MAB225 – Computação II – Aula Prática 5

Objetivo: iteradores, geradores e tratamento de exceção

1. Desejamos um programa que, dada uma string e um segmento (onde segmento é outra variável do tipo string), permita percorrer a string em um loop de forma que a cada iteração seja retornada a posição do segmento na string. O loop deve acabar quando a string tiver sido inteiramente percorrida. Ex.: para a string “engenharia” e o segmento “n” o valor da primeira iteração deve ser 1 e da segunda iteração deve ser 4 (o n aparece nas posições 1 e 4 da string). Se, para a mesma string, o segmento for “en”, o valor da primeira iteração deve ser 0 e da segunda deve ser 3 . Comece a procurar o segmento seguinte a partir do final do primeiro. Ex.: Para a string “bbbbbb” e o segmento “bb”, o valores de retorno devem ser 0, 2 e 4, na primeira, segunda e terceira iterações respectivamente. Resolva esse problema de duas formas diferentes:
    a. Crie uma função geradora cujos argumentos são a string e o segmento.
    b. Crie uma classe que define um objeto iterável. Essa classe deve possuir o método construtor e os dois métodos necessários para tornar o objeto iterável (__iter__ e __next__). O método construtor deve possuir como argumentos a string e o segmento e devem ser criados atributos para ambos argumentos recebidos. Os métodos necessários para a iteração devem respeitar o protocolo de iteração e retornar a posição correta a cada iteração. (Não se preocupe se aparecer a palavra None em algumas das iterações – é possível resolver sem None, mas não é necessário no exercício.)
2. Crie uma classe para definir uma conta bancária. Essa classe deve possuir os seguintes métodos e atributos (obs.: os métodos depósito e saque são muito parecidos, mas é aconselhável não copiar e colar para praticar a estrutura do tratamento de exceção):
    a. Método construtor – recebe como argumento uma string com o nome do cliente e um número que representa o valor inicial depositado na conta. O número deve ter valor default igual a zero. Crie um argumento para o nome e outro para o valor.
    b. Método de depósito – utiliza tratamento de exceção e interage com o usuário (utilizando a função input do Python 3 – equivalente à raw_input no Python 2) para pedir o valor que deve ser depositado. Em seguida, tente transformar o valor passado através do input em um float. Caso não seja possível realizar a transformação, o Python vai gerar a exceção ValueError. Se for possível, verifique se o número é maior que zero. Se não for, gere a exceção ValueError do Python. Tente realizar a soma do valor convertido com o valor do atrituto. Se a soma não for possível, a exceção TypeError (do Python) será gerada. Se ValueError tiver sido gerada, imprima uma mensagem informando ao usuário que o valor sacado deve ser um número maior que zero. Se TypeError for gerada, imprima uma mensagem informando ao usuário que ele deve procurar o gerente da conta. Se o usuário interromper o programa (Ctrl+C), a exceção KeyboardInterrupt, do Python é gerada. Acrescente um except também para essa exceção. Se KeyboardInterrupt for gerada, imprima uma mensagem dizendo que o usuário encerrou o programa e a transação não foi realizada. Se nenhuma exceção for gerada (else), modifique o atributo de

valor de forma que o novo valor seja igual à soma do que já havia no atributo com o valor depositado.

c. Método de saque – utiliza tratamento de exceção para interage com o usuário para pedir o valor que será sacado. Em seguida, tente transformar o valor passar através do input em um float. Caso não seja possível realizar a transformação, o Python vai gerar a exceção ValueError. Se for possível, verifique se o número é maior que zero. Se não for, gere a exceção ValueError do Python. Tente realizar a subtração entre o valor do atributo e o valor convertido para float. Se a subtração não for possível, a exceção TypeError (do Python) será gerada. Verifique então se o resultado da subtração é maior ou igual a zero. Se não for, gere a sua própria exceção ErroLimite. Adicione o código necessário para que a exceção seja reconhecida. Se ValueError for gerada, imprima uma mensagem informando ao usuário que o valor sacado deve ser um número maior que zero. Se TypeError for gerada, imprima uma mensagem informando ao usuário que ele deve procurar o gerente da conta. Se ErroLimite for gerada, imprima uma mensagem de erro informando que o valor retirado é maior que o valor guardado na conta. Imprima também o valor guardado. Acrescente um except para a exceção KeyboardInterrupt. Se KeyboardInterrupt for gerada, imprima uma mensagem dizendo que o usuário encerrou o programa e a transação não foi realizada. Se nenhuma exceção for gerada (else), modifique o atributo de valor de forma que o novo valor seja igual ao que já havia no atributo menos o valor sacado.