

A decorative graphic on the left side of the slide, consisting of a grid of squares in various shades of green and one red square. The squares are arranged in a non-uniform pattern, with some being larger than others. The colors range from light green to dark green, with one prominent red square in the middle-left area.

# Estrutura de Dados

Conjunto em Python

Prof. Dr. Danilo Barbosa

# O que vamos ver nessa aula?

- ▶ Conjuntos



# Conjuntos

- ▶ Matematicamente, um conjunto é uma coleção de itens que não estão em nenhuma ordem específica. Um `set` em Python é semelhante a esta definição matemática com as condições adicionais abaixo.
  - ▶ Os elementos do conjunto não podem ser duplicados.
  - ▶ Os elementos do conjunto são imutáveis (não podem ser modificados), mas o conjunto como um todo é mutável.
  - ▶ Não há índice anexado a nenhum elemento em um `set` em Python. Portanto, eles não suportam nenhuma operação de indexação ou fatiamento.
- ▶ Eles são usados normalmente para operações matemáticas como união, interseção, diferença, etc.



# Conjuntos

- ▶ Criando um conjunto
  - ▶ `a = set()`
  - ▶ `diasUteis = set(["seg", "ter", "qua", "qui", "sex"])`
  - ▶ `mesesVerao = {"dez", "jan", "fev", "mar"}`
  - ▶ `nums = {1, 2, 3}`
- ▶ Determinando o tamanho do conjunto
  - ▶ `len(a)`
  - ▶ `len(diasUteis)`



# Conjuntos

- ▶ Acessando itens
  - ▶ Não podemos acessar os itens por índice ou chaves
    - ▶ `for e in diasUteis: print(e)`
    - ▶ `print("seg" in diasUteis)`
- ▶ Como não podemos acessar os itens individualmente, não conseguimos modificá-los, mas conseguimos adicionar e remover elementos
- ▶ Adicionando um elemento ao conjunto
  - ▶ `nums.add(1)`
  - ▶ `nums.add(10)`



# Conjuntos

- ▶ Adicionando um conjunto
  - ▶ `nums.update({10,20,30})`
- ▶ Adicionando listas, tuplas e dicionários
  - ▶ `nums.update({"AA":11, "BB":22, "CC":33})`
  - ▶ `nums.update((44,55,66))`



# Conjuntos

- ▶ Removendo um elemento
  - ▶ `nums.remove(10)` #se o item não existe **gera um erro**
  - ▶ `nums.discard(10)` #se o item não existe **não gera um erro**
- ▶ Esvaziando o conjunto
  - ▶ `nums.clear()`
- ▶ Deletando um conjunto completamente
  - ▶ `del nums`



# Conjuntos

- ▶ União de dois conjuntos ( operador | )
  - ▶ `union()` retorna um novo conjunto com todos os itens de ambos os conjuntos
    - ▶ `set1 = {"a", "b", "c"} set2 = {1, 2, 3}`  
`set3 = set1.union(set2) print(set3)`
  - ▶ `update()` insere os itens de set2 em set1
    - ▶ `set1 = {"a", "b", "c"} set2 = {1, 2, 3}`  
`set1.update(set2) print(set1)`
- ▶ Ambos `union()` e `update()` vão excluir qualquer item duplicado





# Conjuntos

- ▶ Interseção de dois conjuntos ( operador &)
  - ▶ `intersection()` retorna um novo conjunto que contém os itens que estão presentes em ambos os conjuntos
    - ▶ `x = {"apple", "banana", "cherry"}`  
`y = {"google", "microsoft", "apple"}`  
  
`z = x.intersection(y)`  
  
`print(z)`
  - ▶ `intersection_update()` mantém somente os itens presentes em ambos os conjuntos
    - ▶ `x = {"apple", "banana", "cherry"}`  
`y = {"google", "microsoft", "apple"}`  
  
`x.intersection_update(y)`  
  
`print(x)`



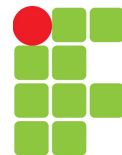
# Conjuntos

- ▶ Diferença de dois conjuntos ( operador - )
  - ▶ `difference()` retorna um novo conjunto que contém a diferença entre os dois conjuntos
    - ▶ `x = {"apple", "banana", "cherry"}`  
`y = {"google", "microsoft", "apple"}`  
  
`z = x.difference(y)`  
  
`print(z)`
    - ▶ `difference_update()` remove os itens que existem em ambos os conjuntos
      - ▶ `x = {"apple", "banana", "cherry"}`  
`y = {"google", "microsoft", "apple"}`  
  
`x.difference_update(y)`  
  
`print(x)`

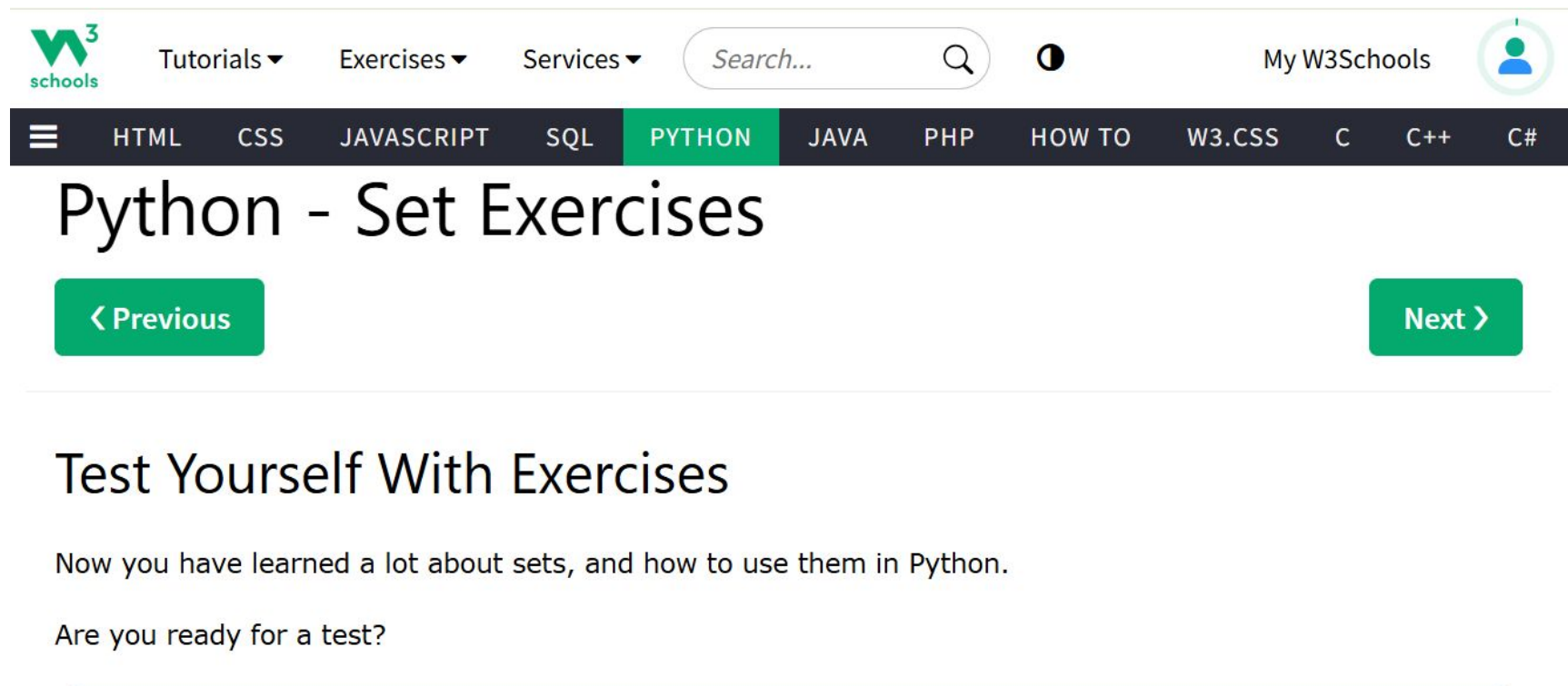


# Conjuntos

- ▶ Diferença simétrica de dois conjuntos ( operador  $\wedge$  )
  - ▶ `symmetric_difference()` retorna um novo conjunto que contém os elementos que não estão presentes em ambos os conjuntos
    - ▶ `x = {"apple", "banana", "cherry"}`  
`y = {"google", "microsoft", "apple"}`  
  
`z = x.symmetric_difference(y)`  
  
`print(z)`
  - ▶ `symmetric_difference_update()` mantém os elementos que não estão presentes em ambos os conjuntos
    - ▶ `x = {"apple", "banana", "cherry"}`  
`y = {"google", "microsoft", "apple"}`  
`x.symmetric_difference_update(y)`  
  
`print(x)`



# Atividade de conjuntos



The screenshot shows the W3Schools website interface. At the top, there is a navigation bar with links for Tutorials, Exercises, and Services, along with a search bar and a user profile icon. Below this is a dark blue menu bar with various programming topics: HTML, CSS, JAVASCRIPT, SQL, PYTHON (highlighted in green), JAVA, PHP, HOW TO, W3.CSS, C, C++, and C#. The main heading is "Python - Set Exercises". Below the heading are two green buttons: "< Previous" on the left and "Next >" on the right. The content area has the heading "Test Yourself With Exercises" followed by the text "Now you have learned a lot about sets, and how to use them in Python." and "Are you ready for a test?".

W3schools Tutorials Exercises Services Search... My W3Schools

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C++ C#

## Python - Set Exercises

< Previous

Next >

### Test Yourself With Exercises

Now you have learned a lot about sets, and how to use them in Python.

Are you ready for a test?

[https://www.w3schools.com/python/python\\_sets\\_exercises.as](https://www.w3schools.com/python/python_sets_exercises.as)



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
PERNAMBUCO

# Atividade de conjuntos

**Exercício 6.19** Escreva um programa que compare duas listas. Utilizando operações com conjuntos, imprima:

- os valores comuns às duas listas
- os valores que só existem na primeira
- os valores que existem apenas na segunda
- uma lista com os elementos não repetidos das duas listas.
- a primeira lista sem os elementos repetidos na segunda

**Exercício 6.20** Escreva um programa que compare duas listas. Considere a primeira lista como a versão inicial e a segunda como a versão após alterações. Utilizando operações com conjuntos, seu programa deverá imprimir a lista de modificações entre essas duas versões, listando:

- os elementos que não mudaram
- os novos elementos
- os elementos que foram removidos



# Referência

PUGA, Sandra; RISSETTI, Gerson. Lógica de Programação e Estruturas de Dados-Com Aplicações em Java. 3ª edição. 2016.

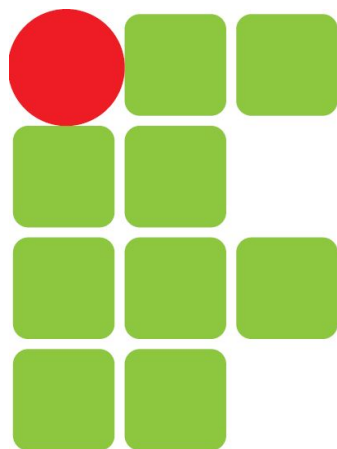
GOODRICH, Michael T.; TAMASSIA, Roberto. **Estruturas de dados & algoritmos em Java**. Bookman Editora, 2013.

CASTILHO, Marcos; SILVA, Fabiano; WEINGAERTNER, Daniel. Algoritmos e Estruturas de Dados I. 2011.

Curso de programação em Python do prof. Gustavo Guanabara. Último acesso em: 01 Set 2024.

Vídeo sobre dicionários: <https://youtu.be/ZWj8o692qGY>





**INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**  
**PERNAMBUCO**