Desenvolva um módulo em Python designado por **epp2.py**. Neste módulo deverá definir as funções descritas abaixo.

Um número *m* diz-se divisor próprio de um número *n*, quando *n/m* é um número inteiro e *n* é diferente de *m*. Um número perfeito é aquele que é igual à soma dos seus divisores positivos, excluindo o próprio número. Por exemplo, o número 28 é um número perfeito porque a soma dos seus divisores próprios (1, 2, 4, 7, 14) é igual a 28.

**Função soma\_divisores\_proprios(n) [8 valores]**

Escreva uma função em Python, chamada **soma\_divisores\_proprios**, que recebe um número inteiro positivo **n** e devolve a soma de todos os divisores próprios de **n**.

Exemplos

**>>> soma\_divisores\_proprios(1)  
0  
>>> soma\_divisores\_proprios(12)  
16  
>>> soma\_divisores\_proprios(28)  
28  
>>> soma\_divisores\_proprios(496)  
496**

**Função perfeito(n) [4 valores]**

Escreva uma função em Python, chamada **perfeito**, que recebe um número inteiro positivo **n** e devolve **True**, caso esse número seja um número perfeito, ou devolve **False**, caso contrário. Na implementação desta função deverá chamar a função **soma\_divisores\_proprios**.

Exemplos

**>>> perfeito(1)  
False  
>>> perfeito(12)  
False  
>>> perfeito(28)  
True  
>>> perfeito(496)  
True**

**Função multiplica\_perfeitos(a,b) [8 valores]**

Escreva uma função em Python, chamada **multiplica\_perfeitos**, que recebe dois números inteiros positivos **a** e **b** e devolve o produto de todos os números perfeitos que existem entre **a** e **b** (incluindo ambos, caso sejam perfeitos). Na implementação desta função deverá chamar a função **perfeito**. Caso não exista nenhum número perfeito entre **a** e **b**, a função deverá retornar **1**.

Exemplos

**>>> multiplica\_perfeitos(1, 5)  
1  
>>> multiplica\_perfeitos(1, 12)  
6  
>>> multiplica\_perfeitos(1, 28)  
168  
>>> multiplica\_perfeitos(1, 496)  
83328**

Notas Importantes

* O seu trabalho **não** deve conter nenhuma instrução **print** nem **input**. As suas funções devem devolver valores e não escrevê-los. Assim, não deve utilizar instruções **print**, mas sim instruções **return**.
* Os valores apresentados nos exemplos são apenas exemplificativos: as suas funções devem aceitar quaisquer valores para os respetivos parâmetros, dentro do tipo e gama especificados.
* O programa pedido deverá ser submetido num ficheiro com o nome **epp2.py**. Na primeira linha do ficheiro deverá constar, em comentário, o seu nome e número de aluno.

Critérios de Avaliação  
Cada uma das funções será classificada individualmente, para a cotação indicada, de acordo com os seguintes critérios.

* Resultados corretos: 70%
  + *Se a função consegue produzir o resultado esperado, incluíndo, caso seja pedido, se consegue produzir as exceções corretas para parâmetros de entrada inválidos.*
* Qualidade do código: 30%
  + *Se o código da função está corretamente comentado (não em excesso), se está bem estruturado e se é de fácil compreensão.*