Atividade Avaliativa #04: Dicionários e Tratamento de Exceções

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: 30 pontos**) O código a seguir faz uma requisição a API do Banco Central e armazena na variável **dictMoedas** as moedas disponíveis na API e em **dictCotacoes** os dados relativos a cotações de compra e de venda do dólar em um determinado período. No exemplo pode-se ver que o período compreende o ano de 2021 completo.

```
import requests

strURL  = 'https://olinda.bcb.gov.br/olinda/servico/PTAX/versao/v1/odata'
strURL += '/Moedas?$top=100&$format=json'

dictMoedas = requests.get(strURL).json()

strURL  =  'https://olinda.bcb.gov.br/olinda/servico/PTAX/versao/v1/odata/'
strURL +=  'CotacaoMoedaPeriodo(moeda=@moeda,dataInicial='
strURL +=  '@dataInicial,dataFinalCotacao=@dataFinalCotacao)?'
strURL += f'@moeda=%27USD%27&@dataInicial=%2701-01-2023%27&'
strURL += f'@dataFinalCotacao=%2712-31-2023%27&$format=json'

dictCotacoes = requests.get(strURL).json()
```

Para se executar o código acima faz-se necessária a verificação se a biblioteca **REQUESTS** está instalada e, caso não esteja, deve-se instalá-la seguindo os passos a seguir (os comandos deverão ser digitados no terminal da IDE):

- Verificar se a biblioteca **REQUESTS** está instalada:
  ```
  pip list
  ```

- Caso a biblioteca **REQUESTS** não esteja instalada, precisamos instalá-la:
  ```
  pip install requests --user
  ```

Com base no trecho de código fornecido pede-se que seja implementado um programa que siga as instruções abaixo (*não esqueça de copiar o trecho do código fornecido para dentro do seu programa*):

i) Solicite ao usuário o ano desejado (o ano informado não pode ser superior ao ano atual, para isso deve-se usar as funções/métodos da biblioteca **DATETIME** para obter/validar o ano atual;

ii) Solicite ao usuário a moeda desejada (deve haver a validação se a moeda é válida ou não com base na variável **dictMoedas**);

iii) Com base no ano informado e na moeda informada efetuar os devidos ajustes no código para que seja feita a requisição relativa ao ano e moeda informados;

iv) Uma vez que a requisição retorne o dicionário com os dados solicitados, deve-se montar um dicionário onde as chaves serão os nomes dos meses e os respectivos valores serão dicionários cujas chaves serão **mediaCompra** e **mediaVenda** e os valores dessas chaves serão as médias de compra e de venda da moeda de cada mês (utilizar apenas os valores de fechamento da moeda em cada data para se calcular a média mensal). Fixar as médias com 5 casas decimais.

v) Uma vez que o dicionário do item **iv** foi gerado, deve-se salvá-lo, no formato de dicionário, em um arquivo com o nome **medias_cotacoes_MMM_AAAA.json** (substitua **MMM** pela sigla da moeda e **AAAA** pelo ano informado);

vi) Deve-se criar um segundo arquivo chamado **medias_cotacoes_MMM_AAAA.csv** (substitua **MMM** pela sigla da moeda e **AAAA** pelo ano informado) onde a primeira linha será: **moeda;mes;mediaCompra;mediaVenda;** e cada linha a seguir será a sigla da moeda na primeira coluna, o nome de cada mês na segunda coluna, a média de compra (com 5 casas decimais) na terceira coluna e a média de venda (com 5 casas decimais) na quarta coluna, lembrando de separar cada informação por **;** (ponto-e-vírgula);

vii) Lembre-se de tratar as devidas exceções no programa (conversões de valores, requisições na WEB, manipulação de arquivos, ...). **Elas são obrigatórias**;

ITEM BÔNUS (Valor: 10 pontos) Exibir um gráfico de linha demonstrando como as médias se comportam ao longo dos meses.

i) Coloque como título do gráfico: **Média Cotações MMM – Ano AAAA**
(substitua **MMM** pela sigla da moeda e **AAAA** pelo ano informado)

ii) Gerar uma linha para cada média (compra e venda);

iii) Adicionar uma legenda ao gráfico

2. **(Valor: 30 pontos)** O código a seguir faz uma requisição ao servidor do CartolaFC e armazena na variável **dictCartola** os dados relativos do CartolaFC do ano atual.

```
import requests

strURL = 'https://api.cartolafc.globo.com/atletas/mercado'

dictCartola = requests.get(strURL).json()
```

Para se executar o código acima faz-se necessária a verificação se a biblioteca **REQUESTS** está instalada e, caso não esteja, deve-se instalá-la seguindo os passos a seguir (os comandos deverão ser digitados no terminal da IDE):

- Verificar se a biblioteca **REQUESTS** está instalada:
  `pip list`

- Caso a biblioteca **REQUESTS** não esteja instalada, precisamos instalá-la:
  `pip install requests --user`

Com base no trecho de código fornecido pede-se que seja implementado um programa que siga as instruções abaixo (*não esqueça de copiar o trecho do código fornecido para dentro do seu programa*):

i) Solicite ao usuário o ano desejado (o ano informado não pode ser superior ao ano atual, para isso deve-se usar as funções/métodos da biblioteca **DATETIME** para obter/validar o ano atual;

ii) Com base no ano informado deve-se **(a)** requisitar na WEB (conforme código disponibilizado) caso seja informado o ano atual ou **(b)** carregar o dicionário do arquivo do CartolaFC relativo ao ano informado caso seja informado anos anteriores ao ano atual (os arquivos encontram-se disponíveis no Moodle);

iii) Uma vez que a variável **dictCartola** esteja com os dados do CartolaFC, o programa deverá solicitar o esquema tático desejado. Os esquemas táticos possíveis são: **3-4-3**, **3-5-2**, **4-3-3**, **4-4-2**, **4-5-1**, **5-3-2** ou **5-4-1**;

iv) Para cada esquema tático, deve-se selecionar a seguinte quantidade de jogadores por posição (vide tabela a seguir):

Esquema	Quantidade de Jogadores:
3-4-3	3 zagueiros / 0 laterais / 4 meias / 3 atacantes
3-5-2	3 zagueiros / 0 laterais / 5 meias / 2 atacantes
4-3-3	2 zagueiros / 2 laterais / 3 meias / 3 atacantes
4-4-2	2 zagueiros / 2 laterais / 4 meias / 2 atacantes
4-5-1	2 zagueiros / 2 laterais / 5 meias / 1 atacantes
5-3-2	3 zagueiros / 2 laterais / 3 meias / 2 atacantes
5-4-1	3 zagueiros / 2 laterais / 4 meias / 1 atacantes

v) Independente do esquema, todos terão de ter 1 goleiro e 1 técnico;

vi) A escolha dos atletas de cada posição será através dos atletas que tiverem a maior pontuação (**média de pontos x quantidade de partidas**) em cada posição (zagueiro, lateral, meia, atacante, goleiro, técnico). Atente para a quantidade de atletas em cada posição em função do esquema tático selecionado;

vii) Gerar um dicionário no formato **k:v** onde a chave **k** será o valor da chave será o id do atleta e o valor **k** será um dicionário no formato **k:v** onde as chaves ficam a critério de vocês e os valores serão: o id do atleta, o nome do atleta, o apelido do atleta, a URL da foto do atleta, o nome do clube, o escudo do clube, o id da posição, o nome da posição e a pontuação do atleta (**média de pontos x quantidade de partidas**). Esse dicionário deverá ser salvo no formato JSON no mesmo diretório/pasta do programa;

ATENÇÃO: Na URL da foto do jogador nem sempre tem a *string* `_220x220_`, as vezes vem a *string* `_FORMATO_` ou `_FORMATO`. Logo, quando aparecer a *string* `_FORMATO_` ela deve ser substituída pela *string* `_220x220_` e quando aparecer a *string* `_FORMATO` ela deve ser substituída pela *string* `_220x220`.

viii) O programa deverá exibir, após todo o processamento necessário, a seleção do Cartola FC com as seguintes informações: o nome da posição (obedecer a seguinte ordem: goleiro, zagueiro, lateral, meia, atacante, técnico), o nome do atleta e seu apelido, o nome do time que o atleta joga e a pontuação (**média x quantidade de partidas**) do atleta;

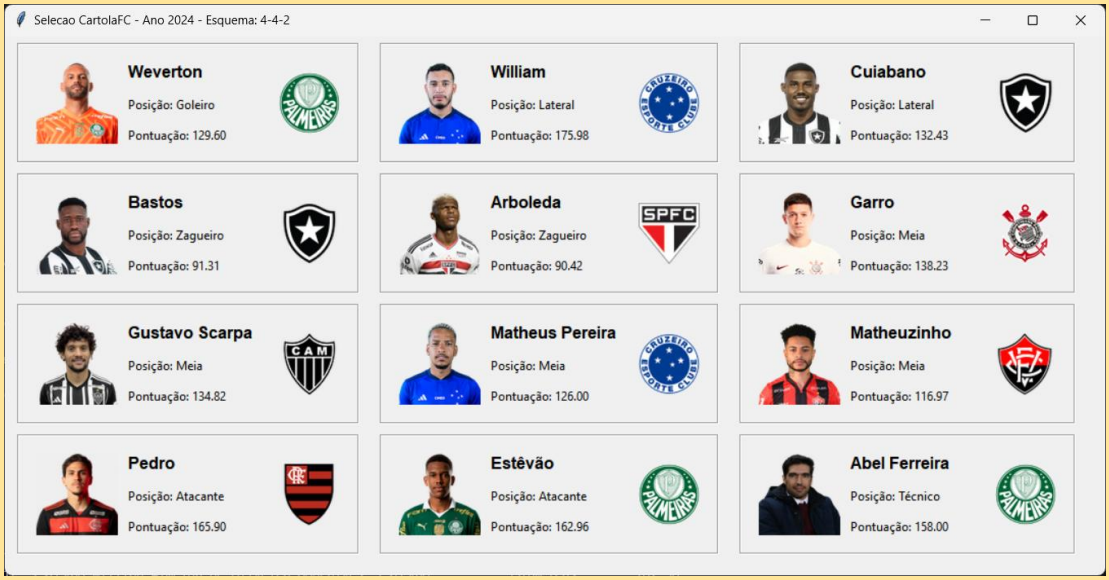
ix) Lembre-se de tratar as devidas exceções no programa (conversões de valores, requisições na WEB, manipulação de arquivos, ...). **Elas são obrigatórias**;

ITEM BÔNUS (Valor: 15 pontos) Exibir a escalação da seleção em uma janela gráfica usando a biblioteca Tkinter.

i) Documentação:
`https://docs.python.org/pt-br/3/library/tkinter.html`

ii) Além dos dados obtidos na questão, deve-se trazer também a foto do jogador e o escudo do time do jogador.

A seguir tem uma sugestão de como poderia ser essa tela, mas você pode elaborar o seu layout:



3. (Valor: 40 pontos) Implementar um programa de Bingo com as seguintes especificações:

i) O programa deverá apresentar as seguintes opções para o usuário (em negrito). Após cada opção está descrita a função a ser criada:

ii) Nas opções de gerar cartela, salvar cartela, ler cartela e imprimir cartela deverão ser criadas funções para as suas referidas tarefas. Para facilitar para vocês segue a especificação de cada uma delas:

1) **Gerar Cartelas**: Deverá ser criada uma função chamada **gerarCartelas**. Essa função deverá receber um valor inteiro correspondente a quantidade de cartelas a serem geradas e irá retornar uma tupla onde o primeiro elemento será **True** quando o dicionário for gerado com sucesso ou **False** caso tenha ocorrido algum erro na geração do dicionário e o segundo elemento será o dicionário gerado (quando o primeiro retorno for **True**) ou a mensagem de qual erro ocorreu (quando o primeiro retorno for **False**). A validação do argumento informado na função (quantidade de cartelas a serem geradas) deve ser implementada dentro da função. As especificações de como e quando o arquivo deverá ser lido estão descritas no item **ii** desta questão;

2) **Salvar Cartelas**: Deverá ser criada uma função chamada **salvarCartelas**. Essa função deverá receber um dicionário que corresponde as cartelas geradas e o nome do arquivo a ser salvo e irá retornar uma tupla onde o primeiro elemento será **True** quando o arquivo for gerado com sucesso ou **False** caso tenha ocorrido algum erro e o segundo elemento será uma mensagem de que o arquivo foi salvo (quando o primeiro retorno for **True**) ou a mensagem de qual erro ocorreu (quando o primeiro retorno for **False**). As validações dos argumentos informados na função (o dicionário a ser salvo e o nome do arquivo) devem ser implementadas dentro da função. As especificações de como e quando o arquivo deverá ser salvo estão descritas no item **iii** desta questão;

3) **Ler Cartelas**: Deverá ser criada uma função chamada **lerCartelas**. Essa função deverá receber o nome do arquivo a ser lido (arquivo que contém os dados das cartelas salvas) e irá retornar uma tupla onde o primeiro elemento será **True** quando o arquivo for lido com sucesso ou **False** caso tenha ocorrido algum erro na leitura e o segundo elemento será uma mensagem de que o arquivo foi lido com sucesso (quando o primeiro retorno for **True**) ou a mensagem de qual erro ocorreu (quando o primeiro retorno for **False**). A validação do argumento informado na função (o nome do arquivo a ser lido) deve ser implementada dentro da função. As especificações de como e quando o arquivo deverá ser lido estão descritas no item **iv** desta questão;

4) **Imprimir Cartelas**: Deverá ser criada uma função chamada **imprimirCartelas**. Essa função deverá receber o número da cartela a ser impressa e irá retornar uma tupla onde o primeiro elemento será **True** quando a cartela existir ou **False** caso tenha ocorrido algum erro na busca da cartela e o segundo elemento será uma *string* com o formato da cartela a ser impresso (quando o primeiro retorno for **True**) ou a mensagem de qual erro ocorreu (quando o primeiro retorno for **False**). A validação do argumento informado na função (o número da cartela a ser impressa) deve ser implementada dentro da função. As especificações de como a cartela será impressa está descrito no item **v** desta questão;

5) **Sair do Programa**

iii) A opção gerar cartelas deverá gerar n cartelas (n deverá ser solicitado pelo usuário e deverá ser positivo e no máximo permitir que sejam geradas 10.000 cartelas). A geração de cartelas deverá obedecer aos seguintes critérios:

a. Cada cartela é composta por 25 números aleatórios distribuídos nas letras **B** (1 a 15), **I** (16 a 30), **N** (31 a 45), **G** (46 a 60) e **O** (61 a 75), onde cada letra da cartela (**B**, **I**, **N**, **G**, **O**) possui 5 números distintos e ordenados de forma crescente;

b. **NÃO** poderá haver cartelas **EXATAMENTE** iguais;

c. Cada cartela deverá possuir um número identificador (esse número deverá ser aleatório e único dentro da faixa de 1 a 100.000) composto por **CART_xxxxxx**, onde **xxxxxx** é o número da cartela com 6 dígitos (complementar com zeros a esquerda quando necessário);

d. A estrutura do dicionário para armazenar as cartelas deverá ser a seguinte:

```
{
   numero_cartela: [[b1, b2, b3, b4, b5], [i1, i2, i3, i4, i5],
   [n1, n2, n3, n4, n5], [g1, g2, g3, g4, g5], [o1, o2, o3, o4, o5]],

   numero_cartela: [[b1, b2, b3, b4, b5], [i1, i2, i3, i4, i5],
   [n1, n2, n3, n4, n5], [g1, g2, g3, g4, g5], [o1, o2, o3, o4, o5]],
   ...
}
```

iv) A opção de salvar cartelas deverá salvar as cartelas geradas no item **ii** em um arquivo chamado **cartelas.txt** onde em cada linha irá conter o número da cartela e os números que foram gerados para ela. Utilizar **;** (ponto e vírgula) como separador. Lembre-se de **(i)** essa opção só poderá ser executada se houve cartelas geradas e **(ii)** o arquivo deverá ser salvo na mesma pasta/diretório do arquivo **.py**;

v) A opção de ler cartelas deverá ler o arquivo **cartelas.txt** e remontar o dicionário com os dados das cartelas salvas. Lembre-se que o arquivo deve existir, caso contrário, o usuário deverá ser informado de que o arquivo não existe;

vi) A opção de imprimir cartelas deverá solicitar ao usuário um valor **n** correspondente ao número da cartela (lembrar dos limitantes da geração das cartelas), buscar a cartela no dicionário e imprimi-la conforme modelo a seguir. Lembre-se de que essa opção só poderá ser executada se houve cartelas geradas:

ITEM BÔNUS (Valor: 15 pontos) Criar uma opção no menu para sortear dezenas.

i) O sorteio inicial deverá ser de 25 dezenas sem repetições entre 1 e 75;

ii) Após o primeiro sorteio deverá ser verificada se houve uma ou mais cartelas batidas;

iii) Caso não tenha havido cartela batida, deve-se sortear uma nova dezena (ela deve ser acrescentada as dezenas já sorteadas e não deve ser repetida). Faz-se novamente a checagem se houve uma ou mais cartelas batidas.

iv) Caso não tenha havido cartela(s) batidas no item **iii**, deve-se repeti-lo até que se tenha cartela(s) batida(s)

v) Uma vez que se tenha(m) cartela(s) batida(s), listar os números sorteados e a as cartelas batidas.