Atividade Avaliativa #03: Listas e Arquivos

Observações:

1. Os programas deverão ser desenvolvidos em linguagem PYTHON;
2. Cada questão deverá ser respondida em arquivos em separado;
3. As respostas deverão ser submetidas no link correspondente a essa lista disponível no Moodle;
4. Atentem para o prazo de submissão. Não serão aceitos envios posteriores a data limite.

1. **(Valor: 5 pontos)** Desenvolva um programa que solicite ao usuário dois valores inteiros: o primeiro representando a quantidade de listas na matriz e o segundo indicando a quantidade de elementos em cada lista. Com base nesses valores, o programa deverá gerar aleatoriamente os elementos da matriz (utilizar ***List Comprehensions***), exibir a matriz original e, em seguida, calcular e apresentar a matriz transposta.

   A matriz transposta de uma matriz **M** com m linhas e n colunas é obtida trocando as linhas pelas colunas e vice-versa, resultando em uma matriz **Mt** com **n** linhas e **m** colunas.

2. **(Valor: 5 pontos)** Faça um programa que leia dois valores: **x** e **n** (**x** e **n** deverão ser solicitados ao usuário), onde **x** é a quantidade de elementos que a lista deverá armazenar positivo e **n** serão os valores inteiros a serem inseridos na lista, o programa deve terminar a leitura dos números quando for informado o valor 0 (o valor 0 não deverá fazer parte da lista). A lista só deverá armazenar os **x** menores números informados, seguindo a lógica abaixo:

```
Informe a quantidade de elementos na lista: 6
Informe um valor: 5
[5]
Informe um valor: 8
[5, 8]
Informe um valor: 6
[5, 6, 8]
Informe um valor: -2
[-2, 5, 6, 8]
Informe um valor: -7
[-7, -2, 5, 6, 8]
Informe um valor: 9
[-7, -2, 5, 6, 8, 9]
Informe um valor: 2
[-7, -2, 2, 5, 6, 8]
Informe um valor: 1
[-7, -2, 1, 2, 5, 6]
Informe um valor: 0
```

3. **(Valor: 15 pontos)** Fazer um programa para gerar automaticamente uma lista de dimensão de **n** elementos (**n** deverá ser solicitado ao usuário, positivo e entre 7 e 60, inclusive), cada elemento dessa lista será um número aleatório entre 1 e 60 (inclusive) sem repetição. Após a lista ser gerada, as seguintes operações deverão ser implementadas:

   a) Deverá ser criada uma segunda lista com todas as combinações possíveis. Cada combinação deverá ser uma sub-lista. Cada combinação deverá estar ordenada do menor número para o maior;

   b) A lista de números escolhidos deverá ser salva em um arquivo chamado numeros_escolhidos.txt (salvar no mesmo diretório/pasta em que o programa está salvo). Nesse arquivo os números deverão estar em apenas 1 linha. Utilize ; para separar os valores na linha;

   c) A lista de combinações deverá ser salva em um arquivo chamado combinações.txt (salvar no mesmo diretório/pasta em que o programa está salvo). Nesse arquivo cada combinação deverá estar em 1 linha. Utilize ; para separar os valores na linha;

   d) No final deverão ser exibidos na tela quantas combinações foram geradas, e quais as probabilidades de se acertar a sena, a quina e a quadra.

4. **(Valor: 15 pontos)** Fazer um programa para gerar automaticamente uma lista (utilizar ***List Comprehensions***) de dimensão de **n** elementos (**n** deverá ser solicitado ao usuário e ser positivo), com os elementos na faixa dos números inteiros entre 0 e 99 (inclusive), gerados aleatoriamente.

   Uma vez a lista gerada, implementar as seguintes operações:

   a) A média dos valores dos elementos da lista;

   b) A mediana dos valores dos elementos da lista;

   c) A variância populacional dos valores dos elementos da lista;

   d) O desvio-padrão populacional dos valores dos elementos da lista.

   ATENÇÃO:

   i) NÃO USAR a biblioteca statistics.py para implementar o que foi pedido anteriormente;

   ii) Para fins de conferência, o aluno poderá utilizar as funções **mean()**, **median()**, **pvariance()** e **pstdev()** da biblioteca **statistics.py**.

5. **(Valor: 15 pontos)** Utilizando as listas abaixo...

```
gabarito = ['A', 'C', 'B', 'A', 'E', 'D', 'D', 'C', 'A', 'A']
lista_alunos = [ ['Aluno 01', 'B', 'D', 'E', 'E', 'C', 'D', 'A', 'B', 'C', 'D'],
                 ['Aluno 02', 'C', 'D', 'A', 'B', 'D', 'A', 'A', 'C', 'B', 'E'],
                 ['Aluno 03', 'A', 'A', 'B', 'D', 'C', 'E', 'E', 'A', 'A', 'C'],
                 ['Aluno 04', 'B', 'B', 'C', 'C', 'D', 'E', 'D', 'D', 'E', 'E'],
                 ['Aluno 05', 'B', 'B', 'D', 'A', 'A', 'E', 'B', 'D', 'E', 'C'],
                 ['Aluno 06', 'C', 'C', 'D', 'E', 'B', 'B', 'C', 'D', 'E', 'A'],
                 ['Aluno 07', 'B', 'A', 'A', 'B', 'B', 'C', 'D', 'E', 'A', 'B'],
                 ['Aluno 08', 'D', 'E', 'A', 'B', 'B', 'C', 'C', 'D', 'A', 'A'],
                 ['Aluno 09', 'A', 'A', 'A', 'C', 'B', 'D', 'D', 'E', 'D', 'C'],]
                 ['Aluno 10', 'B', 'B', 'D', 'E', 'C', 'D', 'C', 'E', 'B', 'A'] ]
```

Fazer um programa que realize as seguintes orientações:

a) O programa deverá adicionar no final de cada sub-lista do item (b) a quantidade de acertos do aluno;

b) A lista deverá ser ordenada pela quantidade de acertos de forma decrescente;

c) O programa deverá exibir no final o gabarito e os nomes de cada aluno, as opções que cada um marcou e a nota obtida.

6. **(Valor: 15 pontos)** A partir do arquivo **CotacoesDolar2023.csv**, fazer um programa que:

a) Monte uma lista onde cada posição deverá ser uma sub-lista. A primeira posição de cada sub-lista deverá ser o mês (Janeiro, Fevereiro,..., Dezembro), a segunda posição deverá ser a maior cotação de venda do respectivo mês e a terceira posição deverá ser a data relativa a essa maior cotação.

b) Montar uma segunda lista onde cada posição deverá ser uma sub-lista. A primeira posição de cada sub-lista deverá ser o mês (Janeiro, Fevereiro,..., Dezembro), e a segunda posição deverá ser a média das cotações de venda do respectivo mês;

c) Teste depois para o arquivo **CotacoesDolar2024.csv**;

ATENÇÃO:

i) Nos arquivos CSV, a ordem das informações são: **DATA**, **N/C**, **N/C**, **N/C**, **VALOR COMPRA**, **VALOR VENDA**, **N/C**, **N/C**;

ii) Deverão ser utilizadas as funções **MAP()**, **SORT()**, **FILTER()** e ***List Comprehensions*** quando possível,

7. **(Valor: 15 pontos)** A partir do arquivo **alunos_ifrn.csv**, fazer um programa que:

   a) Montar uma lista onde cada posição deverá ser uma sub-lista. A primeira posição de cada sub-lista deverá ser a sigla do campus e a segunda a quantidade de alunos daquele campus, no final deverá adicionada a cada sub-lista o percentual correspondente de alunos do campus em relação ao total de alunos do IFRN (limitar a 2 casas decimais). Essa lista deverá ser salva em um arquivo chamado **alunos_ifrn_campus.csv** (esse arquivo deverá ser salvo no mesmo diretório/pasta do programa). onde cada linha deverá ser os dados de cada sub-lista separados por **;**;

   b) Montar uma lista onde cada posição deverá ser uma sub-lista. A primeira posição de cada sub-lista deverá ser o ano de ingresso do aluno e a segunda posição a quantidade de alunos que ingressaram naquele ano. Essa lista deverá ser salva em um arquivo chamado **alunos_ifrn_ano.csv** (esse arquivo deverá ser salvo no mesmo diretório/pasta do programa). onde cada linha deverá ser os dados de cada sub-lista separados por **;**;

   c) Liste os campus, peça ao usuário para escolher um e montar uma segunda lista onde cada posição deverá ser uma sub-lista. A primeira posição de cada sub-lista deverá ser o nome do curso e a segunda posição deverá ser quantidade de alunos daquele curso naquele campus. Essa lista deverá ser salva em um arquivo chamado **alunos_ifrn_campus_curso.csv** (esse arquivo deverá ser salvo no mesmo diretório/pasta do programa). onde cada linha deverá ser os dados de cada sub-lista separados por **;**;

8. **(Valor: 15 pontos)** Se você executar o comando `tracert -d4 www.uol.com` em um terminal do Windows, terá o caminho que sua máquina faz até atingir o portal UOL. Um exemplo de resposta do comando está a seguir (delimitado por hifens):

```
Rastreando a rota para amazonas.uol.com.br [200.147.35.224] com no máximo 30 saltos:
1     9 ms     9 ms     9 ms  192.168.0.1
2    28 ms    21 ms    20 ms  10.17.0.1
3    22 ms    21 ms    24 ms  177.195.26.21
4     *        *       25 ms  201.57.195.101
5     *        *        *     Esgotado o tempo limite do pedido.
6     *        *        *     Esgotado o tempo limite do pedido.
7    83 ms    83 ms     *     200.230.251.2
8    86 ms    79 ms    81 ms  200.244.216.122
9    85 ms    88 ms    88 ms  200.211.219.210
10    80 ms    82 ms    82 ms  186.234.26.65
11    89 ms    80 ms    87 ms  200.147.26.30
12    80 ms    82 ms    79 ms  200.147.35.224
Rastreamento concluído.
```

É possível executar esse comando dentro de um programa em Python e obter um resultado como uma *string*, com toda a resposta.

Para tanto usando os seguintes comandos (ao final toda o resultado da rota está na varável **strCaminho** (as linhas estão separadas por **\r\n**):

```
import subprocess
strCMD = 'tracert -d4 www.uol.com'
caminho = subprocess.run (strCMD, capture_output=True).stdout.decode('latin1')
```

Faça um programa que:

a) Solicite ao usuário um nome de uma máquina de destino (HOST);

b) Em seguida realize o rastreamento e salve a resposta em um arquivo chamado **rastreio.txt**;

c) Indique o menor tempo em que cada uma das máquinas no caminho foi atingida. Ignore os tempos que não puderam ser determinados, aqueles com pelo menos um * na saída do *tracert*.