int(intput())
if numero1 < numero2:
     print("O menor número é:", numero1)
else:
     print("O menor número é:", numero2)
```

Comandos Condicionais

- Note que o **if** é um comando, e como tal pode aparecer dentro do bloco de comandos de outro **if**.
- **Exemplo:** Um programa que lê um número e verifica em qual dos seguintes casos o número se enquadra:
  - Par e menor que 100
  - Par e maior ou igual a 100
  - Ímpar e menor que 100
  - Ímpar e maior ou igual a 100

```
numero = int(intput(“Digite um número”))
if (numero % 2 == 0): # se o número for par
     if (numero < 100):
          print("O número é par e menor que 100")
     else:
          print("O número é par e maior ou igual que 100")
else: # se o número for ímpar
     if (numero < 100):
          print("O número é ímpar e menor que 100")
     else:
          print("O número é ímpar e maior ou igual que 100")
```

Comandos Condicionais

- Exemplo anterior, só que usando operadores lógicos

```
numero = int(intput(“Digite um número”))
if (numero % 2 == 0) and (numero < 100):
     print("O número é par e menor que 100")
if (numero % 2 == 0) and (numero >= 100):
     print("O número é par e maior ou igual que 100")
if (numero % 2 != 0) and (numero < 100):
     print("O número é ímpar e menor que 100")
if (numero % 2 != 0) and (numero >= 100):
     print("O número é ímpar e maior ou igual que 100")
```

Obrigado!