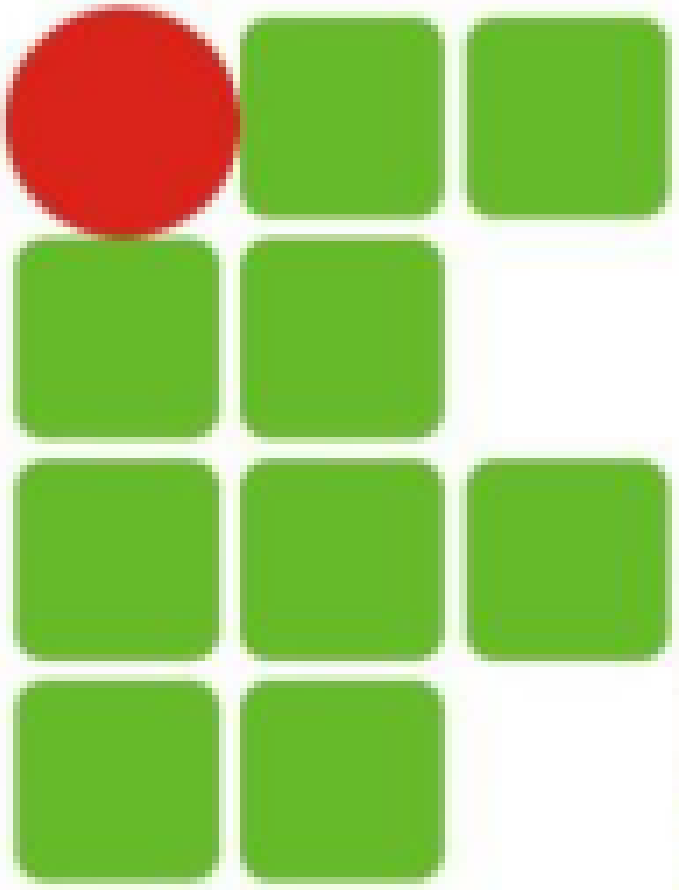
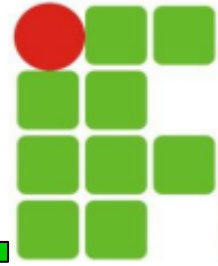


Python: Variáveis e expressões

Galileu Batista de Sousa
Galileu.batista -at +ifrn -edu +br

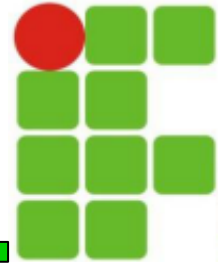
Constantes



- São valores fixos em um programa
  - Seus valores não mudam !!!
- Podem ser:
  - Inteiros (ou int): 123
  - Reais (ou float): 12.3
  - Texto (ou str): "abc"
    - Podem ser com apóstofos.

```
>>> print (123)
123
>>> print (12.3)
12.3
>>> print ("ola mundo!!!")
Ola mundo!!!
```

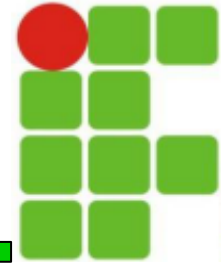
Variáveis



- São posições de memória onde guardamos valores
  - Lembrar o endereço seria terrível
    - Você sabe o número telefônico de sua mãe?
- Ao invés de lembrar endereços
  - Python permite dar um nome ao endereço
  - Uma espécie de agenda
- Comando de armazenamento (=)
  - **Atribuição** é o jargão usado

```
>>> x = 2
>>> y = x + 2.3
>>> print (y)
4.3
>>> x = 10
>>> print (x)
10
```

Nomes de variáveis



- Devem iniciar com letra ou _
- Podem ter letras, números ou _
- São sensíveis maiúsculos / minúsculos

_

OK:	`x`	`nome2`	`salario`	`cpf`
Ruim:	`x'`	`x.2`	`2x`	`#cpf`
Distintos:	`nome`	`Nome`	`NOME`	`noME`

Escolha bons nomes - mnemônicos

```
>>> xy76rx46_2 = 10
>>> r4j5ntd23_ = 5
>>> r4j5mtd23_ = (r4j5ntd23_ * xy76rx46_2) / 2
>>> print (r4j5mtd23_)
```

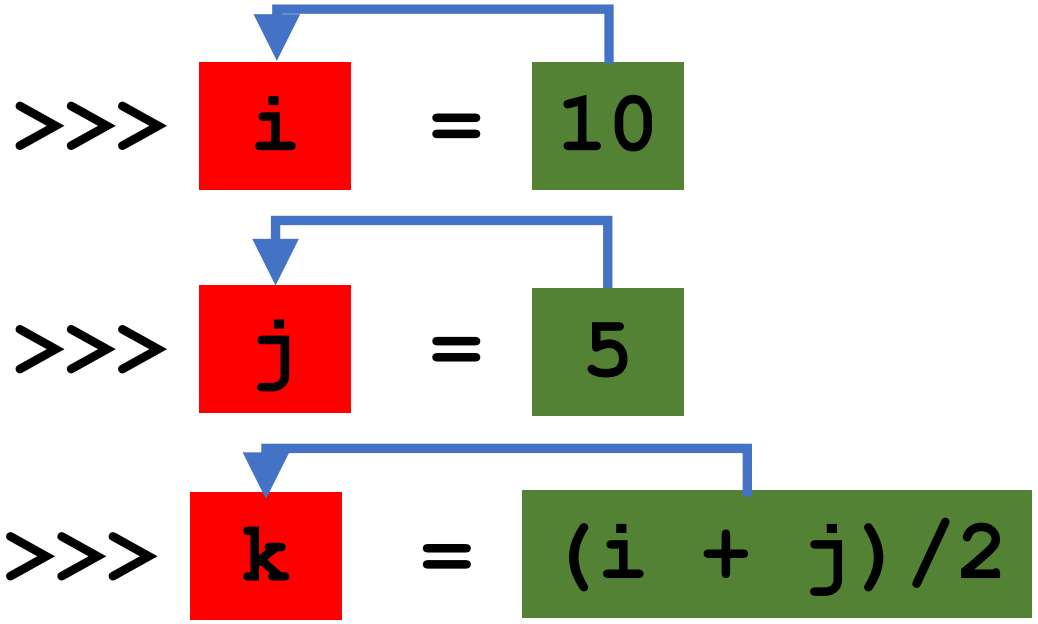
```
>>> i = 10
>>> j = 5
>>> k = (i * j) / 2
>>> print (k)
```

E agora?

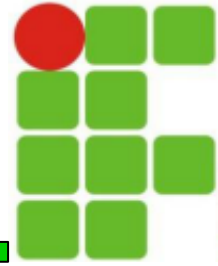
```
>>> base = 10
>>> altura = 5
>>> area_triangulo = (base * altura) / 2
>>> print (area_triangulo)
```

Atribuição

- Guarda o valor de uma expressão na memória
  - Na prática coloca um valor numa variável
  - O valor anterior é perdido
- Usa o símbolo =
  - Por tradição – não é igualdade
- O lado direito do = é **sempre** resolvido antes



Expressões aritméticas



- Avaliadas antes do seu uso:
  - Em uma atribuição por exemplo:
- Principais operadores:

+	soma
-	subtração
*	multiplicação
/	divisão
//	divisão inteira
%	resto da divisão
**	potenciação

Precedência

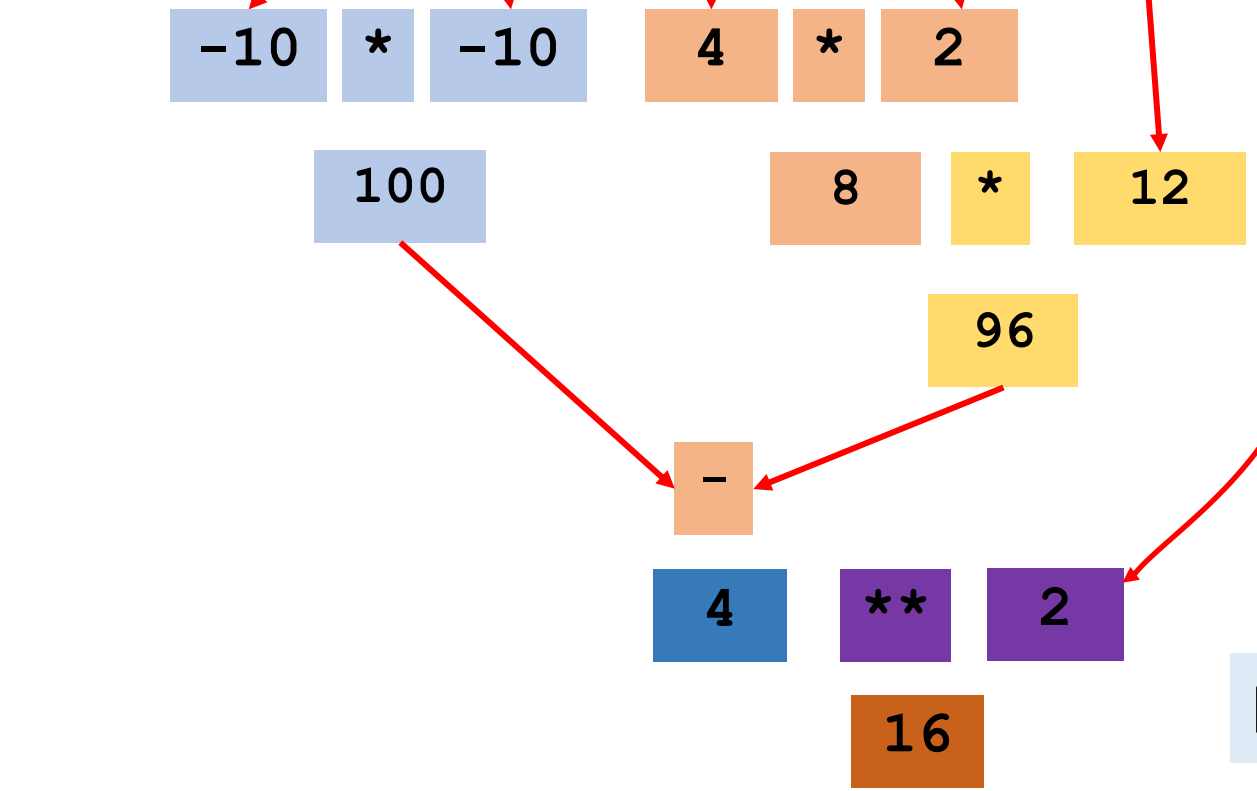
()	**é a maior**
**	a seguinte
* / //	a seguinte
+ -	é a menor

Associatividade

**	à direita
outros	à esquerda

Expressões aritméticas

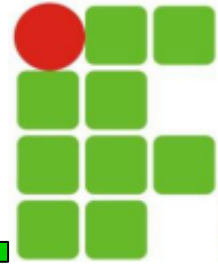
```
>>> a = 2
>>> b = -10
>>> c = 12
>>> x = (b * b  -  4  *  a  *  c) ** 2
```



Tente:
k = 6 * 6 / 6 * 6

Finalmente, armazena 16 em x

Exibição de dados



- No interpretador se uma expressão não é armazenada, é exibida

```
>>> 5 + 2
7
```

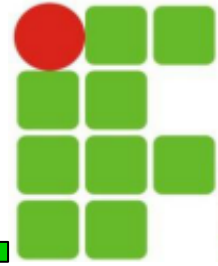
```
>>> 6 * 6 / 6 * 6
36
```

- Em *scripts* deve-se usar **print()**

```
>>> res = 6 * 6 / 6 * 6
>>> print ("Resultado: ", res)
Resultado: 36
>>> res
36
```

  - Textos são mostrados literalmente, variáveis os seus conteúdos
  - Várias expressões podem ser exibidas, separar parâmetros por **,**

Entrada de dados



- O comando para ler um dado do console é: **input()**

```
>>> print ("Digite seu nome")
>>> nome = input()
>>> print ("Olá ", nome, ", tudo bem?")
```

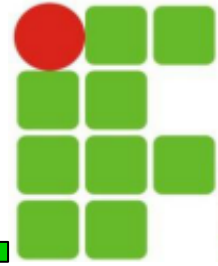
```
>>> nome = input("Digite seu nome")
>>> print ("Olá ", nome, ", tudo bem?")
```

- O resultado de input() é sempre do tipo texto (str)

```
>>> num = input ("Digite um número: ")
>>> dobro = 2 * num
>>> print ("O dobro de", num, "é", dobro)
```

Erro ?
Por que?

Conversão de tipos



- É possível converter dados de um tipo para outro

Função	Significado	Exemplo
`int (...)`	Converte de texto (string) para inteiro	**num = int (“123”)**
`float (...)`	Converte de texto (string) para real (float)	**num = float (“12.3”)**
`str (...)`	Converte int/float/... para texto (string)	**text = str(12.3)**

- Bastante usado em conjunto com input()

```
>>> num = int(input (“Digite um número: ”))
>>> dobro = 2 * num
>>> print (“O dobro de”, num, “é”, dobro)
```