MAC2166 - Introdução à Computação

Prof. Dr. Helder Oliveira

Agenda

- Começando a conhecer Python
- Console
- Comentários
- Tipos
- Variáveis, expressões e comandos
- Comando de atribuição
- Escrita na tela

Imprimindo no Console

- A função print() é responsável por imprimir uma mensagem.
- A função print() pode ser utilizada para informar o usuário sobre:
  - A resposta de um processamento.
  - O andamento da execução do programa.
  - Comportamentos inesperados do programa.
  - Outros motivos em que o usuário precise ser informado sobre algo.

Imprimindo no Console

- Com o ambiente interativo do Python carregado, também chamado de console, digite o seguinte comando:

```
1 print("Hello world!")
```

  - Como resposta desse comando, na linha seguinte do console, deve aparecer a mensagem:

```
1 Hello world!
```

Imprimindo no Console

- Iremos estudar posteriormente como criar nossas próprias funções, mas agora vamos aprender um pouco mais sobre a função print.
- Como todas as funções, a sintaxe para a função de impressão começa com o nome da função (que neste caso é print), seguida de uma lista de argumentos, incluída entre parênteses.

```
1 print("Argumento 1", "Argumento 2", "Argumento 3")
```

```
1 Argumento 1 Argumento 2 Argumento 3
```

Imprimindo no Console

- Note que, quando informamos mais de um argumento para a função print, eles são automaticamente separados por um espaço.

```
print("Hello", "world!")
```

```
Hello world!
```

- Podemos modificar isso utilizando o parâmetro sep.

```
print("Hello", "world!", sep = "+")
```

```
Hello+world!
```

Imprimindo no Console

- Os comandos a seguir produzem o mesmo resultado:

```
1 print("Hello world!")
2 print("Hello", "world!")
3 print("Hello", "world!", sep = " ")
```

- Resposta obtida:

```
1 Hello world!
2 Hello world!
3 Hello  world!
```

Imprimindo no Console

- A função print imprime automaticamente o caractere de quebra de linha (\n) no fim de cada execução.

```
1 pr i nt ("USP")
2 print("MAC2166!")
```

```
1 USP
2 MAC2166!
```

- Também podemos modificar isso utilizando o parâmetro end.

```
1 print("USP", end = "")
2 print("MAC2166!")
```

```
1 USPMAC2166!
```

Imprimindo no Console

- Sem o caractere de controle de quebra de linha (\n) no fim:

```
print("MAC2166", "USP", "2025", sep = " - ", end = "!")
print("Novo Texto!")
```

```
MAC2166 - USP - 2025!Novo Texto!
```

- Com o caractere de controle de quebra de linha (\n) no fim:

```
print("MAC2166", "USP", "2025", sep = " - ", end = "!\n")
print("Novo Texto!")
```

```
MAC2166 - USP - 2025!
Novo Texto!
```

Comentários

- Em Python é possível adicionar um comentário utilizando o caractere #, seguido pelo texto desejado.
- Os comentários não são interpretados pela linguagem, isso significa que todo texto após o caractere #é desconsiderado.
- Exemplo:

```
1 print("Hello world!") # Exemplo de função print
```

- Como resposta para o código acima obtemos apenas:

```
1 Hello World!
```

Comentários

- Vantagens de comentar o seu código:
  - Comentários em trechos mais complexos do código ajudam a explicar o que está sendo realizado em cada passo.
  - Torna mais fácil para outras pessoas que venham a dar manutenção no seu código ou mesmo para você relembrar o que foi feito.

```
1 # Parâmetros importantes da função print
2 # sep: Texto usado na separação dos argumentos recebidos.
3 # end: Texto impresso no final da execução da função.
4 print("MAC2166", "USP", sep = " - ", end = "!\n")
5 # MAC2166 - USP!
```

Comentários

- O caractere # é utilizado para comentar uma única linha.
- É possível comentar múltiplas linhas utilizando a sequência de caracteres ''' no início e no fim do trecho que se deseja comentar.

```
'''
Parâmetros importantes da função print
sep: Texto usado na separação dos argumentos recebidos. end: Texto impresso no
final da execução da função.
'''

print("MAC2166", "USP", sep = " - ", end = "!\n")
# MAC2166 - USP!
```

Tipos

- Em Python existem diferentes tipos de dados.
- Podemos ter dados no formato:
  - Numérico.
  - Textual.
  - Lógico.
- Para isso, em Python, temos alguns tipos:
  - **int** Números inteiros (Exemplos: -3, 7, 0, 2020).
  - **float** Números reais (Exemplos: -3.2, 1.5).
  - **str** Cadeia de caracteres/Strings (Exemplos: "USP" e "MAC2166").
  - **bool** Valores booleanos: True (Verdadeiro) e False (Falso).

Tipos

- A função type pode ser utilizada para mostrar o tipo de um dado.
- Essa função recebe um argumento que terá o tipo identificado.
- Como resposta, a função informa o tipo do dado fornecido como argumento.
- Exemplo da estrutura da função:

```
type(<argumento>)
```

Exemplos de Tipos

```
print(type(10))  #
<class 'int'>
```

```
print(type(10.0))
 # <class 'float'>
```

```
print(type("10"), type("10.0"))
# <class 'str'> <class 'str'>
```

```
print(type(True), type(False), type("True"), type("False"))
# <class 'bool'> <class 'bool'> <class 'str'> <class 'str'>
```

Variáveis

- Ao escrevermos um código, surge a necessidade de armazenarmos valores de maneira temporária, para isso temos as variáveis.
- Em Python, o caractere =é utilizado para atribuir um valor a uma variável.
- Exemplo:

```
1 pi = 3.1416
2 print (pi)
3 # 3.1416
```

Variáveis

- Também é possivel, utilizando o caractere =, atribuir um mesmo valor para múltiplas variáveis num único comando.
- Exemplo:

```
a = b = c = 3
print(a, b, c)
# 3 3 3
```

  - É possivel também atribuir valores diferentes para múltiplas variáveis com um único comando.
  - Exemplo:

```
a, b, c = 1, 2, 3
print(a, b, c)
# 1 2 3
```

Regras para Nomes de Variáveis

- Nomes de variáveis devem começar com uma letra (maiúscula ou minúscula) ou um sublinhado (_).
- Nomes de variáveis podem conter letras maiúsculas, minúsculas, números ou sublinhado.
- Cuidado: a linguagem Python é *case sensitive*, ou seja, ela diferencia letras maiúsculas de minúsculas.
- Por exemplo, as variáveis c1 e C1 são consideradas diferentes:

```
1 c1 = 0
2 C1 = "1"
3 print(c1, type(c1), C1, type(C1))
4  # 0 <class 'int'> 1 <class 'str'>
```

Exemplos de Variáveis

- Exemplo de variáveis do tipo int e float:

```
nota_1 = 10
nota_2 = 7.8
nota_final = 8.75
```

```
print(nota_1, type(nota_1))  # 10
<class 'int'>
```

```
print(nota_2, type(nota_2))  # 7.8
<class 'float'>
```

```
print(nota_final, type(nota_final))  # 8.75 <class
'float'>
```

Exemplos de Variáveis

- Exemplo de variáveis do tipo str:

```
1 UFABC = "Universidade de São Paulo"
2 print(UFABC, type(UFABC))
3 # Universidade de São Paulo <class 'str'>
```

```
1 pi_2023_1s = "PI"
2 print(pi_2023_1s, type(pi_2023_1s))
3 # PI <class 'str'>
```

Exemplos de Variáveis

- Exemplo de variáveis do tipo bool:

```
1  verdadeiro = True
2  falso = False
3  print(verdadeiro, type(verdadeiro), falso, type(falso))
4  # True <class 'bool'> False <class 'bool'>
```

Operadores Aritméticos

Sequência de escape	Descrição
\\	Barra invertida (*Backslash*). Insere uma barra invertida.
\'	Aspas simples (*Single quote*). Insere uma aspas simples.
\"	Aspas duplas (*Double quote*). Insere uma aspas duplas.
\n	Nova linha (*NewLine*). Move o cursor para o início da próxima linha.
\t	Tabulação horizontal (*Horizontal tab*). Move o cursor uma tabulação para frente.

Operadores Matemáticos - Adição

```
1 + 1
# 2
```

```
1.5 + 2
# 3.5
```

```
a = 10
a + 10
# 20
```

```
a = 10
b = 20
a + b
# 30
```

Operadores Matemáticos - Subtração

```
5 - 1.5
# 3.5
```

```
a = 100
a - 50
# 50
```

```
a = 1000
b = 0.1
b - a
# -999.9
a - b
# 999.9
```

Operadores Matemáticos - Multiplicação

```
11 * 13
# 143
```

```
2.5 * 2.5
# 6.25
```

```
3 * 0.5
# 1.5
```

```
a = 11
b = 17
a * b
# 187
```

Operadores Matemáticos - Divisão

```
7 / 2
# 3.5
```

```
a = 10
a / 7
# 1.4285714285714286
```

Operadores Matemáticos - Exponenciação

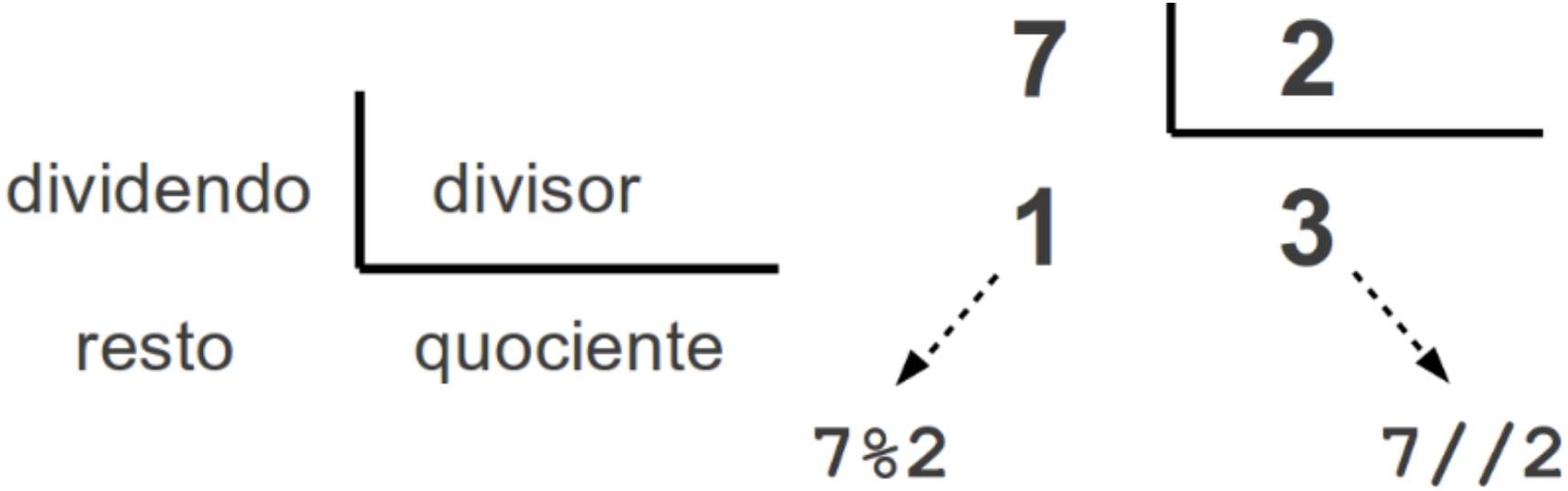
```
2 ** 2
# 4
```

```
a = 10
2 ** a
# 1024   a
** 2
# 100
```

```
2.5 ** 3.5
# 24.705294220065465
```

```
3.5 ** 2.5
# 22.91765149399039
```

Operadores Matemáticos



Operadores Matemáticos - Divisão

- Divisão:

```
7 / 2
# 3.5
```

```
a = 10
a / 7
# 1.4285714285714286
```

- Divisão Inteira:

```
7 // 2
# 3
```

```
a = 10
a // 3.4
# 2.0
```

Atualizações Compactas

- Para os operadores matemáticos, é possível utilizar uma forma compacta para atualizar o valor de uma variável.
  - x += y é equivalente a x = x + y.
  - x -= y é equivalente a x = x - y.
  - x *= y é equivalente a x = x * y.
  - x /= y é equivalente a x = x / y.

Operadores Matemáticos - Ordem de 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 de precedência:
  - Exponenciação.
  - Operadores unários (+ ou −).
  - Multiplicação e divisão (na ordem em que aparecem).
  - Módulo.
  - Adição e subtração (na ordem em que aparecem).
- Podemos controlar a ordem com que as expressões são avaliadas com o uso de parênteses.
- Procure usar sempre parênteses em expressões para deixar claro em qual ordem as mesmas devem ser avaliadas.

Precedência de Operadores Matemáticos

```
1 print(2 + 2 / 2)
2 # 3.0
```

```
1 print((2 + 2) / 2)
2 # 2.0
```

Erros Comuns com Operadores Matemáticos

- Divisão por zero:

```
10 / 0
# ZeroDivisionError: division by zero
```

```
10 / 0.0
# ZeroDivisionError: float division by zero
```

Erros Comuns com Operadores Matemáticos

```
3 + * 3
# SyntaxError: invalid syntax
```

```
2 + % 3
# SyntaxError: invalid syntax
```

```
5 - / 2
# SyntaxError: invalid syntax
```

```
-2 * * 2
# SyntaxError: invalid syntax
```

Quais os Resultados destas Operações?

```
3 * + 3
# 9
```

```
2 % + 3
# 2
```

```
5 / - 2
# -2.5
```

```
-2 ** 2
# -4
```

Operadores com Strings - Concatenação

```
"Hello" + " World"
# 'Hello World'
```

```
USP = "Universidade" + " de" + " São Paulo"
print(USP)
# Universidade de São Paulo
```

```
nome = "Fulano"
mensagem = ", você está na turma de MAC2166!"
print(nome + mensagem)
# Fulano, você está na turma de MAC2166!
```

Operadores com Strings - Replicação

```
"USP" * 3
# 'USPUSPUSP'
```

```
print(4 * "USP ")
# USP USP USP USP
```

```
letra = "Z"  n =
10
print(letra * n)
 # ZZZZZZZZZZ
```

Strings - Ordem de Precedência

- A ordem de precedência dos operadores com strings é a seguinte:
  - Replicação
  - Concatenação
- Podemos controlar a ordem com que as expressões são avaliadas com o uso de parênteses.
- Exemplos:

```
"a" + "b" * 3
# 'abbb'
```

```
("a" + "b") * 3
# 'ababab'
```

Strings vs. Números

```
4 + 5
# 9
```

```
"4" + "5"
# '45'
```

```
"4" + 5
# TypeError: can only concatenate str (not "int") to str
```

```
4 + "5"
# TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Comparações Numéricas

```
5 < 4
# False
```

```
5 > 4
# True
```

```
5 <= 4
# False
```

```
5 <= 5
# True
```

```
5 >= 4
# True
```

Comparações Numéricas

```
5 != 4
# True
```

```
5 == 4
# False
```

```
5 == 5.0
# True
```

```
5 == 5.000001
# False
```

```
5 == "5"
# False
```

Comparações de Strings

- Ordem considerada para os caracteres do alfabeto:
  - 0...9ABC...XYZabc...xyz

```
"a" > "b"
# False
```

```
"a" < "b"
# True
```

```
"a" == "a"
# True
```

```
"a" == "A"
# False
```

Comparações de Strings

```
"A" < "a"
# True
```

```
"A" > "a"
# False
```

```
"Z" < "a"
# True
```

```
"z" < "a"
# False
```

Comparações de Strings

```
"Araraquara" < "Araras"
# True
```

```
"Maria" < "Maria Clara"
# True
```

```
"maria" < "Maria Clara"
# False
```

```
"Marvel" > "DC"
 # True
```

Comparações de Strings

- Para obter a ordem relativa de outros caracteres, consulte a Tabela ASCII:
  - https://pt.wikipedia.org/wiki/ASCII

```
"senha" > "s3nh4"
# True
```

```
"aa aa" >= "aaaa"
# False
```

```
"@mor" < "amor"
 # True
```

```
"21+7" < "2+31"
# False
```

Conversões de Tipos

- Alguns tipos de dados permitem que o seu valor seja convertido para outro tipo (cast).
- Para isso, podemos usar as seguintes funções:
  - int() converte o valor para o tipo int (número inteiro).
  - float() converte o valor para o tipo float (número real).
  - str() converte o valor para o tipo str (string).
  - bool() converte o valor para o tipo bool (booleano).

Caracteres de escape

Operador	Descrição	Exemplo	Avalia para
+	Adição	7 + 3	10
-	Subtração	7 - 3	4
*	Multiplicação	7 * 3	21
/	Divisão (Real)	7 / 3	2.3333333333333335

Caracteres de escape

```
a = "Camões escreveu \"Os Lusíadas\" no século XVI."
print(a)
Camões escreveu "Os Lusíadas" no século XVI.
b = "Primeira linha.\nSegunda linha."
print(b)
Primeira linha.
Segunda linha.
c = "\tCom tabulação.\nSem tabulação."
 print(c)
        Com tabulação.
Sem tabulação.
d = "Barra invertida: \\"
print(d)
Barra invertida: \
e = "\\\\\\"   #Imprime três barras invertidas.
print(e)
\\\
```

Dúvidas

Exercício

- Crie três variáveis: uma para armazenar um **número inteiro**, outra para um **número decimal** e uma terceira para um **texto**. Em seguida, imprima os valores dessas variáveis na tela.

Exercício

- Crie três variáveis: uma para armazenar um **número inteiro**, outra para um **número decimal** e uma terceira para um **texto**. Em seguida, imprima os valores dessas variáveis na tela.

```
# Definição das variáveis
numero_inteiro = 10
numero_decimal = 3.14
texto = "Olá, mundo!"

# Exibindo os valores das variáveis
print("Número inteiro:", numero_inteiro)
print("Número decimal:", numero_decimal)
print("Texto:", texto)
```

Exercício

- Crie duas variáveis numéricas e calcule a soma, subtração, multiplicação e divisão entre elas. Depois, exiba os resultados.

Exercício

- Crie duas variáveis numéricas e calcule a soma, subtração, multiplicação e divisão entre elas. Depois, exiba os resultados.

```
# Definição das variáveis
a = 8
b = 4
texto = "Olá, mundo!"

# Operações matemáticas
print("Número inteiro:", numero_inteiro)
soma = a + b
subtracao = a - b
multiplicacao = a * b
divisao = a / b
```

Exercício

- Crie uma variável com um valor inicial e depois altere o valor dessa variável, imprimindo seu valor antes e depois da modificação.

Exercício

- Crie uma variável com um valor inicial e depois altere o valor dessa variável, imprimindo seu valor antes e depois da modificação.

```
# Definição das variáveis
idade = 20
print("Idade inicial:", idade)
# Atualizando o valor da variável
idade = 25
print("Idade após atualização:", idade)
```

Exercício

- Crie duas variáveis x e y com valores diferentes e troque os valores entre elas sem usar uma variável temporária.

Exercício

- Crie duas variáveis x e y com valores diferentes e troque os valores entre elas sem usar uma variável temporária.

```
# Definição das variáveis
x = 5
y = 10
# Exibindo valores antes da troca
print("Antes da troca: x =", x, ", y =", y)
# Trocando os valores
x, y = y, x
# Exibindo valores após a troca
print("Após a troca: x =", x, ", y =", y)
```

Referencias

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