

# Python 2 - Aula 1

## ▼ Slide aula 1

DJANGO

---



# Bora Revisar?

DJANGO

---



1º Qual a saída de `print(3 + 2 * 2)`?

- (a) 10      (b) 7      (c) 9

2º Qual função embutida converte uma string para inteiro?

- (a) `int()`      (b) `str()`      (c) `float()`

3º Como você cria um comentário em Python?

- (a) `// comentário`      (b) `/* comentário */`      (c) `# comentário`



## DJANGO

---



4º Qual das opções é um tipo de dado imutável em Python?

- (a) Lista      (b) Tupla      (c) Dicionario

5º Qual função interrompe um loop em Python?

- (a) break      (b) continue      (c) exit

6º Qual a saída de len([1, 2, 3, 4])?

- (a) 3      (b) 4      (c) 5



---

## DJANGO

---



## DJANGO

---



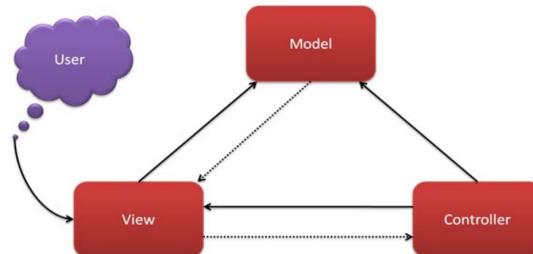
- Django é um **framework** web de código aberto, escrito em Python, que permite a criação de aplicativos web de forma rápida, escalável e segura.
- É uma ferramenta poderosa para desenvolver aplicativos complexos, como redes sociais, plataformas de comércio eletrônico e sistemas de gerenciamento de conteúdo.
- Segue o padrão **Model-View-Template (MVT)** e vem com um grande número de bibliotecas e recursos úteis incorporados que permitem que os desenvolvedores construam aplicações de forma mais rápida e eficiente.

Fuctura

## DJANGO

---

Fuctura



**FRAMEWORK**

GIL



## DJANGO

---

- **Batteries Included:** Funcionalidades integradas para desenvolvimento completo.
- **DRY (Don't Repeat Yourself):** Reuso de código e eficiência.
- **Rápido Desenvolvimento:** Criação de aplicativos de forma ágil.
- **Segurança:** Proteções robustas contra vulnerabilidades.
- **Escalabilidade:** Suporte a projetos de todos os tamanhos.
- **Modularidade:** Uso de componentes necessários apenas.
- **Clean Design:** Código organizado e legível.
- **Django apenas para backend, e para backend e frontend?**

"Framework web para perfeccionistas com pequenos prazos"



## DJANGO - WEB

---



## DJANGO

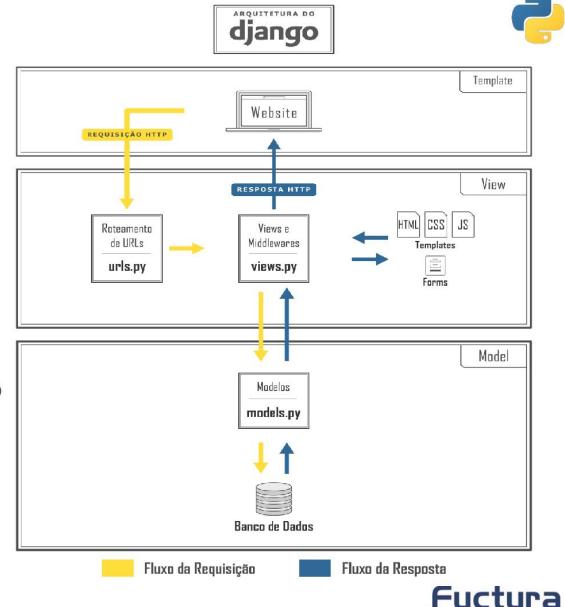


O Django é dividido em **três camadas**:

- A Camada de **Models**.
- A Camada de **Views**.
- A Camada de **Templates**.

**MVC:** O Controller gerencia a interação entre o Model e a View, processando entradas do usuário.

**MVT:** A View trata a lógica de negócios, enquanto os Templates cuidam da apresentação dos dados ao usuário.



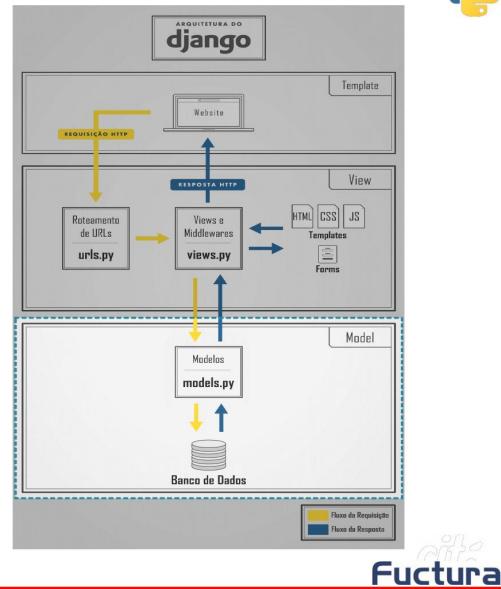
## DJANGO



### Model

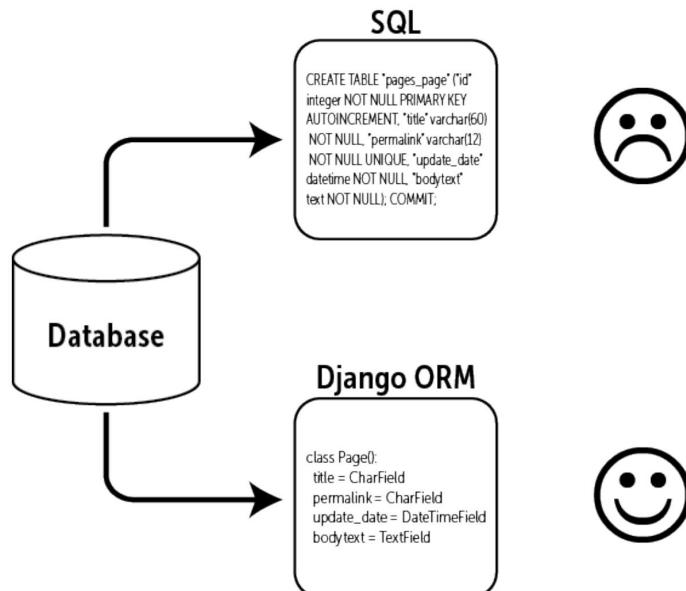
O Model representa os dados do aplicativo e define como eles são armazenados no [banco de dados](#). O Django usa o ORM (Object-Relational Mapping) para mapear os objetos Python para tabelas no banco de dados.

Por baixo dos panos é usada a linguagem SQL, mas com o ORM fica tudo mais simples.



## DJANGO

---



an  
Futura

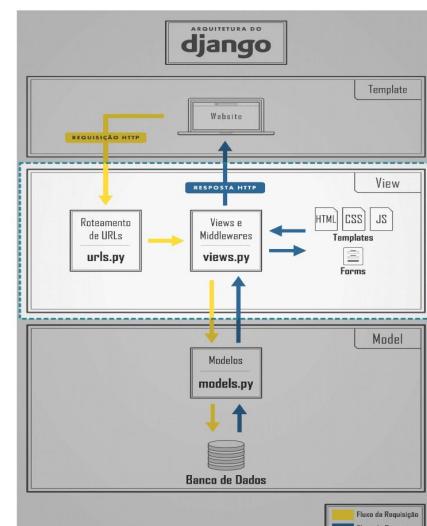
## DJANGO

---

### View

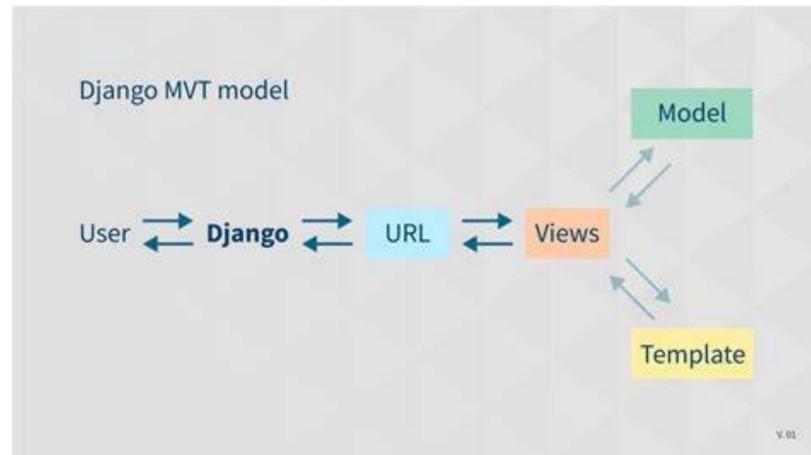
A View é responsável por processar as solicitações do cliente e retornar uma resposta. A View processa as informações que foram enviadas pelo usuário e as envia para o Model para que as informações possam ser armazenadas no banco de dados.

Aqui é onde vamos usar e abusar de orientação a objetos.



an  
Futura

## DJANGO



Info Fuctura

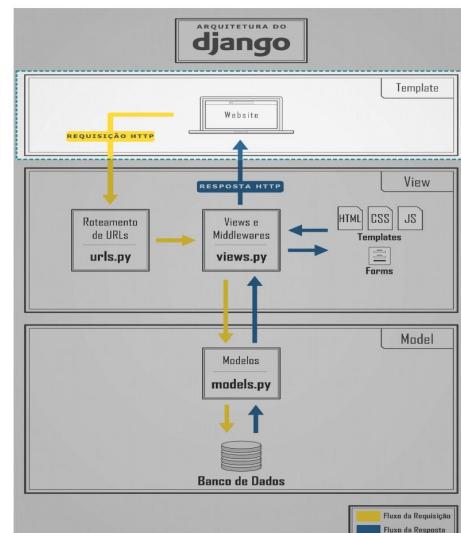
## DJANGO



### Templates

O Template é responsável por renderizar os dados para o usuário. Ele define a aparência do aplicativo e como as informações são exibidas ao usuário.

Nessa parte é onde vamos utilizar os códigos HTML.



Info Fuctura



## DJANGO

---

No decorrer do módulo, nós vamos desbravar as possibilidades do Django. Nesse momento, vamos conhecer mais um pouco sobre o SQL. Como já foi falado, o Django utiliza o SQL, mas de uma forma ‘[indireta](#)’. Tudo é feito de forma muito simples, e não voltaremos a utilizar os códigos SQL durante o curso. Porém, é de extrema importância o [conhecimento](#) e é super indicado um posterior [aprofundamento](#) nesse assunto!



## SQL

---



SQL (Structured Query Language) é uma linguagem de programação utilizada para gerenciar e manipular dados em bancos de dados [relacionais](#). Com o SQL, é possível criar, modificar e consultar tabelas e registros em um banco de dados, além de definir restrições de integridade para garantir a consistência dos dados. O SQL é uma linguagem padronizada e é suportada por vários [sistemas de gerenciamento de bancos de dados](#), como Oracle, MySQL, SQL Server e PostgreSQL.





## SQLite

---

O SQLite é uma biblioteca de banco de dados SQL de código aberto, que oferece uma solução rápida e fácil para armazenar dados em aplicativos.

É extremamente leve e não requer um servidor dedicado para funcionar, pois todo o banco de dados é armazenado em um **único arquivo**.

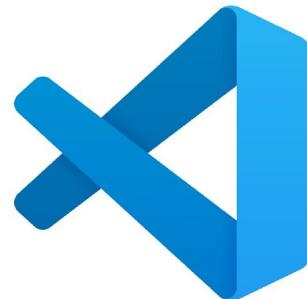
O Python possui uma biblioteca padrão integrada chamada **sqlite3**, que fornece um conjunto completo de funções para trabalhar com o SQLite.



---

IDE que vamos usar!

---



[Visual Studio Code](#)

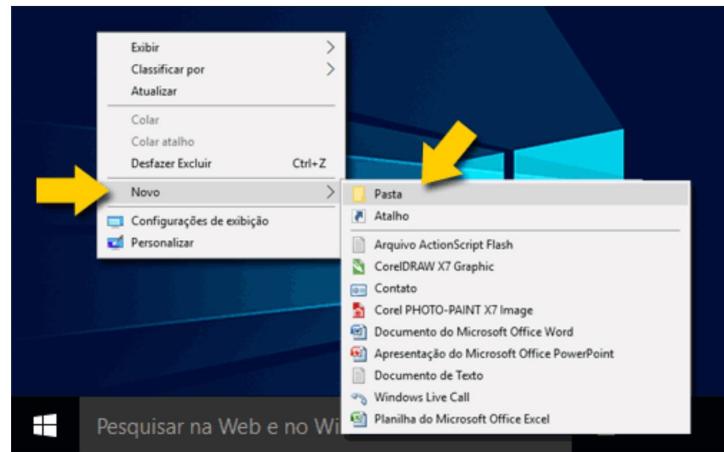




## Importante!!



Sempre que formos iniciar um novo projeto, É importante criarmos uma nova pasta. Dessa forma, nosso projeto fica 'limpo'.

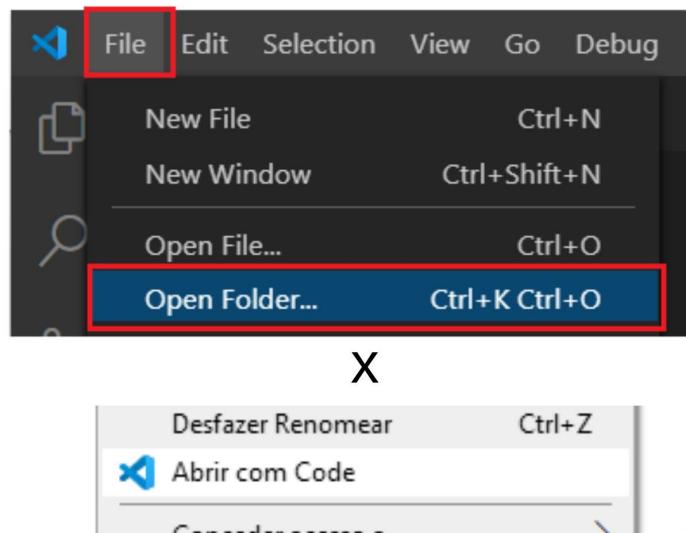


Fuctura

## Importante!!

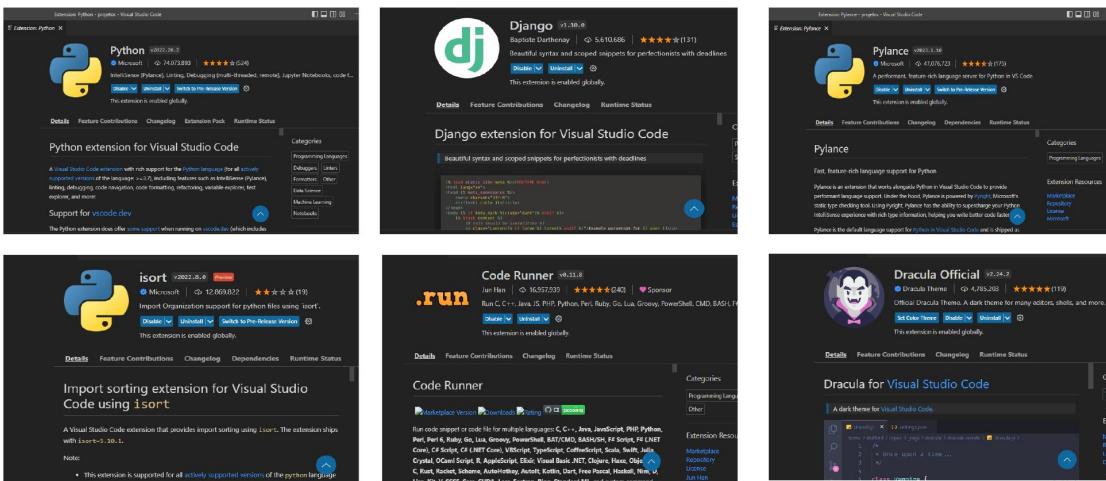


O próximo passo, é abrir o vs code e localizar a pasta que iremos trabalhar. Vamos em 'file', depois clicamos em 'open folder' e procuramos nossa pasta. Ou Clicar com o botão direito dentro da pasta e selecionar 'Abrir com Code'



Fuctura

# Extensões

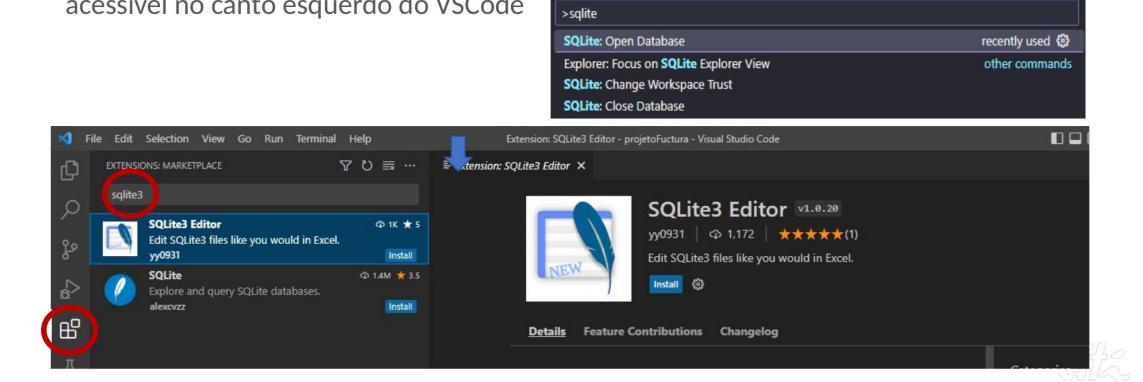


Fuctura

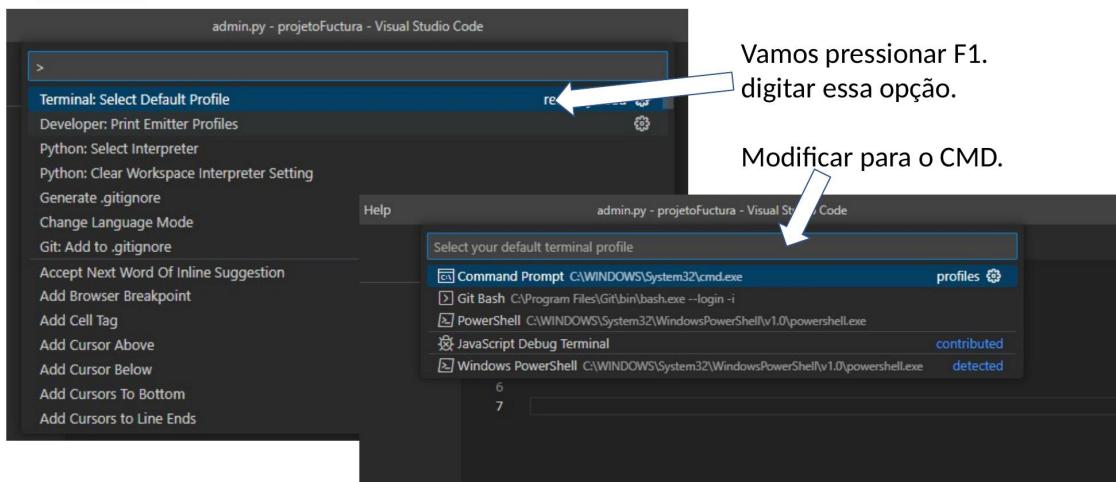
## Django- Visualização de dados no SQLite



Para visualizar nossos bancos, podemos utilizar a extensão do Vs Code: **SQLite3 Editor**. Após instalar execute CTRL + SHIFT + P e digite sqlite. Selecione OpenDatabase e escolha o banco criado na aula (myDatabase) ele ficara visivel e acessivel no canto esquerdo do VSCode

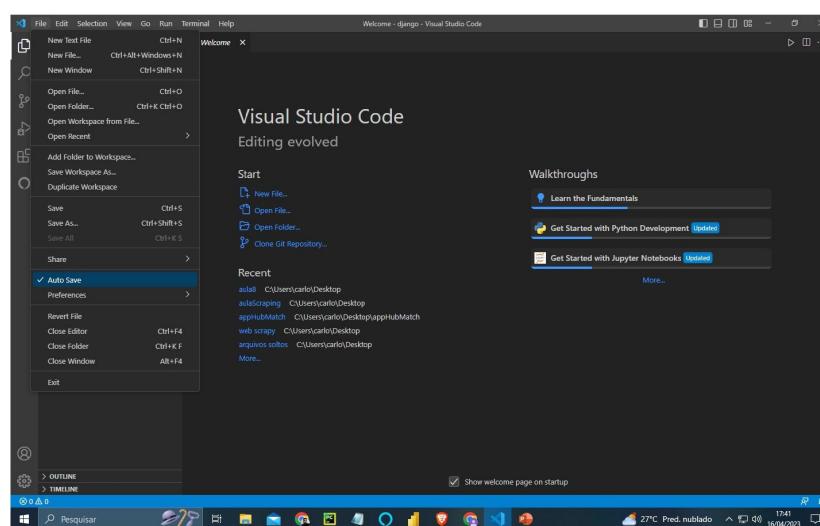


## CMD



Fuctura

## Auto Save - Opcional



Fuctura

Hora de codar!

---

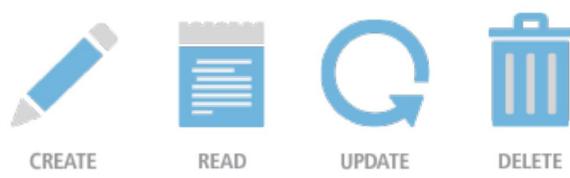


---

info  
Fuctura

**CRUD**

---



C R U D

---

info  
Fuctura



## Criando banco e tabela

```
File Edit Selection View Go Run Terminal Help
OPEN EDITORS
ARQUIVOS ...
    > __pycache__
        arg2.py
        aula3.py
    mydatabase.db
    rest.py
    SqliteConec.py
    testeBl.pbix
EXPLORER
SqliteConec.py X
SqliteConec.py > ...
1 import sqlite3
2
3 # Criar uma conexão com o banco de dados
4 conn = sqlite3.connect('mydatabase.db')
5
6 # Criar uma tabela
7 conn.execute('''CREATE TABLE stocks (date text, trans text, symbol text, qty real, price real)''')
8
9 # Salvar as alterações
10 conn.commit()
11
12 # Fechar a conexão com o banco de dados
13 conn.close()
```

Este código cria um banco de dados chamado 'mydatabase.db' e uma tabela chamada 'stocks' com as colunas 'date', 'act', 'symbol', 'qty' e 'price'. O comando '`conn.commit()`' é usado para salvar as alterações na tabela e o '`conn.close()`' é usado para fechar a conexão com o banco de dados.



## Inserindo dados

Agora que o **banco de dados** e a **tabela** foram criados, podemos adicionar dados a ela. Crie um outro arquivo e escreve o código a seguir:

```
File Edit Selection View Go Run Terminal Help
OPEN EDITORS
ARQUIVOS SOLTOS
    > __pycache__
        arg2.py
        aula3.py
    mydatabase.db
    rest.py
    SqliteConec.py
    SqliteInser.py
    testeBl.pbix
EXPLORER
SqliteConec.py SqliteInser.py X
SqliteInser.py > ...
1 import sqlite3
2
3 # Criar uma conexão com o banco de dados
4 conn = sqlite3.connect('mydatabase.db')
5
6 # Inserir dados na tabela
7 conn.execute("INSERT INTO stocks VALUES ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)")
8 conn.execute("INSERT INTO stocks VALUES ('2006-03-28', 'BUY', 'IBM', 1000, 45.00)")
9 conn.execute("INSERT INTO stocks VALUES ('2006-04-06', 'SELL', 'IBM', 500, 53.00)")
10
11 # Salvar as alterações
12 conn.commit()
13
14 # Fechar a conexão com o banco de dados
15 conn.close()
```

Este código insere dados na tabela 'stocks'. Para fazer isso, é usada a instrução '**INSERT INTO**', seguida pelo nome da tabela e os valores que deseja inserir na tabela.





## Lendo dados

Agora que inserimos dados na tabela, podemos ler esses dados. Crie um novo arquivo, e escreva o código a seguir:



Este código seleciona todos os dados da tabela 'stocks' usando a instrução '**SELECT \***' e armazena os dados em um objeto de cursor. Em seguida, os dados são iterados usando um loop for e impressos na saída padrão.

```
File Edit Selection View Go Run Terminal Help SqliteLer.py - arquivos soltos

OPEN EDITORS
ARQUIVOS SOLTOS
    > _pycache_
        __init__.py
        aula3.py
        mydatabase.db
        rest.py
        SqliteConect.py
        SqliteInserir.py
        SqliteLer.py
        testeBL.xlsx

SqliteLer.py > ...
1 import sqlite3
2
3 # Criar uma conexão com o banco de dados
4 conn = sqlite3.connect('mydatabase.db')
5
6 # Selecionar todos os dados da tabela
7 cursor = conn.execute("SELECT * from stocks")
8
9 # Iterar sobre os dados e exibi-los
10 for row in cursor:
11     print(row)
12
13 # Fechar a conexão com o banco de dados
14 conn.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[Done] exited with code=0 in 0.254 seconds

[Running] python -u "c:\Users\carlo\Desktop\arquivos soltos\SqliteLer.py"
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
('2006-03-28', 'BUY', 'IBM', 1000.0, 45.0)
('2006-04-06', 'SELL', 'IBM', 500.0, 53.0)
```



## Atualizando dados

Também podemos **atualizar dados** na tabela. Crie um novo arquivo e escreva o código a seguir:



Este código atualiza o preço das ações da empresa 'RHAT' para **53.00**.

```
File Edit Selection View Go Run Terminal Help SqliteUpdate.py - arquivos soltos - Visual Studio Code

OPEN EDITORS
ARQUIVOS SOLTOS
    > _pycache_
        __init__.py
        aula3.py
        mydatabase.db
        rest.py
        SqliteConect.py
        SqliteInserir.py
        SqliteLer.py
        SqliteUpdate.py
        testeBL.xlsx

SqliteUpdate.py > ...
1 import sqlite3
2
3 # Criar uma conexão com o banco de dados
4 conn = sqlite3.connect('mydatabase.db')
5
6 # Atualizar dados na tabela
7 conn.execute("UPDATE stocks SET price = 53.00 WHERE symbol = 'RHAT'")
8
9 # Salvar as alterações
10 conn.commit()
11
12 # Fechar a conexão com o banco de dados
13 conn.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[Running] python -u "c:\Users\carlo\Desktop\arquivos soltos\SqliteUpdate.py"
('2006-01-05', 'BUY', 'RHAT', 100.0, 53.0)
('2006-03-28', 'BUY', 'IBM', 1000.0, 45.0)
('2006-04-06', 'SELL', 'IBM', 500.0, 53.0)
```





## Deletando dados

Também podemos deletar dados da tabela. Crie um novo arquivo e escreva o código a seguir:



Este código deleta todas as linhas da tabela '**stocks**' que possuem o símbolo 'IBM'.

```
File Edit Selection View Go Run Terminal Help
SqliteDelete.py - arquivos soltos - Visual Studio Code

EXPLORER
> OPEN EDITORS
> ARQUIVOS SOLTOS
    +_pycache_
        +arp2.py
        +aula3.py
        mydatabase.db
        rest.py
        SqliteConec.py
        SqliteDelete.py <-- selected
        SqliteInser.py
        SqliteLer.py
        SqliteUpdate.py
        testeB1pbix

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[Running] python -u "c:\Users\carlo\Desktop\arquivos soltos\SqliteLer.py"
('2006-01-05', 'BUY', 'RHAT', 100.0, 53.0)
```



## SQLite

- **Criação de banco de dados SQLite3:** Aprendemos a criar um banco de dados SQLite3.
- **Inserção, leitura, atualização e exclusão de dados:** Operações básicas para manipulação de dados nas tabelas.
- **SQLite:** Ótima opção para armazenamento de dados simples em aplicativos Python.
- **Biblioteca sqlite3:** Facilita o trabalho com bancos de dados SQLite diretamente no código Python.





## JOIN

Uma das **principais funcionalidades** dos bancos de dados relacionais é a habilidade de **unir** informações de diferentes tabelas através do comando **JOIN**. O SQLite3 suporta diferentes tipos de JOIN, incluindo **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, e **OUTER JOIN**.

Para demonstrar o uso do JOIN no SQLite3 com Python, vamos criar uma chamada '**companies**', que irá armazenar informações sobre as empresas cujas ações estamos acompanhando. Crie um novo arquivo e escreva o código a seguir:

```
File Edit Selection View Go Run Terminal Help SqliteUltraTabela.py - arquivos soltos - Visual Studio Code
OPEN EDITORS
ARQUIVOS SOLTOS
mydatabase.db
SqliteConect.py
SqliteDelete.py
SqliteInseri.py
SqliteLeri.py
SqliteOutraTabela.py
SqliteUpdate.py
SqliteUltraTabela.py
mydatabase.db
1 import sqlite3
2
3 # Criar uma conexão com o banco de dados
4 conn = sqlite3.connect('mydatabase.db')
5
6 # Criar a tabela 'companies'
7 conn.execute('''CREATE TABLE companies (symbol text, name text, sector text)''')
8
9 # Inserir dados na tabela 'companies'
10 conn.execute("INSERT INTO companies VALUES ('RHAT','Red Hat Inc.','Software')")
11 conn.execute("INSERT INTO companies VALUES ('IBM','International Business Machines Corp.','Technology')")
12 conn.execute("INSERT INTO companies VALUES ('AAPL','Apple Inc.','Technology')")
13
14 # Salvar as alterações
15 conn.commit()
16
17 # Fechar a conexão com o banco de dados
18 conn.close()
19
```



## JOIN

Agora que a tabela '**companies**' foi criada e populada com algumas informações, podemos usar o JOIN para unir as informações das tabelas '**stocks**' e '**companies**' e obter uma visão mais completa dos dados.

Adicione o seguinte código para fazer um **INNER JOIN** entre as tabelas '**stocks**' e '**companies**':

```
File Edit Selection View Go Run Terminal Help SqliteInner.py - arquivos soltos - Visual Studio Code
OPEN EDITORS
ARQUIVOS SOLTOS
mydatabase.db
SqliteConect.py
SqliteDelete.py
SqliteInseri.py
SqliteLeri.py
SqliteOutraTabela.py
SqliteUpdate.py
SqliteUltraTabela.py
SqliteInner.py
mydatabase.db
1 import sqlite3
2
3 # Criar uma conexão com o banco de dados
4 conn = sqlite3.connect('mydatabase.db')
5
6 # Selecionam os dados de stocks e companies usando INNER JOIN
7 cursor = conn.execute("SELECT stocks.date, stocks.trans, stocks.qty, stocks.price, companies.name \
8 FROM stocks \
9 INNER JOIN companies ON stocks.symbol=companies.symbol")
10
11 # Iterar sobre os dados e exibi-los
12 for row in cursor:
13     print(row)
14
15 # Fechar a conexão com o banco de dados
16 conn.close()
17
```

[Done] exited with code=0 in 0.17 seconds

```
[Running] python -u "c:\Users\carlo\Desktop\arquivos soltos\SqliteInner.py"
('2006-01-05', 'BUY', 100.0, 53.0, 'Red Hat Inc.')
('2006-01-05', 'BUY', 100.0, 35.14, 'Red Hat Inc.')
('2006-03-28', 'BUY', 1000.0, 45.0, 'International Business Machines Corp.')
('2006-04-06', 'SELL', 500.0, 53.0, 'International Business Machines Corp.')
```





## JOIN

- **Uso do JOIN:** Unir informações de diferentes tabelas em um banco de dados SQLite3 usando Python.
- **Visão completa dos dados:** O JOIN permite integrar dados de várias tabelas, proporcionando uma visão mais abrangente.
- **Análises sofisticadas:** A junção de tabelas facilita a realização de análises mais avançadas sobre os dados armazenados.



## JOIN



- **INNER JOIN:** Retorna apenas as linhas que têm correspondência em ambas as tabelas com base em uma coluna em comum.
- **LEFT JOIN:** Retorna todas as linhas da tabela da esquerda e as correspondentes da tabela da direita. Se não houver correspondência, as colunas da tabela da direita terão valores nulos.
- **RIGHT JOIN:** Similar ao LEFT JOIN, mas retorna todas as linhas da tabela da direita e as correspondentes da tabela da esquerda. Se não houver correspondência, as colunas da tabela da esquerda terão valores nulos.
- **OUTER JOIN:** Refere-se a JOINs que retornam linhas de ambas as tabelas, independentemente de haver correspondência. Existem dois tipos principais: LEFT OUTER JOIN e RIGHT OUTER JOIN.





## EXERCÍCIO!

### LIVRO x ESCRITOR x EDITORA

**Tabela Livro:** id = Chave primária, titulo = Nome do livro,  
escritor\_id = Chave estrangeira para a tabela **Escritor**,  
editora\_id = Chave estrangeira para a tabela **Editora**.

**Tabela Escritor:** id = Chave primária, nome = Nome do escritor.

**Tabela Editora:** id = Chave primária, nome = Nome da editora.

- Crie um INNER JOIN com 3 novas tabelas mais tabelas
- Use as mesmas tabelas para fazer um LEFT JOIN, RIGHT JOIN, e um OUTER JOIN



## Ambientes virtuais



- **Ambiente virtual:** Uma "pasta" isolada no computador onde o conteúdo não afeta o restante dos programas e vice-versa.
- **Isolamento de dependências:** Ferramentas e bibliotecas usadas no ambiente virtual são isoladas do sistema principal.
- **Módulo venv:** Utilizado para isolar dependências de projetos, permitindo que cada projeto tenha suas próprias bibliotecas separadas.
- **Gestão de versões:** Facilita a utilização de diferentes versões de ferramentas (ex: Django 2.1 e a mais recente) sem conflitos entre projetos.
- **Replicação do ambiente:** Fácil de replicar o mesmo ambiente virtual em outros computadores.



## ▼ Exercícios De logica de Programação

### 1. Calculadora de Juros Simples

- Solicite ao usuário o valor de um empréstimo, a taxa de juros anual e o período de tempo (em anos). Calcule e imprima o valor dos juros simples e o valor total a ser pago no final.

### 2. Verificação de Estoque em Loja

- Crie um programa que gerencie o estoque de uma loja. O programa deve permitir adicionar ou retirar produtos, além de verificar a quantidade atual de determinado produto. Use um dicionário para armazenar os produtos e suas quantidades.

### **3. Cálculo de Despesas de Viagem**

- Solicite ao usuário o custo diário de uma viagem (hospedagem, alimentação, transporte, etc.) e o número de dias que ele pretende viajar. Calcule o valor total da viagem.

### **4. Conversor de Temperatura (Fahrenheit para Celsius)**

- Solicite uma temperatura em Fahrenheit e converta para Celsius. A fórmula para conversão é:  $C = \frac{F - 32}{9}$ .

$$C = \frac{F - 32}{9}$$

### **5. Simulador de Saldo Bancário**

- Crie um programa que gerencie o saldo de uma conta bancária. O usuário pode fazer depósitos e saques, e o programa deve exibir o saldo atual após cada operação.

### **6. Calculadora de Média Ponderada**

- Solicite ao usuário as notas de três provas e seus respectivos pesos. Calcule a média ponderada das notas e informe se o aluno foi aprovado ( $\text{média} \geq 7$ ), reprovado ( $\text{média} < 4$ ) ou em recuperação ( $\text{média entre } 4 \text{ e } 7$ ).

### **7. Simulador de Compra com Parcelamento**

- Solicite o valor de uma compra e a quantidade de parcelas. Calcule e mostre o valor da parcela mensal, considerando um juros de 5% ao mês sobre o valor total.

### **8. Cálculo de Imposto sobre a Renda**

- Solicite ao usuário o salário mensal e calcule o imposto de renda baseado nas faixas: até R\$ 1.500 (isento), de R\$ 1.501 a R\$ 3.000 (15%) e acima de R\$ 3.000 (25%).

### **9. Contagem de Caractere em um Arquivo**

- Crie um programa que leia um arquivo de texto e conte quantas vezes uma palavra ou caractere específico aparece nele.

## **10. Verificador de Idade para Venda de Álcool**

- Crie um programa que verifique se uma pessoa pode comprar bebida alcoólica em um estabelecimento, solicitando sua idade. O programa deve permitir apenas a venda para pessoas com 18 anos ou mais.

## **11. Calculadora de Valor de Conta de Restaurante**

- Solicite ao usuário o valor de uma refeição e calcule a gorjeta (10%) e o valor total a ser pago.

## **12. Calculadora de Retorno de Investimento**

- Solicite ao usuário o valor investido, a taxa de retorno anual e o tempo (em anos) do investimento. Calcule o valor final do investimento com juros compostos.

## **13. Simulador de Carregamento de Celular**

- Crie um programa que simule o tempo de carregamento de um celular. Solicite ao usuário a capacidade da bateria e o tempo que o celular está carregando. O programa deve calcular a porcentagem de carga que foi adicionada ao longo do tempo.

## **14. Simulador de Cartão de Crédito**

- Crie um programa que simule o pagamento de uma fatura de cartão de crédito. Solicite o valor da fatura, o valor pago e calcule o saldo devedor com a aplicação de uma taxa de juros de 2% ao mês.

## **15. Calculadora de Custo de Viagem por Carro**

- Solicite a distância da viagem, o consumo de combustível do carro (km por litro) e o preço do combustível. Calcule o custo total da viagem.

## **16. Conversor de Quilos para Libras**

- Solicite ao usuário um valor em quilogramas e converta para libras ( $1\text{ kg} = 2.20462\text{ libras}$ ).

## **17. Simulador de Compra Online**

- Crie um programa que simule a compra de produtos online. Solicite o preço de vários produtos e o número de itens a ser comprado. No

final, calcule o valor total da compra com um desconto de 5% para compras acima de R\$ 300.

#### **18. Simulador de Taxa de Câmbio**

- Solicite ao usuário um valor em reais e a taxa de câmbio para dólares. Converta o valor em reais para dólares e imprima o valor convertido.

#### **19 Simulador de Imóvel para Aluguel**

- Solicite ao usuário o valor do aluguel mensal e o tempo de contrato. Calcule o valor total a ser pago ao longo do contrato, considerando que um aluguel de 12 meses tem 5% de desconto no valor total.

#### **20. Simulador de Desconto por Fidelidade**

- Crie um programa para um supermercado que aplique um desconto de 10% em compras acima de R\$ 100 para clientes fidelidade. Solicite o valor da compra e informe se o cliente tem direito ao desconto ou não.

#### **21. Calculadora de Salário**

- Solicite o salário bruto de um empregado e calcule o salário líquido após descontos de INSS (8%) e IR (15%). Imprima o valor do salário líquido.

#### **22. Calculadora de IPVA**

- Solicite ao usuário o valor de um veículo e calcule o valor do IPVA (5% do valor do veículo). Imprima o valor a ser pago.

#### **23. Simulador de Despesas Mensais**

- Crie um programa que receba a lista de despesas mensais de um usuário (aluguel, alimentação, transporte, etc.) e calcule o total gasto no mês. Se o total ultrapassar um limite definido, imprima uma mensagem de alerta.

#### **24. Analisador de Nota de Aluno**

- Solicite ao usuário as notas de um aluno (duas notas) e calcule a média. Em seguida, imprima se o aluno foi aprovado (média  $\geq 7$ ), reprovado (média  $< 4$ ) ou em recuperação (média entre 4 e 7).

#### **25. Simulador de Compras no Supermercado**

- Crie um programa que simule compras em um supermercado. Solicite o nome e o preço de vários produtos e calcule o valor total da compra. Se o total ultrapassar R\$ 200, aplique um desconto de 10%.

## 26. Conversor de Quilômetro para Milha

- Solicite ao usuário um valor em quilômetros e converta para milhas (1 quilômetro = 0.621371 milhas).

## 27. Simulador de Divisão de Conta entre Amigos

- Crie um programa que receba o valor total de uma conta de restaurante e dívida entre os amigos. Solicite ao usuário o número de amigos e calcule o valor que cada um deverá pagar.

## 28. Calculadora de Tempo de Viagem

- Solicite a distância em quilômetros de uma viagem e a velocidade média do veículo. Calcule o tempo estimado da viagem (tempo = distância / velocidade).

## 29. Controle de Temperatura

- Solicite a temperatura em Celsius e converta para Fahrenheit ( $F = (C * 9/5) + 32$ ). Imprima o valor convertido.

## 30. Verificador de Horário de Funcionamento

- Crie um programa que verifique se um estabelecimento está aberto ou fechado com base no horário. O programa deve perguntar o horário atual (ex: 14:00) e verificar se o horário está dentro do intervalo de funcionamento (ex: 09:00 - 18:00).

## 31. Organizador de Compras Online

- Crie um programa que organize uma lista de compras, onde o usuário pode adicionar ou remover itens. O programa deve permitir que o usuário veja todos os itens na lista, além de mostrar o total de itens.

## 32. Controle de Temperatura de Ar Condicionado

- Crie um programa que receba a temperatura desejada do ar condicionado e o valor atual. Caso o valor atual seja maior do que o desejado, o ar-condicionado deve esfriar; se for menor, deve aquecer. Imprima a ação do ar-condicionado.

### **33. Calculadora de Área de Terreno**

- Solicite as medidas de um terreno (largura e comprimento) e calcule sua área. Caso a área seja maior do que  $500\text{m}^2$ , imprima "Terreno Grande"; caso contrário, imprima "Terreno Pequeno".

### **34. Calculadora de Rendimento de Investimento**

- Solicite ao usuário o valor investido e a taxa de rendimento mensal de um investimento. Calcule o valor após 6 meses e informe ao usuário.

### **35. Cálculo de Quantidade de Tinta**

- Solicite ao usuário as dimensões de uma parede (largura e altura) e calcule a quantidade de tinta necessária para pintar a parede (considerando que 1 litro de tinta cobre  $10\text{m}^2$ ).

### **36. Cálculo de Preço de Ingresso para Cinema**

- Solicite a idade do usuário e informe o preço do ingresso de cinema: R\$ 10 para crianças até 12 anos, R\$ 20 para adultos de 13 a 60 anos e R\$ 15 para idosos acima de 60 anos.

### **37. Simulador de Pontuação de Jogo**

- Crie um programa que calcule a pontuação de um jogador em um jogo. O jogador começa com 0 pontos, ganha 10 pontos por vitória e perde 5 pontos por derrota. O programa deve perguntar ao usuário o número de vitórias e derrotas, e calcular a pontuação final.

### **38. Cálculo de Aposentadoria**

- Solicite o ano de nascimento e o ano atual de um usuário. Calcule a idade do usuário e verifique se ele já pode se aposentar (idade  $\geq 65$  anos).

### **39. Simulador de Atendimento ao Cliente**

- Crie um sistema simples de atendimento ao cliente que permita que o cliente faça uma reclamação, solicite uma informação ou solicite um serviço. O programa deve pedir o tipo de serviço solicitado e imprimir a resposta correspondente (por exemplo, "Agradecemos pela reclamação, estamos trabalhando para resolver" ou "Informações sobre o serviço estarão disponíveis em breve").

#### **40. Simulador de Calculadora de Frete**

- Solicite o peso de uma encomenda e calcule o valor do frete. Se o peso for inferior a 1 kg, o frete será de R\$ 5,00; de 1 kg a 5 kg será R\$ 10,00; de 5 kg a 10 kg será R\$ 15,00; e acima de 10 kg será R\$ 20,00. Imprima o valor do frete.

#### **41. Controle de Estoque de Produtos**

- Crie um programa que simule o controle de estoque de uma loja. O programa deve permitir ao usuário adicionar ou remover produtos e mostrar a quantidade atual de cada produto. A quantidade de estoque de cada produto deve ser armazenada em um dicionário (produto: quantidade).

#### **42. Calculadora de Desconto**

- Solicite ao usuário o valor de uma compra e o percentual de desconto que ele pode aplicar. Calcule e imprima o valor final da compra com o desconto aplicado.

#### **43. Verificador de Vencimento de Conta**

- Dada uma data de vencimento de uma conta (formato dd/mm/aaaa), crie um programa que verifique se a conta está vencida ou não, considerando a data atual. O programa deve imprimir uma mensagem informando se a conta está vencida ou ainda dentro do prazo.

#### **44. Conversor de moeda**

- Crie um programa que converta um valor em reais para outra moeda (como dólar ou euro). O valor de conversão deve ser fornecido pelo usuário. O programa deve calcular e mostrar o valor convertido.

#### **45. Simulador de Empréstimo**

- Solicite ao usuário o valor do empréstimo, a taxa de juros mensal e o número de parcelas. Calcule e mostre o valor da parcela mensal utilizando a fórmula de juros compostos. (Para simplificar, pode-se usar a fórmula do cálculo da parcela fixa:  $P=PV \times i / (1 + i)^n$ , onde PV é o valor do empréstimo, i é a taxa de juros mensal e n é o número de parcelas).

$$P = PV \times i / (1 + i)^n$$

PVPV

ii

nn

#### **46. Calculadora de Consumo de Combustível**

- O programa deve solicitar ao usuário a distância percorrida em quilômetros e o consumo de combustível (em litros). Em seguida, deve calcular e exibir o consumo de combustível por quilômetro.

#### **47. Verificador de Idade para Entrada em Evento**

- Crie um programa que verifique se a pessoa tem idade suficiente para entrar em um evento com restrição de idade. Solicite a idade da pessoa e informe se ela pode ou não entrar no evento (idade mínima de 18 anos, por exemplo).

#### **48. Organização de Lista de Contatos**

- Solicite ao usuário o nome e o telefone de várias pessoas. Armazene as informações em uma lista de tuplas (nome, telefone). Ao final, permita que o usuário busque um contato pelo nome e imprima o telefone correspondente. Se o nome não existir, o programa deve informar ao usuário.

#### **49. Cálculo de Impostos**

- Solicite ao usuário o valor de uma compra e calcule o imposto a ser pago com base em uma alíquota de 10%. O programa deve calcular o valor do imposto e o valor final da compra (valor original + imposto).

#### **50. Simulador de Caixa Eletrônico**

- Crie um programa que simule um caixa eletrônico. O programa deve permitir que o usuário insira um valor de saque e, em seguida, informar quantas cédulas de 100, 50, 20, 10, 5 e 1 reais ele receberia. O programa deve ser capaz de informar a quantidade mínima de cédulas necessárias para o saque.

