ALGORITMOS E ESTRUTURAS DE DADOS - IF969

2018.2

LISTA DE MONITORIA III

Q1. Para a questão 3 você deve implementar uma lista duplamente encadeada contendo os atributos e métodos descritos a seguir:

*\_\_init\_\_*: o método construtor da sua lista duplamente encadeada pode receber um objeto iterável (lista, tupla, dicionário, string, arquivo) e criar a lista a partir dos objetos contidos nesse iterável ou iniciar a lista vazia (é importante que o método construtor tenha as duas possibilidades).

*\_\_str\_\_*: o método str da sua lista duplamente encadeada deve retornar uma string válida para representação da lista. Para isso, siga o exemplo a seguir:

Uma lista contendo três nós com as strings “Algoritmos”, “Cálculo” e “Estatística” respectivamentes deve produzir como retorno no método str a seguinte string “[‘Algoritmos’;’Cálculo’;’Estatística’]”.

*anexar*: o método anexar recebe como parâmetro um valor, seja ele qual for, e armazena esse valor em um nó no final da lista.

*inserir*: o método inserir recebe como parâmetros um índice e um valor respectivamente. Esse método insere na posição informada um novo nó contendo o valor informado, “empurrando para a direita” todos os demais valores.

Nesse método, quando um índice maior que o tamanho da lista é informado, o valor é inserido na última posição da lista.

*tirar*: o método tirar recebe como parâmetro o índice de um dos elementos da lista, sendo por padrão o último. Ele remove o nó daquela posição, mantendo a estrutura da lista, e retorna o valor que o nó guardava.

Caso o índice não exista, o método levanta a exceção IndexError.

*remover*: o método remover recebe como parâmetro um valor e remove da lista o nó que contém esse valor. Caso o valor apareça mais de uma vez, apenas o primeiro nó é removido.

Se o valor não existir, o método levanta a exceção ValueError.

*procurar*: recebe como parâmetro um valor e deve retornar o índice do nó que contém esse valor. Caso o valor não esteja presente, o método levanta a exceção ValueError.

*eliminar*: o método eliminar funciona como o método remover, no entanto, ele remove todos os nós que contenham o valor informado como parâmetro.

Se o valor não existir, o método levanta a exceção ValueError.

*copiar*: esse método deve retornar um novo objeto do tipo lista duplamente cópia exata da própria instância, contendo os mesmos elementos na mesma ordem.

*trocar*: o método trocar recebe como parâmetro os valores de dois índices de uma lista e troca as células de posição.

*estender*: o método estender adiciona ao fim da lista os elementos de um iterável passado como parâmetro.

*\_\_repr\_\_*: o método repr, como sempre deve retornar uma forma válida de instanciar o objeto. Tendo como exemplo a lista duplamente encadeada do exemplo anterior, temos que o método repr deve ter o seguinte retorno: “ListaDupla([“Algoritmos”, “Cálculo”, “Estatística”]).

*\_\_len\_\_*: o método len deve retornar um número inteiro que representa o tamanho da lista. Você pode optar por salvar o tamanho da lista em um atributo privado e retornar o valor desse atributo quando esse método for chamado ou implementar um método de contagem, mas fique atento as consequência da implementação escolhida nos métodos seguintes.

*\_\_getitem\_\_*: implemente o método getitem, de modo a indexar os nós da lista. Assim, o método deve retornar o valor guardado no primeiro nó quando o parâmetro índice for 0, o segundo quando for 1 e assim sucessivamente.

Normalmente não é necessário implementar uma forma de acesso com índices negativos, visto que o próprio python encontra essa forma a partir dos métodos getitem e len, mas caso queira tentar, é uma boa forma de praticar suas noções de computação.

*\_\_setitem\_\_*: o método setitem atualiza o valor de um determinado nó, dado um índice e um novo valor a substituir o antigo. Caso o índice informado não pertença a lista, o método levanta uma exceção IndexError.

*\_\_contains\_\_*: o método contains é o método que controla o comportamento da keyword in. Esse método recebe como um argumento um valor, e retorna True ou False, dependendo de o argumento estar, ou não, contido no objeto.

Por fim, implemente fora da lista uma função *concatenar*, que recebe como parâmetro duas listas duplamente encadeadas e concatena os elementos da segunda ao fim da primeira.

Q2. Objetos indexados por números inteiros, como listas, tuplas e sets, precisam apenas do método *\_\_getitem\_\_* para serem iteradas pelo for. Isto é, quando você utiliza o for, o python entende que você está preocupado em acessar os itens de índices 0, 1, 2… Apesar de funcionar, essa é uma forma lenta de se iterar pelo objeto, uma vez que numa lista encadeada, diferente de um vetor, o ponto de partida é sempre o início da lista. Assim, acessar o índice 0 é acessar o primeiro elemento, acessar o índice 1 é acessar o primeiro elemento e a partir dele chegar ao segundo, acessar o índice 2 é acessar o primeiro elemento, chegar no segundo a partir dele e então ir para o terceiro.

Para tornar esse processo mais dinâmico, usam-se objetos iteradores, que são responsáveis por apontar a posição atual de cada um dos elementos.

*\_\_init\_\_*: a classe de iterador, ou ponteiro, recebe como parâmetro o objeto pelo qual ela vai iterar. Nesse caso, receberá como parâmetro uma lista duplamente encadeada.

Dentro do método construtor você deve salvar a primeira posição da lista como a posição atual do ponteiro.

*\_\_next\_\_*: o método next será o responsável por retornar a posição do ponteiro e seguir para a próxima posição. Para isso você deve verificar se a posição atual do ponteiro é válida. Se sim, salve o valor da posição atual, mova o ponteiro para a próxima posição e retorne o valor salvo.

Se a posição atual não for válida, levante uma exceção StopIteration.

Ter um ponteiro é mais eficiente como iterador por permitir uma iteração mais dinâmica pelos elementos da lista, assim como permitir múltiplas iterações. Note, por exemplo, que no exercício da classe *Polinomios* da lista de monitoria 2, a forma como a iteração era feita não permitia múltiplos iteradores (tente, por exemplo, usar dois laços for em um polinômio).

Caso os atributos e métodos da sua lista sejam privados, você pode escrever a sua classe ponteiro interna a lista (na mesma indentação dos métodos).

O que entregar:

Você deve implementar um código de classe para a lista duplamente encadeada e o seu ponteiro, devendo entregar os dois códigos preferencialmente num arquivo contendo o modelo de cabeçalho presente no fim desse documento.

Quando entregar:

A entrega deverá ser feita até, no máximo, às 16h da próxima quarta-feira (12). As entregas devem ser feitas via Classroom.

“O mundo está cheio de coisas óbvias que ninguém jamais observa.”

— *Sherlock Holmes*