Gravity Sort

Aluno: _____

Entregue este material no final do encontro, para que os professores possam acompanhar o aprendizado da turma. Ele será devolvido depois.

Parte A: Você entende a gravidade da situação?

Para começar, clone o repositório https://github.com/franciol/Desprog_projeto_final e rode o programa **gravity.exe** (confia que não é vírus). Esse programa é um jogo que vai te ajudar a entender como o gravity sort (ou bead sort) funciona. Uma dica, tem a ver com a gravidade.

Depois de terminar o jogo, escreva abaixo a mensagem secreta que aparece ao terminar a última fase.

Antes de avançar para a próxima página, valide suas respostas com o professor ou um colega validado. Peça para o validador carimbar o nome ao lado.	VALIDADOR

Parte B: Como uma maçã caindo na cabeça de um físico

Agora que você já terminou o jogo, vamos ver se realmente aprendeu alguma coisa. No jogo, apertamos um botão e a gravidade se torna horizontal, de forma que as caixas deixam de ser puxadas para baixo, e passam a ser empurradas para a direita. Explique por que essa mudança é responsável por ordenar os números.

Antes de avançar para a próxima página, valide suas respostas com o professor ou um colega validado. Peça para o validador carimbar o nome ao lado.	VALIDADOR

Parte C: A face por trás da caixa

Mesmo tendo conceitos intuitivos, o método gravity não demonstra a mesma simplicidade na hora da implementação, por isso, aqui está um exemplo para servir como guia.

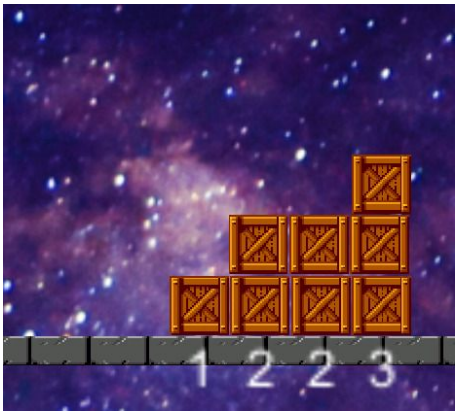
```
def gravity_sort(lista):
      contador = len(lista)
      temp = []
      index = 0
      out = []

      while contador:
      temp.append(contador)
      index += 1
      contador = 0
      for i in lista:
            if i > index:
                  Contador += 1

      index=0
      contador = len(temp)
      while contador:
      out.append(contador)
      index += 1
      contador = 0
      for i in temp:
            if i > index:
                  contador += 1
      return out[::-1]
```

Assumindo que o código acima seja uma função que recebe uma lista de números que deve ser ordenada, vamos analisá-lo passo a passo para entender o que está acontecendo.

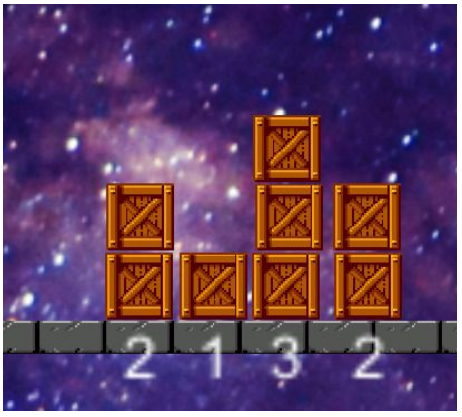
Primeiro, vamos trabalhar com 5 variáveis:

- Lista: lista de números que devem ser ordenados (recebido quando a função é chamada);
- Contador: variável responsável por fazer a varredura de cada valor na lista;
- Temp: lista temporária que armazena quantas caixas temos em casa altura, por assim dizer;
- Index: variável que aumenta de 1 em 1 nos whiles

É importante notar que neste código temos 2 whiles separados, isso acontece pois precisamos montar a lista temp antes de ordenar corretamente os números. Para entender melhor, imagine que recebemos uma lista [1,2,2,3]. No jogo ficaria assim:



Pense que a lista conta as caixas verticalmente, mas a lista temp conta de forma horizontal. Logo, ela seria [4,3,1]. Pode não fazer muita diferença no caso acima, pois as pilhas já estão ordenadas, mas em casos que as caixas estão desordenadas, essa lista é crucial pois ela independe da ordem.



Pense que nessa situação temp irá continuar sendo [4,3,1] e podemos usar isso para ordenar as pilhas. Isso é o que o segundo while do nosso código faz, por isso precisamos primeiro definir temp, para que possamos ordenar corretamente.

Para entender o segundo while, pense em quantos valores de temp são maiores do que zero. A resposta é 3, e assim temos o primeiro valor da nossa lista ordenada. O próximo valor será obtido pela mesma pergunta, mas dessa vez maiores do que um. A resposta, 2. Continuando por essa lógica, também temos 2 valores maiores do que dois, e 1 valor maior do que 3. Nesse ponto, a próxima iteração irá retornar um valor zero, e então sairá do while.

Assim, transformamos uma lista [2,1,3,2] em uma [3,2,2,1]. Parabéns, sua lista está ordenada!