

Funções em Python

- **Funções** são blocos de instrução que podem ser invocados de qualquer parte do nosso código;
- Toda função, por definição, possui um nome, pode receber **parâmetros** e pode **retornar valores**;
- Nem toda função receberá parâmetros, da mesma forma, que nem toda função retornará valores.

Funções em Python

- Funções estão ligadas a uma **rotina**, ou seja, são tarefas que você faz constantemente em seu dia-a-dia. Como por exemplo: acordar, tomar café...
- Agora imagina em um ambiente de programação...

QUE FUNÇÕES VOCÊ FAZ CONSTANTEMENTE EM PYTHON



Funções em Python

Print() ←
→ len()
Input()
int() —
Float()

- Sim, todas são funções que já vem dentro do Python e que utilizamos desde a primeira aula. Mas e se quisermos criar uma nova função com uma nova funcionalidade?

Funções em Python

- Podemos sim, criar uma nova função. Vejamos um exemplo simples, no qual você já deve ter visto em alguns exercícios resolvidos.
- Vamos criar uma função chamada **mostralinha**

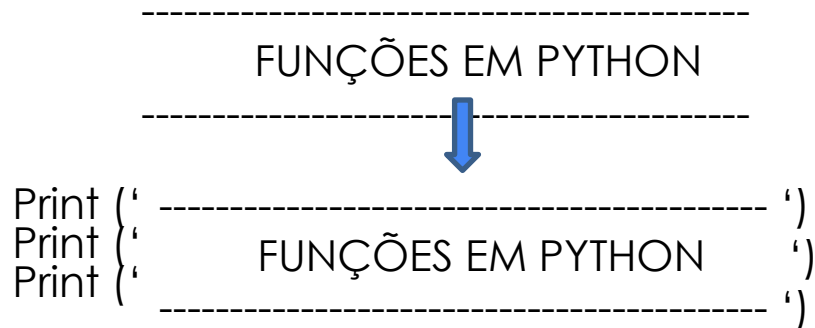
mostralinha() →

Toda função é acompanhada do parênteses ()

Esta é a linha na qual se deseja que a função mostralinha() faça no programa

Funções em Python

- Vejamos a sua representação:



- Em Python para poder expressar as linhas em tela é necessário chamar a função print para cada linha. Mas em funções podemos trabalhar com um comando reservado chamado **def**.

Funções em Python

Comando Def

- Para declararmos uma função utilizamos a palavra reservada **def**.

Declaração da função em Python

Função criada

Definição da função

```
def mostralinha():  
    print('-----')
```

Função sem parâmetro

O QUE SÃO PARÂMETROS ?



Funções em Python

Parâmetros

- Parâmetros são Informações passadas para as funções;
- Em Python trabalhamos com funções com parâmetros e sem parâmetros.

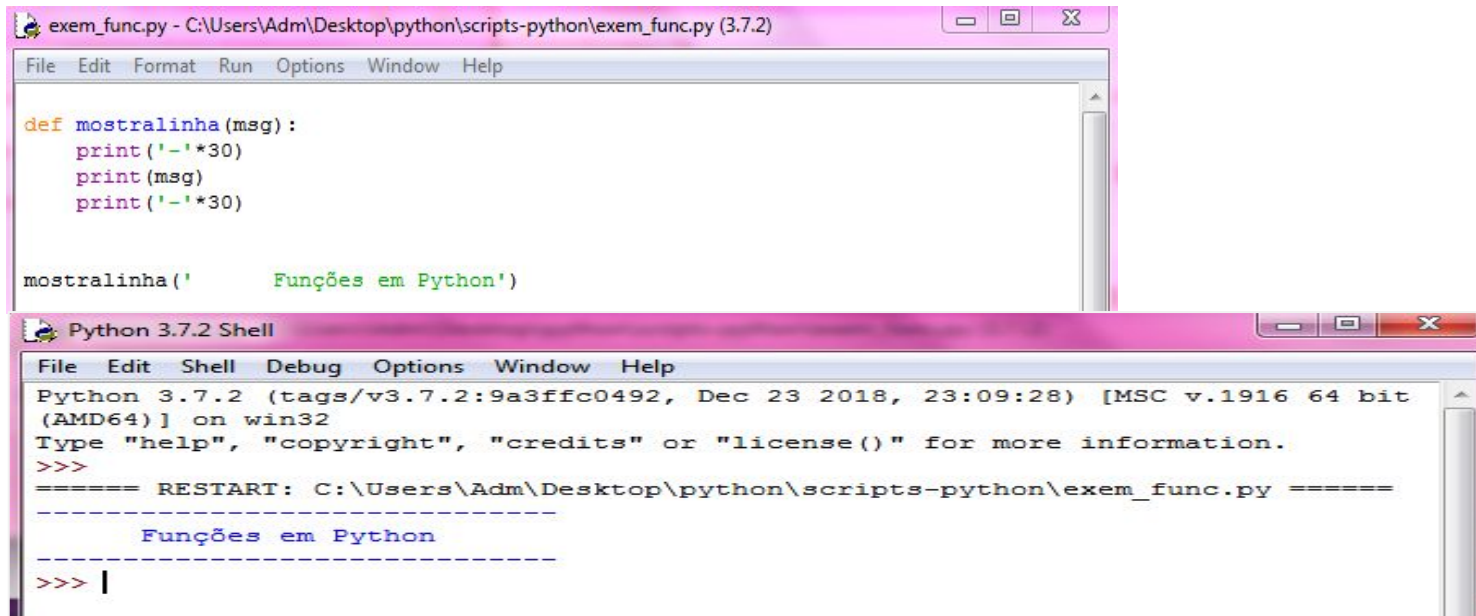
Com parâmetro	Sem parâmetro
mostralinha(msg)	mostralinha()



Msg é o parâmetro adicionado para a função

Funções em Python

- Vamos ver como funciona o programa em Python com parâmetro:



The image shows two windows from a Python IDE. The top window, titled 'exem_func.py - C:\Users\Adm\Desktop\python\scripts-python\exem_func.py (3.7.2)', contains the following Python code:

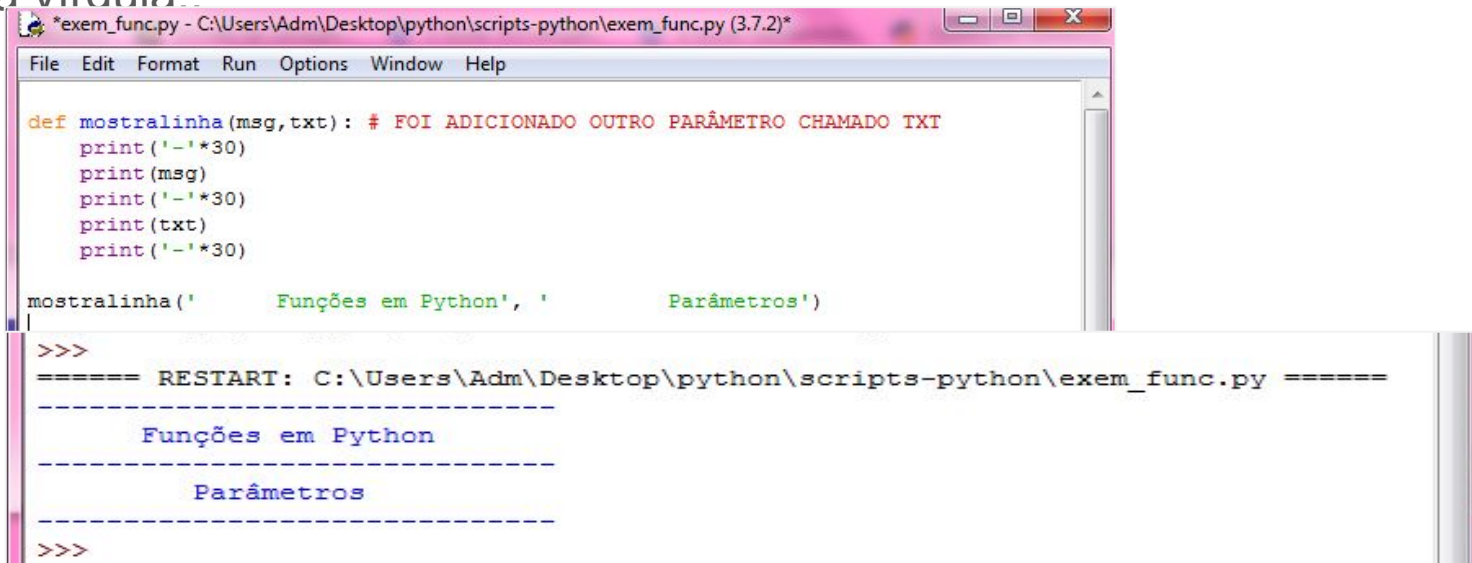
```
def mostralinha(msg):  
    print('-'*30)  
    print(msg)  
    print('-'*30)  
  
mostralinha('      Funções em Python')
```

The bottom window, titled 'Python 3.7.2 Shell', shows the execution of the script. The output is:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit  
(AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\Adm\Desktop\python\scripts-python\exem_func.py =====  
-----  
          Funções em Python  
-----  
>>> |
```


Funções em Python

- Você pode adicionar quantos parâmetros quiser, tem apenas separá-los com uma vírgula :



```
*exem_func.py - C:\Users\Adm\Desktop\python\scripts-python\exem_func.py (3.7.2)*
File Edit Format Run Options Window Help

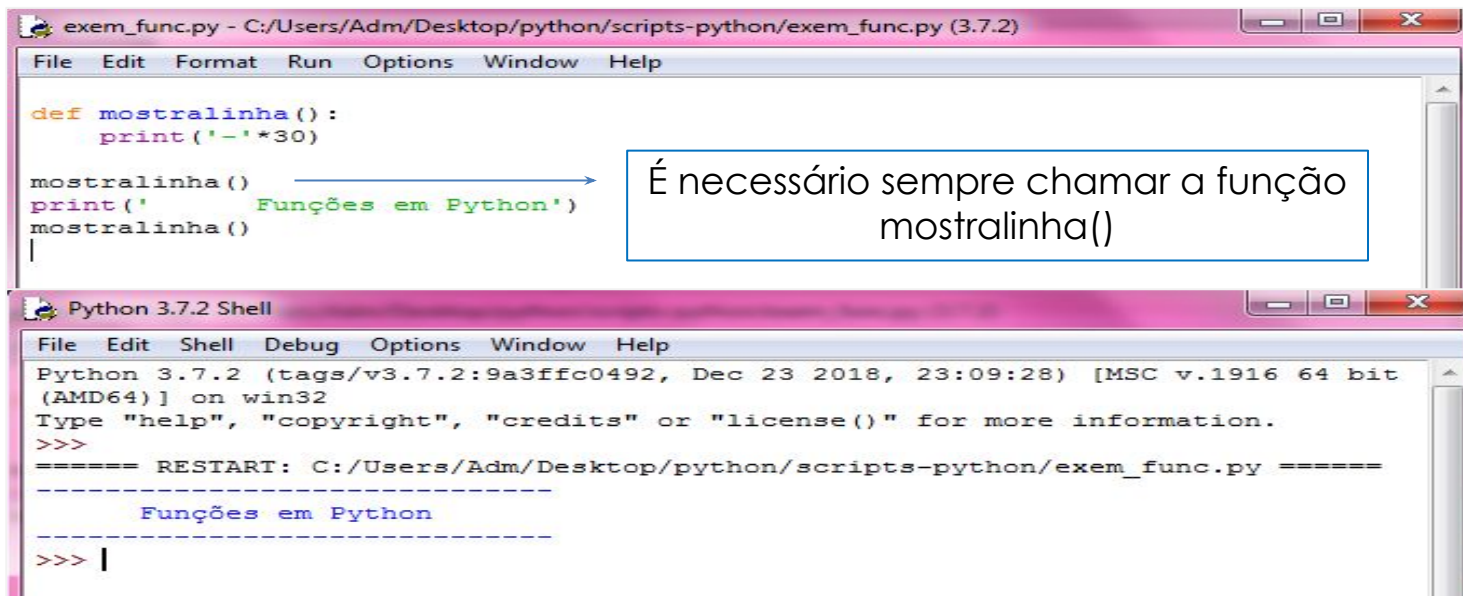
def mostralinha(msg,txt): # FOI ADICIONADO OUTRO PARÂMETRO CHAMADO TXT
    print ('-'*30)
    print (msg)
    print ('-'*30)
    print (txt)
    print ('-'*30)

mostralinha('      Funções em Python', '      Parâmetros')

>>>
===== RESTART: C:\Users\Adm\Desktop\python\scripts-python\exem_func.py =====
-----
          Funções em Python
-----
          Parâmetros
-----
>>>
```

Funções em Python

- Vamos ver como funciona o programa em Python sem parâmetro:



The image shows a screenshot of a Python IDE window titled "exem_func.py - C:/Users/Adm/Desktop/python/scripts-python/exem_func.py (3.7.2)". The code in the editor is as follows:

```
def mostralinha():  
    print('-'*30)  
  
mostralinha()  
print('      Funções em Python')  
mostralinha()  
|
```

An arrow points from the first call to `mostralinha()` to a text box containing the text: "É necessário sempre chamar a função mostralinha()".

Below the editor is a "Python 3.7.2 Shell" window. The output shows the execution of the script:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit  
(AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/Adm/Desktop/python/scripts-python/exem_func.py =====  
-----  
      Funções em Python  
-----  
>>> |
```

Funções em Python

Retornando Valores

- O objetivo de toda função é o processamento de alguma informação e o retorno do dado processado. Desta forma, temos que toda função pode, por definição, retornar valores;
- Para retornarmos valores por funções, temos que utilizar uma notação específica para este fim que é, a instrução **return**.

Funções em Python

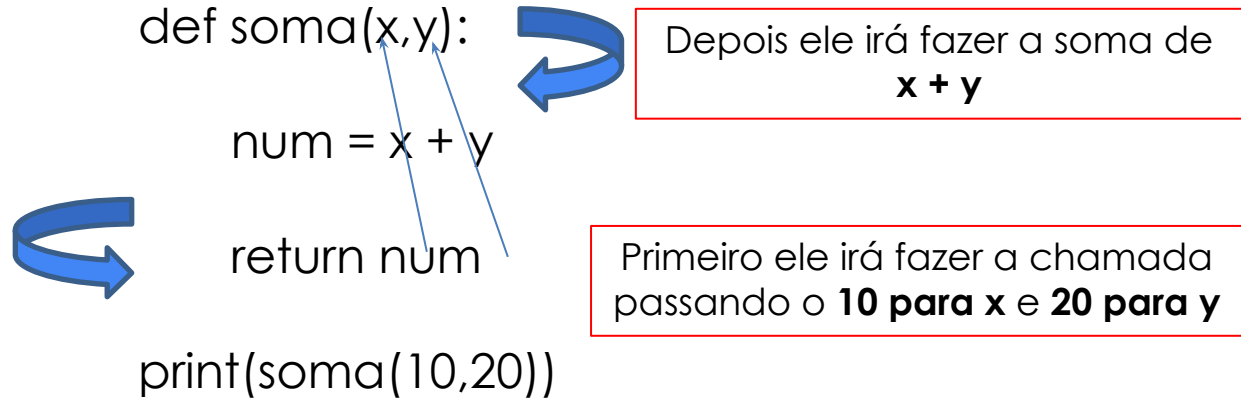
Retornando Valores `return`

- A palavra-chave `return` é utilizada para declarar a informação a ser retornada pela função;
- A mesma funciona também para finalizar a execução do bloco de instrução da função, retornando assim, o valor que estiver à sua frente;

Funções em Python

Retornando Valores `return`

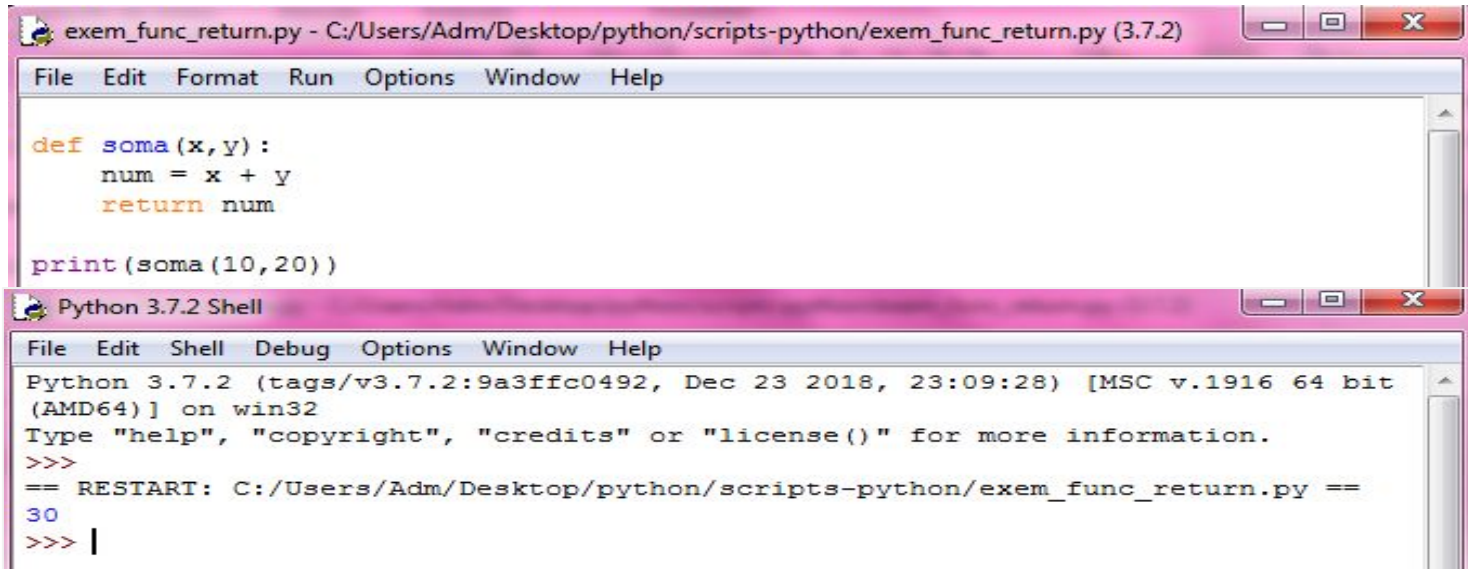
- Vamos entender melhor como funciona `return` com um exemplo:



Funções em Python

Retornando Valores `return`

- Vamos ver como funciona o programa em Python.



The image shows a screenshot of a Python IDE window titled "exem_func_return.py - C:/Users/Adm/Desktop/python/scripts-python/exem_func_return.py (3.7.2)". The window contains the following Python code:

```
def soma(x,y):  
    num = x + y  
    return num  
  
print(soma(10,20))
```

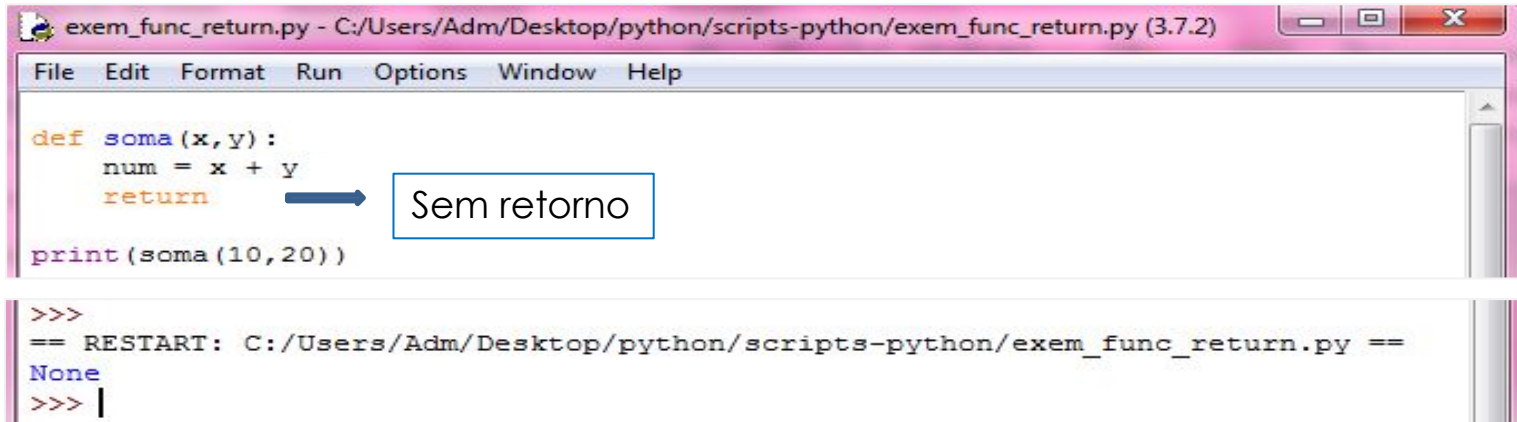
Below the code editor is a "Python 3.7.2 Shell" window. The shell output shows the execution of the script:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit  
(AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
== RESTART: C:/Users/Adm/Desktop/python/scripts-python/exem_func_return.py ==  
30  
>>> |
```

Funções em Python

Retornando Valores `return`

- Quando a instrução `return` é utilizada sem a definição de um valor a ser retornado, por padrão, o valor que será retornado é **None**.



The screenshot shows a Python IDE window titled "exem_func_return.py - C:/Users/Adm/Desktop/python/scripts-python/exem_func_return.py (3.7.2)". The code in the editor is:

```
def soma(x, y):  
    num = x + y  
    return  
  
print(soma(10, 20))
```

An arrow points from the `return` statement to a box containing the text "Sem retorno". Below the code editor, the output of the program is shown in a terminal window:

```
>>>  
== RESTART: C:/Users/Adm/Desktop/python/scripts-python/exem_func_return.py ==  
None  
>>> |
```

Exercícios Práticos

- Exercício 12:
- ✓ Faça um programa que tenha uma função chamada `área()`, que receba as dimensões de um terreno retangular (largura e comprimento) e mostre a área do terreno.

Exercícios Práticos

- Exercício 13:
- ✓ Faça um programa que tenha uma função chamada contador(), que receba três parâmetros: início, fim e passo. Seu programa tem que realizar três contagens através da função criada:
 - de 1 até 10, de 1 em 1;
 - de 10 até 0, de 2 em 2.

Exercícios Práticos

- Exercício 14:
- ✓ Faça um programa que tenha uma função chamada maior(), que receba vários parâmetros com valores inteiros. Seu programa tem que analisar todos os valores e dizer qual deles é o maior.

Exercícios Práticos

- Exercício 15:
- ✓ Faça um programa que tenha uma lista chamada números e duas funções chamadas sorteia() e somaPar(). A primeira função vai sortear 5 números e vai colocá-los dentro da lista e a segunda função vai mostrar a soma entre todos os valores pares sorteados pela função anterior.

Exercícios Práticos

- Exercício 16:
- ✓ Crie um programa que tenha uma função fatorial() e retorne o valor fatorial de três variáveis em tela.

Exercícios Práticos

- Exercício 17:
- ✓ Crie um programa que tenha uma função `par()` e verifique se o número digitado é par ou não.

Exercícios Práticos

- Exercício 18:
- ✓ Faça um programa, com uma função que necessite de um argumento. A função retorna o valor de caractere 'P', se seu argumento for positivo, e 'N', se seu argumento for zero ou negativo.

Exercícios Práticos

- Exercício 19:
- ✓ Faça um programa com uma função chamada `somaImposto`. A função possui dois parâmetros formais: `taxaImposto`, que é a quantia de imposto sobre vendas expressa em porcentagem e `custo`, que é o custo de um item antes do imposto. A função “altera” o valor de `custo` para incluir o imposto sobre vendas.

Exercícios Práticos

- Exercício 20:
- ✓ Faça um programa que converta da notação de 24 horas para a notação de 12 horas. Por exemplo, o programa deve converter 14:25 em 2:25 P.M. A entrada é dada em dois inteiros. Deve haver pelo menos duas funções: uma para fazer a conversão e uma para a saída. Registre a informação A.M./P.M. como um valor 'A' para A.M. e 'P' para P.M. Assim, a função para efetuar as conversões terá um parâmetro formal para registrar se é A.M. ou P.M. Inclua um loop que permita que o usuário repita esse cálculo para novos valores de entrada todas as vezes que desejar.