#1.Faça um programa que leia um número N e gereum arquivo com N nomes e idades aleatórios Faça uso de duas listas criadas na mão: uma que contenha 20 nomes e outra que contenha 20 sobrenomes Cada linha do arquivo resultante deve conter um nome completo e a sua idade

```
def gerar_nomes_aleatorios(qnt_de_numeros, arquivos_com_nomes, arquivos_com_sobrenomes, arquivo_com_idades):
    nomes = open(arquivos_com_nomes, 'r')
    sobrenomes = open(arquivos_com_sobrenomes, 'r')
    idades = open(arquivo_com_idades, 'r')
    for i in range(qnt_de_numeros):
        nome = str(nomes.readline())
        sobrenome = str(sobrenomes.readline())
        idade = str(idades.readline())
        print(f"NOME: {nome}")
        print(f"SOBRENOME:{sobrenome}")
        print(f"IDADE: {idade}")
    nomes.close()
    sobrenomes.close()
    idades.close()
```

#2.Estenda o exemplo do cadastro para considerar também a altura da pessoa

```
def gerar_nomes_aleatorios(qnt_de_numeros, arquivos_com_nomes, arquivos_com_sobrenomes, arquivo_com_idades, arquivo_com_altura):
    nomes = open(arquivos_com_nomes, 'r')
    sobrenomes = open(arquivos_com_sobrenomes, 'r')
    idades = open(arquivo_com_idades, 'r')
    altura = open(arquivo_com_altura, 'r')
    for i in range(qnt_de_numeros):
        nome = str(nomes.readline())
        sobrenome = str(sobrenomes.readline())
        idade = str(idades.readline())
        altura = str(altura.readline())
        print(f"NOME: {nome}")
        print(f"SOBRENOME:{sobrenome}")
        print(f"IDADE: {idade}")
        print(f"ALTURA: {altura}")
    nomes.close()
    sobrenomes.close()
    idades.close()
    altura.close()
```

#3.Escreva uma função que recebe dois nomes de arquivos e copia o conteúdo do primeiro arquivo para o segundo arquivo.Considere que o conteúdo do arquivo de origem é um texto.Sua função não deve copiar linhas comentadas (que começam com //)

```
def copiar_arquivo_sem_comentarios(origem, destino):
    try:
        with open(origem, 'r') as arquivo_origem, open(destino, 'w') as arquivo_destino:
            for linha in arquivo_origem:
                if not linha.strip().startswith('//'):
                    arquivo_destino.write(linha)
        print(f'Conteúdo do arquivo copiado de {origem} para {destino} (comentários removidos).')
    except FileNotFoundError:
        print(f'Arquivo {origem} não encontrado.')
    except Exception as e:
        print(f'Ocorreu um erro: {str(e)}')

origem = 'arquivo_origem.txt'
```

```
destino = 'arquivo_destino.txt'
copiar_arquivo_sem_comentarios(origem, destino)
```

#4. Faça um programa contendo uma função que recebe comoargumentos os nomes de dois arquivos. O primeiro arquivo contém nomes de alunos e o segundo arquivo contém as notas dos alunos. No primeiro arquivo, cada linha corresponde ao nome de um aluno e no segundo arquivo, cada linha corresponde às notas dos alunos (uma ou mais). Assuma que as notas foram armazenadas como strings, e estão separadas umas das outras por espaços em branco. Leia os dois arquivos e gere um terceiro arquivo que contém o nome do aluno seguido da média de suas notas.

```
def calcular_media(notas):
    notas = notas.split()
    total = 0
    contador = 0
    for nota in notas:
        total += float(nota)
        contador += 1
    media = total / contador
    return media

def gerar_relatorio(arquivo_alunos, arquivo_notas, arquivo_saida):
    with open(arquivo_alunos, 'r') as f_alunos, open(arquivo_notas, 'r') as f_notas, open(arquivo_saida,
'w') as f_saida:
        for nome_aluno, notas_aluno in zip(f_alunos, f_notas):
            nome_aluno = nome_aluno.strip()
            notas_aluno = notas_aluno.strip()
            media = calcular_media(notas_aluno)
            f_saida.write(f"{nome_aluno}: {media:.2f}\n")

arquivo_alunos = "nomes_alunos.txt"
arquivo_notas = "notas_alunos.txt"
arquivo_saida = "relatorio_notas.txt"
gerar_relatorio(arquivo_alunos, arquivo_notas, arquivo_saida)
```

#5.Faça um programa para alterar uma das notas de um aluno (usando os arquivos do exercício anterior). O programa deve ter uma função que recebe o nome do aluno, a nota velha e a nova nota. A função deve fazer a alteração no arquivo.

```
def alterar_nota(nome_aluno, nota_antiga, nota_nova, arquivo_notas):
    with open(arquivo_notas, 'r') as f_notas:
        linhas = f_notas.readlines()

    with open(arquivo_notas, 'w') as f_notas:
        for linha in linhas:
            partes = linha.split()
            if partes[0] == nome_aluno:
                for i in range(1, len(partes)):
                    if partes[i] == nota_antiga:
                        partes[i] = nota_nova
                nova_linha = partes[0] + ' ' + ' '.join(partes[1:]) + '\n'
                f_notas.write(nova_linha)
            else:
                f_notas.write(linha)

arquivo_notas = "notas_alunos.txt"

nome_aluno = "João"
nota_antiga = "7.5"
```

```
nota_nova = "8.0"

alterar_nota(nome_aluno, nota_antiga, nota_nova, arquivo_notas)

#6.Faça uma função que leia um arquivo texto contendo uma lista de endereços IP e gere dois outros arquivos, um contendo os endereços IP válidos e outro contendo os endereços inválidos. O formato de um endereço IP é num1.num.num.num, onde num1 vai de 1 a 255 e num vai de 0 a 255.
def validar_endereco_ip(ip):
    partes = ip.split('.')
    if len(partes) != 4:
        return False
    for parte in partes:
        if not 0 <= int(parte) <= 255:
            return False
    return 1 <= int(partes[0]) <= 255

def separar_enderecos(ip_file):
    validos = []
    invalidos = []

    with open(ip_file, 'r') as f_in:
        for linha in f_in:
            ip = linha.strip()
            if validar_endereco_ip(ip):
                validos.append(ip)
            else:
                invalidos.append(ip)

    with open('ip_validos.txt', 'w') as f_valid:
        f_valid.write('\n'.join(validos))

    with open('ip_invalidos.txt', 'w') as f_invalid:
        f_invalid.write('\n'.join(invalidos))

arquivo_ip = "lista_ip.txt"
separar_enderecos(arquivo_ip)
```