TRABALHO FINAL
Introdução à Programação para Design
Prof. Anderson Lemos

A seguir são definidas as atividades referentes ao trabalho prático. Todas as implementações devem ser feitas em Python. O trabalho tem como objetivo o uso de listas. As duplas deverão escolher somente um dos três trabalhos e implementá-lo.

A parte de interface com o usuário pode ser simples, feita no terminal (linha de comando normal, como utilizamos em sala). Se desejar algo mais elaborado, pode usar alguma biblioteca. Quanto mais coisa o trabalho tiver, melhor será avaliado. Inclusive, se você fizer mais funcionalidades que as pedidas, melhor ainda. Seja criativo, crie um sistema que você gostaria de usar!

1. **Trabalhos**

a) Lista de supermercado.

Você deve implementar um sistema que gerencia listas de compras em um supermercado. Você deve criar dois tipos de produtos: comidas e bebidas (tuplas, definição de tupla abaixo). Uma comida deve ter um nome, uma quantidade e um booleano que diz se é vegetariana ou não (uma tupla com três valores). Uma bebida deve ter uma quantidade, um nome e um booleano que diz se é bebida alcoólica ou não (uma tupla com três valores). Além disso, o sistema vai guardar o histórico das listas de compras. Deve ser possível criar uma nova lista de compra do tipo bebida ou comida (comidas devem ser separadas de bebidas), e nessas, deve ser possível adicionar e remover produtos. Deve haver um comando para salvar a lista (depois de salva, uma lista não pode ser alterada). Deve ser possível buscar a última vez que um produto foi comprado. Deve haver um comando para apagar o histórico de listas de compras. Por fim, deve ser possível encerrar o programa. Dicas:

- Uma tupla é como uma lista, só que tem um tamanho fixo, depois de criada ela não pode ser alterada e geralmente inserimos coisas de tipos diferentes. A diferença de tupla para lista, é que declaramos usando parênteses e não colchetes. Por exemplo, uma lista seria algo to tipo: *lista1 = [1, 2, 3, 4]*. Uma tupla que representa uma comida, guarda o nome, a quantidade e um booleano que diz se a comida é vegetariana ou não, por exemplo: *comida1 = ("macarrão", 5, True)*. Nesse caso, *comida1* é uma tupla, que é uma variável que representa a comida *"macarrão"*, a quantidade que deve ser comprada é *5,* e que a comida é vegetariana. Para acessar elementos de uma tupla, é igual à uma lista, usamos o nome da variável e a posição, por exemplo, *comida1[0]* me dá o valor da posição *0* da tupla, que é *"macarrão"*. Mais detalhes sobre tuplas: http://www3.ifrn.edu.br/~jurandy/fdp/doc/aprenda-python/capitulo_09.html.
- Para guardar o histórico de listas de compras, use outra lista, que vai ser uma lista de listas. Isto é, uma lista que guardará as listas de compras.

b) Jogo da cobrinha.

Todo mundo quando criança jogou o famoso jogo da cobrinha. Pra quem não jogou, basicamente se trata de um jogo 2D, onde uma cobrinha tem que comer as comidinhas que aparecem na tela. A cobra se movimenta automaticamente, sem parar e o jogador pode apenas mudar a direção dela. A cada comidinha que ela come, a pontuação aumenta e a cobra aumenta de tamanho. O jogo acaba quando a cobrinha morre, que acontece quando ela bate em uma parede (lateral da tela) ou nela mesma. Há versões em que você passa de fases ao comer uma determinada quantidade de comidinhas, e a cada fase a dificuldade aumenta (com mais obstáculos no meio da tela ou com a cobrinha ficando mais rápida). Mais informações em https://goo.gl/skmZxt. O Google disponibiliza uma versão para jogar em https://goo.gl/V8DB7F. Agora você tem que implementar a sua versão do jogo da cobrinha (dica: a biblioteca pygames pode ajudar muito na interface gráfica, talvez seja mais fácil até do que utilizar o terminal como interface).

c) Dicionário.

Você deve implementar um dicionário virtual, onde você digita uma palavra que você quer saber o significado e o sistema mostra o significado da palavra. Para isso, você vai utilizar uma estrutura conhecida como **HashMap**. Eles servem para armazenamento e obtenção rápida de dados, e se baseia em chave e valor. Você insere neles, valores que estão relacionados a alguma chave, e quando você quer buscar tal valor, você usa essa chave. No dicionário, a palavra é a chave e o significado é o valor, e quando você quer buscar o significado você usa a palavra como chave. Por exemplo, você pode inserir a palavra *"escola"* relacionada com o significado *"estabelecimento público ou privado destinado a ensino coletivo"*, e quando você quer achar o significado de *"escola"*, você diz pro HashMap que você quer o significado de *"escola"* e ele te retorna o significado.

O HashMap é uma estrutura da seguinte forma: há uma lista ***A*** de tamanho ***n***. Essa lista guarda outras listas. Quando você vai inserir algum valor, você passa a chave ***c*** e o valor ***v***, e é calculado o código hash ***h*** da chave (que é um número inteiro entre *0* e *n-1*). Esse código hash vai ser o índice da lista ***A*** que está a lista onde o valor ***v*** é inserido (ou seja, na lista que está em ***A[h]*** se insere ***v***). Como no dicionário, a chave é do tipo string, calcule seu código hash da seguinte forma: some os valores dos caracteres na tabela ASCII (usando a função *ord*) e pegue o resto da divisão dessa soma por ***n***. No seu HashMap, você pode escolher um valor de ***n*** que ache adequado. Seu sistema deve permitir inserir, remover e buscar palavras e significados. Para isso, implemente as seguintes funções no seu HashMap:

- *def put(palavra, significado);* // insere o significado no mapa e retorna se inseriu.
- *def get(palavra);* // retorna o significado relacionad a palavra, ou *None*, se não existir.
- *def containsKey(palavra);* // verifica se existe um significado relacionada a palavra.
- *def remove(palavra);* // remove um significado do mapa e retorna se removeu.
- *def clear();* // remove tudo que há no mapa.
- *def empty();* // retorna se o mapa está vazio.
- *def size();* // retorna quantas palavras há no mapa.

2. Considerações Finais

- As equipes serão formadas por no máximo 2 (DOIS) alunos.
- As notas serão diferenciadas por aluno no momento da apresentação do trabalho em sala.
- A nota do trabalho fará parte da nota final da disciplina, valendo um quarto da nota total.
- O trabalho deverá ser implementado em Python.

3. Dicas

- Sejam criativos! Façam um sistema que vocês gostariam de usar. Funcionalidades a mais serão muito bem vistas.
- A interpretação do trabalho é uma forma de avaliação. Se você acha que algo não ficou bem explicado, tente entender como seria a maneira certa de fazê-lo e resolva o problema. Esse é um problema que vocês vão ter que lidar diariamente no mercado de trabalho.
- Deixem a parte mais difícil por último, concentrem-se em implementar os códigos pouco a pouco, dividindo em funções. As funcionalidades podem ser implementadas depois.
- Procurem dividir o trabalho entre os integrantes.
- Façam o trabalho por partes: leiam com cuidado este documento e vão implementando à medida que forem entendendo.
- Tirem dúvidas com o professor, entre vocês e pela Internet.