Aula 04 – Introdução ao Python

TECNOLOGIA em **Gestão de Dados**

Programação I

Prof. Dr. Wener Sampaio

ReUni DIGITAL · EAD Centro de Educação Aberta e a Distância · UFPI Universidade Federal do Piauí · MEC

Sumário

- Características
- Modos de execução
- Tipos básicos
- Variáveis
- Palavras reservadas
- Operadores
  - Aritméticos, lógicos, relacionais, bit a bit
  - Mistura e conversão de tipos
- Operações com strings
- Sintaxe e semântica
- Atribuições

Características

- Altíssimo nível
- Interpretada
- Rápido aprendizado e codificação
- Grande biblioteca
- Multiparadigma
  - Imperativa, estruturada e orientada a objetos
- Portável

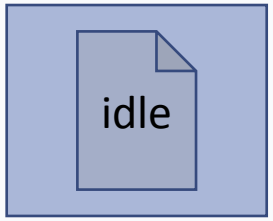
Modos de uso

- Interativo
  - Linha de comando (shell)
  - Interpretador imprime os resultados imediatamente
  - Prompt: >>>
  - Exemplo: python + enter

- Script
  - Escreve-se todo o programa em um editor de texto.
  - Arquivo com extensão “.py”.
  - O interpretador executa o arquivo.
  - Exemplo: IDLE, vscode, codeskulptor

Tipos básicos

- **int**
  - 4, 5, 100, -5
  - Precisão “infinita” (long)
- **float**
  - 3.1415, -6.999999
- **bool**
  - True, False
- **complex**
  - 4+6j
- **str**
  - “ufpi”, ‘CEAD’, “1.000,00”
- **Dica**: comando type(X)



Variáveis

- Não precisam ser declaradas
- Criadas através de atribuição: =
- Nomenclatura:
  - Qualquer tamanho
  - Podem conter: letras (Maiúsculas ou minúsculas), números, _
  - Não devem:
    - Conter espaços ou caracteres especiais: ç, ~, ´, ‘, “, %, &, (,), {, }, [,]
    - Iniciar com número
    - Ser palavras reservadas da linguagem
- Dicas:
  - Usar nomes significativos
  - Usar _ para nomes múltiplos: salario_base

Variáveis

- Exemplos corretos
  - a = 1
  - a= 1.0
  - A = 5+3j
  - print(A, A.real, A.imag)
  - A == a
  - teste = True
  - nome_aluno = ‘Exclebeuton’
- Exemplos incorretos
  - 1a = 1
  - setor funcionario= “RH”
  - return = 0.0
  - C& = “tilt”

Palavras reservadas

- 29 palavras reservadas

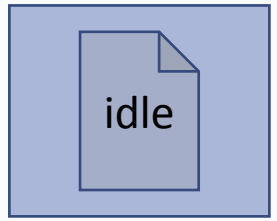
and def exec if not return
assert del finally import or try
break elif for in pass while
class else from is print yield
continue except global lambda raise

- Dica:

```
import keyword
print(keyword.kwlist)
```

Operadores aritméticos

Operador	Descrição	Exemplo
**	Potenciação	a**b
*	Multiplicação	a*b
/	Divisão real	a/b
//	Divisão inteira	a//b
%	Resto da divisão inteira	a%b
+	Soma	a+b
-	Subtração	a-b
-	Negativo	-a

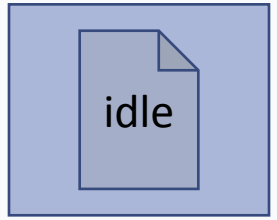


Mistura de tipos

- Conversão para o mais complexo
- >>> 1 / 3 (div. real)
  - 0.33333333
- >>> 1.0 / 3
  - 0.3333333333333331
- >>> 1.0 // 3
  - 0.0
- >>> 1 % 3
  - 1
- >>> 1.0 % 3
  - 1.0

Operadores binários

Operador	Descrição	Exemplo
~	Complemento bit a bit	~101 = 010
<<	Deslocar bits à esquerda	101 << 1 = 1010
>>	Deslocar bits à direita	101 >> 1 = 010
&	e bit a bit	101 & 011 = 001
^	ou bit a bit exclusivo	101 ^ 011 = 110
|	ou bit a bit	101 ^ 011 = 111



Operadores relacionais

Operador	Descrição	Exemplo
<	menor que	i < 100
<=	menor que ou igual a	i <= 100
>	maior que	i > 100
>=	maior que ou igual a	i >= 100
==	igualdade	i == 100
!=	diferente de (também <>)	i != 100

Operadores lógicos

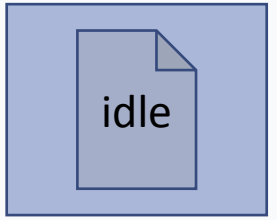
Operador	Descrição	Exemplo
not	negação lógica	not b
and	e lógico	(i <= 10) and (b == True)
or	ou lógico	(i < 100) or (f > 100.1)

Prioridades de operadores

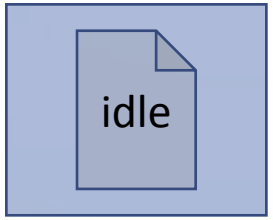
Prioridade	Operador	Descrição
1	**	Potenciação
2	~x	Operador bit a bit NOT
3	+x, -x	Positivo, Negativo
4	*, /, %	Multiplicação, Divisão e Resto
5	+, –	Adição e subtração
6	<<, >>	Deslocamentos de bits
7	&	Operador bit a bit AND
8	^	Operador bit a bit XOR
9	|	Operador bit a bit OR
10	<, <=, >, >=, !=, ==	Comparações
11	is, is not	teste de identidade
12	in, not in	teste de membros
13	not x	Operador boleano NOT
14	and	Operador boleano AND
15	or	Operador boleano OR

Operações com strings

- texto=“Olá mundo!”
- Strings não são números (erros)
  - texto -1
  - "Alô"/123
  - texto *"Alô“
  - "15"+2
- Concatenação (+)
  - x = "banana"
  - y = " com canela"
  - print(x + y)
- Repetição (*)
  - “blá ”*10



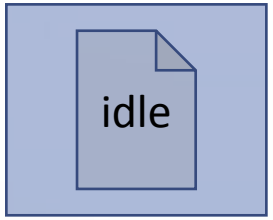
Atribuições



- Simples
  - x = 12
- Múltiplas
  - x = y = 100
  - a, b , c = 5, “UFPI”, 5-4j
- Compostas:
  - operação seguida de atribuição
  - x+=12
  - x-=2

op	Descrição
+=	Atualiza com a soma
-=	Atualiza com a subtração
*=	Atualiza com a multiplicação
/=, //=	Atualiza com a divisão
%=	Atualiza com resto

Atribuições



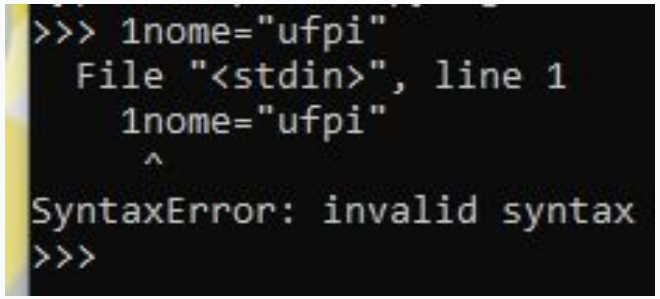
- Condicional
  - Várias linhas

```
valor = ""
if(teste):
    valor = 10
else:
    valor = -10
```

  - Uma linha
    - <variável> = <valorV> if(True) else <valorF>
    - x = 10 if(teste) else -10

Sintaxe e semântica

- Programar é a arte de corrigir erros!
  - Depurar (debug)
- Sintaxe
  - Refere-se à estrutura de um programa com suas regras
  - Em português: uma frase deve começar com uma letra maiúscula e terminar com um ponto.
  - Em python: 1nome=“ufpi”
- Na ocorrência
  - O programa será abortado.
  - Fácil encontrar.



Sintaxe e semântica

- Semântica
  - É a lógica do programa. Tem que fazer sentido.
  - Em português:
    - Um lenço caiu, e pelo barulho o mesmo é verde.
  - Em python:
    - salario_liquido= salario_bruto+descontos
- Na ocorrência
  - O programa tem a sintaxe perfeita, então será executado.
  - Não fará o que o programador pretendia.
  - Difícil encontrar.

Sintaxe e semântica

- Erros de runtime
  - São casos especiais e específicos de erros semânticos.
  - São erros que só ocorrem quando o programa está executando.
  - Exemplos:
    - Apagar um arquivo que está aberto.
    - Raiz quadrada de número negativo.
    - Divisão por zero.
    - Abrir um link inválido.
    - Acessar uma posição inexistente de um vetor.
  - São erros mais raros.

Comentários

- Mensagens ao programador.
- São ignorados pelo interpretador.
- Tipos:
  - Linha simples e linhas múltiplas

```
"""
  Este é um exemplo de um comentário de múltiplas
  linhas em Python. Note que são três aspas duplas
  no ínicio e final do comentário.
"""

# declara as variáveis auxiliares
valor1, valor2 = 5, 2

# efetua o cálculo
soma = valor1 + valor2

# exibe o resultado
print(soma) # um comentário aqui também
```

Quebras de linha

```
# Uma linha quebrada por contra-barra
a = 7 * 3 + \
5 / 2

# Uma lista (quebrada por vírgula)
b = ['a', 'b', 'c',
'd', 'e']

# Uma chamada de função (quebrada por vírgula)
c = range(1,
11)
```