Open in Colab

Slide 3 - Conjuntos

1. Listagem de Elementos de um Conjunto

Como você pode representar o conjunto (V = {a, e, i, o, u}) em Python e imprimir seus elementos?

```
V = {'a', 'e', 'i', 'o', 'u'}
print("Elementos do conjunto V:", V)
```

```
Elementos do conjunto V: {'e', 'i', 'u', 'a', 'o'}
```

2. Verificação de Elementos em um Conjunto

Dado o conjunto (V = {a, e, i, o, u}), como você pode verificar se o elemento 'a' está presente em (V)?

```
print("'a' está em V?", 'a' in V)
```

```
'a' está em V? True
```

3. Criação de um Conjunto com Propriedades Específicas

Como você pode criar um conjunto (B) em Python que contém todos os números inteiros pares maiores que 10 e menores que 20?

```
B = {x for x in range(11, 20) if x % 2 == 0}
print("Conjunto B:", B)
```

```
Conjunto B: {16, 18, 12, 14}
```

4. Comparação de Conjuntos

Se (V = {a, e, i, o, u}) e (C = {i, o, u}), como você pode verificar se todos os elementos de (C) também estão em (V)?

```
V = {'a', 'e', 'i', 'o', 'u'}
C = {'i', 'o', 'u'}
```

```
print("Todos os elementos de C estão em V?", C.issubset(V))
```

```
Todos os elementos de C estão em V? True
```

5. Descrição de Conjuntos por Compreensão

Como você pode representar um conjunto (D) em Python que contém todos os números inteiros de 1 a 10 que são divisíveis por 3?

```
D = {x for x in range(1, 11) if x % 3 == 0}
print("Conjunto D:", D)
```

```
Conjunto D: {9, 3, 6}
```

6. União de Conjuntos

Dados dois conjuntos (A = {1, 2, 3}) e (B = {3, 4, 5}), como você pode obter a união de (A) e (B)?

```
A = {1, 2, 3}
B = {3, 4, 5}
uniao = A.union(B)
print("União de A e B:", uniao)
```

```
União de A e B: {1, 2, 3, 4, 5}
```

7. Interseção de Conjuntos

Dado os conjuntos (A = {1, 2, 3}) e (B = {3, 4, 5}), como você pode encontrar a interseção entre (A) e (B)?

```
A = {1, 2, 3}
B = {3, 4, 5}
intersecao = A.intersection(B)
print("Interseção de A e B:", intersecao)
```

```
Interseção de A e B: {3}
```

8. Diferença entre Conjuntos

Como você pode determinar os elementos que estão em (A = {1, 2, 3}) mas não estão em (B = {3, 4, 5})?

```
A = {1, 2, 3}
B = {3, 4, 5}
diferenca = A.difference(B)
print("Elementos em A que não estão em B:", diferenca)
```

```
Elementos em A que não estão em B: {1, 2}
```

9. Simetria de Diferença entre Conjuntos

Se (A = {1, 2, 3}) e (B = {3, 4, 5}), como você pode obter a diferença simétrica entre (A) e (B)?

```
A = {1, 2, 3}
B = {3, 4, 5}
dif_simetrica = A.symmetric_difference(B)
print("Diferença simétrica entre A e B:", dif_simetrica)
```

```
Diferença simétrica entre A e B: {1, 2, 4, 5}
```

10. Subconjuntos e Superconjuntos

Dado (A = {1, 2, 3}) e (B = {1, 2, 3, 4, 5}), como você pode verificar se (A) é um subconjunto de (B) e se (B) é um superconjunto de (A)?

```
A = {1, 2, 3}
B = {1, 2, 3, 4, 5}
print("A é subconjunto de B?", A.issubset(B))
print("B é superconjunto de A?", B.issuperset(A))
```

```
A é subconjunto de B? True
B é superconjunto de A? True
```

11. Números Pares Maiores que 10

$$B = \{x : x \text{ é um número par}, x > 10\}$$

```
B = {x for x in range(11, 100) if x % 2 == 0}
print("Conjunto B:", B)
```

```
Conjunto B: {12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92
```

12. Números Primos Menores que 20

$$P = \{x : x \text{ é um número primo}, x < 20\}$$

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            return False
    return True

P = {x for x in range(2, 20) if is_prime(x)}
print("Conjunto P:", P)
```

```
Conjunto P: {2, 3, 5, 7, 11, 13, 17, 19}
```

13. Números Ímpares Divisíveis por 3 até 30

$$I = \{x : x \text{ é um número ímpar divisível por } 3, x \leq 30\}$$

```
I = {x for x in range(1, 31) if x % 2 != 0 and x % 3 == 0}
print("Conjunto I:", I)
```

```
Conjunto I: {3, 9, 15, 21, 27}
```

14. Quadrados Perfeitos Menores que 100

$$Q = \{x^2 : x \text{ é um número inteiro}, x^2 < 100\}$$

```
Q = {x**2 for x in range(-10, 10) if x**2 < 100}
print("Conjunto Q:", Q)
```

```
Conjunto Q: {64, 1, 0, 36, 4, 9, 16, 81, 49, 25}
```

15. Múltiplos de 5 entre 10 e 50

$$M = \{x : x \text{ é um múltiplo de } 5, 10 < x < 50\}$$

```
M = {x for x in range(11, 50) if x % 5 == 0}
print("Conjunto M:", M)
```

```
Conjunto M: {35, 40, 45, 15, 20, 25, 30}
```

15. Subconjunto próprio e não próprio

Dados dois conjunto em caa um dos cenários abaixo, escreva um script em python para verificar:

- se A é subconjunto de B e se C é subconjunto próprio de D
- Gerar um diagrama de Venn que ilustre as relações entre os conjuntos em cada um dos cenários
- Explicar porque o diagrama pode mostrar "0"e o que isso significa em termos dos elementos dos conjuntos.

Cenário 1

- Considere (A = {1, 2}) e (B = {1, 2, 3}).

Cenário 2

- Considere (C = {1, 2, 3}) e (D = {1, 2, 3}).

```
import matplotlib.pyplot as plt
from matplotlib_venn import venn2

# Definindo os conjuntos
A = {1, 2}
B = {1, 2, 3}
C = {1, 2, 3}
D = {1, 2, 3}

# Verificando subconjunto próprio
is_subset_proper_A_B = A < B
print(f"A é subconjunto próprio de B: {is_subset_proper_A_B}")

# Verificando subconjunto não próprio
is_subset_C_D = C.issubset(D) and C == D
print(f"C é subconjunto não próprio (igual) de D: {is_subset_C_D}")

# Gerando diagrama de Venn para subconjunto próprio
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
venn2([A, B], ('A', 'B'))
plt.title("A é subconjunto próprio de B")

# Gerando diagrama de Venn para subconjunto não próprio
plt.subplot(1, 2, 2)
venn2([C, D], ('C', 'D'))
plt.title("C é subconjunto não próprio (igual) de D")

plt.show()
```

```
A é subconjunto próprio de B: True
C é subconjunto não próprio (igual) de D: True
```