Nome(s): Rodrigo da Silva Alves

**As respostas devem ser um print do código e da saída em python**

**1) Vamos criar um conjunto:**

A = {1,2,3,4,5,6}

print(A)

**Saída:**

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

**2) Vamos criar um conjunto a partir de uma lista**

lista = [“bananas”, “peras”, “laranjas”, “abacates”]

B = set(lista)

print(B)

**Saída:**

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

**3) Seguindo a mesma lógica do item anterior:**

lista = [“bananas”, “peras”, “laranjas”, “limões”, “bananas”, “bananas”, “abacates”, “laranjas”]

B = set(lista)

print(B)

**Saída:**

**Interface gráfica do usuário, Texto, Aplicativo, Site

Descrição gerada automaticamente**

**Comparando os itens 2 e 3, a que conclusão podemos chegar?**

**Resposta:**

Que uma lista pode conter infinitos valores, mas um set, só pode ter valores únicos não repetidos.

**4) Imprima a cardinalidade do conjunto B obtido no item 3 da forma: “A cardinalidade do conjunto B = { ... } é {tamanho}”**

**Dica: utilize a palavra reservada do python “len”**

**Resposta:**

**Tela de celular com aplicativo aberto

Descrição gerada automaticamente**

**5) Teste as relações de pertinência e imprima a resposta (A = {1,2,3,4,5})**

**Dica: utilize a palavra reservada do python “in”**

a)

b)

c)

Tela de computador com letras e números em fundo preto

Descrição gerada automaticamente

**6) Teste a igualdade entre os conjuntos A= {1,2,3} e B = {3,2,1}, A é igual a B? Imprima o resultado**

**Texto

Descrição gerada automaticamente**

**7) Utilize a função issubset() para testar todos os subconjuntos de C = {2,3,4} – imprima os resultados**

**Texto

Descrição gerada automaticamente**

**Agora, faça o teste utilizando o operador de pertinência em python para o seguinte exemplo:**

Texto

Descrição gerada automaticamente

**Qual resultado é esperado? O python respeita esse resultado?**

O esperado era que fosse False, e sim o python respeita

{1,2} é um subconjunto, e um subconjunto não pode pertencer a um conjunto.

**Faça o teste para o conjunto vazio:**

**Qual resultado é esperado? O python respeita esse resultado?**

**Texto

Descrição gerada automaticamente**

O conjunto vazio é um subconjunto de A. Ele não é um elemento isolado para pertencer a A

**8) Crie uma verificação para testar se A = {1,2,3} é subconjunto próprio de C = {1,2,3,4,5} – imprima o código e resultado. Agora reaproveite o código para testar se D = {5,3,4,2,1} é subconjunto próprio de C.**

**Texto

Descrição gerada automaticamente**

**9) Considerando: A = {1,2,3,4,5} e B = {4,5,6,7,8,9,10} faça a conta (mostrando a simbologia matemática e imprima os resultados em python):  
a)**

**b)**

**c) A – B**

**d) B – A**

**Texto

Descrição gerada automaticamente**

**10) Faça um menu que só encerre quando o usuário solicitar (opção de sair) que seja interativo e com as devidas validações de possíveis erros de entrada do usuário. O objetivo é fazer a operação entre 2 conjuntos, ou seja, crie uma forma de pedir dois conjuntos para o usuário (conjuntos A e B – posteriormente esses conjuntos podem ser alterados pelo usuário). As opções de operações são:**

**a) União**

**b) Intersecção**

**c) Diferença**

**d) Produto cartesiano**

**d) Verificação se A é subconjunto de B (submenu: subconjunto ou subconjunto próprio)**

**e) Mesma verificação do item d, mas de B com A.**

def uniao(conjuntoX, conjuntoY):

    return conjuntoX.union(conjuntoY)

def intersecao(conjuntoX, conjuntoY):

    return conjuntoX.intersection(conjuntoY)

def diferenca(conjuntoX, conjuntoY):

    return conjuntoX.difference(conjuntoY)

def produto\_cartesiano(conjuntoX, conjuntoY):

    return set([(x, y) for x in conjuntoX for y in conjuntoY])

def isSubconjunto(conjuntoX, conjuntoY):

    return conjuntoX.issubset(conjuntoY)

def isSubconjuntoProprio(conjuntoX, conjuntoY):

    if conjuntoX.issubset(conjuntoY) and conjuntoX != conjuntoY:

        return True

    else:

        return False

def isAlpha(entrada\_dados):

    for char in entrada\_dados:

        if char.isalpha():

            return True

def parseStrToConjunto(string):

    valores = string.split(',')

    conjunto = {int(valor) for valor in valores}

    return conjunto

def operacoes(option, conjuntoX, conjuntoY):

    match option:

        case 1:

            resultado = uniao(conjuntoX, conjuntoY)

            resposta = "A união dos conjuntos resultou no conjunto: " + str(resultado)

        case 2:

            resultado = intersecao(conjuntoX, conjuntoY)

            resposta = "A interseção dos conjuntos resultou no conjunto: " + str(resultado)

        case 3:

            resultado = diferenca(conjuntoX, conjuntoY)

            resposta = "A diferença dos conjuntos resultou no conjunto: " + str(resultado)

        case 4:

            resultado = produto\_cartesiano(conjuntoX, conjuntoY)

            resposta = "O Produto Cartesiano dos conjuntos resultou no conjunto: " + str(resultado)

        case 6:

            resultado = isSubconjuntoProprio(conjuntoX, conjuntoY)

            resposta = "O Conjunto A é subconjunto do Conjunto B?: " + str(resultado)

    return resposta

def main():

    while True:

        option = int(input('Digite a opção desejada: '))

        if option == 7:

            print('Finalizando o programa...')

            break

        if option in [1,2,3,4,5,6]:

            stringX = input('Digite os valores para o conjunto A (separados por virgula)')

            stringY = input('Digite os valores para o conjunto B (separados por virgula)')

        if isAlpha(stringX):

            print('Não utilize letras, somente valores numéricos')

            break

        else:

            conjuntoX = parseStrToConjunto(stringX)

            # print(conjuntoX)

        if isAlpha(stringY):

            print('Não utilize letras, somente valores numéricos')

            break

        else:

            conjuntoY = parseStrToConjunto(stringY)

            # print(conjuntoY)

        resposta = operacoes(option, conjuntoX, conjuntoY)

        print(resposta)

main()