Mundo 3 - Listas Compostas

1) Faça um programa que leia nome e peso de várias pessoas, guardando tudo em uma lista. No final mostre:

- Quantas pessoas foram cadastradas
- Uma listagem com as pessoas mais pesadas
- Uma contagem com as pessoas mais leves

Guanabara)

Na solução dele ele reduziu o código onde toda a verificação é feito dentro do while, ele não varre a lista depois que ela é criada, então o nós temos:

- Uma lista temporária
- Uma lista principal
- Duas variáveis para maior e menor

Estrutura de repetição

- Enquanto for verdadeiro, ele vai solicitar o nome e dar append na lista temp, juntamente com o peso
- Se a lista prin não tiver nada, o maior e menor terá o valor da primeira temporária
- Se não, entra na estrutura de maior e menor
- No final dá um append à lista principal
- Limpa a lista temporária para fazer o loop novamente
- Existe aquele estrutura do resp para verificar se quer continuar ou não

No final nós temos um tipo de formação para mostrar todos os dados preenchidos, qual foi o maior peso e qual foi o menor.

A parte legal é o for para mostrar o nome a sua esturura basicamente é:

- Varrer a lista print, se a posição do P1 for igual à variável maior, então ele vai printar o P1

```
temp = list()
princ = list()
mai = men = 0
while True:
    temp.append(input('Nome: '))
    temp.append(float(input('Peso: ')))
    if len(princ) == 0:
        mai = men = temp[1]
    else:
        if temp[1] > mai:
            mai = temp[1]
        if temp[1] < men:
```

```
            men = temp[1]
    princ.append(temp[:])
    temp.clear()
    resp = input('Quer continuar [S/N] ?: ')
    if resp in 'Nn':
        break
    print('-='*30)
print(f'Os dados preenchidos foram {princ}')
print(f'O maior peso fo de {mai} Kg do(s) aluno(s): ',end='')
for p in princ:
    if p[1] == mai:
        print(p[0])
print(f'O menor peso foi de {men} Kg do(s) aluno(s):   ',end='')
for p in princ:
    if p[1] == men:
        print(p[0])
```

a)

O Guanabara resolveu de forma diferente, pois determinou apenas os maiores e menores, eu fiz o seguinte:

- Declarado uma lista chamada temp (ela vai servir para armazenar temporariamente os valores)
- Declarado uma lista chamda pessoas (ela vai armazenar todas as listas / vai ser a composta)
- Declarado listas pesadas e leves para armazenar o nome e peso das pessoas
- Declarado listas auxiliares para realizarmos os append pra dentro delas.

Partimos então para a estrutura while:

- Damos append para nome e peso na lista temporária
- Damos append dos valores de temp para pessoas
- Limpamos a lista temporária

A cada loop vai ser pedido a resp, se sim, continua, se não, para e se for diferente dá como inválida e não continua

Armazenei numa variável totp o tamanho da lista, para saber a quantidade de registros

Agora na parte que vamos varrer os valores da lista pessoas:

- Num for dentro de pessoas, usamos o iterador p para percorrer TODAS as posições das listas
- Se o P1 da posição macro for maior que 50, então damos append no p[0] e p[1] na auxiliadora
- Depois damos append na lista principal pesadas
- Funcina da mesma forma para a lista de leves, porém o peso tem que ser menor que 50

- No final do programa, ele mostra as listas

```
temp = list()
pessoas = list()
pesadas = list()
aux_pes = list()
leves = list()
aux_lev = list()
while True:
    temp.append(input('Nome: '))
    temp.append(int(input('Peso: ')))
    pessoas.append(temp[:])
    temp.clear()
    resp = input('Quer continuar [S/N] ?: ')
    while resp not in 'SsNn' or resp == '':
        resp = input('Valor inválido! Quer continuar [S/N] ?: ')
    if resp in 'Nn':
        break
    print('-='*30)
totp = len(pessoas)
for p in pessoas:
    if p[1] > 50:
        aux_pes.append(p[0])
        aux_pes.append(p[1])
        pesadas.append(aux_pes[:])
        aux_pes.clear()
    if p[1] < 50:
        aux_lev.append(p[0])
        aux_lev.append(p[1])
        leves.append(aux_lev[:])
        aux_lev.clear()
print(f'No total foram cadastradas {totp} pessoas')
print(f'As mais pesadas são = {pesadas}')
print(f'As mais leves são = {leves}')
```

Outra forma que podemos também simplificar a parte de condicional seria:

```
temp = list()
pessoas = list()
pesadas = list()
aux_pes = list()
leves = list()
aux_lev = list()
while True:
    temp.append(input('Nome: '))
    temp.append(int(input('Peso: ')))
    pessoas.append(temp[:])
    temp.clear()
```

```
    resp = input('Quer continuar [S/N] ?: ')
    while resp not in 'SsNn' or resp == '':
        resp = input('Valor inválido! Quer continuar [S/N] ?: ')
    if resp in 'Nn':
        break
    print('-='*30)
totp = len(pessoas)
for p in pessoas:
    if p[1] > 50:
        pesadas.append([p[0],p[1]])
        aux_pes.clear()
    if p[1] < 50:
        leves.append([p[0],p[1]])
        aux_lev.clear()
print(f'No total foram cadastradas {totp} pessoas')
print(f'As mais pesadas são = {pesadas}')
print(f'As mais leves são = {leves}')
```

Comentário

O que não faz sentido é utilizar apenas um interador para percorrer todas as posições e ele mesmo para mostrar uma posição específica, eu achava que deveria colocar como pessoas[p][1] ou p[0][1], mas desta forma usamos apenas 1.

Consegui também resolver de uma outra forma, nesse script podemos escolher quantas vezes o loop vai ser repetir e o programa utiliza dois fors para percorrer a lista composta.

```
temp = list()
pessoas = list()
pesadas = list()
leves = list()
quant = int(input('Quantas pessoas deseja registrar?: '))
print('-='*30)
for c in range(0,quant):
    temp.append(input('Nome: '))
    temp.append(int(input('Peso: ')))
    pessoas.append(temp[:])
    temp.clear()
    print('-='*30)
totp = len(pessoas)
for c in range(0,quant):
    if pessoas[c][1] > 50:
        pesadas.append([pessoas[c][0],pessoas[c][1]])
    if pessoas[c][1] < 50:
        leves.append([pessoas[c][0],pessoas[c][1]])
print(f'No total foram cadastradas {totp} pessoas')
```

```
print(f'As mais pesadas são = {pesadas}')
print(f'As mais leves são = {leves}')
```

2) Crie um programa onde o usuário possa digitar sete valores numéricos e cadastre-os em uma lista única que mantenha separados os valores pares e impares. No final mostre os valores pares e impares em ordem crescente

a) Um laço que se repete vezes

```
lista_num = [[],[]]
for c in range(0,7):
    n = int(input(f'Digite o {c+1}º valor: '))
    if n % 2 == 0:
        lista_num[0].append(n)
    else:
        lista_num[1].append(n)
print('-='*30)
print(f'Os valores digitados pares foram: {lista_num[0]}')
print(f'Os valores digitados ímpares foram: {lista_num[1]}')
```

b) Um laço que se repete quantas vezes quiser

```
lista_num = [[],[]]
while True:
    n = int(input('Digite um número: '))
    if n % 2 == 0:
        lista_num[0].append(n)
    else:
        lista_num[1].append(n)
    res = input('Quer continuar [S/N]?: ')
    while res not in 'SsNn' or res == '':
        res = input('Valor inválido! Quer continuar [S/N]?: ')
    if res in 'Nn':
        break
print('-='*30)
print(f'Os valores pares digitados são: {lista_num[0]}')
print(f'Os valores ímpares digitados são: {lista_num[1]}')
```

Guanabara)

```
n = 0
valores = [],[]
```

```
for c in range(0,7):
    n = (int(input(f'Digite o {c+1}º valor: ')))
    if n % 2 == 0:
        valores[0].append(n)
    else:
        valores[1].append(n)
    valores[0].sort()
    valores[1].sort()
print('-='*30)
print(f'Os valores pares digitados foram = {valores[0]}')
print(f'Os valores impares digitados foram = {valores[1]}')
```

3) Crie um programa que crie uma matriz de dimensão 3 x 3 e preencha com valores lidos pelo teclado. Mostre a matriz na tela com a formatação correta.

a) Minha solução foi um pouco diferente do Guanabara, na verdade eu criei um script que cria uma matriz conforme a nossa necessidade e no final solicita o valor de cada posição

```
colunas = int(input('Quantidade de colunas: '))
linhas = int(input('Quantidade de linhas: '))
matriz = list()

def principal():
    for l in range(0,linhas): # inserir a quantidade de listas que serão as
linhas
        matriz.append([])

    for l in range(0,linhas): # inserir a quantidade de números que serão a
s colunas
        for c in range(0,colunas):
            matriz[l].append(0)

    for l in range(0,linhas):
        for c in range(0,colunas):
            print(f'[{matriz[l][c]:^5}]',end='') # Dar print na matriz cria
da
        print()

def insercao():
    for l in range(0,linhas):
        for c in range(0,colunas):
            matriz[l][c] = int(input(f'Digite o valor para a posição [{l}]
[{c}]: '))
```

```
def mostrar():
    for l in range(0, linhas):
        for c in range(0, colunas):
            print(f'[{matriz[l][c]:^5}]', end='')  # Dar print na matriz cr
iada
        print()

principal()
print('-='*30)
insercao()
print('-='*30)
mostrar()
```

Guanabara)

Na solução do Guanabara o mesmo introduziu a utilização de matrizes, basicamente são listas com linhas e colunas e podemos manipular igual qualquer lista, o que muda na verdade é a forma como ela é mostrada.

Como funciona:

- Declaramos uma lista composta por 3 listas em que cada uma possui 3 valores, podemos considerar cada lista como as linhas de 0 a 2 e cada item como as colunas de 0 a 2 também.
- Damos 2 tipos de fors onde um percorre a macro e a outra percorre a micro, e a cada iteração damos vários inputs para as posições específicas de cada for, eles ditam essas posições
- No final printamos um separador

Para mostrar a matriz:

- Utilizamos o mesmo mecanismo para printar, ele também vai ditar a posição que queremos e para separar em cada linha bonitinho, basta darmos um print no final do for mais interno antes dele pular para o próximo número do indice macro da lista

```
matriz = [0,0,0],[0,0,0],[0,0,0]
for l in range(0,3):
    for c in range(0,3):
        matriz[l][c] = int(input(f'Digite um valor para a posição [{l},
{c}]: '))
print('-='*30)
for l in range(0,3):
    for c in range(0,3):
        print(f'[{matriz[l][c]:^5}]',end='')
    print()
```

4) Aprimore o desafio anterior, mostrando no final:

- A soma de todos os valores pares digitados.

- A soma dos valores da terceira coluna
- O maior valor da segunda coluna

a) Neste modelo na verdade coloquei a soma de todos os valores gerais e não somente dos pares

```
linhas = int(input('Quantidade de linhas: '))
colunas = int(input('Quantidade de colunas: '))
matriz = list()
sum = 0
somat = 0
maior = 0

def principal():
    for l in range(0,linhas): # inserir a quantidade de listas que serão as
linhas
        matriz.append([])

    for l in range(0,linhas): # inserir a quantidade de números que serão a
s colunas
        for c in range(0,colunas):
            matriz[l].append(0)

    for l in range(0,linhas):
        for c in range(0,colunas):
            print(f'[{matriz[l][c]:^5}]',end='') # Dar print na matriz cria
da
        print()

def insercao():
    for l in range(0,linhas):
        for c in range(0,colunas):
            matriz[l][c] = int(input(f'Digite o valor para a posição [{l}]
[{c}]: '))

def mostrar():
    for l in range(0, linhas):
        for c in range(0, colunas):
            print(f'[{matriz[l][c]:^5}]', end='')  # Dar print na matriz cr
iada
        print()

def somatotal():
    global sum
    for l in range(linhas):
        for c in range(colunas):
            sum += matriz[l][c]
    return sum
```

```
def somater():
    global somat
    for l in range(linhas):
        for c in range(colunas):
            if c == 2:
                somat += matriz[l][c]
    return somat

def maior2c ():
    global maior
    for l in range(linhas):
        for c in range(colunas):
            if c == 1:
                if matriz[l][c] > maior:
                    maior = matriz[l][c]
    return maior

principal()
print('-='*30)
insercao()
print('-='*30)
mostrar()
print('-='*30)
valor = somatotal()
tsoma = somater()
maior = maior2c()
print(f'A soma de todos os valores é de {valor}')
print(f'A soma da terceira coluna é de {tsoma}')
print(f'O maior número da segunda coluna é de {maior}')
```

b) Já aqui inseri apenas a soma de todos os números Pares

```
linhas = int(input('Quantidade de linhas: '))
colunas = int(input('Quantidade de colunas: '))
matriz = list()
sum = 0
somat = 0
maior = 0

def principal():
    for l in range(0,linhas): # inserir a quantidade de listas que serão as
linhas
        matriz.append([])

    for l in range(0,linhas): # inserir a quantidade de números que serão a
s colunas
```

```
        for c in range(0,colunas):
            matriz[l].append(0)

    for l in range(0,linhas):
        for c in range(0,colunas):
            print(f'[{matriz[l][c]:^5}]',end='') # Dar print na matriz cria
da
        print()

def insercao():
    for l in range(0,linhas):
        for c in range(0,colunas):
            matriz[l][c] = int(input(f'Digite o valor para a posição [{l}]
[{c}]: '))

def mostrar():
    for l in range(0, linhas):
        for c in range(0, colunas):
            print(f'[{matriz[l][c]:^5}]', end='')  # Dar print na matriz cr
iada
        print()

def somatotal():
    global sum
    for l in range(linhas):
        for c in range(colunas):
            if matriz[l][c] % 2 == 0:
                sum += matriz[l][c]
    return sum

def somater():
    global somat
    for l in range(linhas):
        for c in range(colunas):
            if c == 2:
                somat += matriz[l][c]
    return somat

def maior2c ():
    global maior
    for l in range(linhas):
        for c in range(colunas):
            if c == 1:
                if matriz[l][c] > maior:
                    maior = matriz[l][c]
    return maior

principal()
```

```
print('-='*30)
insercao()
print('-='*30)
mostrar()
print('-='*30)
valor = somatotal()
tsoma = somater()
maior = maior2c()
print(f'A soma de todos os valores pares é de {valor}')
print(f'A soma da terceira coluna é de {tsoma}')
print(f'O maior número da segunda coluna é de {maior}')
```

Guanabara)

No caso dele foi mais simples pois ele já tinha uma matriz pré definida, mas o que me chamou atenção foi a forma que ele varreu as variáveis, não precisando de colocar mais de um for, então eu posso reduzir também o meu código anterior

```
matriz = [0,0,0],[0,0,0],[0,0,0]
spar = mai = scol = 0
for l in range(0,3):
    for c in range(0,3):
        matriz[l][c] = int(input(f'Digite um valor para a posição [{l},
{c}]: '))
print('-='*30)
for l in range(0,3):
    for c in range(0,3):
        print(f'[{matriz[l][c]:^5}]',end='')
        if matriz[l][c] % 2 == 0:
            spar += matriz[l][c]
    print()
print('=-'*30)
print(f'A soma de todos os valores pares é = {spar}')
for l in range (0,3):
    scol += matriz[l][2]
print(f'A soma dos valores da terceira coluna é {scol}')
for c in range(0,3):
    if c == 0:
        mai = matriz[1][c]
    elif matriz[1][c] > mai:
        mai = matriz[1][c]
print('-='*30)
```

Meu código resumido)

O que muda é a não necessidade de passar outro for dentro de outro se já temos o número da coluna a ser procurada, isso reduz um pouco o tamanho do script.

```
linhas = int(input('Quantidade de linhas: '))
colunas = int(input('Quantidade de colunas: '))
matriz = list()
sum = 0
somat = 0
maior = 0

def principal():
    for l in range(0,linhas): # inserir a quantidade de listas que serão as
linhas
        matriz.append([])

    for l in range(0,linhas): # inserir a quantidade de números que serão a
s colunas
        for c in range(0,colunas):
            matriz[l].append(0)

    for l in range(0,linhas):
        for c in range(0,colunas):
            print(f'[{matriz[l][c]:^5}]',end='') # Dar print na matriz cria
da
        print()

def insercao():
    for l in range(0,linhas):
        for c in range(0,colunas):
            matriz[l][c] = int(input(f'Digite o valor para a posição [{l}]
[{c}]: '))

def mostrar():
    for l in range(0, linhas):
        for c in range(0, colunas):
            print(f'[{matriz[l][c]:^5}]', end='')  # Dar print na matriz cr
iada
        print()

def somatotal():
    global sum
    for l in range(linhas):
        for c in range(colunas):
            if matriz[l][c] % 2 == 0:
                sum += matriz[l][c]
    return sum

def somater():
    global somat
    for l in range(linhas):
```

```
        somat += matriz[l][2]
    return somat

def maior2c ():
    global maior
    for l in range(linhas):
        if matriz[l][1] > maior:
            maior = matriz[l][1]
    return maior

principal()
print('-='*30)
insercao()
print('-='*30)
mostrar()
print('-='*30)
valor = somatotal()
tsoma = somater()
maior = maior2c()
print(f'A soma de todos os valores pares é de {valor}')
print(f'A soma da terceira coluna é de {tsoma}')
print(f'O maior número da segunda coluna é de {maior}')
```

5) Faça um programa que ajude um jogador da mega sena a criar palpites. O programa vai perguntar quantos jogos serão gerados e vai sortear 6 números entre 1 e 60 para cada jogo, cadastrando tudo em uma lista composta. Lembrando que os números não podem se repetir.

a) Existem algumas formas de resolver, mas eu vou fazer de tal forma que números possam se repetir mas não no mesmo jogo.

Então no meu código eu estou basicamente:

- Criando uma lista onde vai receber outras listas para ficar composta, cada uma será um jogo diferente
- Na estrutura de repetição nós vamos fazer a quantidade que for especificada pelo input de n
- criamos uma lista onde em cada laço ele automaticamente vai estar limpa, não tendo necessidade de fazer o clear
- Após isso temos outra estrutura dentro dela, num range de 6, ou seja vai de 0 a 5
- Criado uma variável temporária que recebe um número aleatório de 0 a 60
- Se temp não estiver na lista jogo, então jogo vai receber temp
- No final do loop de 6 a lista jogo vai pra lista jogos
- Finalizando vamos percorrer a lista composta mostrando a composição de cada uma

Observação:

Neste código podemos ter números iguais mas em jogos diferentes, existem outras formas de mudarmos isso.

```
 from random import randint
from time import sleep
print('='*40)
print('{:^40}'.format('JOGO MEGA SENA'))
print('='*40)
jogos = list()
n = int(input('Quantos jogos deseja sortear?: '))
for j in range(n):
    jogo = []
    for c in range(6):
        temp = randint(0,60)
        if temp not in jogo:
            jogo.append(temp)
    jogos.append(jogo[:])
print('='*40)
print('Sorteando....')
print('='*40)
for i, jog in enumerate(jogos):
    sleep(0.5)
    print('Jogo nº {:<3} = {}'.format(i+1,jog))
```

b) Vou tentar fazer a mesma coisa, mas agora nenhum jogo pode ter números diferentes.

- Diferente da solução anterior eu criei uma lista que salva todos os números que o randint gera para testar se o número já está na lista temporária e se ele tá na lista de todos os números
- Fora isso temos um problema, pois se não dermos condições e se os 60 números forem sorteados, o pc não entende o que é para fazer e fica num loop infinito então fizemos uma condição sendo, se o len de todos os números for 60 então ele mostra uma mensagem e dá um break saindo do while.

```
from random import randint
from time import sleep
print('='*40)
print('{:^40}'.format('JOGO MEGA SENA'))
print('='*40)
jogos = list()
todos_numeros = list()
n = int(input('De 1 a 10, quantos jogos deseja sortear?: '))
for j in range(n):
    jogo = []
    while len(jogo) < 6:
        temp = randint(0,60)
        if temp not in jogo and temp not in todos_numeros:
```

```
            jogo.append(temp)
            todos_numeros.append(temp)
    jogos.append(jogo[:])
    if len(todos_numeros) == 60:
        print('Impossível realizar {} combinações, mostrando a quantidade m
áxima'.format(n))
        break
print('='*40)
print('Sorteando....')
print('='*40)
for i, jog in enumerate(jogos):
    sleep(0.5)
    print('Jogo nº {:<3} = {}'.format(i+1,jog))
```

Guanabara)

Esta foi a forma como o professor resolveu e vou estar detalhando passo a passo para tentar resolver de outra forma.

Bibliotecas utilizadas:

- Random: importou o randint para gerar um número aleatório
- Time: importou o sleep para dar um delay no momento que mostra os resultados na tela
- Declarado duas listas, uma para ser a temporária chamada lista () e outra para a principal onde vai guardar todos os jogos chamada jogo ()
- Deu uns prints apenas para ficar um pouco mais bonito
- Input da quantidade de jogos a serem sorteados
- Declaração da variável tot sendo o contador macro do código, ele vai estabelecer o limite junto com o quant da quantidade de listas
- Enquanto tot for menor ou igual a quant aí ele entra no bloco de código
- declaração de um contador micro valendo 0 no início
- Enquanto verdadeiro (loop infinito)
- Num recebe randint de 1 a 60
- se o num não estiver na lista temporária, então damos um append de num
- contador recebe mais 1
- se o contador for maior ou igual a 6, ele para o código (aqui ditamos a quantidade de números na lista)
- Organiza os dados da lista com um sort
- variável maior recebe todos os valores de lista temporária
- Apagamos o conteúdo da lista temporária
- tot recebe mais 1

Para finalizar:

- damos o print informando quantos jogos estamos sorteando
- com o for em enumerate, mostramos o número do jogo e quais os números

```
from random import randint
from time import sleep
lista = list()
jogos = list()
print('-'*50)
print(f'{'JOGOS DA MEGA SENA':^50}')
print('-'*50)
quant = int(input('Quantos jogos você quer que eu sorteie?: '))
tot = 1
while tot <= quant:
    cont = 0
    while True:
        num = randint(1,60)
        if num not in lista:
            lista.append(num)
            cont+=1
        if cont >= 6:
            break
    lista.sort()
    jogos.append(lista[:])
    lista.clear()
    tot+=1
print('-='*3, f'SORTEANDO {quant} JOGOS...','-='*3)
for i, c in enumerate(jogos):
    print(f'Jogo {i+1}: {jogos[i]}')
    sleep(0.5)
print('-='*3,'< BOA SORTE >','-='*3)
```

6) Crie um programa que leia nome e duas notas de vários alunos e guarde tudo em uma lista composta. No final, mostre um boletim contendo a média de cada um e permita que o usuário possa mostrar as notas de cada aluno individualmente.

a) Esta foi a forma como eu pensei em resolver, vou listar o passo a passo:

- Criei duas listas, uma para salvar todos os alunos com as notas, sendo o identificado para cada aluno o próprio índice
- As médias serão o seu nome e a média já calculada
- Vamos para o primeiro loop True:
    - nome, n1, n2 vão receber valores digitados pelo usuário

    - média será calculada a cada loop
    - alunos recebe em lista as duas notas
    - medias recebe em lista o nome e a média calculada
    - Ficará sempre no final do loop a pergunta se quer continuar ou não e o break funciona dependendo da resposta do usuário
- Saindo do loop vamos para a parte principal onde é feita uma listagem com todos os alunos e suas médias
- Entramos no segundo loop True:
    - o valor da escolha de qual aluno mostrar dependerá do seu índice mostrado na tela
    - varremos a lista alunos e verificamos o seguinte:
        - se o indice for igual à escolha, então ele printa o nome do aluno e a média
        - se o valor digitado for maior que o tamanho da lista então ele mostrará que o aluno não existe e dá o break para escolher novamente
- Se esc for 999 ele sai do programa e finaliza

```
alunos = list()
medias = list()
while True: # primeiro laço para receber as informações
    nome = input('Nome: ')
    n1 = float(input('Nota 1: ')) # Dados iniciais digitados
    n2 = float(input('Nota 2: '))
    media = (n1+n2)/2
    alunos.append([n1,n2])
    medias.append([nome,media])
    resp = input('Quer continuar [S/N]?: ')
    while resp not in 'SsNn' or resp == '':
        resp = input('Valor inválido ! Quer continuar [S/N]:? ') # Validaçã
o para continuar ou não
    if resp in 'Nn':
        break
print('-='*40)
print('{:<5}{:<20}{:>4}'.format('Nº','NOME','MÉDIA'))
print('-'*40)
for i,c in enumerate(medias):
    print('{:<5}{:<20}{:<4}'.format(i,medias[i][0],medias[i][1]))
print('-'*40)
while True:
    esc = int(input('Mostrar notas de qual aluno? (999 interrompe): '))
    for i, p in enumerate(alunos):
        if i == esc:
            print('Aluno {} teve as notas: {}'.format(medias[i][0],p))
        if esc > len(alunos)-1:
            print('Aluno não existe! Tente novamente...')
            break
```

```
    if esc == 999:
        break
print('Programa Finalizado!')
```

Guanabara)

O professor na verdade fez de uma forma mais simples que eu não pensei na época da minha solução, basicamente estruturou o molde de cada ficha composta, ficando mais fácil para salvar e buscar os dados, no meu caso eu separei em duas fichas compostas, mas vamos ver como ele resolveu.

- Criado uma lista chamada ficha
- Entramos no primeiro loop True:
  - criado variáveis para usuário preencher chamadas nome, nota1, nota2
  - média é calculada de forma automática
  - lista ficha recebe uma lista com a seguinte estrutura [nome, [nota1,nota2],media] então podemos criar uma lista dentro de outra lista e enviar para outra lista, e as próximas seguem o mesmo padrão, sendo mais fácil para a consulta de informação
  - estrutura de pergunta se o usuário deseja continuar ou não, dando break se ele escolher n
- Print para mostrar nº nome e media na tela
- abaixo varremos a lista ficha mostrando o índice, nome e média.
- Entramos no segundo loop True:
- perguntamos ao usuário qual aluno o mesmo quer mostrar se digitar 999 o loop finaliza e é mostrado “volte sempre”
- se a opção for menor que o len menos 1 (que é o máximo do range da lista), ele mostra a ficha[opc] que é qual lista interna e a casa 0 que é o nome, depois a posição 1 que são a lista dentro dela sendo as notas individuais

```
ficha = list()
while True:
    nome = input('Nome: ')
    nota1 = float(input('Nota 1: '))
    nota2 = float(input('Nota 2: '))
    media = (nota1+nota2)/2
    ficha.append([nome,[nota1,nota2],media ])
    resp = input('Quer continuar [S/N]: ')
    if resp in 'Nn':
        break
print('-='*30)
print(f'{'Nº':<4}{'NOME':<10}{'MEDIA':>8}')
print('-'*26)
for i, a in enumerate(ficha):
```

```
    print(f'{i:<4}{a[0]:<10}{a[2]:>8.1f}')
while True:
    print('-'*35)
    opc = int(input('Mostrar notas de qual aluno ? (999 interrompe): '))
    if opc === 999:
        print('Finalizando...')
        break
    if opc <= len(ficha)-1:
        print(f'Notas de {ficha[opc][o]} são {ficha[opc][1]}')
print('<<<< Volte Sempre >>>>')
```