

# DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON

Tema 3 - Aula 07 : Python com Banco  
de dados (Parte III).




*Prof. Ronaldo Candido*  
*ronaldo.candido@estacio.br*



**Estácio**

2023.2

1


DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON 

### Objetivos da aula

- **Aplicar as funcionalidades para inserção, remoção e atualização de registros em tabelas.**

### Conteúdo Programático

1. Situação-problema.
2. Inserção de registros.
3. Atualização de registros.
4. Remoção de registros.
5. Atividade verificadora de aprendizagem.
6. Aprenda+.



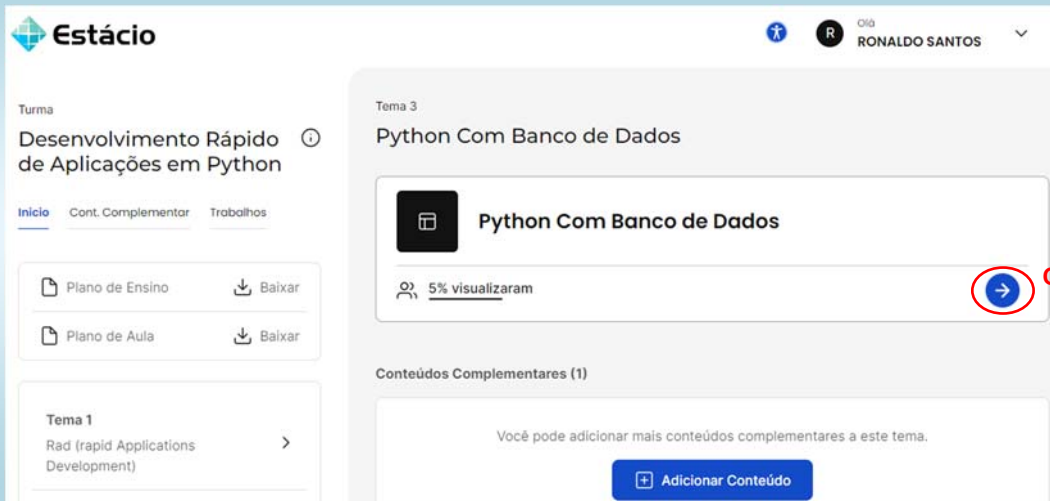
Prof. Ronaldo Candido AULA 07

2

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON

**Estácio**

## Sala de Aula Virtual (SAVA)



The screenshot shows the Estácio SAVA interface. On the left, under 'Turma', is 'Desenvolvimento Rápido de Aplicações em Python' with tabs for 'Início', 'Cont. Complementar', and 'Trabalhos'. Below are links to 'Plano de Ensino' and 'Plano de Aula', each with a 'Baixar' button. Under 'Tema 1' is 'Rad (rapid Applications Development)'. On the right, under 'Tema 3', is 'Python Com Banco de Dados' with a '5% visualizaram' indicator. A red circle highlights a blue arrow icon next to the text 'Conteúdo digital'. Below this is a section for 'Conteúdos Complementares (1)' with a message 'Você pode adicionar mais conteúdos complementares a este tema.' and an 'Adicionar Conteúdo' button.

Prof. Ronaldo Candido

AULA 07


3

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON

**Estácio**

## Situação-problema

- Na aula anterior vimos como criar tabelas a um Banco de Dados usando um SGBD. Para implementar funcionalidades de Bancos de Dados em nosso sistema de registro de notas, precisaremos saber manipular dados nas tabelas. **Como realizar esta tarefa usando Python ?**



The image shows a group of five diverse students, three boys and two girls, sitting around a table in a classroom setting. They are all looking at a tablet held by one of the girls, appearing to be engaged in a collaborative learning activity.

Prof. Ronaldo Candido

AULA 07

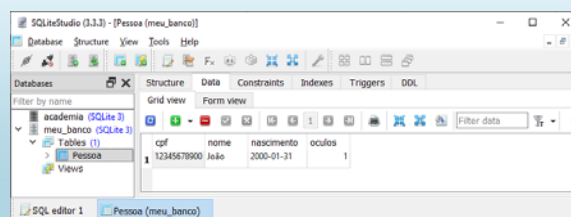
4

## Inserção de registros

- Para inserir registros em uma tabela, utilizamos o comando **INSERT INTO**, do **SQL**.
- Utilizaremos um exemplo para inserir o seguinte registro na tabela Pessoa :

**CPF: 12345678900**  
**Nome: João**  
**Data de Nascimento: 31/01/2000**  
**Usa óculos: Sim (True)**

```
INSERT INTO Pessoa (cpf, nome, nascimento, olhos)
VALUES (12345678900, 'João', '2000-01-31', 1);
```



Prof. Ronaldo Candido

AULA 07

5

## Exemplo de código no SQLite

```
#Aula07_exemplo01
import sqlite3 as conector #apelido
def criar_tabela():
    try: #abertura da conexão
        conexao = conector.connect('meu_banco.db')
        cursor = conexao.cursor()
        sql = """CREATE TABLE if not exists Pessoa (
            cpf INTEGER NOT NULL,
            nome TEXT NOT NULL,
            nascimento DATE NOT NULL,
            olhos BOOLEAN NOT NULL,
            PRIMARY KEY (cpf)
        );"""
```

```
cursor.execute(sql)
#efetivação do comando
conexao.commit()
print('Banco de dados ok')
except conector.DatabaseError as err:
    print('Erro de banco de dados',err)
finally:
    #fechamento das conexões
    if(conexao):
        cursor.close()
        conexao.close()
```

Prof. Ronaldo Candido

AULA 07

6

### Exemplo de código no SQLite (continuação)

```
def inserir_registro():
    try:
        conexao = conector.connect('meu_banco.db')
        cursor = conexao.cursor()
        sql = "INSERT INTO Pessoa (cpf, nome,
nascimento, olhos) VALUES
(12345678900, 'João', '2000-01-31', 1);"
        cursor.execute(sql)
        conexao.commit()
        print('Registro inserido')
    except conector.DatabaseError as err:
        print('Erro de banco de dados',err)

finally:
    #fechamento das conexões
    if(conexao):
        cursor.close()
        conexao.close()

    #executar funções
    criar_tabela()
    inserir_registro()

    #encerrando
    print("Fim do programa")
```

Prof. Ronaldo Candido

AULA 07

7

### Inserção de dados com queries dinâmicas

- É muito comum reutilizarmos uma mesma **string** para inserir diversos registros em uma tabela, alterando apenas os dados a serem inseridos.
- Para realizar esse tipo de operação, o método **execute**, da classe **Cursor**, prevê a utilização de parâmetros de consulta, que é uma forma de criar comandos **SQL** dinamicamente.
- Essa concatenação é realizada de forma correta, evitando brechas de segurança, como a **SQL Injection**, e convertendo os dados para os tipos e formatos esperados pelo banco de dados. Como resultado final, temos um comando pronto para ser enviado ao banco de dados.
- Para indicar que a **string** de um comando contém parâmetros que precisam ser substituídos antes da sua execução, utilizamos delimitadores. Esses delimitadores podem ser: **"?"**, **"%s"**, entre outros. Na biblioteca do **SQLite**, utilizamos o delimitador **"?"**. Exemplo:  
comando = "INSERT INTO tabela1 (atributo1, atributo2) VALUES (?, ?);"

Prof. Ronaldo Candido

AULA 07

8

## Inserção de dados com queries dinâmicas (continuação)

```
#Aula07_exemplo02.py
import sqlite3 as conector #apelido
#Classe Pessoa
class Pessoa:
    def __init__(self,cpf,nome,data_nascimento,usa_olculos):
        self.cpf = cpf
        self.nome = nome
        self.data_nascimento = data_nascimento
        self.usa_olculos = usa_olculos
#função inserir dados com parâmetros
def inserir_dados():
    try:
        conexao = conector.connect('meu_banco.db')
        cursor = conexao.cursor()
```

**A classe Pessoa será usada como modelo na orientação a objetos.**

## Inserção de dados com queries dinâmicas (continuação)

```
    pessoa = Pessoa(20000000044,'José','1999-11-15',True) #Criação dos dados de Pessoa
    #definição com query parameter
    sql = 'INSERT INTO Pessoa (cpf, nome, nascimento, olculos) VALUES (?,?,?,?)'
    cursor.execute(sql,(pessoa.cpf,pessoa.nome,pessoa.data_nascimento,pessoa.usa_olculos))
    conexao.commit()
    print('Dados inseridos!!!')
except conector.DatabaseError as err:
    print('Erro de banco de dados: ',err)
finally:
    if(conexao):
        cursor.close()
        conexao.close()
#executar função
inserir_dados()
print("Fim do programa")
```

DB Browser for SQLite - C:\2022\_win10pro\estacio\2022.1\Desenv\_Rápido\_Python\aulas\zips\

Arquivo Editar Exibir Ferramentas Ajuda

Novo banco de dados Abrir banco de dados Escrever modificações

Estrutura do banco de dados Navegar dados Editar pragmas Executar SQL

Tabela: Pessoa

	cpf	nome	nascimento	olculos
Filtro	Filtro	Filtro	Filtro	Filtro
1	10000000099	Maria	1990-01-31	0
2	12345678900	João	2000-01-31	1
3	20000000044	José	1999-11-15	1



## Atualização de registros

- Para atualizar um registro, utilizamos o comando **SQL UPDATE**. Sintaxe:  
UPDATE tabela1 SET coluna1 = valor1, coluna2 = valor2... WHERE [condição];
- Assim como no comando **INSERT**, podemos montar o comando **UPDATE** de três formas. Uma **string** sem delimitadores, uma **string** com o delimitador "?" ou uma **string** com argumentos nomeados.
- Também podemos utilizar os delimitadores na condição da cláusula **WHERE** :

```
#Aula07_exemplo03.py
import sqlite3 as conector
#função alterar
def alterar_dados():
    try:
        conexao = conector.connect('meu_banco.db')
        cursor = conexao.cursor()
```

11

## Atualização de registros (continuação)

```
#definição e execução dos comandos nas três formas
comando1 = 'UPDATE Pessoa SET olhos=1;'
cursor.execute(comando1)
comando2 = 'UPDATE Pessoa SET olhos=? WHERE cpf=10000000099;'
cursor.execute(comando2, (True,))
comando3 = 'UPDATE Pessoa SET olhos=:usa_olhos WHERE cpf=:cpf;'
cursor.execute(comando3, {'usa_olhos': False, 'cpf': 20000000044})
conexao.commit()
print('Dados alterados!!!')
except conector.DatabaseError as err:
    print('Erro de banco de dados: ', err)
finally:
    if(conexao):
        cursor.close()
        conexao.close()
#executar função
alterar_dados()
print("Fim do programa")
```

12

## Remoção de registros

- Para remover um registro, utilizamos o comando **SQL DELETE**. Sintaxe:

DELETE FROM tabela1 WHERE [condição];

- Exemplo:

#Aula07\_exemplo04.py

import sqlite3 as conector #apelido

def excluir\_dados():

try:

conexao = conector.connect('meu\_banco.db')

cursor = conexao.cursor()

comando = "DELETE FROM Pessoa WHERE  
cpf=12345678900;"

cursor.execute(comando)

conexao.commit()

print('Registro excluído!!!')

except conector.DatabaseError as err:  
print('Erro de banco de dados: ',err)

finally:

if(conexao):

cursor.close()

conexao.close()

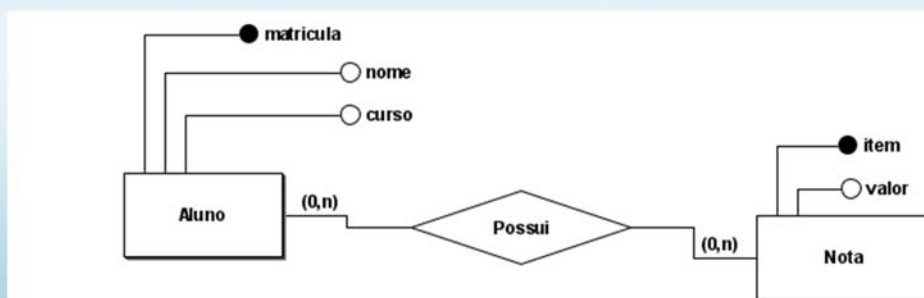
#executar função

excluir\_dados()

print("Fim do programa")

## Atividade verificadora de aprendizagem

- Voltando ao cenário que trata de um sistema de registro de notas de alunos em uma pequena instituição de ensino, implementar as funcionalidades de inserção, exclusão e modificação dos dados. Usar o seguinte modelo relacional (BRModelo. Disponível em: <<https://www.sis4.com/brModelo/>>. Acesso em: 28 mar. 2023.



### Aprenda +

- Assistir o vídeo: FOROGH, P. "Python Sqlite Database | For Beginners | CRUD Operations". Disponível em <<https://www.youtube.com/watch?v=wglgdlyltlc>>. (Ative as legendas e a tradução para português). Acesso em: 30 mar. 2023.
- Assistir o vídeo: Ignorância Zero. "Aulas Python - 062 - POO IV: Herança, Super e Polimorfismo". Disponível em <<https://www.youtube.com/watch?v=Z2qAbhqNEXE>>. Acesso em: 30 mar. 2023.
- Atividade Autônoma Aura:

Questão 1) A Linguagem de programação Python permite a inserção em massa de várias linhas em uma tabela de banco de dados usando uma única consulta SQL. Pode-se fazer isso usando uma consulta parametrizada com o comando:

- a) insertmany()
- b) execute()
- c) inserirmuitos()
- d) executemany()
- e) writeln()

### Aprenda + (continuação)

Questão 2) Após a execução de comandos de manipulação de registros, tem-se que realizar a finalização da transação, e para isso usamos o comando:

- a) commit()
- b) rollback()
- c) persist()
- d) execute()
- e) run()





### Para a próxima aula...

- Leitura e resolução dos exercícios propostos no livro "BANIN, S. L. Python 3 Conceitos e Aplicações. Uma Abordagem Didática". Capítulo 8: Python 3 com Banco de Dados SQLite.
- Conteúdo digital da disciplina, Tema 3 - "PYTHON COM BANCO DE DADOS", Módulo 4 "Funcionalidades para recuperação de registros em tabelas".
- Estudar a Aula 08 de DESENVOLVIMENTO RÁPIDO EM PYTHON no SAVA previamente.



Prof. Ronaldo Candido

AULA 07

17

### Referências

BANIN, S. L. Python 3 Conceitos e Aplicações. Uma Abordagem Didática. 1a. ed. São Paulo: Érica, 2018. Páginas 329-359. Disponível em: <<https://integrada.minhabiblioteca.com.br/#/books/9788536530253/>>. Acesso em: 16 fev. 2023.

ESTÁCIO. Conteúdo digital da disciplina, Tema 3 - "Python com banco de dados", Módulo 3 "Inserção, remoção e atualização de registros em tabelas". Disponível na Sala de aula virtual da disciplina.

PERKOVIC, L. Introdução à Computação Usando Python - Um Foco no Desenvolvimento de Aplicações. 1a. ed. Rio de Janeiro: LTC, 2016. Disponível em: <<https://integrada.minhabiblioteca.com.br/reader/books/9788521630937/>>. Acesso em: 16 fev. 2023.



Prof. Ronaldo Candido

AULA 07

18

# DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON

Dúvidas, sugestões ou análises ???



*Prof. Ronaldo Candido*  
*ronaldo.candido@estacio.br*



**Estácio**