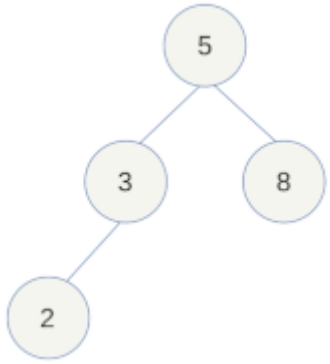
Inserção na Árvore Binária de Busca

Elabore um programa para fazer inserção de chaves inteiras em uma árvore binária de busca. A árvore deve ser representada como um dicionário que contém três elementos com as seguintes chaves:

- `'valor'`: armazena o valor da chave na raiz da árvore;
- `'esq'`: um dicionário que representa a sub-árvore esquerda da raiz;
- `'dir'`: um dicionário que representa a sub-árvore direita da raiz.

Uma sub-árvore vazia deve ser representada com o objeto `None`.

Por exemplo, considere a árvore mostrada na figura a seguir:



A representação dessa árvore como um dicionário em Python é a seguinte:

```
T = { 'valor': 5,
      'esq': { 'valor': 3,
               'esq': { 'valor': 2,
                        'esq': None,
                        'dir': None
                      },
               'dir': None
             },
      'dir': { 'valor': 8,
               'esq': None,
               'dir': None
             },
    }
```

Cada par de chaves representa um dicionário. A notação `'string': objeto` significa que `'string'` é uma chave que está associada a algum objeto. Os elementos do dicionário podem ser acessados com uma notação semelhante à que usamos para indexar listas. Por exemplo, `T['valor']` contém o valor da chave na raiz da árvore, `T['esq']` é a sub-árvore esquerda (que contém o nós `3` e 2) e `T['dir']` é a sub-árvore direita (que contém apenas o nó `8`). Observe que sub-árvores vazias são representados como `None`. Então,

como o nó de valor `3` só possui filho esquerdo, seu filho direito é `None`. E como os nós folhas não possuem filhos, tanto a chave `'esq'` quanto a chave `'dir'` estão associadas ao valor `None` nesses casos.

Complete o código inicial, criando a função `insere(arvore, valor)` para inserir um elemento na árvore binária de busca. Você deverá representar a árvore de acordo com a especificação acima, de modo que a função `imprime_arvore`, no código inicial, possa ser usada para imprimir toda a estrutura da árvore a partir do dicionário.

Entrada

A entrada conterá vários casos de teste.

Cada caso de teste contém uma lista de valores inteiros. Estes são os elementos que devem ser inseridos em uma árvore inicialmente vazia.

A entrada termina quando o caso de teste for vazio. Esse caso não deverá ser processado.

Saída

Para cada caso de teste, imprima primeiro uma linha contendo o padrão `Arvore #`, sendo `#` o número da árvore para facilitar a depuração.

Em seguida, imprima o conteúdo da árvore como no exemplo da descrição. Use a função `imprime_arvore`, disponibilizada no código inicial.

Imprima uma linha em branco após cada caso de teste.

Dicas

Use a função `cria_no(valor)` para criar um novo nó com a chave `valor` e a função `cria_arvore` para criar uma árvore vazia.

Não é necessário preocupar-se com alocação/desalocação de memória, pois Python possui coleta de lixo.

Se uma árvore `T` for vazia, a expressão `T is None` deve ser verdadeira. Use o operador `is` para verificar se você chegou a uma sub-árvore vazia no algoritmo de inserção.

Para usar a função `imprime_arvore`, basta passar o dicionário da árvore como primeiro argumento. Teste o exemplo abaixo no Python:

```
arvore = {'valor': 1, 'esq': None, 'dir': {'valor': 2, 'esq': None, 'dir':
{'valor': 3, 'esq': None, 'dir': None}}}
imprime_arvore(arvore)
```

Exemplos de Entrada e Saída

Entrada

```
[5, 3, 2, 8]
[8, 5, 3, 2]
[2, 3, 5, 8]
[]
```

Saída

```
Arvore 1
{ 'valor': 5,
  'esq': { 'valor': 3,
           'esq': { 'valor': 2,
                    'esq': None,
                    'dir': None,
                  },
           'dir': None,
         },
  'dir': { 'valor': 8,
           'esq': None,
           'dir': None,
         },
},

Arvore 2
{ 'valor': 8,
  'esq': { 'valor': 5,
           'esq': { 'valor': 3,
                    'esq': { 'valor': 2,
                             'esq': None,
                             'dir': None,
                           },
                    'dir': None,
                  },
           'dir': None,
         },
  'dir': None,
},

Arvore 3
{ 'valor': 2,
  'esq': None,
  'dir': { 'valor': 3,
           'esq': None,
           'dir': { 'valor': 5,
                    'esq': None,
                    'dir': { 'valor': 8,
                             'esq': None,
                             'dir': None,
                           },
                  },
         },
},
```

Entrada

```
[5]
[5, 4, 7]
[5, 4, 7, 2, 6]
[5, 4, 7, 2, 6, 1, 3]
```

```
[]
```

Saída

```
Arvore 1
{ 'valor': 5,
  'esq': None,
  'dir': None,
},

Arvore 2
{ 'valor': 5,
  'esq': { 'valor': 4,
           'esq': None,
           'dir': None,
         },
  'dir': { 'valor': 7,
           'esq': None,
           'dir': None,
         },
},

Arvore 3
{ 'valor': 5,
  'esq': { 'valor': 4,
           'esq': { 'valor': 2,
                    'esq': None,
                    'dir': None,
                  },
           'dir': None,
         },
  'dir': { 'valor': 7,
           'esq': { 'valor': 6,
                    'esq': None,
                    'dir': None,
                  },
           'dir': None,
         },
},

Arvore 4
{ 'valor': 5,
  'esq': { 'valor': 4,
           'esq': { 'valor': 2,
                    'esq': { 'valor': 1,
                             'esq': None,
                             'dir': None,
                           },
                    'dir': { 'valor': 3,
                             'esq': None,
                             'dir': None,
                           },
                  },
           'dir': None,
         },
  'dir': { 'valor': 7,
           'esq': { 'valor': 6,
                    'esq': None,
                    'dir': None,
                  },
           'dir': None,
         },
},
```