Roteiro

Primeiros Passos

Primeiros Comandos em Python

Tipos e Variáveis

Operadores

Conversões de Tipos

Entrada de Dados

Primeiros Passos

Versão do Python

- Neste curso, aprenderemos como programar utilizando a versão 3 da linguagem Python.
- Você pode verificar a versão do Python instalada no seu computador abrindo o terminal e digitando o comando:

```
python3 --version
```

- A resposta esperada para o comando deve ser:

```
Python 3.x.x
```

Ambiente Interativo do Python

- Nesse ambiente, é possível fornecer um comando ou bloco de comandos e verificar o resultado da execução.
- Para abrir o ambiente interativo basta digitar no terminal:

```
python3
```

- Quando o ambiente interativo é carregado algumas informações são exibidas e o Python fica aguardando algum comando para ser executado:

```
>>>
```

Primeiros Comandos em Python

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:

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

- Resposta obtida:

```
Hello world!
Hello world!
Hello world!
```

Imprimindo no Console

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

```
print("Unicamp")
print("MC102!")
```

```
Unicamp
MC102!
```

- Também podemos modificar isso utilizando o parâmetro `end`.

```
print("Unicamp", end = "")
print("MC102!")
```

```
UnicampMC102!
```

Imprimindo no Console

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

```
print("MC102", "Unicamp", "2022", sep = " - ", end = "!")
print("Novo Texto!")
```

```
MC102 - Unicamp - 2022!Novo Texto!
```

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

```
print("MC102", "Unicamp", "2022", sep = " - ", end = "!\n")
print("Novo Texto!")
```

```
MC102 - Unicamp - 2022!
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:

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

- Como resposta para o código acima, obtemos apenas:

```
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("MC102", "Unicamp", sep = " - ", end = "!\n")
5 # MC102 - Unicamp!
```

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("MC102", "Unicamp", sep = " - ", end = "!\n")
# MC102 - Unicamp!
```

Tipos e Variáveis

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`, `2022`).
`float`	Números reais (Exemplos: `-3.2`, `1.5`, `1e-8`, `3.2e5`).
`str`	Cadeia de caracteres/Strings (Exemplos: `"Unicamp"` e `"MC102"`).
`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:

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

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

```
1 a, b, c = 1, 2, 3
2 print(a, b, c)
3 # 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 Unicamp = "Universidade Estadual de Campinas"
2 print(Unicamp, type(Unicamp))
3 # Universidade Estadual de Campinas <class 'str'>
```

```
1 mc102_2022_2s = "MC102"
2 print(mc102_2022_2s, type(mc102_2022_2s))
3 # MC102 <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'>
```