

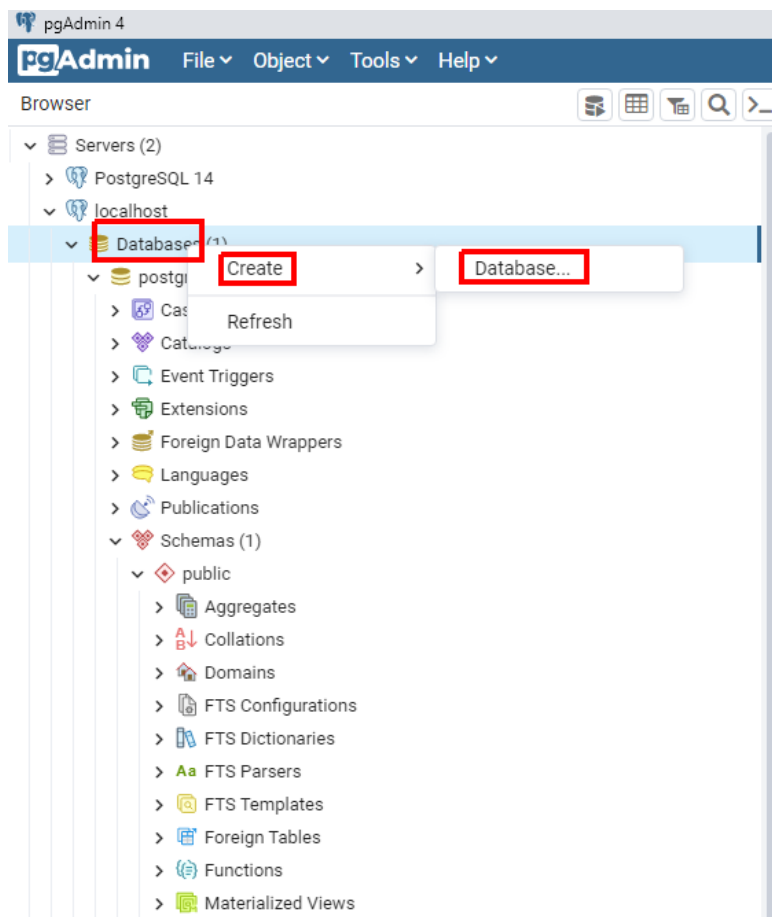
1 Introdução

Neste material desenvolveremos uma aplicação com Python 3 que se conecta a uma instância do PostgreSQL a fim de realizar um “login”, ou seja, verificar se um determinado usuário caracterizado por dados de login e senha informados pelo usuário existe na base.

2 Desenvolvimento

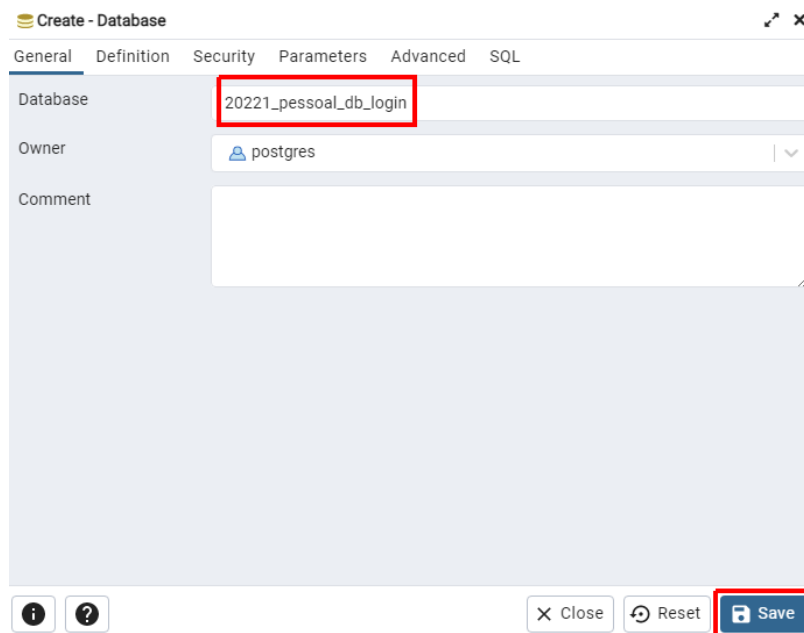
2.1 (Criando um database no PostgreSQL) Comece criando uma base de dados própria para este material. Utilizando o pgAdmin4, você pode clicar com o direito em **Databases** e escolher **Create >> Database...** como na Figura 2.1.1.

Figura 2.1.1



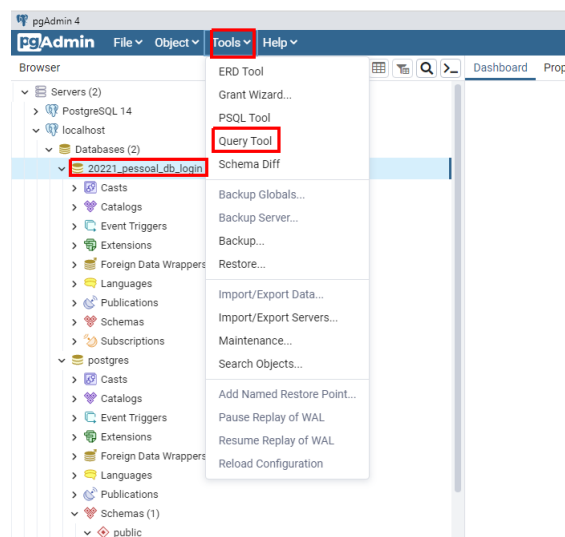
Na tela seguinte, exibida pela Figura 2.2.2, escolha um bom nome para a base e clique em **Save**.

Figura 2.2.2



2.3 (Criando a tabela para dados de usuários) Para criar a tabela que armazenará os dados dos usuários, mantenha a base que acaba de criar “selecionada” no menu à esquerda. Clique em **Tools >> Query Tool** para abrir o editor em que poderá digitar seus comandos SQL. Veja a Figura 2.3.1.

Figura 2.3.1



Crie a tabela e insira alguns usuários utilizando o código exibido pelo Bloco de Código 2.3.1. Por fim, verifique se os dados foram de fato inseridos com o comando SELECT. Use o botão **Execute** para executar. Veja a Figura 2.3.2.

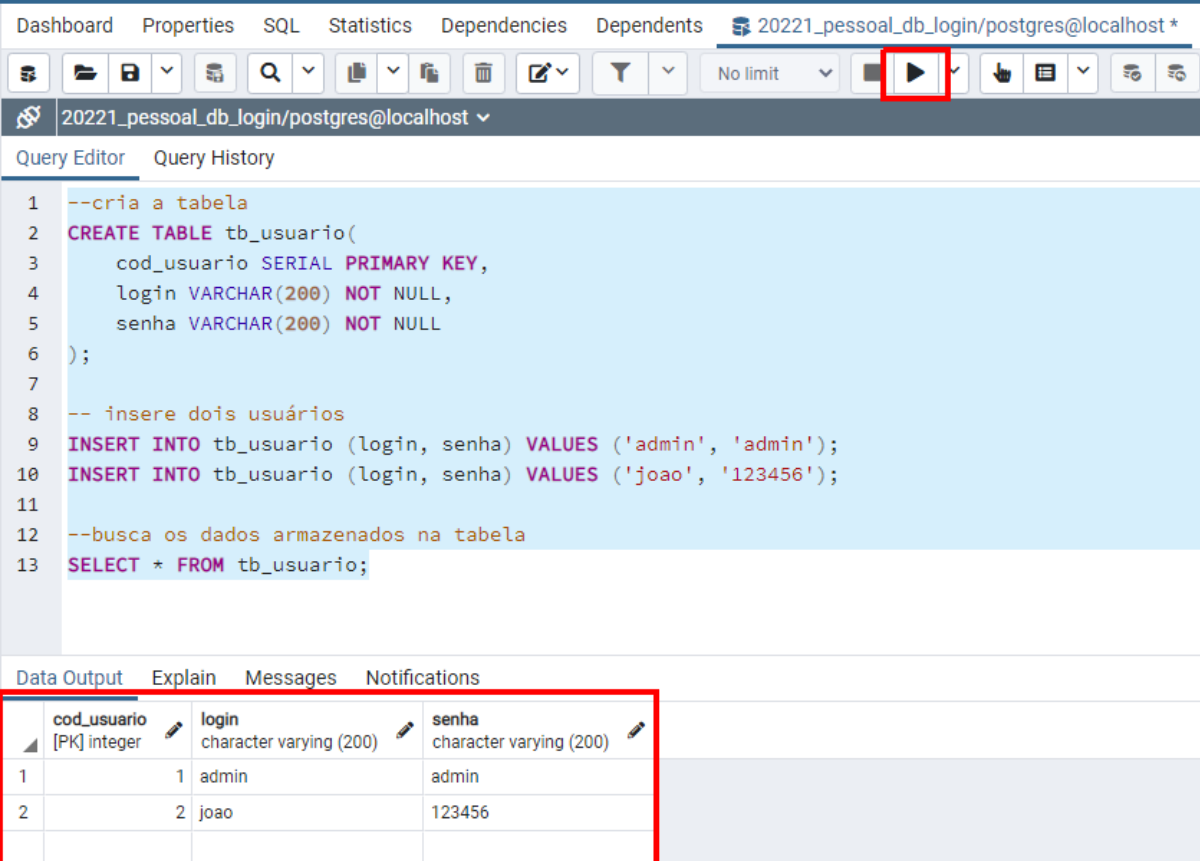
Bloco de Código 2.3.1

```
--cria a tabela
CREATE TABLE tb_usuario(
    cod_usuario SERIAL PRIMARY KEY,
    login VARCHAR(200) NOT NULL,
    senha VARCHAR(200) NOT NULL
);

-- insere dois usuários
INSERT INTO tb_usuario (login, senha) VALUES ('admin', 'admin');
INSERT INTO tb_usuario (login, senha) VALUES ('joao', '123456');

--busca os dados armazenados na tabela
SELECT * FROM tb_usuario;
```

Figura 2.3.2

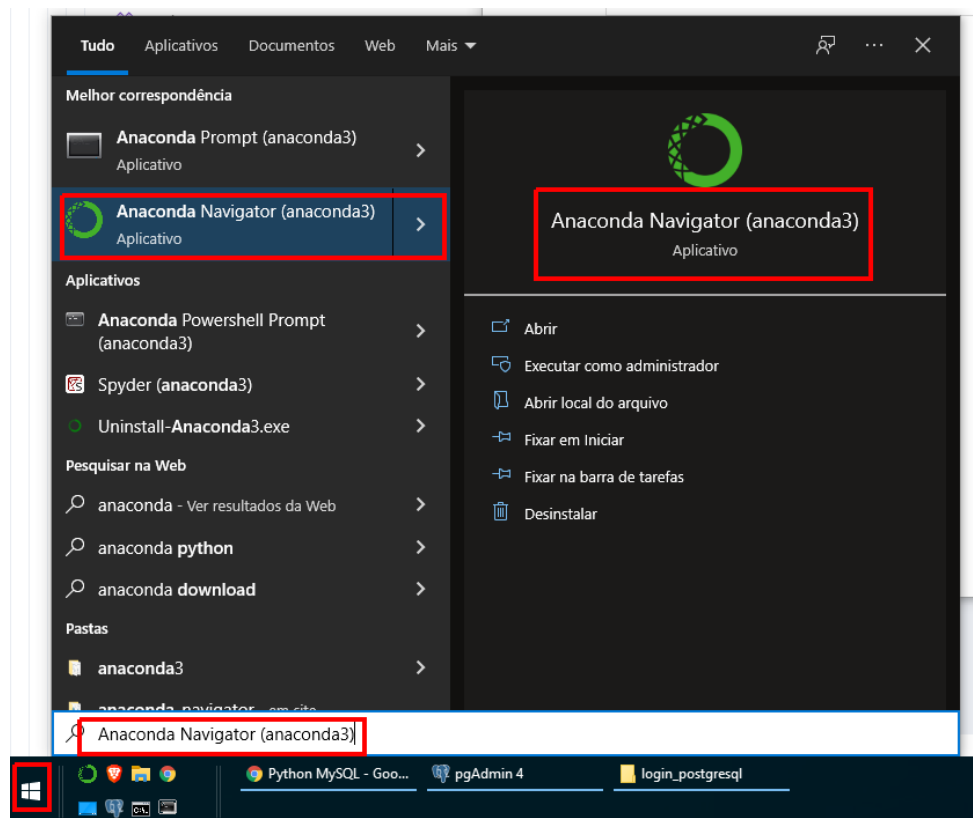


The screenshot shows a database management interface with a top navigation bar (Dashboard, Properties, SQL, Statistics, Dependencies, Dependents) and a connection bar (20221_pessoal_db_login/postgres@localhost *). Below the connection bar is a toolbar with various icons, including a red box highlighting the 'Execute' button (a play icon). The main area is the 'Query Editor', which contains the SQL code from Bloco de Código 2.3.1. Below the editor are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the results of the SELECT query. The table has four columns: an index, 'cod_usuario' (integer, primary key), 'login', and 'senha'. It contains two rows of data: (1, 'admin', 'admin') and (2, 'joao', '123456').

	cod_usuario [PK] integer	login character varying (200)	senha character varying (200)
1	1	admin	admin
2	2	joao	123456

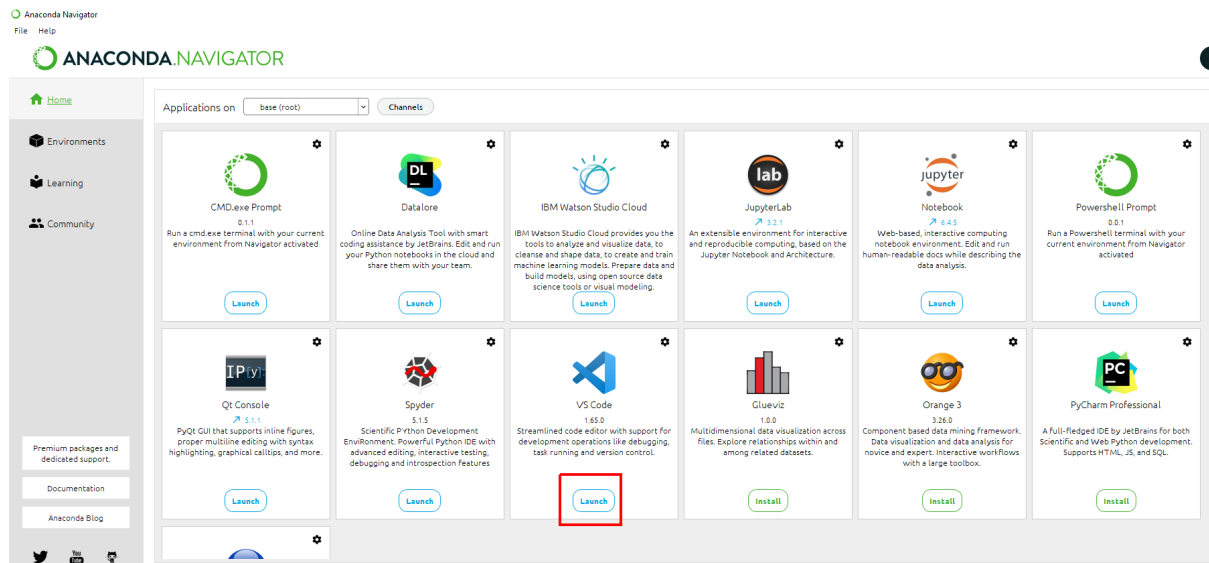
2.4 (Abrindo o VS Code com o Anaconda) Neste material, estamos utilizando o ambiente Anaconda e o IDE VS Code. O próprio Anaconda, em seu processo de instalação, sugere que seus diretórios do tipo “bin” não sejam adicionados à variável de ambiente path do sistema operacional. O procedimento recomendado é abrir as ferramentas desejadas por meio do **Anaconda Navigator**. Ele se encarrega de inicializar um ambiente Python e disponibilizá-lo para os IDEs que forem abertos por meio dele. Por isso, busque por Anaconda Navigator na lista de aplicativos de seu sistema operacional e abra-o, como na Figura 2.4.1.

Figura 2.4.1



Por meio de sua interface gráfica, abra o VS Code, como mostra a Figura 2.4.2.

Figura 2.4.2



Nota. Caso o VS Code ainda não esteja instalado, você deverá ver um botão chamado **Install** no lugar do botão **Launch**. Basta clicar sobre ele para fazer a instalação. A partir daí, você deverá ver a opção **Launch**.

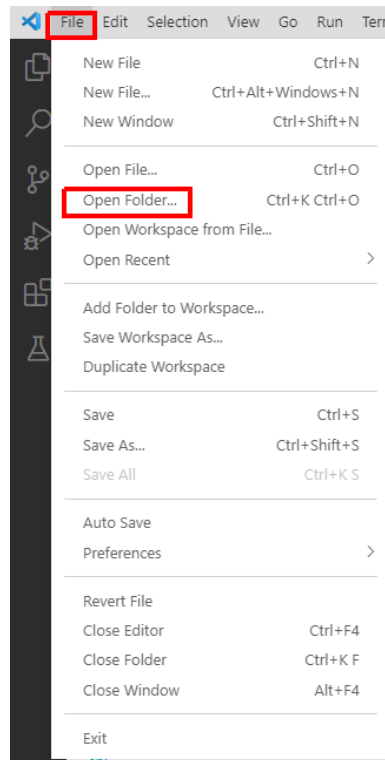
2.5 (Criando uma pasta para o projeto e vinculando o VS Code a ela) É muito importante criar uma pasta em seu sistema de arquivos para abrigar os arquivos referentes a este projeto. Além disso, é muito importante vincular o VS Code a ela uma vez que ela tenha sido criada. Utilize o navegador de arquivos de seu sistema operacional (Windows Explorer, por exemplo: aquela pastinha amarela) e crie uma pasta de acordo com as seguintes boas práticas:

- Não escolha um diretório que possua espaços em branco ou acentos no nome
- Não coloque espaços em branco e/ou acentos nos nomes dos arquivos que criar
- No Windows, para evitar problemas com o mecanismo de permissão do sistema operacional, procure usar uma pasta como

C:\Users\seuUsuario\Documents\meus_projetos_python

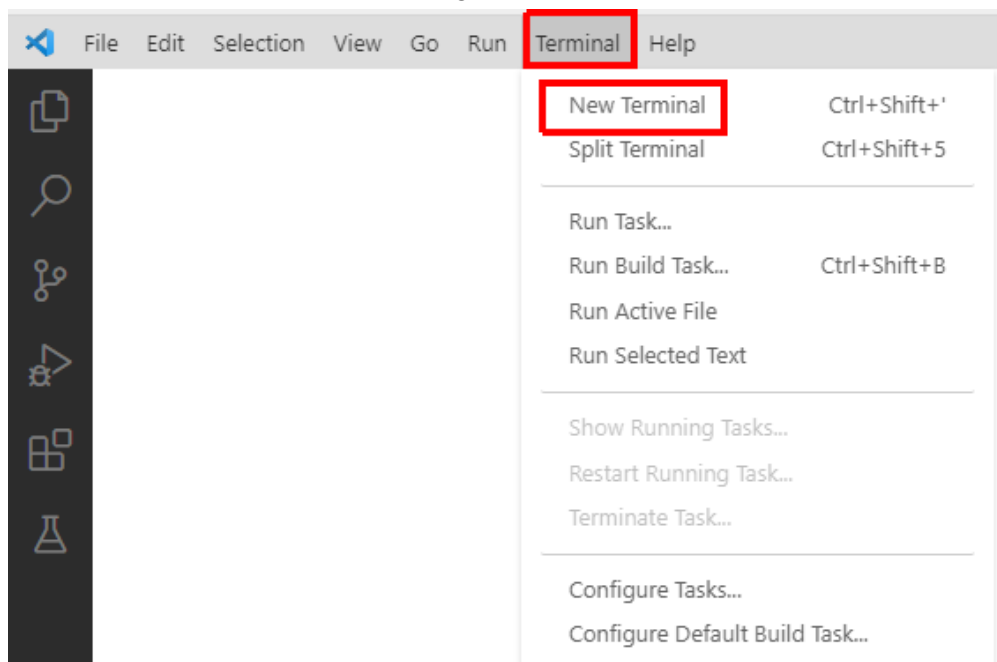
Uma vez que tenha criado a pasta, clique **File >> Open Folder** no VS Code e navegue no sistema de arquivos até encontrá-la para que fiquem vinculados. Veja a Figura 2.5.1.

Figura 2.5.1



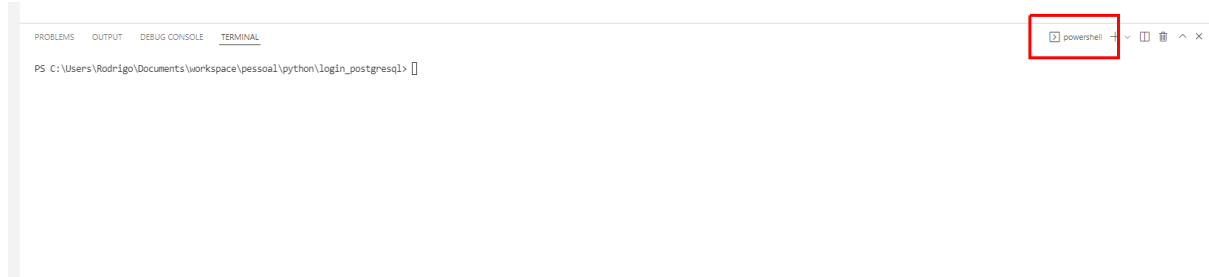
2.6 (Abrindo um terminal interno do VS Code) Os trabalhos poderão ficar mais simples caso utilizemos um terminal interno do VS Code. Para tal, basta clicar **Terminal >> New Terminal**, como mostra a Figura 2.6.1.

Figura 2.6.1



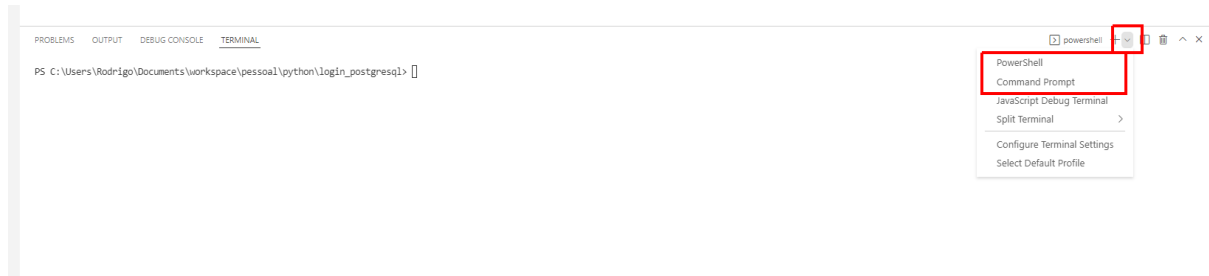
É possível que você obtenha uma instância do **powershell** do Windows ou do prompt de comando comum. Essa informação é importante para o passo seguinte. Verifique qual você está usando no canto inferior direito da tela, como destaca a Figura 2.6.2.

Figura 2.6.2



Observe que há uma opção para que você possa utilizar o que preferir. Veja a Figura 2.6.3.

Figura 2.6.3



2.7 (Diretório “bin” do PostgreSQL na variável de ambiente “path” do sistema operacional) No Windows, o módulo que implementa a especificação de conexão com bancos de dados que utilizaremos depende de um arquivo **.DLL** chamado **libpq.dll**. Ele pode ser encontrado no diretório “bin” do PostgreSQL. Caso ele tenha sido instalado no diretório padrão, o diretório bin pode ser encontrado em

C:\Progra~1\PostgreSQL\14\bin

É preciso colocar esse diretório na variável de ambiente “path” do sistema operacional para que o arquivo **libpq.dll** possa ser encontrado. Isso pode ser feito utilizando-se o terminal do próprio VS Code. É importante estar ciente de que esta configuração será válida apenas para esta instância atual do terminal em uso. Por isso, quando for executar o programa, utilize sempre o terminal em que fez a configuração.

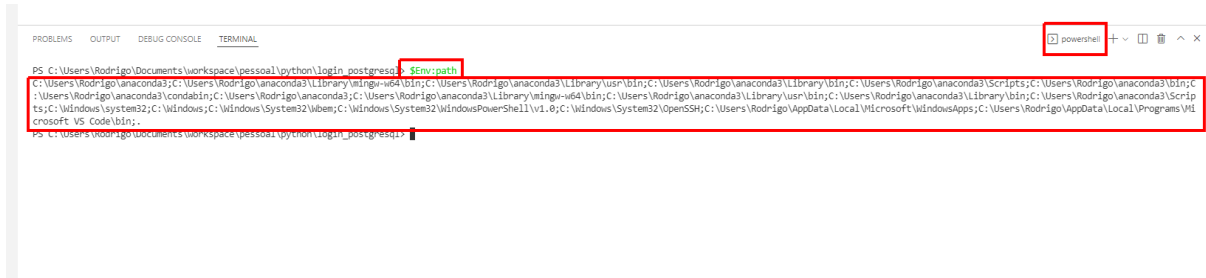
Nota. Caso tenha privilégios de administrador no computador, pode ser interessante fazer a configuração de maneira definitiva.

Se estiver usando o Powershell, use

\$Env:path

para visualizar o que você possui em sua variável de ambiente path no momento. O resultado deve ser parecido com o que exibe a Figura 2.7.1.

Figura 2.7.1



Use

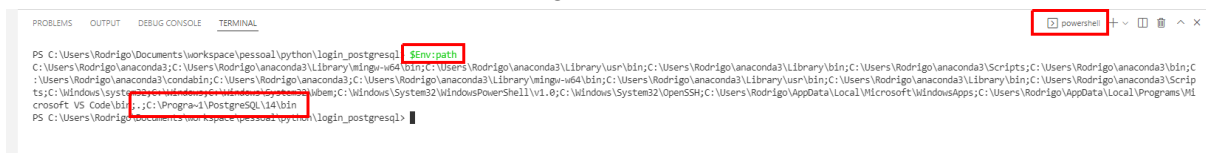
\$Env:path += ';C:\Progra~1\PostgreSQL\14\bin'

para adicionar o diretório **bin** do PostgreSQL à variável de ambiente path do sistema operacional. Use

\$Env:path

novamente para certificar-se de que a configuração foi feita com sucesso. Veja o resultado esperado na Figura 2.7.2.

Figura 2.7.2

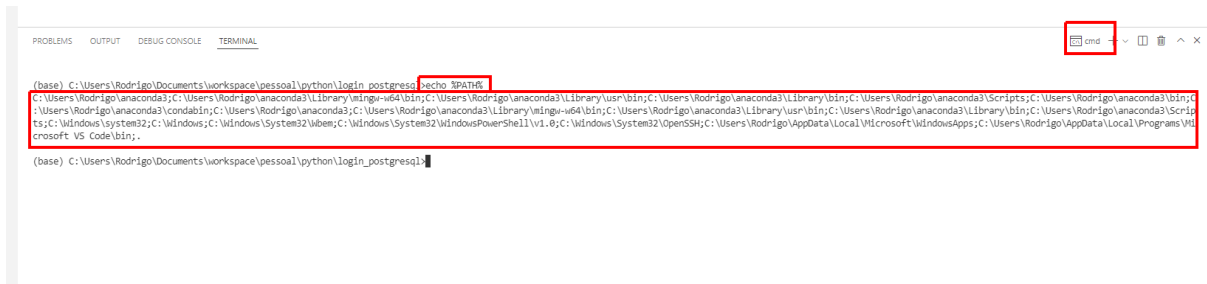


Se estiver usando o prompt de comando comum, use

echo %PATH%

para visualizar o que você possui em sua variável de ambiente path no momento. O resultado deve ser parecido com o que exibe a Figura 2.7.3.

Figura 2.7.3



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cmd
(base) C:\Users\Rodrigo\Documents\workspace\pessoal\python\login_postgres>echo %PATH%
C:\Users\Rodrigo\anaconda3;C:\Users\Rodrigo\anaconda3\Library\mingw-w64\bin;C:\Users\Rodrigo\anaconda3\Library\usr\bin;C:\Users\Rodrigo\anaconda3\Library\bin;C:\Users\Rodrigo\anaconda3\Scripts;C:\Users\Rodrigo\anaconda3\bin;C:\Users\Rodrigo\anaconda3\condabin;C:\Users\Rodrigo\anaconda3;C:\Users\Rodrigo\anaconda3\Library\mingw-w64\bin;C:\Users\Rodrigo\anaconda3\Library\usr\bin;C:\Users\Rodrigo\anaconda3\Library\bin;C:\Users\Rodrigo\anaconda3\Scripts;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Users\Rodrigo\AppData\Local\Microsoft\WindowsApps;C:\Users\Rodrigo\AppData\Local\Programs\Microsoft VS Code\bin;
(base) C:\Users\Rodrigo\Documents\workspace\pessoal\python\login_postgres>
```

Use

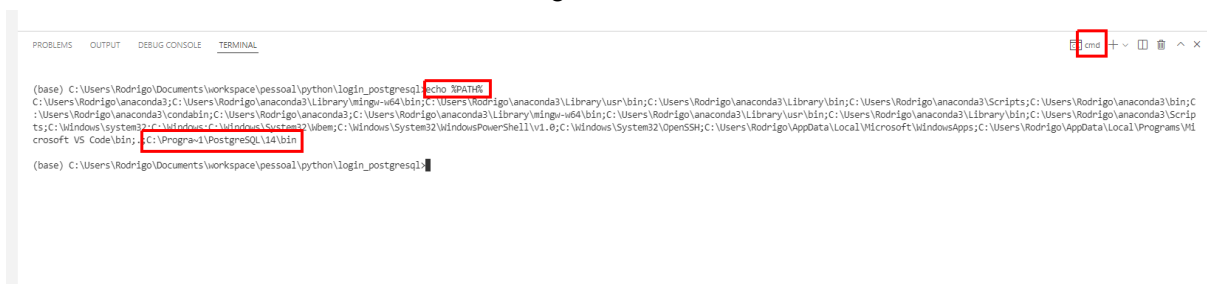
set PATH=%PATH%;C:\Progra~1\PostgreSQL\14\bin

para adicionar o diretório **bin** do PostgreSQL à variável de ambiente path do sistema operacional. Use

echo %PATH%

novamente para certificar-se de que a configuração foi feita com sucesso. Veja o resultado esperado na Figura 2.7.4.

Figura 2.7.4



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cmd
(base) C:\Users\Rodrigo\Documents\workspace\pessoal\python\login_postgres>echo %PATH%
C:\Users\Rodrigo\anaconda3;C:\Users\Rodrigo\anaconda3\Library\mingw-w64\bin;C:\Users\Rodrigo\anaconda3\Library\usr\bin;C:\Users\Rodrigo\anaconda3\Library\bin;C:\Users\Rodrigo\anaconda3\Scripts;C:\Users\Rodrigo\anaconda3\bin;C:\Users\Rodrigo\anaconda3\condabin;C:\Users\Rodrigo\anaconda3;C:\Users\Rodrigo\anaconda3\Library\mingw-w64\bin;C:\Users\Rodrigo\anaconda3\Library\usr\bin;C:\Users\Rodrigo\anaconda3\Library\bin;C:\Users\Rodrigo\anaconda3\Scripts;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Users\Rodrigo\AppData\Local\Microsoft\WindowsApps;C:\Users\Rodrigo\AppData\Local\Programs\Microsoft VS Code\bin;C:\Progra~1\PostgreSQL\14\bin
(base) C:\Users\Rodrigo\Documents\workspace\pessoal\python\login_postgres>
```

A partir de agora, não feche mais o terminal que utilizou para configurar esta variável de ambiente. Caso o faça, será necessário refazer o procedimento utilizando um novo terminal.

2.8 (O módulo psycopg) O acesso a bases de dados relacionais por aplicativos Python é feito utilizando-se mecanismos previstos na especificação “**Python Database API Specification v2.0**”. Há diferentes implementações para esta especificação. Uma das mais comuns e utilizadas se chama **psycopg**. Veja a sua página oficial no Link 2.8.1.

Link 2.8.1

<https://www.psycopg.org/>

Seu uso requer a instalação de um módulo Python. No terminal interno do VS Code, use

pip install psycopg[binary]

para fazer a instalação.

2.9 (Novo arquivo e teste da psycopg) No VS Code Clique em **File >> New File** e crie um arquivo chamado **login.py**. Faça como o Bloco de Código 2.9.1 mostra para verificar se o módulo está configurado corretamente.

Bloco de Código 2.9.1

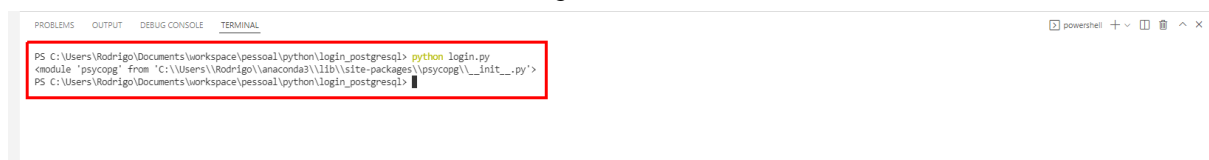
```
import psycopg
print (psycopg)
```

No terminal interno do VS Code - o mesmo em que você configurou a variável de ambiente path - use

python login.py

para testar. O resultado esperado se parece com aquele que a Figura 2.9.1 exhibe.

Figura 2.9.1



2.10 (Classe para representar usuários) Na aplicação Python, usuários serão representados por objetos do tipo “Usuario”, graças a uma classe que definimos, como mostra o Bloco de Código 2.10.1.

Bloco de Código 2.10.1

```
import psycopg
print (psycopg)
class Usuario:
    def __init__(self, login, senha):
        self.login = login
        self.senha = senha
```

2.11 (Método que abre conexão com o PostgreSQL e verifica se usuário existe) O método do Bloco de Código 2.11.1 desempenha as seguintes tarefas:

- Abre uma conexão com o PostgreSQL
- Por meio da conexão, obtém uma abstração do tipo “cursor”
- Por meio do cursor, executa um comando SELECT
- Usa um método do cursor para verificar o resultado do comando SELECT
- Devolve True ou False, dependendo do resultado

Bloco de Código 2.11.1

```
import psycopg
print (psycopg)
class Usuario:
    def __init__(self, login, senha):
        self.login = login
        self.senha = senha

#definição do método: ele recebe um objeto do tipo Usuario
def existe (usuario):
    #Abre a conexão
    with psycopg.connect(
        host="localhost",
        port=5432,
        dbname="20221_pessoal_db_login",
        user="postgres",
        password="postgres"
    ) as conexao:
        #obtem um cursor
        with conexao.cursor() as cursor:
            #executa o comando
            cursor.execute('SELECT * FROM tb_usuario WHERE
login=%s AND senha=%s', (f'{usuario.login}',
f'{usuario.senha}'))
            #obtem o resultado
            result = cursor.fetchone()
            #verifica se o resultado é diferente de None, o
            que indica que o usuário existe na base
            return result != None
```

2.12 (Menu de opções) A seguir, escrevemos uma função que dá as seguintes opções ao usuário:

- 0: Sair
- 1: Fazer login
- 2: Fazer logoff

Veja o Bloco de Código 2.12.1. Repare que na **última linha** tomamos o cuidado de **chamar a função menu**.

Bloco de Código 2.12.1

```
import psycopg
print (psycopg)
class Usuario:
    def __init__(self, login, senha):
        self.login = login
        self.senha = senha
#definição do método: ele recebe um objeto do tipo Usuario
def existe (usuario):
    #Abre a conexão
    with psycopg.connect(
        host="localhost",
        port=5432,
        dbname="20221_pessoal_db_login",
        user="postgres",
        password="postgres"
    ) as conexao:
        #obtem um cursor
        with conexao.cursor() as cursor:
            #executa o comando
            cursor.execute('SELECT * FROM tb_usuario WHERE login=%s AND senha=%s', (f'{usuario.login}',
f'{usuario.senha}'))
            #obtem o resultado
            result = cursor.fetchone()
            #verifica se o resultado é diferente de None, o que indica que o usuário existe na base
            return result != None
#definição da função menu
def menu():
    #texto a ser exibido
    texto = "0-Fechar Sistema\n1-Login\n2-Logoff\n"
    #usuário ainda não existe
    usuario = None
    #capturamos a opção do usuário
    op = int (input (texto))
    #enquanto ele não digitar zero
    while op != 0:
        #se digitar 1, capturamos login e senha e verificamos se o usuário existe na base
        if op == 1:
            login = input ("Digite seu login\n")
            senha = input ("Digite sua senha\n")
            usuario = Usuario (login, senha)
            print ("Usuário OK!!!" if existe(usuario) else "Usuário NOK!!!")
            #se ele digitar 2, configuramos o usuario como "None" novamente
            elif op == 2:
                usuario = None
                print ("Logoff realizado com sucesso")
                op = int (input (texto))
            else:
                #se digitar zero, dizemos adeus. Observe que esse else está associado ao while
                print ("Até mais")
#chamamos a função menu
menu()
```

Exercício

Adicione a opção número 3 ao menu. Ela deve permitir que o usuário faça a inserção de novos usuários na base gerenciada pelo PostgreSQL.

Bibliografia

HEUSER, Carlos Alberto. Projeto de Banco de Dados. 6a Ed., Bookman, 2008.