



*Faculdade de  
Tecnologia SENAI  
“Roberto Mange”*



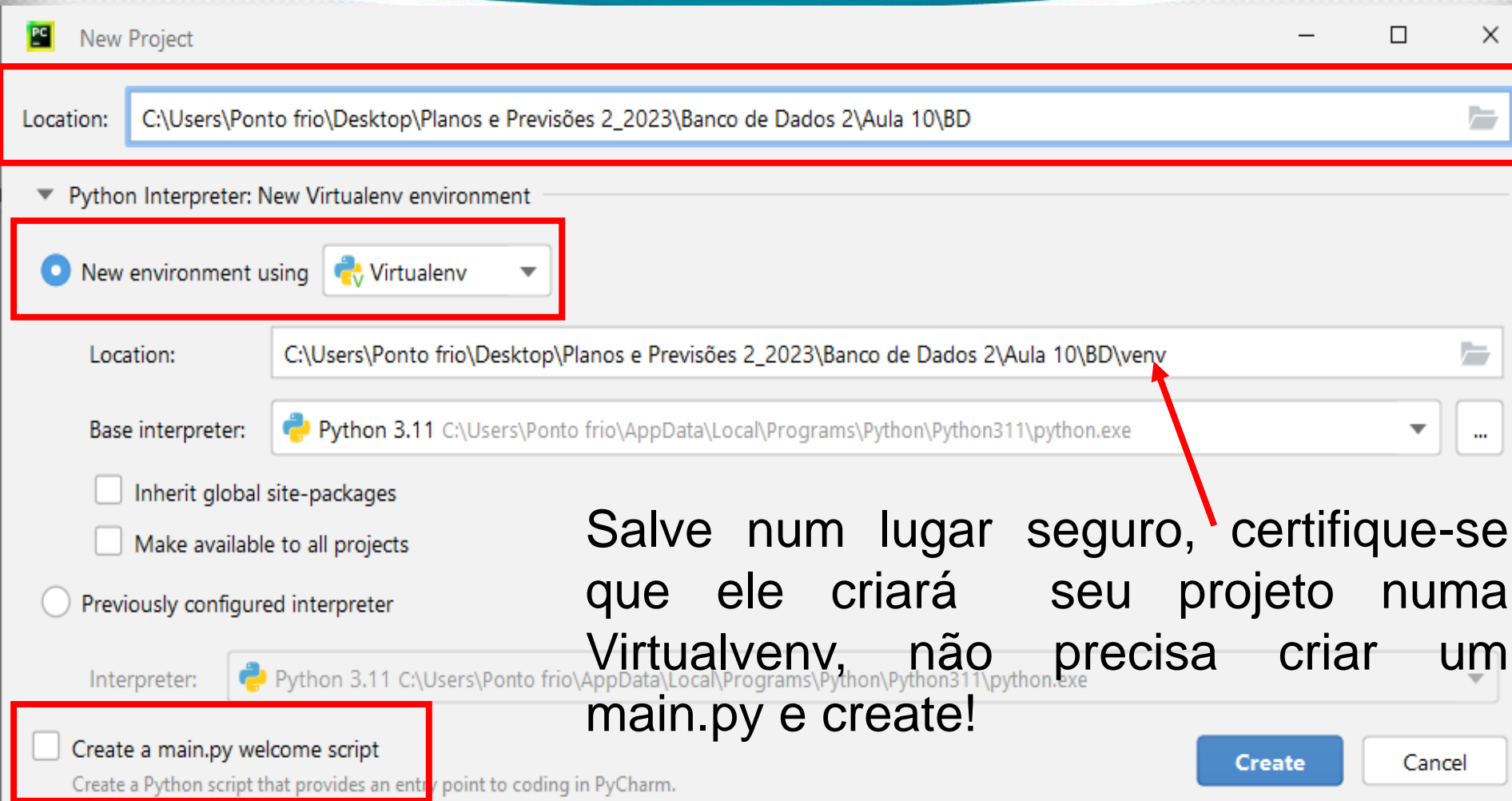
***Aula 10 – CRUD – Create, Retrieve,  
Update e Delete com python***

# *CRUD com Python*

Vamos criar uma aplicação para criar, listar, atualizar e apagar tabelas de um banco de dados com uma aplicação em python. Esse conceito é básico mas imprescindível para a interação de sistemas diversos com banco de dados.



# CRUD com Python



The image shows the 'New Project' dialog in PyCharm. The 'Location' field is set to 'C:\Users\Ponto frio\Desktop\Planos e Previsões 2\_2023\Banco de Dados 2\Aula 10\BD'. The 'Python Interpreter' section is set to 'New Virtualenv environment'. The 'New environment using' dropdown is set to 'Virtualenv'. The 'Location' for the virtual environment is 'C:\Users\Ponto frio\Desktop\Planos e Previsões 2\_2023\Banco de Dados 2\Aula 10\BD\venv'. The 'Base interpreter' is 'Python 3.11 C:\Users\Ponto frio\AppData\Local\Programs\Python\Python311\python.exe'. The 'Create a main.py welcome script' checkbox is checked. The 'Create' button is highlighted.

New Project

Location: C:\Users\Ponto frio\Desktop\Planos e Previsões 2\_2023\Banco de Dados 2\Aula 10\BD

Python Interpreter: New Virtualenv environment

New environment using Virtualenv

Location: C:\Users\Ponto frio\Desktop\Planos e Previsões 2\_2023\Banco de Dados 2\Aula 10\BD\venv

Base interpreter: Python 3.11 C:\Users\Ponto frio\AppData\Local\Programs\Python\Python311\python.exe

☐ Inherit global site-packages

☐ Make available to all projects

☐ Previously configured interpreter

Interpreter: Python 3.11 C:\Users\Ponto frio\AppData\Local\Programs\Python\Python311\python.exe

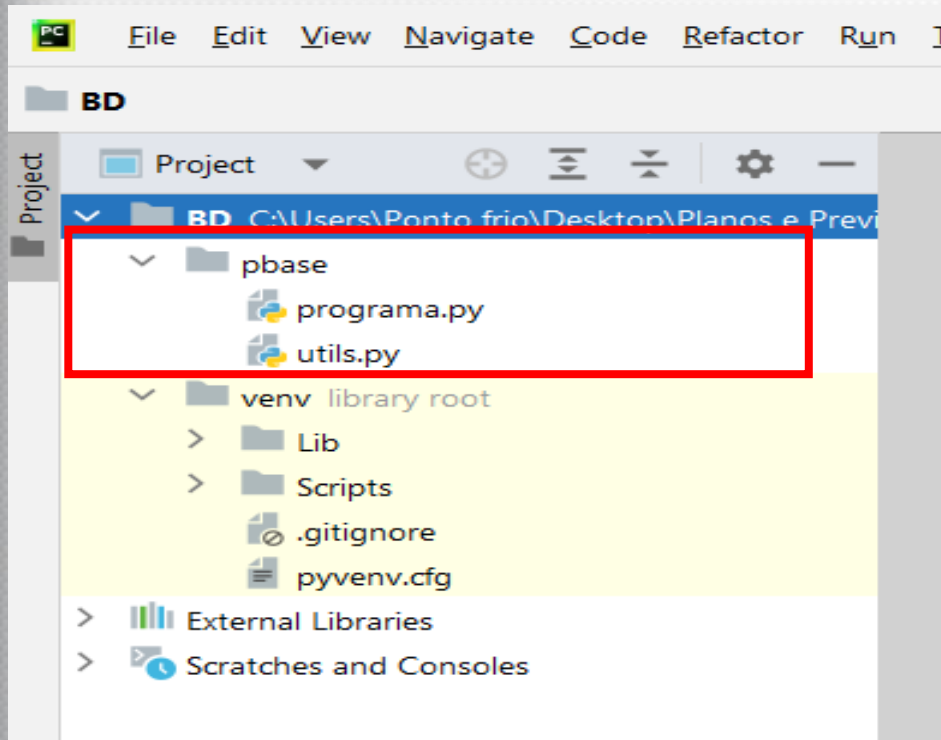
☒ Create a main.py welcome script  
Create a Python script that provides an entry point to coding in PyCharm.

Create Cancel

Salve num lugar seguro, certifique-se que ele criará seu projeto numa Virtualenv, não precisa criar um main.py e create!

# CRUD com Python

Uma vez criado o projeto, pegue a pasta pbase que enviei e salve ela dentro do diretório e pasta que você acabou de criar o seu projeto



É provável que ele já consiga entender essas bibliotecas e já inclua na árvore de projetos. Se não conseguir informe ao professor que irá te ajudar.

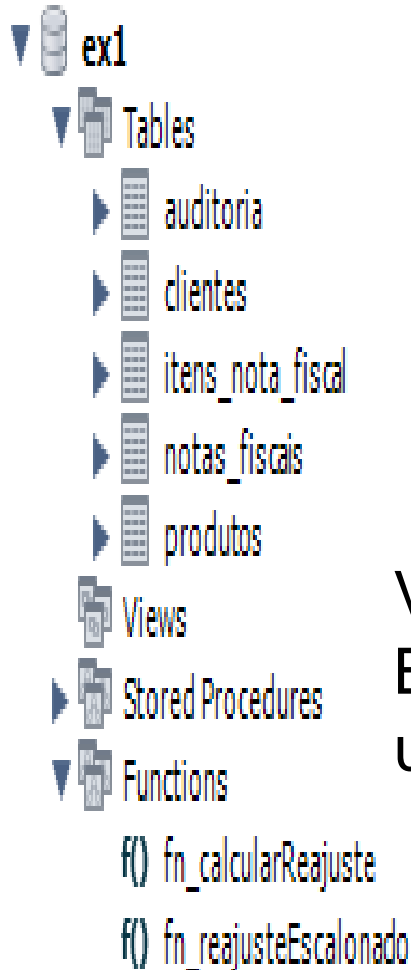
# CRUD com Python

A biblioteca programa será somente para rodar o programa, enquanto a utils terá as rotinas de execução.

```
programa.py X
1 from utils import menu
2
3 if __name__ == '__main__':
4     menu()
```

```
utils.py X
1 def conectar():
2     """
3     Função para conectar ao servidor
4     """
5     print('Conectando ao servidor...')
6
7 def desconectar():
8     """
9     Função para desconectar do servidor.
10    """
11    print('Desconectando do servidor...')
12
13
14 def listar():
15     """
16     Função para listar os produtos
17     """
18    print('Listando produtos...')
```

# Banco de dados de acesso



## Manage Server Connections

MySQL Connections

Local instance MySQL80

BD2

Connection Name: BD2

Connection

Remote Management

System Profile

Connection Method: Standard (TCP/IP)

Method to use to connect to the RDBMS

Parameters

SSL

Advanced

Hostname: 127.0.0.1

Port: 3306

Name or IP address of the server host - and TCP/IP port.

Username: bd2

Name of the user to connect with.

Password:

Store in Vault ...

Clear

The user's password. Will be requested later if it's not set.

Default Schema:

The schema to use as default schema. Leave blank to select it later.

Vamos usar um BD, conexão e usuário já existente



# Instalando a biblioteca mySQL-client

Ache o pacote e instale mysql-client

The screenshot displays the PyCharm IDE interface. On the left, the 'File' menu is open, with 'Settings...' selected. The 'Settings' dialog is open, showing the 'Python Interpreter' section for 'Project: BD'. In the 'Available Packages' list, 'mysql-client' is highlighted. The 'Package' field in the 'Python Interpreter' section is also visible.

File Edit View Navigate Code Refactor R Settings

BD > New Project...  
New...  
New Scratch File  
Open...  
Save As...  
Attach project...  
Recent Projects  
Close Project  
Rename Project...  
Settings... Ctrl+Alt+S  
File Properties  
Local History  
Save All Ctrl+S

Appearance & Behavior  
Keymap  
Editor  
Plugins  
Version Control  
Project: BD  
Python Interpreter  
Project Structure  
Build, Execution, Deployment  
Languages & Frameworks  
Tools  
Settings Sync  
Advanced Settings

Project: BD > Python Interpreter

Python Interpreter: Python 3.11 (BD) C:\Users\PC

Package

Available Packages

mysql

mysql  
mysql-autodoc  
mysql-batch  
mysql-binlog-explorer  
mysql-cli  
mysql-client

# Instalando a biblioteca mySQL-client

E também o pacote  
mysql-  
conector-  
python

The screenshot shows an IDE interface with the 'File' menu open, displaying options like 'New Project...', 'Open...', and 'Settings...'. The 'Settings' window is open, showing the 'Python Interpreter' section for 'Project: BD'. A list of installed packages is shown, including 'mysql-client', 'mysql-connector-python' (highlighted), 'pip', 'protobuf', 'setuptools', and 'wheel'.

File Edit View Navigate Code Refactor Run Settings

BD > New Project... New... New Scratch File Open... Save As... Attach project... Recent Projects Close Project Rename Project... Settings... Ctrl+Alt+S File Properties Local History Save All Ctrl+S

Appearance & Behavior Keymap Editor Plugins 1 Version Control Project: BD Python Interpreter Project Structure Build, Execution, Deployment Languages & Frameworks

Project: BD > Python Interpreter

Python Interpreter: Python 3.11 (BD)

Package

- mysql-client
- mysql-connector-python
- pip
- protobuf
- setuptools
- wheel



# Estabelecendo a conexão

```
1 # importar a biblioteca de drive para o python
2 import mysql.connector
3 # individualizar o erro de conexão se houver
4 from mysql.connector import errorcode

def conectar():
    """
    Função para conectar ao servidor
    """
    try:
        conn = mysql.connector.connect(host='127.0.0.1', user='bd2', password='1234', database='ex1');
        return conn, "Conexão realizada com BD!", "Usuário: ", conn.user, "Banco de Dados: ", conn.database;

    except mysql.connector.Error as error:
        if error.errno == errorcode.ER_BAD_DB_ERROR:
            print("BD não existe!")
        elif error.errno == errorcode.ER_ACCESS_DENIED_ERROR:
            print("Usuário ou password errados!")
        else:
            print(error)
```

# Estabelecendo a conexão

```
programa.py x
1 from utils import menu
2
3 if __name__ == '__main__':
4     menu()
```

Execute somente a função print (conectar()) em programa.py, comente o restante do código passado e verifique o resultado!

Run: programa x

```
"C:\Users\Ponto frio\Desktop\Planos e Previsões 2_2023\Banco de Dados 2\Aula 10\BD\venv\Scripts\python.exe" "C:\Users\Ponto frio\Desktop\Planos e Previsões 2_2023\Banco de
Dados 2\Aula 10\BD\pbase\programa.py"
(<mysql.connector.connection_cext.CMySQLConnection object at 0x00000250D6B77C10>, 'Conexão realizada com BD!', 'Usuário: ', 'bd2', 'Banco de Dados: ', 'ex1')
Process finished with exit code 0
```

# Estabelecendo a conexão

```
programa.py X
1 from utils import menu
2
3 if __name__ == '__main__':
4     menu()
```

Erre usuário, host ou nome do banco pra testar! A integridade da conexão é a parte mais importante do software.

Run: programa X

```
"C:\Users\Ponto frio\Desktop\Planos e Previsões 2_2023\Banco de Dados 2\Aula 10\BD\venv\Scripts\python.exe" "C:\Users\Ponto frio\Desktop\Planos e Previsões 2_2023\Banco de
Dados 2\Aula 10\BD\pbase\programa.py"
Usuário ou password errados!
None
Process finished with exit code 0
```

# Desconectar

```
def desconectar(conn):  
    """  
    Função para desconectar do servidor.  
    """  
    if conn:  
        conn.close();
```

Sempre depois de realizar uma transação entre aplicação e banco de dados, deverá ser realizada a desconexão dentre eles.



# Listando os dados do banco

Query 1 x

Limit to 1000 rows

```
1 • SELECT *FROM produtos;
```

Result Grid

	id	discrimanacao	p_unitario
▶	1	Notebook	8999.99
	2	Laptop	9558.99
	3	MEMÓRIA RAM	1068.66
	4	Celular SAMSUNG	2778.59
•	NULL	NULL	NULL

Vamos usar o banco de dados ex1 e sua tabela produtos como exemplo.

# *método listar()*

```
def listar():  
    """  
    Função para listar (SELECT) os produtos  
    """  
    conn = conectar()  
  
    """  
    Cursores são recursos importantes em Banco de dados para varrer tabelas ou linhas  
    São mais eficazes que o SELECT em alguns casos onde são necessárias varias operações  
    """  
    cursor = conn.cursor();  
    cursor.execute('SELECT *FROM produtos'); #função que executa o comando SQL no python  
    produtos = cursor.fetchall() #metodo fetchall retorna numa lista o conteudo de cursor  
    if len(produtos) > 0 :  
        print("----- LISTANDO PRODUTOS -----")  
        for produto in produtos:  
            print(f'ID={produto[0]} DISCRIMINAÇÃO={produto[1]} PREÇO UNITÁRIO={produto[2]}')  
        print("-----")  
    else:  
        print("----- NÃO EXISTEM PRODUTOS NA LISTA -----")  
    desconectar(conn)
```



# *Testar método listar()*

```
Run: programa x
"C:\Users\Ponto frio\Desktop\Planos e Previsões 2_2023\Banco de Dados 2\Aula
  Dados 2\Aula 10\BD\pbase\programa.py"
=====Gerenciamento de Produtos=====
Selecione uma opção:
1 - Listar produtos.
2 - Inserir produtos.
3 - Atualizar produto.
4 - Deletar produto.
Digite a opção: 1
Conexão realizada com BD! Usuário: bd2 Banco de Dados: ex1
----- LISTANDO PRODUTOS -----
ID=1 DISCRIMINAÇÃO=Notebook PREÇO UNITÁRIO=8999.99
ID=2 DISCRIMINAÇÃO=Laptop PREÇO UNITÁRIO=9558.99
ID=3 DISCRIMINAÇÃO=MEMÓRIA RAM PREÇO UNITÁRIO=1068.66
ID=4 DISCRIMINAÇÃO=Celular SAMSUNG PREÇO UNITÁRIO=2778.59
-----

Process finished with exit code 0
```

# *método inserir()*

```
def inserir():  
    """  
    Função para inserir um produto  
    """  
  
    conn = conectar()  
    cursor = conn.cursor();  
    discriminacao = input("Informe a discriminação do produto: ")  
    preco = float(input("Informe o preço do produto: "))  
  
    cursor.execute(f"INSERT INTO produtos (discriminacao,p_unitario) VALUE ('{discriminacao}','{preco}')" )  
    conn.commit()  
    if cursor.rowcount == 1 :  
        print(f"O produto {discriminacao} foi inserido corretamente")  
    else:  
        print("Não foi possível cadastrar o produto!")  
    desconectar(conn)
```

# *método inserir()*

Run:



programa x



```
"C:\\Users\\Ponto frio\\Desktop\\Planos e Previsões 2_2023\\Banco de Dados 2\\Aula 10  
  Dados 2\\Aula 10\\BD\\pbase\\programa.py"
```

```
=====Gerenciamento de Produtos=====
```

```
Selecione uma opção:
```

```
1 - Listar produtos.
```

```
2 - Inserir produtos.
```

```
3 - Atualizar produto.
```

```
4 - Deletar produto.
```

```
Digite a opção: 2
```

```
Conexão realizada com BD! Usuário: bd2 Banco de Dados: ex1
```

```
Informe a discriminação do produto: CI
```

```
Informe o preço do produto: 1.98
```

```
0 produto CI foi inserido corretamente
```

# Testar método inserir()



```
SELECT *FROM produtos;
```

Result Grid



Filter Rows:

	id	discrimanacao	p_unitario
▶	1	Notebool	8999.99
	2	Laptop	9558.99
	3	MEMÓRIA RAM	1068.66
	4	Celular SAMSUNG	2778.59
	33	CI	1.98
●	NULL	NULL	NULL

Run: programa

```
"C:\Users\Ponto frio\Desktop\Planos e Previsões 2_2023\Banco de Da  
Dados 2\Aula 10\BD\pbase\programa.py"
```

```
=====Gerenciamento de Produtos=====
```

```
Selecione uma opção:
```

- 1 - Listar produtos.
- 2 - Inserir produtos.
- 3 - Atualizar produto.
- 4 - Deletar produto.

```
Digite a opção: 1
```

```
Conexão realizada com BD! Usuário: bd2 Banco de Dados: ex1
```

```
----- LISTANDO PRODUTOS -----
```

```
ID=1 DISCRIMINAÇÃO=Notebool PREÇO UNITÁRIO=8999.99
```

```
ID=2 DISCRIMINAÇÃO=Laptop PREÇO UNITÁRIO=9558.99
```

```
ID=3 DISCRIMINAÇÃO=MEMÓRIA RAM PREÇO UNITÁRIO=1068.66
```

```
ID=4 DISCRIMINAÇÃO=Celular SAMSUNG PREÇO UNITÁRIO=2778.59
```

```
ID=33 DISCRIMINAÇÃO=CI PREÇO UNITÁRIO=1.98
```

```
-----
```

```
Process finished with exit code 0
```

# ***Atividade Somativa***

Observação:

- Atividade em duplas;
- Atividade deverá ser demonstrada ao professor até o dia 05 de dezembro;



# *Atividade Somativa*

- 1) Realizar a transação de UPDATE via aplicativo (25 pontos);
- 2) Realizar a transação de DELETE via aplicativo (25 pontos);
- 3) Permitir a escolha de qualquer tabela e comando de alteração CRUD via aplicativo para o banco de dados do ex1 (25 pontos);
- 4) Toda operação de UPDATE e DELETE de alguma tabela deverá ser disparado um trigger e que seja cadastrado numa tabela a ser criada nesse banco de dados, com a data do ocorrido, usuário envolvido, tipo do comando e identificação da chave primária da tabela (25 pontos)

