IBGE
Instituto Brasileiro de Geografia e Estatística

Escola Nacional de
Ciências Estatísticas

Introdução à Programação

Aula 02: Programação em Python:
Variáveis e Entrada / Saída Básicas

Prof. Eduardo Corrêa

Data 12/03/2024

Tópicos da Aula

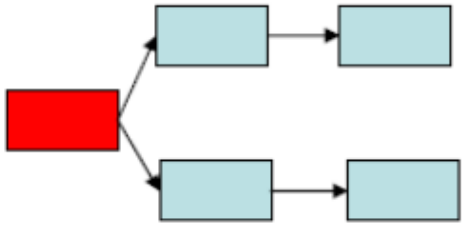
- Temas desta aula:
  - Revisão – Algoritmos
  - Variáveis
  - Comentários
  - Operadores Aritméticos (introdução)
  - Entrada e Saída Básicas

Algoritmos (1/3)

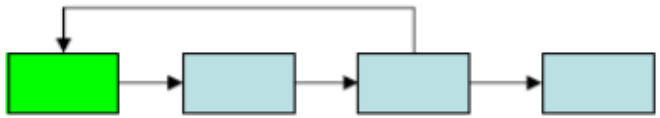
- Um algoritmo é uma **sequência** de passos. Cada passo é de um dos três tipos:

1. Uma operação elementar.

2. Uma operação de controle especificando uma **seleção** entre uma sequência de passos.



3. Uma operação de controle especificando uma **repetição** entre uma sequência de passos.



Algoritmos (2/3)

- Através dos algoritmos, na última aula iniciamos nosso contato com a **lógica de programação**.
  - Lógica de programação = forma de encadear corretamente pensamentos e separá-los em uma sequência de passos que sejam capazes de solucionar um problema.

Algoritmos (3/3)

- Um programa de computador é um algoritmo escrito em uma dada linguagem de programação.
- A partir desta aula, vamos começar a utilizar a linguagem Python para escrever programas ("materializar" algoritmos).
- Iniciaremos com programas simples, sem alternativa ou repetição, ou seja, compostos apenas por operações elementares.

Programação Python

- **Python**: linguagem de alto nível desenvolvida no início da década de 1990.
- Características Principais:
  - ☺ É uma linguagem de **propósito geral** (podemos fazer qualquer tipo de programa com ela)
  - ☺ É, talvez, a mais **didática** de todas as linguagens.
  - ☺ É uma das **mais utilizadas** em sistemas comerciais e sistemas acadêmicos.
  - ☺ É uma das mais utilizadas para **estatística**.

Estrutura de um Programa Python

- Um programa Python é uma sequência de instruções.

```
instrução1
instrução2
...
instruçãon
```

- As instruções serão **executadas** pelo computador **na ordem** em que estiverem dispostas no programa.
- Quase sempre, cada linha do programa corresponde a uma instrução.
  - Dá para colocar mais de uma instrução em uma única linha (separando-as por ponto-e-vírgula), mas isso raramente é feito, pois prejudica a legibilidade do programa.

Variáveis (1/3)

- O conceito de **variável** é um dos mais importantes relacionados à programação de computadores.
- Entenda uma variável da seguinte forma:

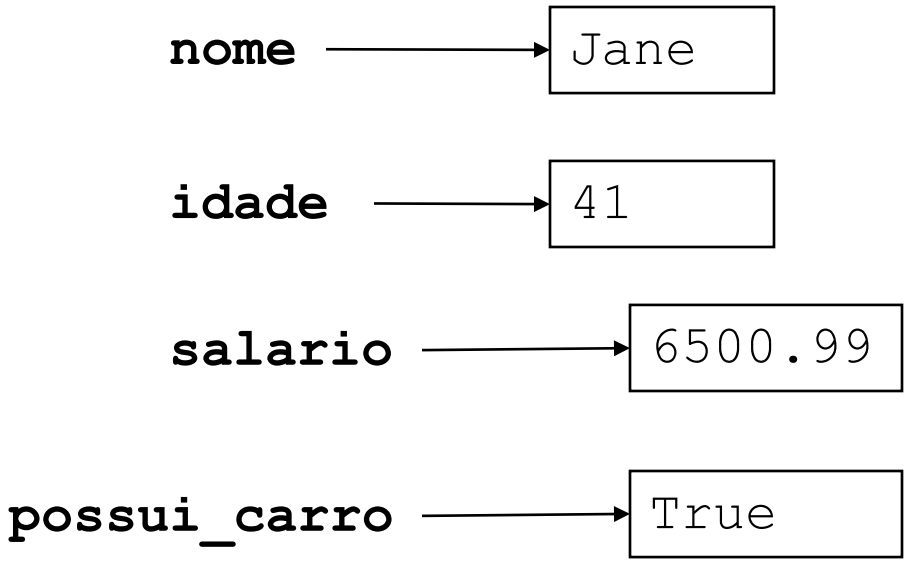
  Um local onde é possível guardar algum valor.

- Este valor pode:
  - Ter sido especificado como **entrada** pelo usuário (ex.: valor do raio da esfera, digitado pelo teclado).
  - Ter sido resultante de um **processamento** do programa (ex.: valor calculado do volume da esfera).

Variáveis (2/3)

- Para simplificar ainda mais:

A variável é como uma caixinha que possui um nome e guarda algum valor.

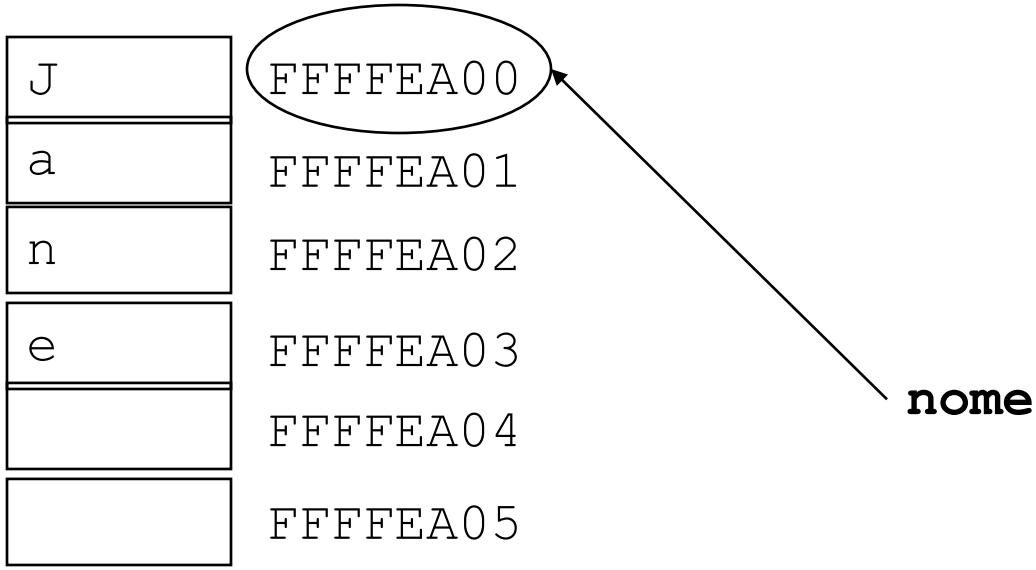


Variáveis (3/3)

- Do ponto de vista da arquitetura do computador:

  A variável representa um endereço da memória principal do computador em que um valor está armazenado

- Ou seja: é um "apelido" para uma posição de memória.



Atribuição

- Atribuição é a operação elementar que consiste em armazenar um dado (valor) em uma variável.
  - Operador de atribuição: =

```
nome = 'Jane'
idade = 41
salario = 6500.99
possui_carro = True
```

Atribuição – Programa Exemplo (1/6)

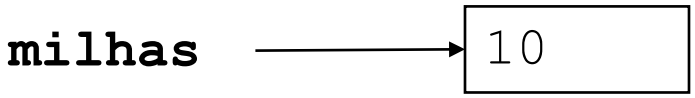
```
milhas = 10
km = milhas * 1.61
milhas = 20
km = milhas * 1.61
```

- Este é um programa "bobo" que não recebe nenhum dado como entrada e nem gera saída.
- Ele serve apenas para demonstrar o que acontece quando eu declaro e atribuo valores a variáveis.
- A seguir apresentaremos uma simulação da execução do programa.

Atribuição – Programa Exemplo (2/6)

```
milhas = 10
km = milhas * 1.61
milhas = 20
km = milhas * 1.61
```

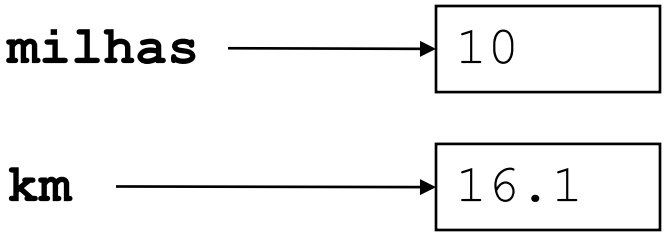
- Nesta linha a variável milhas é criada e tem o valor 10 atribuído.
- Quando uma variável é criada, você pode **imaginar** que o computador define uma "caixinha".
  - O nome da caixinha é o nome da variável e o conteúdo é o seu valor.



Atribuição – Programa Exemplo (3/6)

```
milhas = 10
km = milhas * 1.61
milhas = 20
km = milhas * 1.61
```

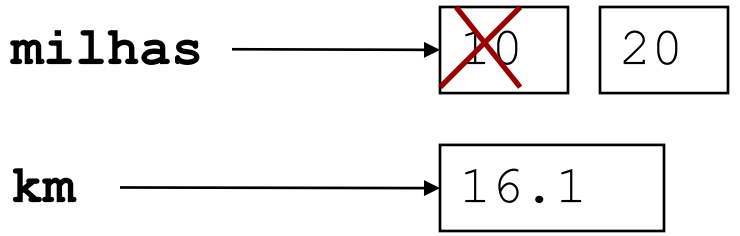
- Nesta linha atribui-se o valor milhas * 1.61 para uma nova variável, chamada km.
- Como o conteúdo atual de milhas é 10, então o valor de km é obtido por 10 x 1.61 = 16.1.
- Este é um exemplo de atribuição via operação aritmética.



Atribuição – Programa Exemplo (4/6)

```
milhas = 10
km = milhas * 1.61
milhas = 20
km = milhas * 1.61
```

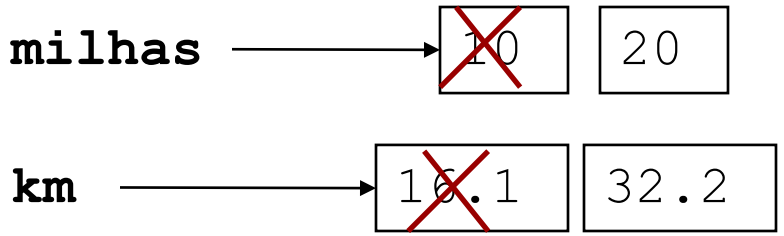
- Esta linha muda o conteúdo de milhas de 10 para 20.
- O conteúdo de uma variável **pode ser modificado** em qualquer passo (linha) do programa.
- Quando eu atribuo um novo conteúdo para uma variável, o conteúdo antigo é destruído.



Atribuição – Programa Exemplo (5/6)

```
milhas = 10
km = milhas * 1.61
milhas = 20
km = milhas * 1.61
```

- Esta linha muda o conteúdo de km.
- Como o valor atual de milhas é 20, o novo valor de km. passa a ser 20 x 1.61 = 32.2.
- Nunca esqueça: os comandos do programa são executados na sequência em que são especificados no programa.



Atribuição – Programa Exemplo (6/6)

- Se desejar imprimir o conteúdo das variáveis, utilize a função print() – que é a função de saída do Python.
  - Explicaremos o uso básico de print() ainda nessa aula.

```
milhas = 10
km = milhas * 1.61
print(milhas, km)

milhas = 20
km = milhas * 1.61
print(milhas, km)
```

Saída:

```
10 16.1
20 32.2
```

Nomenclatura de Variáveis (1/3)

- Para escolher o nome de uma variável, é preciso respeitar as seguintes regras:
  - Os nomes podem conter **letras**, **números** ou o caractere especial **_** (**sublinhado** ou *underscore*).
  - **Não** podem ser iniciados por um número.
  - E até podem ser iniciados por *underscore,* mas apenas em situações específicas (portanto, evite fazer isso por enquanto!)
- Nomes **Válidos**:
  - peso, altura, x, y, z, km, Milhas, zzz, PI
  - salario_anual, tot_renda, z_z_z
  - n1, n2, z10, H999
- Nomes **Inválidos**:
  - 1n, 10_abc, 2altura (começam com número)
  - *peso, s@al@rio, C&A, z$ (possuem caractere especial)
  - salario anual (possui espaço em branco)

Nomenclatura de Variáveis (2/3)

- Há **diferenciação** entre letras **maiúsculas** e **minúsculas**.
  - Ou seja: se você nomear uma variável como x, não poderá referenciá-la como X.

- As palavras a seguir são **palavras reservadas** do Python e **não podem** ser utilizadas como nomes de variáveis.

```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

- **Não** crie uma variável com o nome de uma função ou tipo do Python, como print, input ou bool. Se você fizer isso, você vai anular a função ou tipo e terá que reiniciar o Python.

Nomenclatura de Variáveis (3/3)

- A documentação do Python recomenda que todas as variáveis devam ter o nome especificado em minúsculo e que *underscore* (_) seja utilizado para separar nomes compostos (padrão conhecido como ***snake case***)
  - idade
  - renda_media_anual
  - codigo_ocupação
  - produtos_2024
  - *Não é obrigação, apenas uma recomendação*.
- Outra recomendação é **evitar** o uso de **caracteres acentuados** no nome das variáveis.
  - *No conteúdo não tem problema, mas no nome da variável não é uma boa ideia*.

Comentários (1/3)

- **Comentários** são textos ou frases definidos com o uso do símbolo **#**.
  - Eles são ignorados pelo interpretador Python na hora de gerar o programa em linguagem de máquina.
  - Servem apenas para **documentação do programa** (para serem lidos pelos humanos que estiverem analisando ou alterando um programa).

```
# programa usado para
# introduzir o conceito de variável
milhas = 10
km = milhas * 1.61
milhas = 20
km = milhas * 1.61
```

Comentários (2/3)

- Na prática, é interessante utilizar comentários para:
  - **Descrever o objetivo** de um programa (neste caso, podem ser colocados nas linhas iniciais);
  - **Explicar partes** mais **complicadas**;
  - **Documentar** as **variáveis**.
- Comentários são uma excelente ferramenta para deixar o seu programa bem organizado e documentado,
  - Mas utilize-os com bom senso! Você não precisa exagerar. Por exemplo, o comentário abaixo é desnecessário, pois é óbvio:

```
x = 0    # atribui o valor 0 para a variável x
```

Comentários (3/3)

- Também é possível especificar comentários entre três aspas: """ (veja o exemplo abaixo)
  - Isso é especialmente útil para comentários que envolvem múltiplas linhas.

```
"""
programa usado para
introduzir o conceito de variável
"""
milhas = 10
km = milhas * 1.61
milhas = 20
km = milhas * 1.61
```

Tipo de Dado

- Conforme visto, a declaração de uma variável consiste em especificar seu **nome** e o seu **valor** (conteúdo, que também é chamado de **literal**).
  - O valor determinará o **tipo** (ou **tipo de dado**) da variável.
  - Toda variável possui um tipo que indica a **espécie de dado** que ela armazena.

```
nome = 'Jane'
idade = 41
salario = 6500.99
possui_carro = True
```

- Em Python há 4 tipos básicos: **str**, **int**, **float** e **bool**.

tipo str

- As variáveis do tipo **str** – abreviação de string – podem armazenar qualquer **informação alfanumérica**.
  - Ou seja: **textos** de qualquer tamanho, incluindo textos que possuem números.
  - **Exemplos**:
    - nome, endereço, profissão, e-mail, placa do carro etc.
- Para definir uma variável do tipo str, realiza-se a atribuição colocando o **valor entre aspas simples** ou **duplas**.

```
letra = 'A'
endereco = "Rua André Cavalcanti, 106"
empresa = 'ACME LTDA'
s = '@'
```

tipo int

- As variáveis do tipo **int** (inteiro) podem armazenar **números inteiros** negativos ou positivos.
  - **Exemplos**:
    - idade, quantidade de filhos, população total, quantidade de gols etc.
- Para criar uma variável int, basta indicar os dígitos que formam o número.
- E como podemos declarar valores negativos?
  - É simples, basta usar o sinal de menos, por exemplo, z = -99.

```
x = 0
y = 1000
k = -1
```

tipo float

- As variáveis do tipo **float** podem armazenar um número real (fracionário), seja ele negativo, nulo ou positivo.
- **Exemplos**:
  - preço, salário, taxa de emprego, percentual de aprovados etc.
- Veja que o ponto "." é utilizado para definir a parte decimal, e não a vírgula "," – as linguagens de programação seguem o padrão do idioma inglês.

```
x = 0.5
constante_de_napier = 2.72
w = 1.888888
preco = 1.99
variacao = -1.1
```

tipo bool

- As variáveis do tipo **bool** (*boolean*) podem armazenar apenas um destes dois valores: True (verdadeiro) ou False (falso).
- Em Português é chamado de **tipo lógico**. Deve ser utilizado quando a informação reflete uma característica que só pode ser verdadeira ou falsa.
  - **Por exemplo**: Possui Carro? Possui Casa Própria? Tem cartão de crédito? Foi aprovado na disciplina?
  - Embora simples, é um tipo de dado muito importante na programação (como você verá nesse curso).
  - **Obs.**: **T**rue deve ser escrito com "T" maiúsculo e sem aspas, assim como **F**alse com "F" maiúsculo e sem aspas.

```
possui_filhos = False
casado = True
```

Operações Aritméticas (1/5)

- Os programas de computador são comumente empregados para a realização de cálculos.
- A tabela abaixo apresenta os principais operadores aritméticos do Python.
  - Eles permitem a realização de cálculos sobre números inteiros (int) ou reais (float).

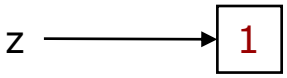
Operador	*Operando*	*Resultado*	*Operador*
Adição	Inteiro ou Real	Inteiro ou Real	+
Subtração	Inteiro ou Real	Inteiro ou Real	-
Multiplicação	Inteiro ou Real	Inteiro ou Real	*
Divisão	Inteiro ou Real	Real	/
Exponenciação	Inteiro ou Real	Inteiro ou Real	**
Quociente da Divisão	Inteiro ou Real	Inteiro ou Real	//
Resto da Divisão	Inteiro ou Real	Inteiro ou Real	%

- Mais detalhes serão fornecidos na próxima aula.

Operações Aritméticas (2/5)

- Simulação de execução de um programa com operações aritméticas.

```
z = 1
z = z + 1
z  = z * 2
z  = z % 3
```



Operações Aritméticas (3/5)

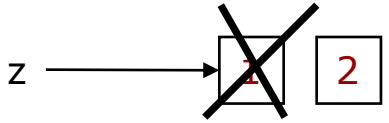
- Esta operação **significa**: somar um ao valor de z e armazenar o resultado na própria variável z.

```
z = 1

z = z + 1  ⇦

z  = z * 2

z  = z % 3
```



- Agora o conteúdo de z é 2.

Operações Aritméticas (4/5)

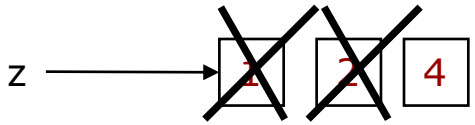
- Esta operação **significa**: multiplicar o valor de z por 2 e armazenar o resultado na própria variável z.

```
z = 1

z = z + 1

z  = z * 2

z  = z % 3
```



- Agora o conteúdo de z é 4.

Operações Aritméticas (5/5)

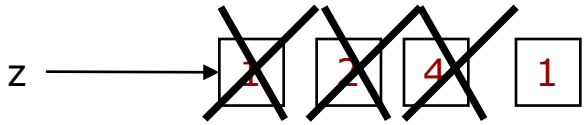
- Esta operação **significa**: obter o resto da divisão de z por 3 e armazenar na própria variável z.

```
z = 1

z = z + 1

z  = z * 2

z  = z % 3
```



- No final do programa, o valor (conteúdo) de z é igual a 1.

Entrada de Dados (1/5)

- Uma instrução de **entrada** (ou **leitura**) serve para permitir como que o programa possa **receber dados** do **ambiente externo** (dados especificados pelo usuário).
  - Ou seja: permite que o usuário digite dados, possibilitando um **diálogo com o computador**.
  - Python possui uma única função para entrada via teclado: **input()**
  - Ler uma informação significa: copiar o valor de entrada para uma variável do programa.

Entrada de Dados (2/5)

nome = **input**()

- Esta instrução significa:
  - "leia um valor para a variável nome".
  - Quando o computador "vê" esse comando, fica esperando o usuário digitar uma informação.
  - O computador não passa para a instrução seguinte do programa enquanto o usuário não digitar.
- A operação de leitura também representa uma atribuição, pois ela cria ou modifica o valor de uma variável.

Entrada de Dados (3/5)

- Abaixo, um programa com input() e um exemplo possível de execução.

```
print('Qual o seu nome?')
nome = input()
print('Hmmm... então você é o famoso', nome)
```

```
>>>
Qual o seu nome?
Rakesh
Hmmm... então você é o famoso Rakesh
```

- A função input() também pode ser utilizada com um texto como argumento, para que seja possível fazer uma pergunta ao usuário sem precisar usar antes a função print().

```
nome = input('Qual o seu nome?')
print('Hmmm... então você é o famoso', nome)
```

Entrada de Dados (4/5)

- **IMPORTANTE**: o valor retornado pela função input() será **sempre uma string** contendo os caracteres que o usuário digitar do teclado.
  - Isso acontecerá mesmo que ele digite apenas números.
  - Logo, a variável usada em conjunto com o input() no lado esquerdo da atribuição será uma variável do tipo str.
  - Por isso, sempre ocorrerá um erro na execução do programa abaixo. Não é possível somar 1 a uma string !

```
# O input() sempre gera uma string !!!
n = input('Digite um número: ')
print('O sucessor do número é:', n + 1)
```

```
>>>
Digite um número: 3
TypeError: can only concatenate str (not "int") to str
```

Entrada de Dados (5/5)

- Para resolver, você deverá usar uma das 2 funções abaixo **em conjunto** com input().
  - int(): recebe uma informação como entrada e tenta convertê-la para o tipo inteiro caso seja possível. Se a conversão falhar, todo o programa também falhará.
  - float(): funciona da mesma maneira, porém converte a informação de entrada para o tipo float.

```
# exemplo de uso da função int() com input()
n = int(input('Digite um número: '))
print('O sucessor do número é:', n + 1)
```

```
>>>
Digite um número: 3
O sucessor do número é: 4
```

Saída de Dados (1/3)

- Uma **instrução de saída** (ou **escrita**) **envia dados** para o usuário através de um dispositivo de saída (monitor, impressora etc.).
- Python possui uma única função para saída na tela: **print()**
- Uma instrução de saída pode escrever 3 coisas:
  - uma **mensagem**
  - ou o **valor de uma variável**
  - ou o **resultado de uma operação aritmética**.

Saída de Dados (2/3)

print('Hello World')

- Imprime a mensagem "*Hello World*" na tela.
- A mensagem a ser impressa tem que estar entre aspas simples.

print(nome)

- Imprime o valor da variável nome na tela (**não** imprime a mensagem "nome"!!!).
- A variável **não pode** estar entre aspas.

print(10 * 2);

- Imprime 20 na tela.
- A operação aritmética **não pode** estar entre aspas.

Saída de Dados (3/3)

- É possível utilizar um mesmo print() para **escrever várias coisas**
  - Basta separá-las por vírgula.
  - Um espaço em branco será utilizado para separar cada informação no momento em que forem exibidas na tela.
- No exemplo abaixo, o mesmo print() é usado para escrever uma mensagem, o valor de uma variável e o resultado de um cálculo.

```
# Escrevendo várias coisas em um único print()
x = 10
print('o cubo de', x, 'é', x ** 3)
```

```
>>>
o cubo de 10 é 1000
```

Exemplo Completo

```
print('* * Programa Conversor - Milhas -> Km * *')
print()
print('Digite o valor da distância em Milhas:')
milhas = float(input())
km = milhas * 1.61
print('O valor equivalente em km é:', km)
```

- Este programa solicita ao usuário o valor de uma distância em **milhas** e o converte para **quilômetros**.
- O programa se "comunica" com o usuário com o uso das funções print() e input().

Exemplo Completo

milhas_para_km.py ×

```
1  print('* * Programa Conversor - Milhas -> Km * *')
2  print()
3  print('Digite o valor da distância em Milhas:')
4  milhas = float(input())
5  km = milhas * 1.61
6  print('O valor equivalente em km é:', km)
7
```

Shell ×

```
>>> %Run milhas_para_km.py
 * * Programa Conversor - Milhas -> Km * *

 Digite o valor da distância em Milhas:
 1000
 O valor equivalente em km é: 1610.0
>>>
```

- Acima um exemplo de execução, onde o usuário digitou o valor 1000 para a distância em milhas.

Exemplo Completo

```
print('* * Programa Conversor - Milhas -> Km * *')
print()
print('Digite o valor da distância em Milhas:')
milhas = float(input())
km = milhas * 1.61
print('O valor equivalente em km é:', km)
```

- Exibe mensagem para o usuário que informa sobre o propósito do programa.

Exemplo Completo

```
print('* * Programa Conversor - Milhas -> Km * *')
print()
print('Digite o valor da distância em Milhas:')
milhas = float(input())
km = milhas * 1.61
print('O valor equivalente em km é:', km)
```

- Esta linha foi colocada por motivos "estéticos".
  - O print() sem parâmetros faz com que uma linha em branco seja impressa na tela.

Exemplo Completo

```
print('* * Programa Conversor - Milhas -> Km * *')
print()
print('Digite o valor da distância em Milhas:')
milhas = float(input())
km = milhas * 1.61
print('O valor equivalente em km é:', km)
```

- Mensagem pedindo que usuário digite o valor da distância em milhas.

Exemplo Completo

```
print('* * Programa Conversor – Milhas -> Km * *')
print()
print('Digite o valor da distância em Milhas:')
milhas = float(input())
km = milhas * 1.61
print('O valor equivalente em km é:', km)
```

- Recebe valor digitado pelo usuário. Em seguida converte esse valor para float (*ver slide 38*)
- Então, o armazena na variável milhas.
- input() é a função para fazer a entrada via teclado. Quando o computador "vê" essa instrução ele "sabe" que deve esperar o usuário digitar um valor no teclado.
- E como o input() está em uma atribuição, ele "sabe" que deve copiar este valor para a variável especificada pelo usuário após sua conversão para float.

Exemplo Completo

```
print('* * Programa Conversor – Milhas -> Km * *')
print()
print('Digite o valor da distância em Milhas:')
milhas = float(input())
km = milhas * 1.61
print('O valor equivalente em km é:', km)
```

- Aqui temos uma operação aritmética e uma atribuição.
  - No lado direito, a operação aritmética que converte o valor digitado em milhas para quilômetros.
  - O resultado é armazenado em uma nova variável chamada km, com o uso do operador de atribuição "="

Exemplo Completo

```
print('* * Programa Conversor – Milhas -> Km * *')
print()
print('Digite o valor da distância em Milhas:')
milhas = float(input())
km = milhas * 1.61
print('O valor equivalente em km é:', km)
```

- Imprime uma mensagem e o valor armazenado na variável km.
- Note que a mensagem é especificada entre aspas, depois usa-se o caractere virgula e o identificador (variável) km (a vírgula é utilizada para separar as coisas que desejamos imprimir no print()).