CONJUNTO

Alunos: Vinicius Ozelim Cavichioli, Luara Alves da Silva
Link Github:
https://github.com/ViniciusCavichioli/MatematicaDiscreta/tree/master/Conjuntos

Conjunto é um agrupamento de objetos, como por exemplo:

$$B = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

Para criar uma classe $Conjunto\ (\ )$ em python podemos definir da seguinte forma:
Um Conjunto possui:

- Elementos: Atribuído como uma lista
- Nome: Nome atribuído ao Conjunto

```
class Conjunto():
    def __init__ (self, nome):
        self.elementos = []
        self.nome = nome
```

Essa classe está definida com as seguintes funcionalidades, sendo elas:

- Adicionar Elemento: Função para adicionar os elementos no conjunto.

```
    def adicionar_Elemento(self,item):
        if isinstance(item, Conjunto):
            self.elementos.append(item)
        else:
            if item not in self.elementos:
                self.elementos.append(item)
```

Essa função verifica se o item passado como parâmetro é do tipo Conjunto, se for ele adiciona o Conjunto todo dentro do Conjunto desejado se não, ele verifica se possui elementos repetidos e insere. Ex:

$A = \{1, 2, 3, 3, 4, 5\}$ ✗

$A = \{1, 2, 3, 4, 5\}$ ✓

$B = \{a, b, c, d, e, e\}$ ✗

$B = \{a, b, c, d, e\}$ ✓

$B = \{a, b, c, d, e, \{a, b, c\}\}$ ✓

- Imprimir: Imprimir o Conjunto criado.

```
def __str__(self):
    return '{' + ', '.join([str(i) for i in self.elementos]) + '}'


def imprimir(self):
    print(self.nome, "=", self)
```

A função __str__ é o método chamado pela função str() para obter o valor do objeto em forma de string;
Nisso atribuímos o padrão que queremos ao sair a impressão do conjunto. Como por exemplo:

$B = \{2, 5, 10, 3, 11, 9\}$

$C = \{1, 2, 5, 10, 3, 11, 9, \{1, 2, 3\}, \{0\}\}$

- Pertence: Método para verificar se um elemento pertence a outro Conjunto.

```
def pertence(self,elemento):
    if elemento in self.elementos:
        return True
    else:
        return False
```

Função percorre a conjunto comparado se o elemento passado como parâmetro pertence ao conjunto desejado.

- eh_subconjunto: Se todos os elementos de um conjunto também são elementos de um outro conjunto.

```
def eh_subconjunto(self,conjunto):
    for i in conjunto:
        if i not in self.elementos:
            return False
    return True
```

Essa função percorre os elementos de um Conjunto passado por parâmetro, e verifica se ele pertence ao Conjunto desejado.

- eh_subconjunto_proprio: Verifica se o conjunto passado por parâmetro é subconjunto do desejado, e se possui elementos diferentes.

```
def eh_subconjunto_proprio(self,conjunto):
    for i in conjunto:
        if i not in self.elementos:
            return False
    return True
    for i in self.elementos:
        if i not in conjunto:
            return True
    return False
```

Essa função verifica se o conjunto passado por parâmetro é subconjunto próprio do desejado, ou seja, ele verifica se é subconjunto, se retornar True, ele verifica se possui elementos diferentes.

- união: Realiza a união de dois conjuntos, sendo o que está chamando a função e o passado por parâmetro.

```
def ElementoNeutroE(self,conjunto):
    if len(self.elementos) == 0 and len(conjunto.elementos) != 0:
        return True
    else:
        return False

def ElementoNeutroD(self,conjunto):
    if len(conjunto.elementos) == 0 and len(self.elementos) != 0:
        return True
    else:
        return False

def Idempotencia(self,conjunto):
    if self.elementos == conjunto.elementos:
        return True
    else:
        return False

def RealizaUniao(self, conjunto):
    conjuntoResultado = []
    for i in self.elementos:
        conjuntoResultado.append(i)
    for i in conjunto.elementos:
        if i not in conjuntoResultado:
            conjuntoResultado.append(i)
    return conjuntoResultado

def uniao(self,conjunto):
    if self.ElementoNeutroE(conjunto) == True:
        return print(self.nome,'U',conjunto.nome,'=',self.nome)
    elif self.ElementoNeutroD(conjunto) == True:
        return print(self.nome,'U',conjunto.nome,'=',conjunto.nome)
    elif self.Idempotencia(conjunto) == True:
        return print(self.nome,'U',self.nome,'=',conjunto.nome)
    else:
        result = ""
        verifica = self.RealizaUniao(conjunto)[-1]
        for i in self.RealizaUniao(conjunto):
            if i != verifica:
                result += str(i) + ","
            else:
                result += str(i)
        return print(self.nome,"U",conjunto.nome,"= " "{",result,"}")
```

Essa função verifica as propriedades de Uniao(Elemento Neutro, Idempotencia) antes de realizá las.

- Intersecção: Realiza a intersecção de dois conjuntos, sendo o que está chamando a função e o passado por parâmetro.

```
def ElementoNeutroI(self,conjunto):
    conjuntoResultado = []
    if conjunto.nome == 'U' or self.nome == 'U':
        for i in self.elementos:
            if i in conjunto.elementos:
                conjuntoResultado.append(i)
    else:
        return False

    if len(conjuntoResultado) == 0:
        return False
    return True

def RealizaInterseccao(self, conjunto):
    conjuntoResultado = []
    for i in self.elementos:
        if i in conjunto.elementos:
            conjuntoResultado.append(i)
    return conjuntoResultado

def Interseccao(self,conjunto):
    if self.ElementoNeutroI(conjunto) == True:
        if self.nome == 'U':
            return print(self.nome,'∩',conjunto.nome,'=',conjunto.nome)
        else:
            return print(self.nome,'∩',conjunto.nome,'=',self.nome)
    if self.Idempotencia(conjunto) == True:
        return print(self.nome,'∩',self.nome,'=',conjunto.nome)
    else:
        result = ""
        try:
            verifica = self.RealizaInterseccao(conjunto)[-1]
        except:
            print('')
        for i in self.RealizaInterseccao(conjunto):
            if i != verifica:
                result += str(i) + ","
            else:
                result += str(i)
        return print(self.nome,"∩",conjunto.nome,"= " "{",result,"}")
```

A função de Intersecção realiza a verificação das propriedades(Elemento Neutro e Idempotência).

- Complemento: Dado um conjunto B, podemos encontrar o conjunto complementar de B que é determinado pelos elementos de um conjunto universo que não pertencem a B.

```
def Complemento (self,conjunto):
    conjuntoResultado = []
    for i in self.elementos:
        if i not in conjunto.elementos:
            conjuntoResultado.append(i)
    result = ""
    verifica = conjuntoResultado[-1]
    for i in conjuntoResultado:
        if i != verifica:
            result += str(i) + ","
        else:
            result += str(i)
    return print('~',conjunto.nome,"= " "{",result,"}")
```

Tendo como conjunto Universo $U = \{0,\ 1,\ 2,\ 3,\ 5,\ 9,\ 10,\ 11\}$ seguido de um conjunto $A = \{1,\ 2,\ 3\}$, o Complemento de A é:
$\sim A = \{$ 1,2,3}

- Diferença: a diferença de conjuntos é representada pelos elementos de um conjunto que não aparecem no outro conjunto.

```
def Diferenca(self, conjunto):
    conjuntoResultado = []
    for i in self.elementos:
        if i not in conjunto.elementos:
            conjuntoResultado.append(i)
    result = ""
    verifica = conjuntoResultado[-1]
    for i in conjuntoResultado:
        if i != verifica:
            result += str(i) + ","
        else:
            result += str(i)
    return print(self.nome,'-',conjunto.nome,"= " "{",result,"}")
```

Se o elemento que está no primeiro conjunto não esteja no segundo é adicionado em um novo conjunto representando a diferença. Como exemplo:

Tendo como conjunto Universo $U = \{0,\ 1,\ 2,\ 3,\ 5,\ 9,\ 10,\ 11\}$ seguido de um conjunto $A = \{1,\ 2,\ 3\}$, A diferença de A é:
$U - A = \{$ 0,5,9,10,11}

- Conjunto Partes: resulta no conjunto de todos os subconjuntos de A.

```
def conjuntoPartes(self):
    conjuntoResultado = []
    tamanhoConjunto = len(self.elementos)
    for x in range(tamanhoConjunto):
        for i in itertools.combinations(self.elementos, x):
            conjuntoResultado.append(i)
    result = ""
    verifica = conjuntoResultado[-1]
    for i in conjuntoResultado:
        if i != verifica:
            result += str(i) + ","
        else:
            result += str(i)
    return print('P(',self.nome,')',"= " "{",result,"}")
```

Segue como exemplo, tendo o conjunto $A = \{1,\ 2,\ 3\}$, seu conjunto partes é:
$P(A) = \{(), (1), (2), (3), (1,2), (1,3), (2,3)\}$