

SSC0800 - Introdução à Ciência de Computação I

Operadores e Expressões

Prof.: Leonardo Tórtoro Pereira

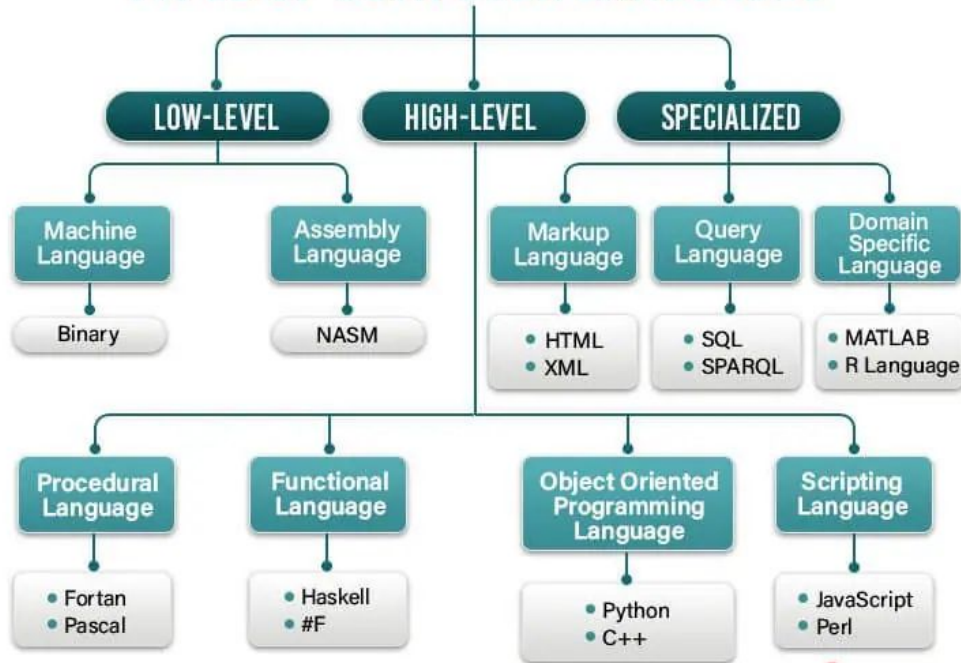
leonardop@usp.br

Baseado no material dos profs Fernando S. Osório e Claudio F.M. Toledo

Na aula passada...



TYPES OF COMPUTER LANGUAGES



O que vamos aprender hoje?



Objetivos

- Estudar os operadores de Python e como utilizá-los para formar expressões
- Utilizar a entrada e saída de dados para interação com o programa

Tópicos da Aula

- Operadores
- Expressões

Operadores

Operadores

→ Símbolos simples ou combinados que representam operações

- ◆ Aritmética
- ◆ Relacional
- ◆ Lógica
- ◆ De bits (bit-a-bit)

→ Cada uma tem uma precedência

→ Parênteses forçam uma ordem

Operadores

→ Podem ser classificados pela quantidade de elementos sob os quais incidem

- ◆ Unários
 - '-' – negativo ($-a$)
- ◆ Binários
 - '+' soma ($a+b$)
- ◆ Ternários...

Operadores Relacionais

Operador	Descrição	Exemplo	Resultado
>	Maior que	$2 > 3$	False
<	Menor que	$2 < 4$	True
>=	Maior igual que	$2 >= 2$	True
<=	Menor igual que	$4 <= 4$	True
!=	Diferente de	$\text{True} != \text{False}$	True
==	Igual a	$\text{True} == 1$	True

Operadores Relacionais

- Resultam em um booleano
 - ◆ True ou False
- Podem ser usados com strings para comparação
- Exemplos

Operadores Lógicos

Operador	Descrição	Exemplo	Resultado
not	Negação lógica	<code>not 2 == 2</code>	False
and	E lógico	<code>2 == 2 and 4==2</code>	False
or	OU lógico	<code>2 == 2 or 4==2</code>	True

Operadores Lógicos

- Resultado booleano (True, False)
- Exemplos

Operadores Aritméticos

Operador	Descrição	Exemplo	Resultado
+	Adição	$2 + 3$	5
-	Subtração	$2 - 2$	0
*	Multiplicação	$2 * 3$	6
/	Divisão	$5/4$	1.25
//	Divisão inteira	$4//3$	1
**	Exponenciação	$5**3$	125
%	Resto de Divisão (mod)	$10\%2$	0

Operadores Aritméticos

→ Resultado numérico (int ou float)

◆ '+' é sobrecarregado para concatenar strings

◆ '*' é sobrecarregado para duplicar a string

→ Exemplos

Operadores bit a bit

Operador	Descrição	Exemplo	Resultado
~	Complemento bit-a-bit	~0010	1101
<<	Deslocar bits à esq.	001 << 1	010
>>	Deslocar bits à dir.	100 >> 2	001
&	E (AND) bit-a-bit	001 & 101	001
^	OU exclusivo (XOR) bit-a-bit	001 ^ 101	100
	OU (OR) bit-a-bit	001 101	101

Operadores bit a bit

- Opera sobre os bits das variáveis
- ◆ Costumam ser mais rápidos que operações matemáticas
- Exemplos

Operadores de atribuição

→ Forma geral:

◆ `variavel = expressao ou constante`

→ Armazena o conteúdo à direita no elemento à esquerda

◆ `salario_minimo = 1500.00`

→ Múltiplas atribuições (todos recebem o mesmo valor)

◆ `x = y = z = 0`

→ Atribuição + Operador

◆ `x += 5`

Precedência de operadores

Operador	Precedência
()	+
**	
-	
* / % //	
+ -	
> >= < <= == !=	
not	
and	
or	
=	-

Expressões

Expressões

→ Expressões são compostas por:

◆ Operandos/Variáveis: `a`, `b`, `x`, `Meu_dado`, `2`, ...

◆ Operadores: `+`, `-`, `%`, ...

◆ Precedência: `()`

◆ Funções da biblioteca: `math.sin(x)`, `math.sqrt(x)`, ...

→ Sempre retornam um valor

◆ `X = 5 + 4` -> Retorna 9

◆ `((5 + 4) == 9)` -> Retorna True



Entrada e Saída de Dados

Entrada e Saída de Dados

- `input()` sempre lê dados como str (string)
 - ◆ `a = input("Valor de a")`
- Para converter em outros tipos, usar conversão explícita
 - ◆ `c = int(a) + int(b)`
 - ◆ `a = float(input("Valor de a"))`

Entrada e Saída de Dados

→ Pode-se exibir os resultados com *print()*

◆ a = "batata"

◆ print(a)

◆ b = 132

◆ print(b)

◆ c = 123.433632

◆ print("{:.3f}".format(c))

◆ print(format(c, ".2f"))

Entrada e Saída de Dados

→ Além da formatação no exemplo anterior, existem caracteres especiais na impressão

- ◆ `'\n'` pula uma linha
- ◆ `'\t'` tab
- ◆ `'\\'` backslash
- ◆ `'\"'` aspas simples - `'\"'` aspas duplas
- ◆ `print('Olá\nmundo')`

Entrada e Saída de Dados

→ Print também pode escrever dados em arquivos externos

```
f=open("teste.txt", "w")  
print("Escrevendo algo aqui", file=f, flush=True)  
f.close()
```

Exemplos

Strings

Strings

- Sequência de caracteres em uma espécie de vetor ou lista
 - ◆ Um ao lado do outro
 - ◆ Começando do índice 0, todas as letras podem ser acessadas com colchetes após a string
 - `a = "string"`
 - `a[0] -> s; a[1] -> t ... a[5] -> g`
 - ◆ Tamanho pode ser dado por `"len(a)"`
 - ◆ Números negativos "começam do fim"

Operações em Strings

- Comparação: verifica caractere por caractere
 - ◆ "Hello" > "Bye" -> True
- Verificar se string tem outra dentro: "in"
 - ◆ "hello" in "hello world" -> True
- '+' concatena as strings
 - ◆ "Hello" + "World" -> "HelloWorld"

Operações em Strings

- `'.'` itera sobre strings
 - ◆ `A = "Hello World";`
 - ◆ `A[:5]` -> "Hello" (do início até...)
 - ◆ `A[6:]` -> "World" (do índice até o fim)
- `'*'` repete "x" vezes uma string
 - ◆ `"Hello" * 3` -> "HelloHelloHello"
- `ord(a[0])` -> retorna inteiro ASCII
- `char(65)` -> retorna caractere ASCII

Exemplos

Referências

Referências

- <https://www.learnpython.org/>
- <https://www.w3schools.com/python/>
- <https://panda.ime.usp.br/cc110/static/cc110/index.html>
- https://www.youtube.com/playlist?list=PLcoJJSvnDgcKpOi_UeneTNTIV0igRQwcn