# Usando Python 3 para conectar no banco de dados Postgres

# *Autor: Moisés Omena*

### *Para realizar a conexão com o banco de dados postgres precisamos instalar o pacote*

*O psycopg2 é um adaptador que possibilitará o acesso a bancos de dados postgres por meio do python.*
*Para realizar a instalação do psycopg2 execute os seguintes procedimentos.*

*a) Instalação do psycopg2*

*In [1]:*

```
!pip install psycopg2-binary
```

```
Requirement already satisfied: psycopg2-binary in ./env35/lib/python
3.5/site-packages (2.7.5)
```

### *Agora precisamos importar a o pacote para podermos utilizá-lo*

*b) para importar a biblioteca psycopg2 execute os seguintes procedimentos*

*In [2]:*

```
import psycopg2
```

***Portanto, agora podemos utilizar o pacote psycopg2 para acessar o banco de dados por meio da linguagem python***

# *Conectando a base de dados*

### *Inicialmente utilizaremos a funcao connect da biblioteca psycopg2 para conectar no banco de dados desejado.*

***Observe a lista de parametros disponíveis para conexão.***

*database: é o nome do banco de dados no qual desejamos conectar*
*user: usuario usado para autenticação*
*password: senha usada para autenticação*
*host: servidor de banco de dados (para bases de dados locais podemos utilizar: localhost)*
*port: o número da porta de conexão que por padrão é 5432 caso não seja informada*

### *Formas de conexão*

*c) observe que passaremos como parametros para a função connect o nome do banco de dados, o usuário e a senha de acesso*

*In [3]:*

```
conn = psycopg2.connect("dbname=atividade_integrado user=postgres password=123456")
```

*Também podemos especificar os parametros separados por vírgula como no exemplo abaixo*

*In [4]:*

```
conn = psycopg2.connect(host="localhost",port="5432",database="atividade_integrado", user="postgres", password="123456")
```

*Se a conexão for ciriada com sucesso retornará um objeto de conexão. Abaixo podemos observar o conteúdo do objeto criado.*

*In [5]:*

```
print(conn)
```

```
<connection object at 0x7ff409d15c28; dsn: 'dbname=atividade_integrado user=postgres password=xxx host=localhost port=5432', closed: 0>
```

*Agora precisamos criar um cursor por meio do método cursor() do objeto conecção. O cursor será utilizado para executar instruções SQL no banco de dados desejado.*

*In [6]:*

```
cur = conn.cursor()
```

*Se desejarmos, podemos verificar o status do cursor por meio de 'statumessage'*

*In [7]:*

```
print(cur.statusmessage)
```

```
None
```

***Uma vez criado nosso cursor, podemos emitir instruções sql para nosso banco de daddos.***

# *Criando tabelas, inserindo e manipulando informações*

*Antes de tudo, precisamos iniciar uma transação no banco de dados. A transação garantirá que ao final do processo as alterações possam ser confirmadas ou desfeitas.*

In [8]:

```
cur.execute('start transaction')
```

*a) Agora que iniciamos a transação, vamos inciar criando nossa tabela. (Observe que para manter a integridade do tutorial estamos eliminando qualquer tabela denominada aluno existente previamente)*

In [9]:

```
cur.execute("drop table if exists aluno") ## caso a tabela exista ela será elimi
nada para evitar inconsistências
cur.execute("create table aluno (matricula int primary key, nome varchar(100) no
t null)")
```

*b) Agora vamos inserir dados em nossa tabela aluno!*

In [10]:

```
cur.execute("insert into aluno (matricula,nome) values (%s,%s)",(1,'Pedro'))
cur.execute("insert into aluno (matricula,nome) values (%s,%s)",(2,'Maria'))
cur.execute("insert into aluno (matricula,nome) values (%s,%s)",(3,'Ana'))
```

*c) Uma vez que os dados foram inseridos, vamos consultar nossa tabela para verificar se os dados foram inseridos corretamente. Sendo assim, vamos executar a instrução select do sql para obter os registros/linhas que estão na tabela aluno. As informações serão armazenadas no proprio cursor.*

In [11]:

```
cur.execute("select * from aluno");
```

*obtem uma das linhas armazenadas no cursor (busca uma única linha por vez)*

In [12]:

```
cur.fetchone()
```

Out[12]:

```
(1, 'Pedro')
```

*obtem todas as linhas disponíveis no cursor (busca todas as linhas restantes)*

In [13]:

```
for i in cur.fetchall():
    print(i)
```

```
(2, 'Maria')
(3, 'Ana')
```

*Agora vamos apagar todas as linhas da tabela aluno que possuem matricula maior que 1*

In [14]:

```
cur.execute("delete from aluno where matricula>1");
```

*Para visualizar as modificações, vamos buscar todas as linhas restantes na tabela aluno. Observe que as linhas onde os valores de matricula são maiores que '1' foram excluídas.*

In [15]:

```
cur.execute("select * from aluno");
```

In [16]:

```
for i in cur.fetchall():
    print(i)
```

```
(1, 'Pedro')
```

### *Confirmando ou desfazendo as alterações*

*Para confirmar todas as alterações podemos aplicar a instrução Commit*

In [17]:

```
cur.execute("commit")
```

*Caso seja necessário **desfazer todas as atividades executadas anteriormente** poderíamos ter executado ao invês do commit, a instrução **rollback**. como exemplo teríamos: **cur.execute("rollback")***

### *Para concluir podemos fechar o cursor e finalizar a conexão.*

In [18]:

```
cur.close()
```

*O Psycopg2 também fornece como forma alternativa de fechamento da conexão juntamente com commit da instruções SQL (ou seja uma operação de multiplas ações), o método de conexão commit(). Observe a próxima instrução:*

In [19]:

```
conn.commit()
```

*Da mesma semelhante o Psycopg2 também fornece a possibilidade de fechamento da conexão juntamente com rollback por meio do método rollback(). Um exemplo seria aplicação de **conn.rollback()***

# *Captura de erros*

*Também é possível verificar a existencia de erros durante a execução e tomar ações com base nestes erros. veja o exemplo abaixo. Observe que neste caso ocorrerão erros pois o cursor já foi finalizado e portanto, para que o procedimento ocorresse de forma correta precisariamos criar um novo cursor.*

In [20]:

```
try:
    cur.execute("SELECT * FROM aluno")
except Exception as e:
    print("OCORRERAM PROBLEMAS!")
    print("Mensagem padrao de erro foi: ",e)
    try:
        cur.execute("Rollback")
        print("foi aplicado rollback")
    except Exception as e2:
        print("foi aplicado rollback, mas tentativa de aplicação falhou")
        print("Mensagem padrao de erro foi: ",e2)
```

```
OCORRERAM PROBLEMAS!
Mensagem padrao de erro foi:  cursor already closed
foi aplicado rollback, mas tentativa de aplicação falhou
Mensagem padrao de erro foi:  cursor already closed
```

# Referências


http://initd.org/psycopg/docs/usage.html (http://initd.org/psycopg/docs/usage.html)

http://initd.org/psycopg/docs/install.html (http://initd.org/psycopg/docs/install.html)

http://www.postgresqltutorial.com/postgresql-python/connect/ (http://www.postgresqltutorial.com/postgresql-python/connect/)

http://www.postgresqltutorial.com/postgresql-python/query/ (http://www.postgresqltutorial.com/postgresql-python/query/)

https://www.fullstackpython.com/blog/postgresql-python-3-psycopg2-ubuntu-1604.html (https://www.fullstackpython.com/blog/postgresql-python-3-psycopg2-ubuntu-1604.html)