

Listas – Tuplas – Dicionários – Collections – Conjuntos

Utilizando apenas condicionais

TUPLAS

- 1- **Verificação de Palíndromo:**
Crie um programa que recebe uma tupla de strings e retorna uma nova tupla contendo apenas as palavras que são palíndromos.
- 2- **Ordenação Condicional:**
Implemente um programa que ordena uma tupla de números de forma decrescente, mas mantenha os números pares no início da tupla e os ímpares no final.
- 3- **Pesquisa de Dados:**
Desenvolva um programa que recebe uma tupla de dicionários representando dados de estudantes (nome, idade, média) e retorna uma lista com os nomes dos estudantes que têm uma média acima de 8.0.
- 4- **Máximo e Mínimo Condicionais:**
Crie um programa que, dada uma tupla de números, encontre o maior e o menor elemento, mas ignore os números negativos.
- 5- **Validação de Senhas:**
Implemente um programa que verifica se uma tupla de strings contém senhas válidas. Considere como válidas as senhas que têm pelo menos 8 caracteres, incluindo pelo menos uma letra maiúscula, uma letra minúscula, e um número.
- 6- **Agrupamento Condicional:**
Desenvolva um programa que recebe uma tupla de números inteiros e os agrupa em sublistas consecutivas, com base na seguinte condição: dois números são do mesmo grupo se a diferença entre eles for menor ou igual a 2.
- 7- **Soma Condicional:**
Crie um programa que, dada uma tupla de números, calcule a soma apenas dos números positivos que estão em posições ímpares.
- 8- **Análise de Sequência:**
Implemente um programa que recebe uma tupla de números e verifica se ela forma uma sequência aritmética, geométrica ou nenhuma das duas. Considere sequências com pelo menos três elementos.
- 9- **Divisão de Números:**
Desenvolva um programa que, dada uma tupla de números, divide os números em dois grupos: os que são divisíveis por 3 e os que não são
- 10- **Filtragem de Strings:**
Crie um programa que recebe uma tupla de strings e retorna uma nova tupla contendo apenas as strings que começam e terminam com a mesma letra.
- 11- **Validação de Data:**
Implemente um programa que recebe uma tupla de datas (ano, mês, dia) e verifica se cada data é válida.
- 12- **Substituição Condicional:**
Desenvolva um programa que recebe uma tupla de strings e substitui todas as ocorrências da palavra "Python" por "Java".

- 13- **Máximo Comum Divisor:**
Crie um programa que, dada uma tupla de números, encontre o máximo comum divisor (MCD) de todos os números.
- 14- **Pesquisa de Palavras:**
Implemente um programa que recebe uma tupla de strings e uma palavra-chave. O programa deve retornar uma lista com as strings que contêm a palavra-chave.
- 15- **Filtragem de Dados:**
Desenvolva um programa que recebe uma tupla de dicionários representando informações de produtos (nome, preço, quantidade) e retorna uma nova lista apenas com os produtos que têm um preço superior a um valor especificado.

DICIONARIOS

- 1- **Validação de Senhas:**
Crie um programa que solicite ao usuário que insira uma senha. O programa deve verificar se a senha atende aos seguintes critérios: pelo menos 8 caracteres, incluindo pelo menos uma letra maiúscula, uma letra minúscula e um número.
- 2- **Verificação de Chave em Dicionário:**
Dado um dicionário que representa informações sobre um estudante (nome, idade, média), crie um programa que verifica se a chave "média" está presente no dicionário.
- 3- **Contagem de Palavras:**
Implemente um programa que recebe uma string e conta a frequência de cada palavra. O resultado deve ser armazenado em um dicionário.
- 4- **Ordenação de Dicionário:**
Crie um programa que recebe um dicionário de alunos (nome, nota) e retorna uma lista de nomes dos alunos ordenados por nota, do maior para o menor.
- 5- **Validação de Data:**
Desenvolva um programa que recebe uma data (dia, mês, ano) como um dicionário e verifica se é uma data válida.
- 6- **Filtragem de Produtos:**
Dado um dicionário de produtos (nome, preço, quantidade), crie um programa que retorna uma lista apenas com os produtos que têm uma quantidade superior a um valor especificado.
- 7- **Jogo de Adivinhação:**
Implemente um jogo simples de adivinhação onde o programa escolhe um número aleatório e o usuário tenta adivinhar. O programa deve fornecer dicas (maior ou menor) e contar o número de tentativas.
- 8- **Validação de E-mail:**
Crie um programa que solicita ao usuário que insira um endereço de e-mail. O programa deve verificar se o e-mail é válido com base em critérios simples (por exemplo, presença de "@" e ".").
- 9- **Verificação de Anagrama:**
Dadas duas strings, crie um programa que verifica se são anagramas. Use dicionários para contar a frequência de cada caractere em ambas as strings.
- 10- **Conversor de Moeda:**
Desenvolva um programa que simula um conversor de moeda. O usuário fornece a moeda original, a moeda desejada e o valor a ser convertido. O programa utiliza um dicionário com as taxas de câmbio.
- 11- **Contagem de Caracteres:**
Crie um programa que conta a frequência de cada caractere em uma string. O resultado deve ser armazenado em um dicionário.
- 12- **Pesquisa de Palavras-chave:**
Implemente um programa que recebe um texto e uma lista de palavras-chave. O programa deve contar quantas vezes cada palavra-chave aparece no texto.
- 13- **Calculadora de Notas:**
Dado um dicionário de notas (nome, nota), crie um programa que calcula a média das notas e atribui uma categoria (A, B, C, D) com base na média.

- 14- **Verificação de Números Primos:**
Desenvolva um programa que recebe um número e verifica se é primo. Utilize um dicionário para armazenar os resultados anteriores e evitar cálculos repetidos.
- 15 **Acesso a Dados:**
Crie um programa que simula o acesso a um sistema com autenticação. Utilize um dicionário para armazenar os dados de usuário e senha.
- 16- **Filtragem de Dados de Estudantes:**
Dado um dicionário de estudantes (nome, idade, curso), crie um programa que retorna uma lista apenas com os estudantes que estão cursando um curso específico.
- 17- **Cálculo de Desconto:**
Implemente um programa que recebe um dicionário representando um produto (nome, preço) e calcula o preço com desconto com base em uma taxa fornecida pelo usuário.
- 18- **Verificação de Subsequência:**
Crie um programa que recebe duas strings e verifica se a primeira é uma subsequência da segunda. Utilize dicionários para armazenar informações sobre as posições dos caracteres.
- 19- **Análise de Texto:**
Desenvolva um programa que recebe um texto e conta a quantidade de palavras únicas. Utilize um dicionário para armazenar as palavras e suas frequências.
- 20- **Controle de Estoque:**
Crie um programa que simula um sistema de controle de estoque. Utilize um dicionário para armazenar os produtos e suas quantidades disponíveis. O programa deve permitir a adição e remoção de produtos do estoque.

LISTAS

- 1- **Média dos Elementos Pares:**
Crie um programa que calcula a média dos elementos pares em uma lista de números.
- 2- **Ordenação Condicional:**
Implemente um programa que ordena uma lista de números de forma crescente, mas mantenha os números ímpares no início da lista e os pares no final.
- 3- **Substituição Condicional:**
Dada uma lista de strings, crie um programa que substitui todas as ocorrências da palavra "Python" por "Java".
- 4- **Filtragem de Números Negativos:**
Desenvolva um programa que recebe uma lista de números e retorna uma nova lista apenas com os números não negativos.
- 5- **Contagem de Elementos:**
Crie um programa que conta quantas vezes um elemento específico aparece em uma lista.
- 6- **Soma Condicional:**
Implemente um programa que calcula a soma dos quadrados dos números ímpares em uma lista.
- 7- **Filtragem de Strings:**
Dada uma lista de strings, crie um programa que retorna uma nova lista apenas com as strings que têm mais de 5 caracteres.
- 8- **Validação de Senhas:**
Desenvolva um programa que solicita ao usuário que insira uma senha. O programa deve verificar se a senha atende a critérios simples, como ter pelo menos 8 caracteres.
- 9- **Cálculo de Desconto:**
Crie um programa que recebe uma lista de preços de produtos e calcula o preço com desconto com base em uma taxa fornecida pelo usuário.
- 10- **Verificação de Anagrama:**
Implemente um programa que verifica se duas strings são anagramas. Utilize listas para contar a frequência de cada caractere em ambas as strings.
- 11- **Filtragem de Elementos Únicos:**
Dada uma lista, crie um programa que retorna uma nova lista apenas com os elementos que não se repetem.
- 12- **Conversor de Moeda:**
Desenvolva um programa que simula um conversor de moeda. O usuário fornece a moeda original, a moeda desejada e o valor a ser convertido. O programa utiliza uma lista com as taxas de câmbio.
- 13- **Validação de E-mail:**
Crie um programa que solicita ao usuário que insira um endereço de e-mail. O programa deve verificar se o e-mail é válido com base em critérios simples (por exemplo, presença de "@" e ".").
- 14- **Pesquisa de Palavras:**
Implemente um programa que recebe uma lista de palavras e uma palavra-chave. O programa deve contar quantas vezes a palavra-chave aparece na lista.
- 15- **Maior Subsequência Crescente:**
Desenvolva um programa que encontra a maior subsequência crescente em uma lista de números.

- 16- **Controle de Estoque:**
Crie um programa que simula um sistema de controle de estoque. Utilize uma lista para armazenar os produtos e suas quantidades disponíveis. O programa deve permitir a adição e remoção de produtos do estoque.
- 17- **Classificação de Idades:**
Implemente um programa que recebe uma lista de idades e classifica cada idade em uma das categorias: criança, adolescente, adulto ou idoso.
- 18- **Soma Acumulada:**
Crie um programa que gera uma lista com a soma acumulada dos elementos de outra lista.
- 19- **Verificação de Sequência Aritmética:**
Desenvolva um programa que verifica se uma lista de números forma uma sequência aritmética.
- 20- **Maior Produto Três a Três:**
Implemente um programa que, dada uma lista de números, encontra o maior produto que pode ser obtido multiplicando três números diferentes entre si.

CONJUNTOS

- 1- **Verificação de Elementos Comuns:**
Crie dois conjuntos e verifique se eles têm algum elemento em comum. Se tiverem, imprima "Conjuntos têm elementos em comum", caso contrário, imprima "Conjuntos não têm elementos em comum".
- 2- **Remoção de Duplicatas:**
Implemente um programa que recebe uma lista de números e utiliza um conjunto para remover as duplicatas, imprimindo a lista resultante.
- 3- **Maior Elemento Comum:**
Dados dois conjuntos, encontre o maior elemento que eles têm em comum. Se não houver elementos em comum, imprima "Nenhum elemento em comum".
- 4- **Soma de Elementos Únicos:**
Crie dois conjuntos e calcule a soma dos elementos únicos presentes em ambos os conjuntos
- 5- **Verificação de Subconjunto:**
Desenvolva um programa que recebe dois conjuntos e verifica se um é subconjunto do outro.
- 6- **União Condicional:**
Crie dois conjuntos e, se a união deles tiver um número par de elementos, imprima a união, caso contrário, imprima a diferença entre eles.
- 7- **Remoção de Elemento Específico:**
Dado um conjunto de caracteres e um caractere específico, remova o caractere do conjunto, se estiver presente.
- 8- **Interseção Não Vazia:**
Implemente um programa que cria dois conjuntos e verifica se a interseção entre eles não é vazia.
- 9- **Criação de Conjunto Condicional:**
Dada uma lista de números, crie um conjunto apenas se a lista contiver números diferentes.
- 10- **Verificação de Conjunto Vazio:**
Crie um conjunto e verifique se está vazio. Se estiver, adicione alguns elementos e imprima o conjunto.
- 11- **Diferença Simétrica:**
Desenvolva um programa que cria dois conjuntos e imprime a diferença simétrica entre eles.
- 12- **Divisibilidade por 3 ou 5:**
Crie um conjunto de números inteiros e imprima os números que são divisíveis por 3 ou 5.
- 13- **Verificação de Igualdade de Conjuntos:**
Crie dois conjuntos e verifique se são iguais. Se forem, imprima "Conjuntos são iguais", caso contrário, imprima "Conjuntos não são iguais".
- 14- **Contagem de Elementos:**
Dada uma lista de strings, conte quantas palavras únicas estão presentes. Utilize conjuntos para realizar a contagem.
- 15- **Verificação de Conjuntos Disjuntos:**
Crie dois conjuntos e verifique se são conjuntos disjuntos. Se forem, imprima "Conjuntos são disjuntos", caso contrário, imprima "Conjuntos não são disjuntos".

COLLECTIONS

- 1- **Contagem de Elementos Únicos:**
Crie um programa que recebe uma lista de números e utiliza um Counter para contar a ocorrência de cada elemento. Imprima apenas os elementos que aparecem mais de uma vez.
- 2- **Ordenação de Lista de Tuplas:**
Dada uma lista de tuplas (nome, idade), ordene-a com base nas idades usando sorted e imprima a lista resultante.
- 3- **Remoção de Elementos Duplicados:**
Implemente um programa que recebe uma lista e cria um conjunto a partir dela, removendo assim os elementos duplicados. Em seguida, imprima a lista resultante.
- 4- **Filtragem de Dados em Dicionário:**
Crie um dicionário com informações sobre produtos (nome, preço) e remova os produtos que têm preço menor que um valor especificado.
- 5- **Maior Elemento em Lista de Números:**
Desenvolva um programa que, dada uma lista de números, utilize a função max para encontrar o maior elemento. Se a lista estiver vazia, imprima uma mensagem indicando isso.
- 6- **Verificação de Elementos em Tupla Nomeada:**
Crie uma NamedTuple para representar informações de um livro (título, autor, ano) e, dado um título, verifique se ele está presente na lista de livros.
- 7- **Criação de Dicionário Condicional:**
Dada uma lista de palavras, crie um dicionário que conte a frequência de cada palavra, mas apenas para palavras com mais de 3 caracteres.
- 8- **Filtragem de Elementos em Conjunto:**
Desenvolva um programa que cria dois conjuntos e imprime os elementos que estão presentes em ambos os conjuntos.
- 9- **Verificação de Maior Elemento em Deque:**
Utilize um deque para criar uma fila de números e imprima o maior elemento na fila. Se a fila estiver vazia, imprima uma mensagem indicando isso.
- 10- **Criação de Dicionário a partir de Listas:**
Crie duas listas, uma com chaves e outra com valores, e utilize zip para criar um dicionário. Imprima o dicionário resultante.
- 11- **Contagem de Elementos Únicos em Conjunto:**
Crie um conjunto de caracteres e, utilizando um defaultdict(int), conte a ocorrência de cada caractere. Imprima apenas os caracteres que aparecem mais de uma vez.
- 12- **União de Dicionários:**
Crie dois dicionários e utilize ChainMap para uni-los. Se uma chave estiver presente em ambos os dicionários, mantenha o valor do primeiro.
- 13- **Filtragem de Elementos em DefaultDict:**
Crie um defaultdict(int) e utilize uma lista de números para incrementar a contagem de cada número no dicionário. Imprima apenas os números que aparecem mais de duas vezes.
- 14- **Verificação de Elemento em Heap:**
Utilize heapq para criar um heap a partir de uma lista de números. Imprima se um determinado número está presente no heap.

- 15- **Criação de Dicionário a partir de Lista de Tuplas:**
Dada uma lista de tuplas (chave, valor), crie um dicionário utilizando um `defaultdict(list)`. Agrupe os valores para cada chave e imprima o dicionário resultante.
- 16- **Ordenação de Dicionário:**
Crie um dicionário com informações de alunos (nome, média) e ordene-o pelo valor da média em ordem decrescente. Imprima o dicionário ordenado.
- 17- **Maior Elemento Comum em Dois Conjuntos:**
Desenvolva um programa que cria dois conjuntos e encontra o maior elemento que ambos têm em comum.
- 18- **Verificação de Elementos em Contador:**
Crie um programa que utiliza Counter para contar a frequência de cada palavra em uma string. Imprima apenas as palavras que aparecem mais de três vezes.
- 19- **Remoção de Elementos em Dicionário:**
Crie um dicionário com informações de produtos (nome, quantidade) e remova os produtos que têm quantidade igual a zero.
- 20- **Filtragem de Dados em DefaultDict:**
Dada uma lista de números, crie um `defaultdict(list)` que agrupe os números por paridade (ímpar ou par). Imprima apenas as listas que contêm mais de três elementos.