**n, m, s = ...**

Essa é uma **atribuição múltipla** em Python. Os três valores gerados pelo gerador são atribuídos, na ordem, às variáveis n, m e s.

**Exemplo:** n = 5, m = 10, s = 2

**input()**

O comando input() lê uma linha de texto inserida pelo usuário.

**Exemplo**: Suponha que o usuário digite 5 10 2.

**.split(" ")**

Divide a string de entrada em partes separadas pelo espaço " "

**Exemplo**: "5 10 2".split(" ") retorna a lista: ["5", "10", "2"].

**tmp**

O nome **tmp** é apenas uma escolha de quem escreveu o código. Ele poderia ser qualquer nome.

**Nome de uma variável temporária** usada dentro da compreensão de gerador, ele funciona como um **iterador** para os elementos da lista.

É usada para armazenar, temporariamente, **cada elemento da lista** enquanto o gerador percorre os valores.

**(int(tmp) for tmp in ...)**

Isso é uma **compreensão de gerador**, que cria um objeto que gera valores sobdemanda.

A expressão int(tmp) converte cada elemento da lista retornada por .split(" ") de string para inteiro.

**Exemplo**: o gerador produzirá os inteiros: 5, depois 10, e por fim 2.

Quando você usa **for tmp in input().split(" ")**, o Python:

1. Pega o **primeiro elemento** da lista gerada pelo split(" ") (uma string).
2. Atribui esse valor à variável **tmp**.
3. Executa a operação **int(tmp)** (converte para inteiro).
4. Passa o resultado para a saída do gerador.

Isso se repete para todos os elementos da lista, um por vez.

**map**

Aplica uma função (neste caso, int) a cada elemento de um iterável (neste caso, a lista retornada por input().split()).

Ela retorna um objeto iterável (um **map object**) que gera os resultados da aplicação da função

Passos:

1. **input()** lê "5 10 2".
2. **.split()** retorna ["5", "10", "2"].
3. **map(int, ...)** aplica **int** diretamente:
   * Converte "5" em 5.
   * Converte "10" em 10.
   * Converte "2" em 2.

**[elemento for \_ in range(n) ]**

É uma **compreensão de lista** em Python, ela cria uma lista contendo n sublistas vazias. É como inicializar uma estrutura de dados com várias listas dentro dela.

Desmontando a linha:

**Elemento**

O elemento pode ser qualquer coisa: um número, string, tupla, dicionário, objeto ou até outra lista com valores específicos. Podendo gerar a sublista com valores fixos ou variáveis dependendo do elemento escolhido.

Quando o **elemento** é uma sublista vazia [], cada iteração adiciona uma nova sublista vazia à lista principal.

\_

Convenção em Python para representar uma **variável ignorada**, neste caso, significa que o valor gerado por **range(n)** não é usado diretamente, só serve para repetir a operação **n** vezes.

**range(n)**

Gera uma sequência de números de **0 a n-1**, usada como um contador para repetir a criação das sublistas.

Essa construção é comum quando queremos criar uma **lista de adjacências** para representar um grafo, onde cada sublista pode conter os vértices adjacentes a um vértice específico, ou se queremos inicializar uma estrutura para armazenar coleções separadas de valores.

*.****sort()***

Função nativa do Python utilizada para **ordenar elementos de uma lista** em ordem crescente (por padrão) ou de acordo com critérios personalizados. É um método **in-place** (não retorna uma nova lista, mas altera a existente).

**Parâmetros**

***key*:** É uma função que especifica como os elementos devem ser comparados. O método aplica essa função a cada elemento da lista antes de compará-los.

***reverse***: É um valor booleano (True ou False). Define se a ordenação será em ordem decrescente (reverse=True) ou crescente (padrão: reverse=False).

***lambda***

Em Python é uma **função anônima**, ou seja, uma função sem nome. É usada quando você precisa de uma função rápida, geralmente simples, e não quer criar uma função completa com ***def***.

***lambda*** argumentos: expressão

**Argumentos**: São os valores que a função recebe.

**Expressão**: É o que a função retorna.