

Aula 07

- Armazenamento Persistente de Dados
- Python DB-API
- Módulo ***psycopg2***
 - Create
 - Insert
 - Update
 - Delete
 - Select

- Python DB-API
 - Padrão da linguagem Python que especifica uma interface entre as aplicações e o banco de dados.
 - PEP 249 -- *Python Database API Specification* v2.0

<https://www.python.org/dev/peps/pep-0249/>

- Python DB-API
 - Suporte para maioria dos banco de dados por meio de conexões ODBC.
 - Além de APIs específicas para:
 - IBM DB2
 - Firebird (and Interbase)
 - Informix
 - MySQL
 - Oracle
 - PostgreSQL
 - Microsoft SQL Server
 - Sybase

- Outras soluções
 - SQLAlchemy
 - Toolkit
 - www.sqlalchemy.org
 - SQLAlchemy ORM
 - <http://www.sqlalchemy.org>

- ***Psycopg***

- Adaptador PostgreSQL mais popular para a linguagem de programação Python.
- Implementa todas as especificações indicadas na Python DB API 2.0.
- Permite o acesso a vários recursos oferecidos pelo SGBD.
- Instalação

```
pip install psycopg2
```



Python com PostgreSQL

- Estabelecendo uma Conexão

```
8     try:
9         con = pg.connect(
10             database="db_livraria",
11             user="postgres",
12             password="postgres",
13             host="127.0.0.1",
14             port="5432"
15         )
16         print("Conexão realizada com sucesso!")
17         con.close()
18     except Exception as erro:
19         print(erro)
```

- Criação de Tabelas (DDL)

```
42     cur = con.cursor()
43     cur.execute(sql)
44     print("Tabelas criadas com sucesso!")
45     con.commit()
46     con.close()
```

Utilizado pelo Python para execução de instruções no PostgreSQL.

- Criação de Tabelas (DDL)

```
42     cur = con.cursor()
43     cur.execute(sql)
44     print("Tabelas criadas com sucesso!")
45     con.commit()
46     con.close()
```

*Função usada para execução de um script SQL:
Create, Insert, Update, Delete, Select, entre outros.*

Python com PostgreSQL

- Criação de Tabelas (DDL)

```
42     cur = con.cursor()
43     cur.execute(sql)
44     print("Tabelas criadas com sucesso!")
45     con.commit()
46     con.close()
```

- *Realiza o commit de todas as pendências para o SGBD.*
- *Por padrão, uma transação é aberta antes da execução de cada instrução, assim, se a função commit não é chamada os efeitos da manipulação serão perdidos.*

- Inserção de dados

```
42  sql = "INSERT INTO tb_editora (nome) values ('FATEC Ribeirão Preto')"  
43  cur = con.cursor()  
44  cur.execute(sql)  
45  print("Operação realizada com sucesso!")  
46  con.commit()  
47  con.close()
```

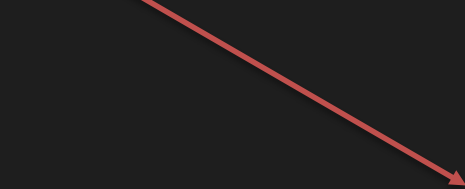
- Inserção de dados
 - Parametrização para evitar *SQL Injection*

```
44  # SQL com parâmetros
45
46  cur = con.cursor()
47
48  sql = "INSERT INTO tb_livro (id_editora, titulo) values (%s,%s)"
49  cur.execute(cur.mogrify(sql,(1,'Python Cookbook'))))
50
51  print("Operação realizada com sucesso!")
52  con.commit()
53  con.close()
```

Python com PostgreSQL

- Inserção de dados
 - Retornar o último id (*last id*)

```
44 # SQL com parâmetros
45
46 cur = con.cursor()
47
48 sql = "INSERT INTO tb_livro (id_editora, titulo) values (%s,%s) RETURNING id"
49 cur.execute(cur.mogrify(sql,(1,'Python Advanced'))
50
51 #Retonar o ID inserido
52 id = cur.fetchone()[0]
53 print("ID = %s" %id)
54
55 print("Operação realizada com sucesso!")
56 con.commit()
57 con.close()
```



Python com PostgreSQL

- Seleção de dados

```
63 cur = con.cursor()
64 sql = "SELECT * FROM tb_editora"
65
66 cur.execute(sql)
67 linhas = cur.fetchall()
68
69 for linha in linhas:
70     print("Id    = %d " %linha[0])
71     print("Nome = %s " %linha[1])
72
73 con.close()
```

Python com PostgreSQL

- Seleção de dados

```
63 cur = con.cursor()
64 sql = "SELECT * FROM tb_editora"
65
66 cur.execute(sql)
67 linhas = cur.fetchall()
68
69 for linha in linhas:
70     print("Id    = %d " %linha[0])
71     print("Nome = %s " %linha[1])
72
73 con.close()
```

- *fetchone*: retorna uma linha
- *fetchmany(n)*: retorna um *n* linhas
- *fetchall*: retorna todas as linhas

Python com PostgreSQL

- Update de dados

```
79  cur = con.cursor()
80
81  sql = "UPDATE tb_editora SET nome=%s WHERE id=%s"
82  cur.execute(cur.mogrify(sql,('FATEC RP',1)))
83  con.commit()
84
85  print("Total linhas atualizadas: %s" %cur.rowcount)
86
87  con.close()
```


- Apagar de dados

```
79     cur = con.cursor()
80
81     sql = "DELETE FROM tb_editora WHERE id=%s"
82     cur.execute(cur.mogrify(sql,"1"))
83     con.commit()
84
85     print("Total linhas atualizadas: %s" %cur.rowcount)
86
87     con.close()
```

- **Exercício 1:** Escreva uma aplicação Python que seja capaz de:
 - Carregar um conjunto de dados no formato JSON. Para isso, utilize o site **generatedata.com**.
 - Criar uma conexão com o PostgreSQL e inserir todos os dados carregados (JSON) e uma tabela.
 - Demonstrar uma operação de seleção com ***fetchall***
 - Demonstrar uma operação de seleção com ***fetchone***
 - Demonstre o uso do operador ***like***

- **Exercício 2:** Escreva uma aplicação Python que exporta todos os dados de uma tabela no PostgreSQL no formato JSON.