|  |  |
| --- | --- |
| **Disciplina:** Estatística Aplicada | **Data:** 22/02/2021 |
| **Professor (a):** Luiz Carlos dos Santos Filho | |
| **Alunos (as)** | **RA** |
| Daniel Fortunato de Castro | 2920482021036 |
| Felipe Almeida Dias | 2920482021001 |

**Hora do Código**

**Medidas de Tendência Central**

Conforme visto na teoria desenvolvida nas aulas, média, moda, mediana (2º quartil), 1º quartil e 3º quartil, são chamadas, em estatística, de medidas de tendência central.  
Elabore um código na linguagem de sua preferência que receba como entrada os dados numéricos de um arquivo TXT ou CSV e apresente como saída o cálculo das medidas de tendência central.

1. Copie e cole neste documento o código e o print da tela de sua execução com os resultados dos cálculos para os dados abaixo.
2. Organize o desenvolvimento desta atividade, pois será solicitado a demonstração do funcionamento para alguns conjuntos de dados que serão informados durante a demonstração.

Dados para o cálculo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 680 | 663 | 700 | 690 | 683 |
| 669 | 658 | 718 | 694 | 735 |
| 719 | 634 | 690 | 660 | 688 |
| 699 | 720 | 681 | 649 | 704 |
| 670 | 690 | 702 | 675 | 672 |
| 710 | 677 | 696 | 701 | 698 |
| 722 | 669 | 692 | 721 | 659 |

Critérios de avaliação:

1. apresentação do trabalho escrito: documentação do código, comentários no código, apresentação dos resultados dos cálculos,
2. formatação da entrada e da saída durante a execução,
3. utilização otimizada de código, por exemplo, clareza, concisão, funções, procedures, etc
4. funcionamento adequado e cálculos corretos,
5. apresentação oral com a demonstração do funcionamento.

**Obs: O objetivo desta atividade é compreender o método de cálculo destas medidas, portanto não use bibliotecas com funções prontas para as medidas de tendência central, caso necessário, construa você a função ou o procedimento. Para cálculos intermediários poderá ser utilizada bibliotecas, por exemplo: ordenação, somatória, etc.**

**╔═════════════════════════════════════╗**

**║                                modfunctions.py                                ║**

**╚═════════════════════════════════════╝**

Este arquivo Python armazena as funções do projeto principal (main.py) que são acessadas por meio de módulos.

- Link GITHUB

<https://github.com/Dev-Castro/EA_FATEC-FRV/blob/master/modfunctions.py>

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Entrada de dados**

Esta função recebe um arquivo txt como entrada e armazena seus valores em uma lista.

def entrada\_de\_dados(arquivo):

    found = False

    # Tente abrir o arquivo e adicionar os valores a lista.

    try:

        valores = []

        # Abre o arquivo de entrada no modo Leitura.

        with open(arquivo, "r") as entrada:

            # Lê um por um dos valores do arquivo.

            for valor in entrada:

                # Analisa a disposição dos valores no arquivo.

                valor = valor.split(" ")

                # Se os valores estiverem separados por espaço, adicione um por um.

                if (type(valor) == list):

                    for i in valor:

                        valores.append(float(i))

                # Se não, adicione o valor.

                else: valores.append(float(valor))

            # Fecha o arquivo de entrada.

            entrada.close()

            found = True

            return found, valores

    # Se não encontrar o arquivo, verifica se o usuário quer continuar no programa ou sair.

    except FileNotFoundError:

        found = False

        print("\n\033[1;35mArquivo não encontrado...\033[m\n")

        print("\033[1;42m[1]\033[m - \033[32mTentar novamente\033[m ")

        print("\033[1;41m[2]\033[m - \033[31mSair\033[m\n")

        resposta = int(input())

        return found, resposta

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Rol de dados**

Esta função recebe a lista de valores e os apresenta no terminal com formatação matricial.

def rolDeDados(valores):

    # Armazena valor de raiz maior ou equivalente.

    raiz = 0

    # Conta o print dos valores.

    conta = 0

    # Descobre raiz maior ou equivalente a quantidade de valores do arquivo (para criar tabela proporcional).

    while(raiz\*\*2 < len(valores)):

        raiz+=1

    # Ordenação dos dados

    valores.sort()

    # Print dos valores do arquivo em tabela.

    print("\n\033[1;35m       Valores do arquivo:\033[m\n")

    while(conta < len(valores)):

        print("\033[1;35m   |\033[m", end = " ")

        for i in range(raiz):

            if(conta==len(valores)):

                break

            print("%.2f" % valores[conta], end = "  ")

            conta+=1

        print("\033[1;35m|\033[m\n")

    print("\n\033[35m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m")

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Média**

Esta função recebe a lista de valores, calcula sua média aritmética e apresenta os resultados no terminal.

def media(valores):

    # MEDIA

    print("\033[33m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m")

    print("""\033[1;33m

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|                         /                                  |

|            |\    /|  \_\_\_\_\_   \_\_\_\_                          |

|            | \  / | |       |    \   0    /\               |

|            |  \/  | |\_\_\_\_   |     |  |   /  \              |

|            |      | |       |     |  |  /----\             |

|            |      | |\_\_\_\_\_  |\_\_\_\_/   | /      \            |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

    \033[m""")

    # Armazena a soma dos valores da lista.

    soma = 0

    # Quantidade de valores na lista.

    quantidade = len(valores)

    # Soma todos os valores da lista(valores).

    for valor in valores:

        soma = soma + valor

    # Calcula a média aritmética dos valores da lista.

    media = soma / quantidade

    # Imprime no terminal o calculo de média.

    print("\n  \033[1;33mMédia\033[m é a soma de todos os valores de um conjunto de dados\n  dividido pela quantidade de dados.")

    print("\n  Somatória = %.2f" % soma)

    print("  Quantidade = %i" % quantidade)

    print("""

    %.2f

    ▬▬▬▬▬▬▬▬ =  \033[1;33m%.2f\033[m = Média

        %i

    """ % (soma, media, quantidade))

    print("\033[33m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m")

    print("\033[31m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m")

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Moda**

Esta função recebe a lista de valores, calcula a frequência de repetição dos valores e apresenta os resultados no terminal.

def moda(valores):

    # MODA

    print("""\033[1;31m

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|                                                            |

|             |\    /|  \_\_\_\_\_   \_\_\_\_                         |

|             | \  / | |     | |    \     /\                 |

|             |  \/  | |     | |     |   /  \                |

|             |      | |     | |     |  /----\               |

|             |      | |\_\_\_\_\_| |\_\_\_\_/  /      \              |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

    \033[m""")

    print("\n\033[1;31mModa\033[m é o valor mais frequente em um conjunto de dados.")

    # Quantidade de valores.

    quantidade = len(valores)

    # Conta qual valor está se repetindo.

    moda = []

    # Conta quantas vezes se repetiu.

    repeticoes = []

    # Classifica o tipo de moda.

    classe = ["AMODAL", "UNIMODAL", "BIMODAL", "MULTIMODAL"]

    # Indice de classificação.

    indice = 0

    # Armazena as maiores ocorrências de repetição.

    maioresRepeticoes = []

    # Armazena os valores que mais repetiram.

    modas = []

    # Armazena valores que se repetem(para o print).

    multimodal = []

    # Todos os valores que repetem e não repetem, porém aparecendo apenas uma vez cada.

    united = []

    # Ordena os valores em ordem crescente.

    valores.sort()

    # Analisa o número de repetições de cada valor e armazena nas listas.

    for i in range(quantidade):

        conta = valores.count(valores[i])

        repeticoes.append(conta)

        moda.append(valores[i])

    # Descobrindo se é Amodal

    ### Se a soma de repetições dos valores divido pela quantidade de valores é igual a 1, significa que não há repetições pois quando um valor não se repete é armazenado o número 1 na lista repetições.

    if(sum(repeticoes)/quantidade == 1):

        # Classificação Amodal

        indice = 0

        # Print info moda.

        print("\nA classificação desse conjunto de dados é: \033[1;31m%s\033[m." % classe[indice])

        print("Quando não há repetições no conjunto de dados.")

    # Quando todos os valores se repetem, porém em mesma frequência.

    elif(sum(repeticoes)/quantidade == max(repeticoes)):

        # Classificação Amodal

        indice = 0

        # Print info moda.

        print("\nA classificação desse conjunto de dados é: \033[1;31m%s\033[m." % classe[indice])

        print("Quando todos os valores se repetem, porém em mesma frequência.")

    else:

        ### Se não for amodal, só pode ser unimodal, bimodal ou multimodal.

        # Buscar e armazenar as três maiores ocorrêcias de repetição(para identificar a classificação do conjunto de dados).

        ### Sabendo as três maiores ocorrências de repetição, podemos saber se a classificação do conjunto de dados é unimodal, bimodal ou multimodal.

        ### (unimodal) A > B , C

        ### (bimodal) A = B > C

        ### (multimodal) A = B = C

        for i in range(3):

            # Busca a maior ocorrência de repetição.

            maior = max(repeticoes)

            # Busca o índice de maior de repetição.

            modinha = repeticoes.index(maior)

            # Armazena a maior ocorrência de repetição.

            maioresRepeticoes.append(maior)

            # Armazena o valor que se repete.

            modas.append(moda[modinha])

            # Deleta a maior ocorrência atual para achar a próxima no próximo ciclo de repetição do For.

            del moda [modinha : modinha + maior]

            del repeticoes [modinha : modinha + maior]

        # Descobrindo se é Multimodal

        if (max(maioresRepeticoes)==min(maioresRepeticoes)):

            # Classificação Multimodal.

            indice = 3

            # Armazena todos os valores sem suas repetições.

            for i in valores:

                if(not i in united):

                    united.append(i)

            # Analisa quais se repetem em maior e mesma frequência.

            for j in united:

                # Descobre quais são os que se repetem mais (os multimodais).

                if(valores.count(j)>=max(maioresRepeticoes)):

                    multimodal.append(j)

            # Print info moda.

            print("\nA classificação desse conjunto de dados é: \033[1;31m%s\033[m." % classe[indice])

            print("Quando há mais de dois valores que se repetem em mesma frequência no\nconjunto de dados.")

            print("\nValores que se repetem: \033[1;31m%a\033[m, que se repetem \033[1;31m%i\033[m vezes." % (multimodal, max(maioresRepeticoes)))

        else:

            # Deleta o menor valor.

            ### Deletando o menor valor, sobram apenas os dois maiores valores. Se ambos tiverem mesmo valor de repetição, então o conjunto de dados é bimodal, se não é modal.

            del modas[maioresRepeticoes.index(min(maioresRepeticoes))]

            del maioresRepeticoes[maioresRepeticoes.index(min(maioresRepeticoes))]

            # Descobrindo se é Bimodal.

            if (max(maioresRepeticoes)==min(maioresRepeticoes)):

                # Classificação Bimodal.

                indice = 2

                # Print info moda.

                print("\nA classificação desse conjunto de dados é: \033[1;31m%s\033[m." % classe[indice])

                print("Quando há dois valores que se repetem em mesma frequência no\nconjunto de dados.")

                print("\nValores que se repetem: \033[1;31m%.2f\033[m e \033[1;31m%.2f\033[m, que se repetem \033[1;31m%i\033[m vezes." % (modas[0], modas[1], max(maioresRepeticoes)))

            # Se não, é Unimodal.

            else:

                # Classificação Uniodal.

                indice = 1

                # Print info moda.

                print("\nA classificação desse conjunto de dados é: \033[1;31m%s\033[m." % classe[indice])

                print("Quando há um valor que se repete em maior frequência no\nconjunto de dados.")

                print("\nO valor que mais se repete é: \033[1;31m%.2f\033[m, que se repete \033[1;31m%i\033[m vezes." % (modas[maioresRepeticoes.index(max(maioresRepeticoes))], max(maioresRepeticoes)))

    print("\n\033[31m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m")

    print("\033[36m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m")

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Mediana**

Esta função recebe a lista de valores, calcula os quartis e apresenta os resultados no terminal.

def mediana(valores):

    # MEDIANA (2° Quartil)

    print("""\033[1;36m

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|                                                            |

|    |\    /|  \_\_\_\_\_   \_\_\_\_                                  |

|    | \  / | |       |    \   0    /\    |\   |    /\       |

|    |  \/  | |\_\_\_\_   |     |  |   /  \   | \  |   /  \      |

|    |      | |       |     |  |  /----\  |  \ |  /----\     |

|    |      | |\_\_\_\_\_  |\_\_\_\_/   | /      \ |   \| /      \    |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

    \033[m""")

    porcent = "%"

    quantidade = len(valores)

    print("A \033[1;36mMediana\033[m é o valor central em um conjunto de dados.\n")

    print("O \033[1;36m1° Quartil\033[m é o valor aos 25%s da amostra ordenada.\n" % porcent)

    print("O \033[1;36m3° Quartil\033[m é o valor aos 75%s da amostra ordenada.\n" % porcent)

    ### O cálculo é simples, porém o maior código foi preocupado em fazer um print intuitivo e bem explicito para o conjunto de dados.

    # Conjunto de dados com total de elementos par.

    if (quantidade%2 == 0):

        # Calcula as posições centrais.

        posicaoSQ = int(quantidade/2)-1

        # Calcula o valor do 2° Quartil.

        segundoQuartil = (valores[posicaoSQ] + valores[posicaoSQ+1]) /2

        # Divide em listas a primeira e segunda metade do conjunto de dados para calcular o 1° e o 3° Quartil.

        primeiroQuartil = valores[0:posicaoSQ+1]

        terceiroQuartil = valores[posicaoSQ+1:quantidade]

        # Par(2°) - Par(1° e 3°)

        if (len(primeiroQuartil)%2==0):

            # Identifica as posições dos quartis e seus valores.

            posicaoPQ = int(len(primeiroQuartil)/2)-1

            primeiroQuartil = (primeiroQuartil[posicaoPQ] + primeiroQuartil[posicaoPQ+1])/2

            posicaoTQ = posicaoPQ+len(terceiroQuartil)+1

            terceiroQuartil = (terceiroQuartil[posicaoPQ] + terceiroQuartil[posicaoPQ+1])/2

            # Posição exata dos Quartis no conjunto de dados.

            valores.insert(posicaoPQ+1, 0)

            valores.insert(posicaoTQ+1, 0)

            valores.insert(posicaoSQ+2, 0)

            # Print avançado do conjunto de dados identificando os Quartis.

            for i in range(len(valores)):

                if(i==posicaoPQ+1):

                    print("\033[1;36m ─────── %.2f 1° Quartil\033[m" % primeiroQuartil)

                elif(i==posicaoSQ+2):

                    print("\033[1;36m ─────── %.2f 2° Quartil\033[m" % segundoQuartil)

                elif(i==posicaoTQ+2):

                    print("\033[1;36m ─────── %.2f 3° Quartil\033[m" % terceiroQuartil)

                else: print("║%.2f" % valores[i])

        # Par(2°) - Impar(1° e 3°)

        else:

            # Identifica as posições dos quartis e seus valores.

            posicaoPQ = int((len(primeiroQuartil)+1)/2)-1

            primeiroQuartil = primeiroQuartil[posicaoPQ]

            posicaoTQ = posicaoPQ+len(terceiroQuartil)+1

            terceiroQuartil = terceiroQuartil[posicaoPQ]

            # Posição exata do 2° Quartil no conjunto de dados.

            valores.insert(posicaoSQ+1, 0)

            # Print avançado do conjunto de dados identificando os Quartis.

            for i in range(len(valores)):

                if(i==posicaoPQ):

                    print("\033[1;36m╠%.2f ─── 1° Quartil\033[m" % valores[i])

                elif(i==posicaoSQ+1):

                    print("\033[1;36m ─────── %.2f 2° Quartil\033[m" % segundoQuartil)

                elif(i==posicaoTQ):

                    print("\033[1;36m╠%.2f ─── 3° Quartil\033[m" % valores[i])

                else: print("║%.2f" % valores[i])

    # Conjunto de dados com total de elementos ímpar.

    else:

        # Calcula a posição central.

        posicaoSQ = int((quantidade+1)/2)-1

        # Calcula o valor do 2° Qaurtil.

        segundoQuartil = valores[posicaoSQ]

        # Divide em listas a primeira e segunda metade do conjunto de dados para calcular o 1° e o 3° Quartil.

        primeiroQuartil = valores[0:posicaoSQ]

        terceiroQuartil = valores[posicaoSQ+1:quantidade]

        # Impar(2°) - Par(1° e 3°)

        if (len(primeiroQuartil)%2==0):

            # Identifica as posições dos quartis e seus valores.

            posicaoPQ = int(len(primeiroQuartil)/2)-1

            primeiroQuartil = (primeiroQuartil[posicaoPQ] + primeiroQuartil[posicaoPQ+1])/2

            posicaoTQ = posicaoPQ+len(terceiroQuartil)+2

            terceiroQuartil = (terceiroQuartil[posicaoPQ] + terceiroQuartil[posicaoPQ+1])/2

            # Posição exata dos Quartis no conjunto de dados.

            valores.insert(posicaoPQ+1, 0)

            valores.insert(posicaoTQ+1, 0)

            # Print avançado do conjunto de dados identificando os Quartis.

            for i in range(len(valores)):

                if(i==posicaoPQ+1):

                    print("\033[1;36m ─────── %.2f 1° Quartil\033[m" % primeiroQuartil)

                elif(i==posicaoSQ+1):

                    print("\033[1;36m╠%.2f ─── 2° Quartil\033[m" % valores[i])

                elif(i==posicaoTQ+1):

                    print("\033[1;36m ─────── %.2f 3° Quartil\033[m" % terceiroQuartil)

                else: print("║%.2f" % valores[i])

        # Impar(2°) - Impar(1° e 3°)

        else:

            # Identifica as posições dos quartis e seus valores.

            posicaoPQ = int((len(primeiroQuartil)+1)/2)-1

            primeiroQuartil = primeiroQuartil[posicaoPQ]

            posicaoTQ = posicaoPQ+len(terceiroQuartil)+1

            terceiroQuartil = terceiroQuartil[posicaoPQ]

            # Print avançado do conjunto de dados identificando os Quartis.

            for i in range(len(valores)):

                if(i==posicaoPQ):

                    print("\033[1;36m╠%.2f ─── 1° Quartil\033[m" % valores[i])

                elif(i==posicaoSQ):

                    print("\033[1;36m╠%.2f ─── 2° Quartil\033[m" % valores[i])

                elif(i==posicaoTQ):

                    print("\033[1;36m╠%.2f ─── 3° Quartil\033[m" % valores[i])

                else: print("║%.2f" % valores[i])

    print("\n\033[36m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m\n")

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**╔═════════════════════════════════════╗**

**║                                 main.py                                ║**

**╚═════════════════════════════════════╝**

Este é o programa principal, ele é responsável pela interface do usuário dentro do terminal e direcionamento do processamento dos dados. Através de um módulo o programa usa funções dentro do arquivo modfuncitons.py.

- Link GITHUB

<https://github.com/Dev-Castro/EA_FATEC-FRV/blob/master/main.py>

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

from os import system

import modfunctions

# Laço de repetição infinito que servirá como chave de controle para fechar ou manter aberto o programa ao término de execução.

while(True):

    system('cls')

    # Entrada do nome do Arquivo.

    print("\n\033[35m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m\n")

    arquivo = str(input("\033[1mDigite o nome do arquivo (txt):\033[m "))

    print("\n\033[35m▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣\033[m\n")

    # Identifica se na entrada o usuário digitou o nome do arquivo com ou sem extensão e corrige.

    if (not ".txt" in arquivo):

        arquivo = arquivo + ".txt"

    found, valores = modfunctions.entrada\_de\_dados(arquivo)

    if(found):

        modfunctions.rolDeDados(valores)

        modfunctions.media(valores)

        modfunctions.moda(valores)

        modfunctions.mediana(valores)

        # Verifica se o usuário quer continuar no programa ou sair.

        print("\033[1;42m[1]\033[m - \033[32mContinuar\033[m")

        print("\033[1;41m[2]\033[m - \033[31mSair\033[m\n")

        resposta = int(input())

        if(resposta==2):

            break

        elif(resposta==1):

            system('cls')

        else:

            print("\n\033[35mEste valor não é válido!\033[m\n")

            break

    else:

        resposta = valores

        if(resposta==2):

            break

        elif(resposta==1):

            system('cls')

        else:

            print("\n\033[35mEste valor não é válido!\033[m\n")

            break

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

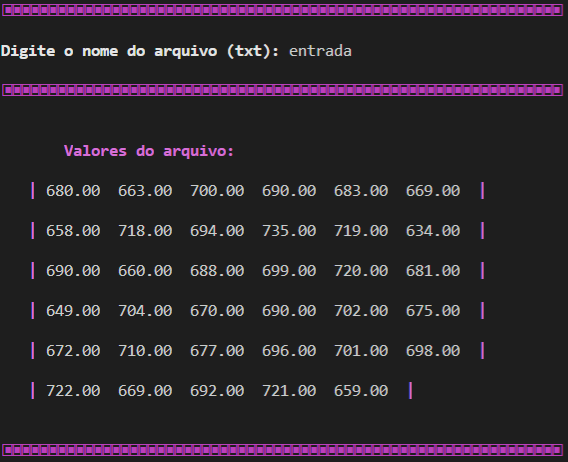
**╔═════════════════════════════════════╗**

**║                               EXECUÇÃO                           ║**

**╚═════════════════════════════════════╝**

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Rol de dados**



▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Média**

****

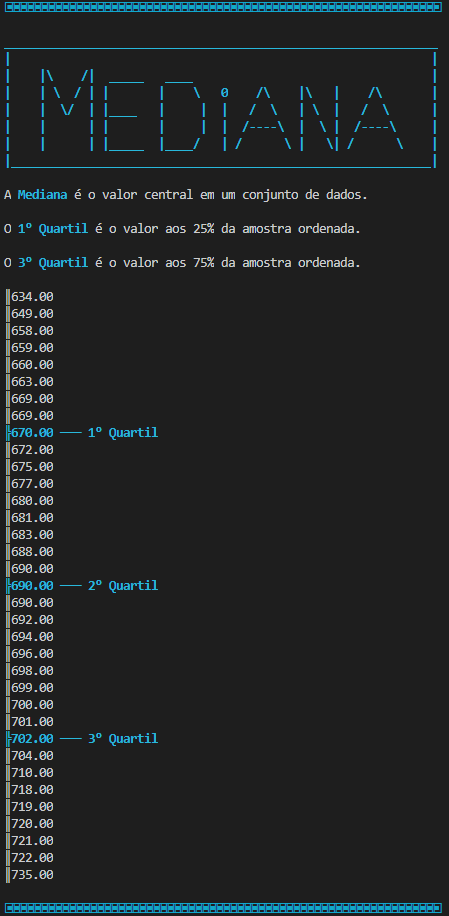
▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Moda**

****

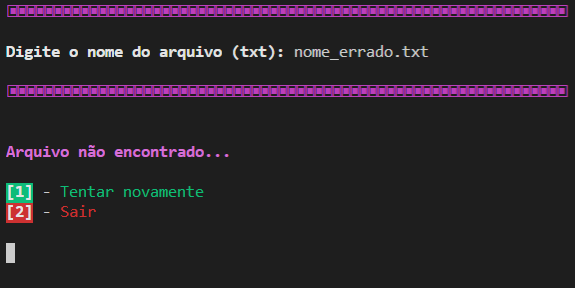
▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Mediana**



▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Arquivo não encontrado**

****

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

**Fim de execução**

****

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬