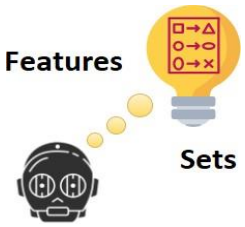
1. **Teoria de conjuntos (sets)**
   1.1. Dados (estruturados e não estruturados)
   1.2. Coleções não ordenadas
   1.3. Descrição
   1.4. Operações

1. Teoria dos Conjuntos (sets)

Conjuntos ou sets são coleções de certos **tipos de dados ou elementos**. A ordenação nos conjuntos não tem relevância, por isso não são acessados através de colchetes que indiquem ordem, mas tão somente um grupo de elementos ali contido. Conjuntos não podem ser divididos e não aceitam a repetição de seus elementos, figura1.



- **Não são ordenados.**
- **Os elementos ou membros são únicos, não se repetem.**
- **Duplicidade em conjunto não é permitida.**
- **O conjunto pode ser mudado, mas os elementos contidos são imutáveis.**

Figura 1 – Características dos conjuntos

Vamos à 3 exemplos de conjuntos de dados ou certos tipos de elementos:

1) A = {1, 3, 5, 7, 9, 11}

2) B = { vermelho, azul, verde, amarelo}

3) C = { 1,vermelho.mp3 , 3,7,triângulo.png, circunferência.pdf, verde}

O que podemos observar nos três conjuntos ?

Os conjuntos A e B são definidos pelos seus próprios elementos, isto quer dizer, que a estrutura do dado ou elemento dessa coleção é quem o define. Então podemos dizer que, o conjunto A é um conjunto finito de dados numéricos ímpares, sua estrutura é reconhecida. O conjunto B é composto de forma finita, por 4 elementos denominados cores ou padrões de cores, também é um padrão estruturado, como string ou cadeia de caracteres que fazem referência as cores.

Já o conjunto C, é composto por elementos de estrutura não definida, mas sim diversos, há numerais, há elementos com referência as cores e há elementos geométricos que podem ser apenas nomes ou até mesmo arquivos com extensão de imagem ou som.

1.1. Dados (estruturados e não estruturados)

Vejamos a seguir como são facilmente identificados quando estruturados (figura 2) e o conjuntos de dados não estruturados (figura 3), requer algumas técnicas mais elaboradas de segmentação e estruturação.

São Paulo Tech School - SPTECH
www.sptech.school / atendimento@sptech.school
R. Haddock Lobo 595, Cerqueira César - São Paulo [SP] 01414-905

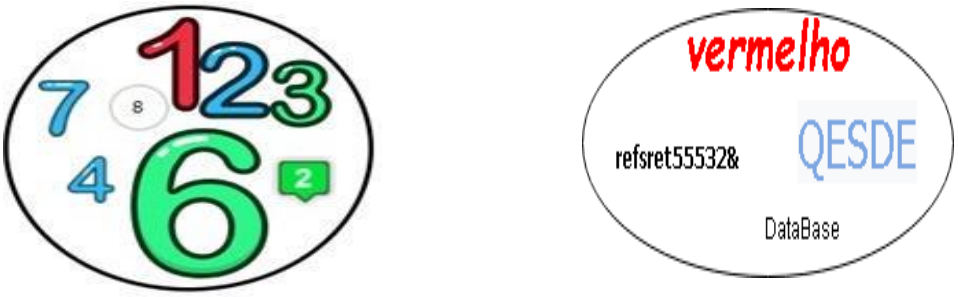


Figura 2 – Dados estruturados: exemplos e características

Os conjuntos da figura 2 são facilmente identificados, porque seus elementos são do mesmo tipo, podem ter atributos distintos, ex: números, datas, strings. A este tipo de conjunto chamamos de dados estruturados. Características:

- Podem ser apresentados em linhas e colunas e permitem relacionamentos;
- Requer menos armazenamento;
- Facilidade de gestão e segurança com soluções legadas.

Agora que já podemos reconhecer os conjuntos ou coleções, conjuntos de dados estruturados e dados não estruturados. Embora os dados não estruturados, sejam parte de conteúdos mais complexos, que não serão aplicados aqui, cabe ressaltar que os dados não estruturados são pré-processado. Segmentados em novas coleções estruturadas ou semi estruturadas, para posterior processamento e aplicação de técnicas aqui estudadas (figura 3).

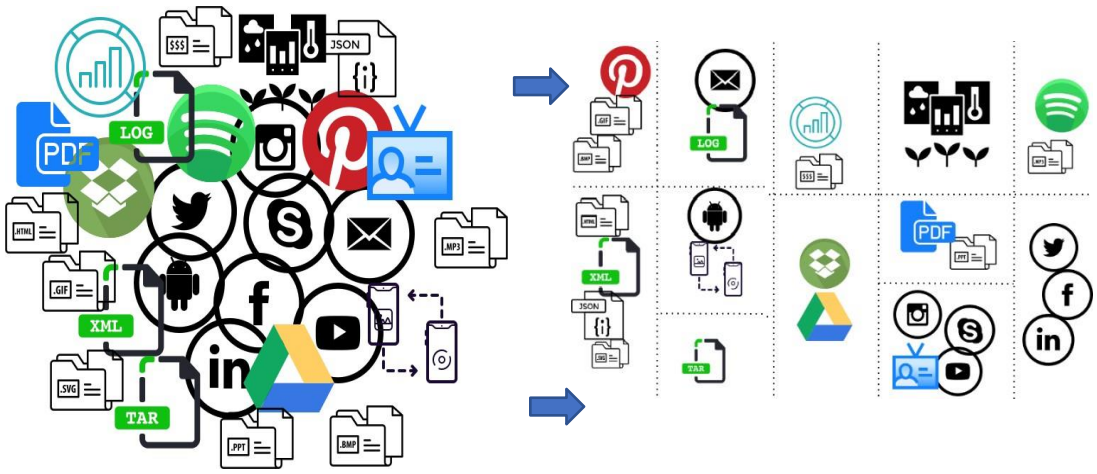


Figura 3 – Dados não estruturados e pré-processamento de segmentação para novos conjuntos de dados estruturados.

Este conjunto não é facilmente identificado, não há uma estrutura definida. Por este motivo são denominados como grandes conjuntos de dados não estruturados. O volume e variedade de dados impossibilitam qualquer análise ou operação de conjuntos. Há necessidade de pré-processamento para segmentar o grande conjunto com dados que possam ter estrutura similar para análises.

Além disso, por sua natureza volumétrica a análise é feita de forma discretizada.

- Os dados não podem ser apresentados em linhas e colunas de bancos relacionais.
- Requer muito espaço de armazenamento
- Gerenciamento e integração com sistemas legados muito difícil.

1.2. Coleções não ordenadas

Uma coleção de dados não ordenados, é um conjunto de dados que não tem uma ordenação por exemplo, ordem alfabética, ordem numérica crescente ou decrescente, ordem por data, etc.

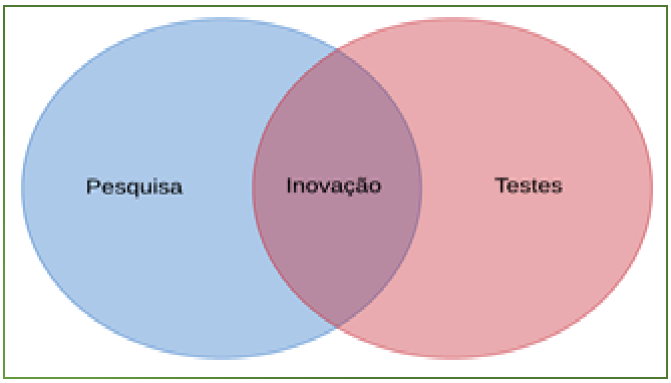
Exemplo: conjunto numerais= {one, three, two}

Podemos ordenar estes conjuntos criando listas, indexando sua ordem, muito comum em estruturas de dados.

1.3 Descrição

Os conjuntos de dados podem ser descritos por meio de um dicionário ou podem ter sua descrição representada por diagramas. A descrição de diagramas em geral permite visualizar as interações do conjunto, unitária ou com outros conjuntos.

Uma classe é representada por uma área plana simples (de formato circular ou oval); se for desejado mostrar o complemento de uma classe, então o círculo ou uma elipse são desenhadas dentro de um retângulo que representa o conjunto universo, figura 4.



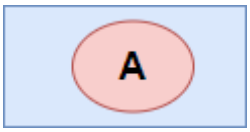


Figura 4 - Conjunto, classe universo ou *universal set*

As relações entre os conjuntos, classes ou sets podem ser representadas graficamente por meio de um dispositivo útil conhecido como Diagrama de Venn. Você descobrirá que os diagramas de Venn são úteis para orientar seu raciocínio sobre conjuntos, e que eles dão mais significado as abstrações teóricas dos conjuntos.

1.3. Operações

Pode-se realizar operações entre conjunto relativa a cada elemento pertencente a eles, figura 5.

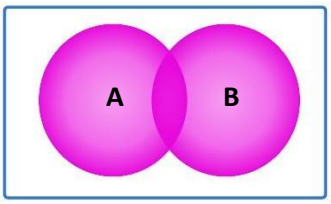
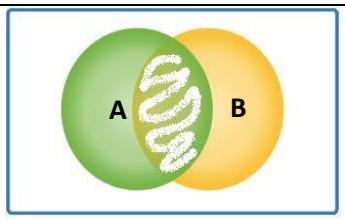
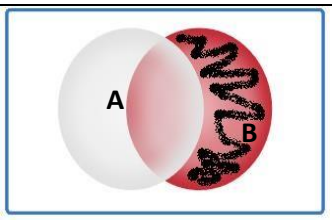
		
União: AUB	**Intersecção: A∩B**	**Diferença: A-B**

Figura 5 – Operações fundamentais de conjuntos

Outras relações a pertinência e a igualdade. Pertinência é a relação entre um determinado conjunto e o elemento que pode ou não pertencer a esse conjunto.

Simbologia pertinência $\in$ (pertence) $\notin$ (não pertence)

Simbologia igualdade $A=B$

Vamos agora praticar estes conceitos com Python, mas para iniciar precisamos criar um conjunto.

A nomenclatura de comandos em Python relacionados as operações de conjuntos estão descritas na tabela 1:

Operação	Comando/operador
Pertence	in
Não Pertence	not in
União	|
Interseção	&
Diferença	-
Igualdade	=
A maior b	A > b

Tabela 1 – Comandos em Python das operações com conjuntos

Em Python existem duas maneiras de criar conjunto, uma usando chaves **{ }** e outra o comando **set.** Vamos usar o interpretador IDLE.

```
>>> conjunto_processador = {'CPU', 'Registrador', 'Core'}
>>> conjunto_processador
{'Registrador', 'Core', 'CPU'}
```

Agora crie esse mesmo conjunto com o comando **set**

```
>>> conjunto_processador = set(['CPU', 'Registrador', 'Core'])
>>> conjunto_processador
{'Registrador', 'Core', 'CPU'}
```

O resultado da criação do conjunto foi o mesmo, mas qual a diferença? O comando **set** cria uma lista de elementos que não se repetem.

Vamos testar:

```
>>> lista_processador = {'CPU', 'Registrador', 'Core', 'CPU'}
>>> conjunto_processador2 = set (lista_processador)
>>> conjunto_processador2
{'Registrador', 'Core', 'CPU'}
```

Com este procedimento podemos garantir a unicidade dos elementos do conjunto e portanto, realizar operações com conjunto, facilitando a aplicação.

Operação de União

Recomendamos que volte na figura 5 para relembrar a representação no Diagrama de Venn da operação União.

A máquina do usuário Thor está sendo monitorada a partir das medidas de 3 elementos de seu hardware. Mas a empresa resolveu ampliar o monitoramento de todas as máquinas, assim economizar no consumo de energia ou encontrar usuários subutilizando os computadores. A primeira medida foi a de capturar os dados de máquinas, e pelo menos dois processos em execução. Verifique o código a seguir:

```
>>> user_Thor = {'mysql','CPU','RAM','SSD1','Google'}
>>> user_Thanos = {'LoL','RAM','CPU','HD','Google'}
>>> user_CA = {'mysql','LOL','RAM','CPU','Firefox'}
>>> user_TS = {'mysql','CPU','RAM','SSD1','Google'}
>>> inventario1 = user_Thor | user_CA
>>> inventario1
{'SSD1', 'LOL', 'mysql', 'RAM', 'CPU', 'Firefox', 'Google'}
```

E se fosse realizada a operação de união com os 4 usuários, o resultado seria como o anterior? Confirme.

```
>>> user_Thor = {'mysql','CPU','RAM','SSD1','Google'}
>>> user_Thanos = {'LoL','RAM','CPU','HD','Google'}
>>> user_CA = {'mysql','LOL','RAM','CPU','Firefox'}
>>> user_TS = {'mysql','CPU','RAM','SSD1','Google'}
>>> inventario1 = user_Thor | user_CA
>>> inventario1
{'SSD1', 'LOL', 'mysql', 'RAM', 'CPU', 'Firefox', 'Google'}
>>> inventario2 = user_Thor | user_CA | user_Thanos | user_TS
>>> inventario2
{'SSD1', 'HD', 'LOL', 'mysql', 'RAM', 'CPU', 'LoL', 'Firefox', 'Google'}
```

O resultado do inventario2 não foi o mesmo de inventario1, você sabe explicar o que ocorreu?

São Paulo Tech School - SPTECH
www.sptech.school / atendimento@sptech.school
R. Haddock Lobo 595, Cerqueira César - São Paulo [SP] 01414-905

Outra forma de executar a operação de união, é usando o comando union.

```
>>> inventario3 = user_Thor.union(user_CA)
>>> inventario3
{'SSD1', 'LOL', 'mysql', 'RAM', 'CPU', 'Firefox', 'Google'}
```

Exercício1: Utilizando o comando **union** em Python, faça com que os quatro conjuntos, user_Thor, user_Thanos, user_CA, user_TS estejam em uma operação de união. Qual é o resultado em python?

Operação de Intersecção

A resultante desta operação & de conjuntos distintos, com elementos finitos, é de se obter um novo conjunto com os elementos comuns em ambos conjuntos de origem.

```
>>> inventariol = user_Thor & user_CA
>>> inventariol
{'mysql', 'RAM', 'CPU'}
>>>
```

Usando o comando intersection:

```
>>> user_Thor = {'mysql','CPU','RAM','SSD1','Google'}
>>> user_Thanos = {'LoL','CPU','RAM','HD','Google'}
>>> inventario3 = user_Thor.intersection(user_Thanos)
>>> inventario3
{'Google', 'RAM', 'CPU'}
```

Exercício2: Faça o mesmo para todos os 4 conjuntos, considerando seus elementos em comum. Utilize o operados & e depois o comando intersection.

Diferença

É a operação resultante dos elementos de uma conjunto que não existem em outro conjunto da relação.

```
>>> inventario5 = user_Thor - user_Thanos
>>> inventario5
{'SSD1', 'mysql'}
```

Exercício3: Faça o mesmo para todos os 4 conjuntos, considerando seus elementos na diferença. Utilize o operados -.

Operação de pertinência

Esta operação exibe como resultante a booleana, verdadeira ou falsa, para elemento pertence ao conjunto. Veja o exemplo:

```
>>> 'CPU' in user_TS
True
>>> 'Firefox' in user_Thanos
False
```


São Paulo Tech School - SPTECH
www.sptech.school / atendimento@sptech.school
R. Haddock Lobo 595, Cerqueira César - São Paulo [SP] 01414-905

.....

```
>>> 'LoL' not in user_Thor
True
>>> 'LoL' not in user_Thanos
False
```

Exercício4: Faça o mesmo para todos os 4 conjuntos, considerando seus elementos na pertinência. Utilize o operados in e not in.

Pertinência com subconjuntos e superconjuntos (subset (<=) e superset(>=))

Um caso bem comum é o de saber se determinado conjunto contém outro conjunto com todos os seus elementos, neste caso temos que um subconjunto está contido em um conjunto.

```
>>> user_Thor.issubset(user_Thanos)
False
```

.....

```
>>> user_Thor.issubset(user_TS)
True
```

Outra forma:

```
>>> user_Thor <= user_TS
True
>>> user_Thor <= user_Thanos
False
```

Agora podemos testar casos contrários, como se um conjunto tem um subconjunto, neste caso, testaremos superconjuntos.

```
>>> user_TS.issuperset(user_Thor)
True
>>> user_TS >=(user_Thor)
True
```

...

Apresente uma solução para todos os 4 conjuntos, user_Thor, user_Thanos, user_CA, user_TS, considerando seus elementos na pertinência de subconjuntos e superconjuntos. Utilize a recursividade, faça testes antes de chegar a uma solução ideal. Tente usar a lógica proveniente da linguagem para chegar a solução. Teste possíveis recursividades.

Igualdade (comparação)

Vimos que conjuntos de dados, tem variedade e volume. Outra propriedade característica dos conjuntos é a **veracidade.**

A veracidade pode ser testada nos conjuntos de dados por sua igualdade, a comparação, até porque ela é dada pelos elementos que pertecem ao conjunto e não por sua classe, o conjunto em si.

```
>>> user_TS ==(user_Thor)
True
>>> user_TS !=(user_Thor)
False
```

...

```
>>> user_TS ==(user_CA)
False
>>> user_TS !=(user_CA)
True
```

Diferença simétrica (relacionado a integridade de dados)

Um novo conjunto surge, contendo todos os elementos que não são comuns aos dois conjuntos.

```
>>> user_TS ^ user_CA
{'Google', 'CPU', 'Firefox', 'LOL'}
```

 **Carimbe seu conhecimento**	A Integridade de dados, de uma forma geral, podeser compreendida pela manutenção e a garantia da precisão e consistência de dados durante todo o ciclo de vida da informação, e é um aspecto crítico para o projeto, implementação e uso de qualquer sistema que armazene, processe ou recupere esses dados.

Exercício5: Você está a 2 anos na empresa, e foi chamado para prover treinamento técnico. Vai receber uma nova equipe de estagiários no departamento de Big Data. Faça uma apresentação com exemplos de conjuntos de dados diferentes dos que vimos aqui, relativo a indicadores de consumo de energia e meio ambiente (relacionado a energia elétrica), cada conjunto deve conter no mínimo 5 elementos distintos, não ordenados. Voce deverá explicar a nova equipe as operações de igualdade e diferença simétrica.

Você pode criar no mínimo 4 e no máximo 6 conjuntos, distintos, porém com repetição de elementos de livre critério.

Pode fazer em dupla.

Suba upload

E prepare o treinamento de 4 minutos

São Paulo Tech School - SPTECH
www.sptech.school / atendimento@sptech.school
R. Haddock Lobo 595, Cerqueira César - São Paulo [SP] 01414-905

Exemplos: teste, corrija, aprenda

Criação conjuntos:

```
>>> frutas = ['Pera', 'Laranja', 'Uva', 'Morango', 'Pera', 'Abacate
', 'Laranja']
>>> cj1 = set(frutas)
>>> print(cj1)
{'Pera', 'Laranja', 'Uva', 'Morango', 'Pera', 'Abacate', 'Laranja'}
```

Há ordenação?

Criação de conjunto vazio (atenção aqui)

```
>>> cj2 = set()
>>> s1 = {}
>>> type(cj2)
<class 'set'>
>>> type(s1)
<class 'dict'>
```

Explique porque foi usado {} em s1 e apareceu a class dictionary.

Resposta: Um conjunto vazio apenas pode ser criado por meio de set() sem argumentos:

cj2 = set()

Usar as chaves vazias para criar um conjunto vazio não é uma forma válida para Python, porque vai interpretar como um dicionário vazio.

Método de adição:

Sintaxe do método é s.add(x)

```
>>> s2 = {'Pera', 'Uva', 'Abacate', 'Laranja', 'Abacaxi'}
>>> s2.add('Maracuja')
>>> s2
{'Maracuja', 'Uva', 'Laranja', 'Pera', 'Abacate', 'Abacaxi'}
```


São Paulo Tech School - SPTECH
www.sptech.school / atendimento@sptech.school
R. Haddock Lobo 595, Cerqueira César - São Paulo [SP] 01414-905

Principais métodos em Python para sets

Sintaxe	Explicação
s.difference_update(t)	Remove os itens que estão no conjunto t do conjunto s.
s.intersection_update(t)	Faz com que o conjunto s contenha a interseção dele com t.
s.symetric_difference_update(t)	Faz com que o conjunto s contenha a diferença simétrica dele com t.
s.isdisjoin(t)	Retorna True se s e t não tem nenhum item em comum.
s.issubset(t) (s <= t)	Retorna True se s é igual a ou um subconjunto de t.
s.issuperset(t) (s >= t)	Retorna True se s é igual a ou t é um subconjunto de s.
s.discard(x)	Remove o item x do conjunto s.
s.remove(x)	Remove o item x se ele estiver em s se não retorna um KeyError.
s.update(t)	Adiciona cada item do conjunto s que não está no conjunto t para o conjunto t.

Pra quem quiser treinar, tem um jogo chamado While True: Learn() , ele explica a história da ciência de dados e tem exercícios práticos sobre conjuntos e algoritmos

São Paulo Tech School - SPTECH
www.sptech.school / atendimento@sptech.school
R. Haddock Lobo 595, Cerqueira César - São Paulo [SP] 01414-905