Técnicas de Programação II – AC5

Tipos Estruturado

1. Compare list com tuple apontado uma semelhança e três diferenças entre elas.
   Ambos são utilizados para armazenar uma coleção de valores que podem ser de tipos básicos e estruturados que são identificados por um índice inteiro. Os itens de uma list são mutáveis enquanto numa tupla são imutáveis. O desempenho no acesso a tuplas é melhor do que no acesso a lists (que as torna melhor para lidar com dados imutáveis) e lists são criadas com uma coleção de dados separados por vírgulas (“,”) e rodeadas por colchetes (“[“ e “]”) enquanto as tuplas podem ser rodeadas por parênteses (“(“ e “)”).

2. Compare set com tuple apontado uma semelhança e três diferenças entre elas.
   Ambos são utilizados para armazenar uma coleção de valores que podem ser de tipos básicos e estruturados. Os itens de ambos (tuples e sets) são imutáveis. Os valores nas tuples podem ser repetidos enquanto que em sets não são permitidas repetições (que as torna melhor para lidar com dados não repetidos, como as chaves de busca em bancos de dados) e sets são criadas com uma coleção de dados separados por vírgulas (“,”) e rodeadas por chaves (“{“ e “}”) enquanto as tuplas podem ser rodeadas por parênteses (“(“ e “)”). Além disso, sets suportam operações de intersecção e união.

3. Em um programa a ser desenvolvido, um conjunto de CPFs de clientes precisa ser armazenado. Que tipo estruturado você escolheria? Justifique a sua resposta.
   O melhor tipo dependeria do uso que fosse dado. Os tipos list, set e tuple poderiam ser usados. A melhor escolha possivelmente seria set pelos dados serem não repetidos (não há diferentes clientes com o mesmo CPF), não serem modificados (clientes não mudam de CPF) e a performance para se procurar por um item em um set ser muito melhor que nos demais ser ordenação.

4. O que é uma estrutura do tipo dict?
   O dict é um tipo estruturado mutável que armazena pares de dados na forma chave:valor onde a chave é única na estrutura e o valor é um valor associado a ela. Ele é criado na forma: { chave_1:valor_1, chave_2:valor_2,..., chave_n:valor_n}.

5. No exemplo abaixo, o que acontece?
   X = “abc”
   Y = X
   Y[0] = “#”
   Erro em tempo de execução, pois str é um tipo imutável

6. No exemplo abaixo, o que acontece?
   X = [“a”, “b”, “c”]
   Y = X
   Y[0] = “#”
   X passa a ter a lista [“#”, “b”, “c”] porque a operação Y = X não é uma operação de cópia (não duplica a lista na memória). Apesar de X[0] ser str ele pode ser mudado de “a” para “#” pois o item está sendo substituído por outro e não modificado (modificação seria fazer Y[0][0] = “#”)

7. No exemplo abaixo:
   x = [1, 2, (3, 4), "xyz"]
   a. x[0] é mutável?
   b. x[1] é mutável?
   c. x[2] é mutável?
   d. x[2][0] é mutável?
   e. x[2][1] é mutável?
   f. x[3][0] é mutável?
   g. x[3][1] é mutável?
   h. x[3] é mutável?

   a. sim
   b. sim
   c. sim
   d. não
   e. não
   f. não
   g. não
   h. sim

Funções

8. O que significa uma função poder ser reentrante? O que é recursão?
   Uma função pode chamar outra função. Essa possibilidade é chamada de reentrância. Não há limite teoricamente imposto pela linguagem para a reentrância. Assim, dentro de uma função f1() pode ter uma chamar para f2() que tem uma chamada para f3() e assim por diante. Na prática, como a chamada de uma função consome recursos da máquina a reentrância deve ser limitada a algumas milhares de chamadas. Recursão é uma situação de reentrância em particular onde se forma um loop de chamadas (por exemplo, f1() chama f2() que chama f3() que chama f1() ou um caso mais simples f() que chama a própria f()). O código que usa recursão tem que conter mecanismo que impeça que o loop permaneça infinitamente, com em qualquer loop for ou while.

9. Qual a principal vantagem do uso de funções?
   Estruturar programas em múltiplas funções permite decompor um programa em diversos módulos que podem ser desenvolvidos e testados individualmente, permitindo a distribuição de programas complexos entre diversos times. Com a separação em módulos, variáveis locais ficam isoladas de uma função para outra, o que reduz a complexidade do desenvolvimento.

10. Que é docstring?
    É um comentário na forma de string de múltiplas linhas (iniciado com “”” e terminado com “””) colocado logo no início de cada módulo e de cada função. Ele é acessado através de uma variável do módulo e de cada função chamada __doc__.

11. Que é faz o comando return?
    Devolve um valor para o comando que chamou a função. Pode retornar valor de qualquer tipo (inclusive estruturado). Mesmo quando a função não tem um comando return, um return None é implicitamente executado no momento em que a função é terminada.

12. Qual é a diferença no uso dos comandos “import” e “from ... import”?
    Com ambos os comandos se traz as funções e variáveis de um módulo em outro arquivo para o programa. A diferença é que no comando “import” é trazido todo o módulo, incluindo o espaço de nomes. Já o comando “from ... import” traz funções e variáveis escolhidas do módulo para o próprio espaço de nomes do módulo que faz a importação.

Parâmetros e Variáveis Locais e Globais

13. O que é um default em um parâmetro de função?
    Uma função pode ser criada com um valor pré-estabelecido para um parâmetro que será usado caso nenhum valor seja usado na hora da chamada. Esse valor é estabelecido com uma atribuição para a variável no cabeçalho da função. Por exemplo:

```
def f(x=1):
    pass
f()  # Sem parâmetro: x recebe o valor default 1
f(2)  # Com parâmetro: x recebe o valor 2
f(x=3)  # Com parâmetro. Forma alternativa: x recebe o valor 3
```

14. Uma função f(x) recebe em um parâmetro um valor de tipo int em uma variável chamada x, alterada para 0 dentro da função. O que acontece com a variável x, do programa principal, quando a função é chamada?

```
def f(x):
    x = 0
x = 1
f(x)
print(x)  # O que é impresso aqui?
```

    Nada. A alteração em x, feita dentro da função não tem efeito fora da função pois parâmetros de função são locais

15. Uma função f(x) recebe em um parâmetro um valor de tipo list em uma variável chamada x, com x[0] alterado para 0 dentro da função. O que acontece com a variável x, do programa principal, quando a função é chamada?

```
def f(x):
    x[0] = 0
x = [1, 2, 3, 4]
f(x)
print(x)  # O que é impresso aqui?
```

    x[0] será alterado para 0. A alteração em x, feita dentro da função tem efeito fora da função pois uma lista passada como parâmetro de função é a lista de fora tornada visível para a função. Isso é feito por default para tomar menos tempo e espaço ao evitar de se sempre produzir uma cópia de uma lista como parâmetro

16. No caso da questão anterior, quando se deseja passar uma cópia de uma lista para uma função usa duas possíveis alternativas de cópia: list.copy() e copy.deepcopy(). Qual a diferença entre as duas? Use o exemplo abaixo para explicar:
    x = [[1, 2], 3, 4]
    y = x.copy()
    z = copy.deepcopy(x)
    No exemplo, com copy() a cópia só é criada para as variáveis de tipo simples. Os itens que são list não são copiados:
    y[0][0] e y[0][1] são os mesmos ítens que x[0][0] e x[0][1]: a mesma list
    y[1] é cópia distinta de x[1]
    y[2] é cópia distinta de x[2]

    Com copy.deepcopy() a cópia é recursiva. Os itens dentro das list também são copiados:
    z[0][0] é cópia distinta de x[0][0]
    z[0][1] é cópia distinta de x[0][1]: z[0] é uma cópia de x[0]
    z[1] é cópia distinta de x[1]
    z[2] é cópia distinta de x[2]

17. O que é uma variável local?
    É uma variável criada dentro de uma função. Ele só pode ser usada dentro da função, não podendo ser usada nem modificada por outras funções do módulo.

18. O que é uma variável global?
    É uma variável colocada dentro de um módulo e fora das funções e que pode ser usada (mas não necessariamente modificada) por todas as funções do módulo (a menos que haja uma variável local com o mesmo nome).

19. O que é o comando global?
    É um colocado dentro de uma função. Ele define uma lista de variáveis globais que podem ser modificadas pela função.

20. O que fazem as funções globals() e locals()?
    Devolvem dicionários contendo como chaves os nomes das variáveis e como valores seus respectivos valores, respectivamente para as variáveis globais e locais.

Algoritmos e Ordenação

21. Cite dois algoritmos de ordenação gulosos.
    Selection sort e Insertion sort. Ambos ordenam conjuntos de dados a partir de uma série de trocas de posição de pares de dados.

22. Cite dois algoritmos de ordenação baseados em divisão e conquista.
    Merge sort e Quick sort. Ambos ordenam conjuntos de dados a partir da divisão sucessiva desses conjuntos em conjuntos menores até que eles fiquem com apenas um ou nenhum elemento e a sua ordenação seja trivial.

23. Complete a tabela abaixo:

Algoritmo	Complexidade Média
Selection Sort	**$O(n^2)$**
Insertion Sort	**$O(n^2)$**
Quick Sort	**O(n.log(n))**
Merge Sort	**O(n.log(n))**
Heap Sort	**O(n.log(n))**

24. Qual a desvantagem do Merge Sort em relação ao Quick Sort e o Heap Sort?
    Merge Sort requer espaço adicional de memória enquanto o Quick Sort e o Heap Sort usam o próprio espaço (in place) ocupado já pelos dados.

25. Qual a situação de pior caso do Quick Sort?
    O Quick Sort tem o pior desempenho quando se deseja colocar em ordem crescente uma lista que já está em ordem decrescente. Nessa situação a complexidade piora para $O(n^2)$. Por esta razão, há implementações de Quick Sort que embaralham os dados antes serem executados.

26. Quando o Insertion Sort é bastante eficiente?
    Em situações em que os dados estão "pouco desordenados" o algoritmo pode ser executado em tempo O(n).

27. Que é ordenação estável?
    Um algoritmo de ordenação é estável quando, ao ordenar um conjunto, se dois itens tem a chave coincidente, a ordem original destes itens é preservada.

Acesso a arquivos

28. Quais são os tipos de arquivos conforme seu conteúdo?
    Os arquivos podem ser em modo texto ou binário. Em modo texto, o conteúdo é limitado a caracteres imprimíveis e é usado para textos, scripts, programas fonte e documentos para serem usados por pessoas. Os arquivos em modo binário armazenam dados usados por programas, assim seu conteúdo pode não ser legível. Arquivos binários podem conter programas e aplicativos executáveis, conteúdo criptografado, comprimido, mídia (som ou imagem) entre outros.

29. Quais são os modos que um arquivo (binário ou texto) pode ser aberto?
    r, m, a, r+, m+, a+ (para arquivos do tipo texto) e rb, wb, ab, rb+, wb+, ab+ (para arquivos do tipo binário).

30. Como é o acesso no modo "r"?
    No modo "r", o arquivo está sendo aberto apenas para leitura. Esse modo permite que outros programas também abram o mesmo arquivo para leitura sem problemas (uma vez que não há concorrência para modificar o arquivo) e restringe o acesso de outros programas que desejam escrever). Um erro ocorre quando o arquivo desejado não existe ou já está aberto para escrita por outro programa ou se executa um comando de escrita para esse arquivo.

31. Como é o acesso no modo "w"?
    No modo "w", o arquivo está sendo criado apenas para escrita. Esse modo NÃO permite que outros programas também abram o mesmo arquivo (seja para leitura ou escrita). Um erro ocorre quando o arquivo desejado já está aberto para escrita por outro programa ou se executa um comando de leitura para esse arquivo. Um arquivo novo é sempre criado – se um arquivo com o mesmo nome já existia ele será apagado.

32. Quando se usa redirecionamento de entrada/saída (funções input() e print() de e para arquivo), onde se especifica os nomes dos arquivos de entrada e saída?
    O redirecionamento de entrada é feito na linha de comando ao executar o programa em python (onde programa.py deve ser substituído pelo nome do programa e entrada.txt deve ser substituído pelo nome do arquivo de entrada) do seguinte modo:
    Python programa.py << entrada.txt

    O redirecionamento de saída é feito na linha de comando ao executar o programa em python (onde programa.py deve ser substituído pelo nome do programa e saida.txt deve ser substituído pelo nome do arquivo de saída) do seguinte modo:
    Python programa.py >> saida.txt

    Ambos podem ser redirecionados do seguinte modo:
    Python programa.py << entrada.txt >> saida.txt

33. Por que um arquivo (binário ou texto) deve ser aberto (open()) pelo programa?
    O acesso a arquivos é mediado pelo sistema operacional, assim antes de ser permitido o acesso ao programa, devem ser feitas cinco tarefas:
    a. **verificar se o arquivo existe**
    b. **verificar se pode ter o tipo de acesso (leitura, escrita, remoção...) pedido**
    c. **se está havendo compartilhamento do arquivo e se ele é possível**
    d. **reservar memória para a área de transferência do arquivo para o programa**
    e. **definir o tipo de bloqueio dos demais programas ao arquivo**

34. Por que um arquivo (binário ou texto) deve ser fechado (close()) pelo programa?
    O acesso ao arquivo deve ser fechado para:
    a. **copiar dados da área de transferência em memória para o arquivo**
    b. **liberar a memória da área de transferência do arquivo para o programa**
    c. **liberar o bloqueio dos demais programas ao arquivo**