

# **Funções, Objetos e Acesso à Base de Dados em Python**

# Agenda

## Tópicos:

- **Funções**
- **Tratamento de Erros**
- **Objetos (Dicionários)**
- **Acesso à Base de Dados**
- **Procurando Ajuda**

# Funções

**Para declarar uma função em Python, utilizamos a instrução `def`, e devemos considerar a indentação, como no `if` ou no `while`**

```
def delta_bhaskara(a, b, c):  
    return (b * b) - (4 * a * c)
```

```
d = delta_bhaskara(2, 7, 3)  
print(d)
```

# Funções

Para declarar uma função em Python, utilizamos a instrução `def`, e devemos considerar a indentação, como no `if` ou no `while`

```
def delta_bhaskara(a, b, c):  
    return (b * b) - (4 * a * c)
```

```
d = delta_bhaskara(2, 7, 3)  
print(d)
```

Diferente de algumas linguagens, e similar ao JavaScript, em Python é possível misturar código com declaração de funções

# Funções

**Para declarar uma função em Python, utilizamos a instrução `def`, e devemos considerar a indentação, como no `if` ou no `while`**

```
def delta_bhaskara(a, b, c):  
    return (b * b) - (4 * a * c)
```

```
d = delta_bhaskara(2, 7, 3)  
print(d)
```

# Funções

Para declarar uma função em Python, utilizamos a instrução `def`, e devemos considerar a indentação, como no `if` ou no `while`

```
def delta_bhaskara(a, b, deltaBhaskara):  
    return (b * b) - (4 * a * deltaBhaskara)  
  
d = delta_bhaskara(2, 7, 3)  
print(d)
```

Em Java, essa função seria chamada de `deltaBhaskara`, mas em Python, usamos `_` para separar palavras (é só uma convenção, o código funcionaria de qualquer jeito):

[www.python.org/dev/peps/pep-0008](http://www.python.org/dev/peps/pep-0008)

# Funções

É possível transformar um parâmetro em opcional, atribuindo a ele um valor padrão

```
def delta_bhaskara(a, b=7, c=3):  
    return (b * b) - (4 * a * c)
```

```
d = delta_bhaskara(2)  
print(d)
```

# Funções

É possível transformar um parâmetro em opcional, atribuindo a ele um valor padrão

```
def delta_bhaskara(a, b=7, c=3):  
    return (b * b) - (4 * a * c)
```

```
d = delta_bhaskara(2)  
print(d)
```

**Depois do primeiro parâmetro opcional, só podem aparecer parâmetros opcionais!**



# Funções

É possível transformar um parâmetro em opcional, atribuindo a ele um valor padrão

```
def delta_bhaskara(a, b=7, c=3):  
    return (b * b) - (4 * a * c)
```

```
d = delta_bhaskara(2, 9)  
print(d)
```

**b valerá 9, e c, 3**

# Funções

É possível transformar um parâmetro em opcional, atribuindo a ele um valor padrão

```
def delta_bhaskara(a, b=7, c=3):  
    return (b * b) - (4 * a * c)
```

```
d = delta_bhaskara(2, 9, 5)  
print(d)
```

**b valerá 9, e c, 5**

# Funções

É possível transformar um parâmetro em opcional, atribuindo a ele um valor padrão

```
def delta_bhaskara(a, b=7, c=3):  
    return (b * b) - (4 * a * c)
```

```
d = delta_bhaskara(2, c=5)  
print(d)
```

**Em Python é possível passar valores para os parâmetros através dos nomes, como aqui, onde b valerá 7, mas c valerá 5**

# Funções

É possível transformar um parâmetro em opcional, atribuindo a ele um valor padrão

```
def delta_bhaskara(a, b=7, c=3):  
    return (b * b) - (4 * a * c)
```

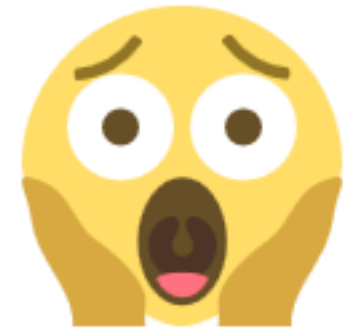
```
d = delta_bhaskara(2, c=5, b=9)  
print(d)
```

**Usando os nomes dos parâmetros, a ordem não importa!**

# Funções

**Aviso:**

**Os slides a seguir contém material avançado, que ocuparia semanas de aulas!**



# Funções

**Assim como no JavaScript, é possível criar funções dentro de funções em Python**

```
def tres_vezes_dobro(a):  
    dobro = 2 * a  
    def imprime():  
        print(f'0 dobro de {a} é {dobro}')    imprime()  
    imprime()  
    imprime()  
  
tres_vezes_dobro(4)
```

# Funções

Assim como no JavaScript, é possível criar funções dentro de funções em Python

```
def tres_vezes_dobro(a):  
    dobro = 2 * a  
    def imprime():  
        print(f'O dobro de {a} é {dobro}')    imprime()  
    imprime()  
    imprime()  
  
tres_vezes_dobro(4)
```

**Saída na tela:**  
**O dobro de 4 é 8**  
**O dobro de 4 é 8**  
**O dobro de 4 é 8**

# Funções

**Para alterar o valor de uma variável da função externa, devemos “declarar a variável” usando nonlocal**

```
def sequenciador(a):  
    def proximo():  
        nonlocal a  
        a = a + 1  
        print(a)  
    proximo()  
    proximo()  
    proximo()  
  
sequenciador(4)
```



# Funções

Para alterar o valor de uma variável da função externa, devemos “declarar a variável” usando nonlocal

```
def sequenciador(a):  
    def proximo():  
        nonlocal a  
        a = a + 1  
        print(a)  
    proximo()  
    proximo()  
    proximo()
```

```
sequenciador(4)
```

**Saída na tela:**

**5**

**6**

**7**

# Funções

Também como no JavaScript, é possível retornar funções e armazenar funções em variáveis

```
def favorito(a):  
    def imprime():  
        print(f'Seu número favorito é {a}')    return imprime  
  
x = favorito(a)  
x()  
x()
```

# Funções

Também como no JavaScript, é possível retornar funções e armazenar funções em variáveis

```
def favorito(a):  
    def imprime():  
        print(f'Seu número favorito é {a}')    return imprime
```

```
x = favorito(a)  
x()  
x()
```

**Saída na tela:**  
**Seu número favorito é 4**  
**Seu número favorito é 4**

# Funções

**Para mais informações:**

**[www.python.org/dev/peps/pep-3104](http://www.python.org/dev/peps/pep-3104)**

# Funções

**Pronto!**

**Acabou a parte avançada!**



# Tratamento de Erros

Lembram do erro causado quando um elemento não existia em uma lista, e pedíamos sua posição???

```
>>> lista = [8, 9, 1, 15]
```

```
>>> lista.index(15)
```

```
3
```

```
>>> lista.index(10)
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
ValueError: 10 is not in list
```



# Tratamento de Erros

Pois bem! Eis como tratar erros!

```
lista = [8, 9, 1, 15]
try:
    print(lista.index(10))
except:
    print('Não estava na lista!')
```

# Tratamento de Erros

É possível, também, acessar uma variável com o que deu errado, e não apenas saber que algo deu errado

```
lista = [8, 9, 1, 15]
try:
    print(lista.index(10))
except Exception as ex:
    print(ex)
```



# Tratamento de Erros

É possível, também, acessar uma variável com o que deu errado, e não apenas saber que algo deu errado

```
lista = [8, 9, 1, 15]
try:
    print(lista.index(10))
except Exception as ex:
    print(ex)
```

**Do mesmo modo que ocorre em outras linguagens, em casos mais avançados, podemos ter mais de um except, cada um com uma classe de exceção diferente, para melhor explicar o significado do erro!**

# Tratamento de Erros

```
lista = [8, 9, 1, 15]
try:
    print(lista.index(10))
except:
    print('Não estava na lista!')
else:
    print('Tudo correu bem!')
```

**Em Python existe o  
else para o try!**

**Ele é executado  
depois do try, mas  
apenas se nada deu  
errado dentro do try!**

# Tratamento de Erros

```
lista = [8, 9, 1, 15]
try:
    print(lista.index(10))
except:
    print('Não estava na lista!')
else:
    print('Tudo correu bem!')
```

**Em Python existe o  
else para o try!**

**Ele é executado  
depois do try, mas  
apenas se nada deu  
errado dentro do try!**

# Tratamento de Erros

```
lista = [8, 9, 1, 15]
try:
    print(lista.index(10))
except:
    print('Não estava na lista!')
else:
    print('Tudo correu bem!')
finally:
    print('Fim!')
```

**Como em outras linguagens, o Python tem o `finally`, que é executado depois de tudo, tendo ocorrido uma exceção ou não, e até mesmo se sairmos do `try` por um `return` ou `break`!!!**

# Tratamento de Erros

## Combinações válidas:

`try / except`

`try / except / else`


`try / except / finally`

`try / except / else / finally`

`try / finally`

# Objetos (Dicionários)

É possível criar objetos em Python utilizando uma notação bastante parecida com a do JSON, assim como fazemos em JavaScript



```
>>> a = {  
...     'nome': 'Rafael',  
...     'idade': 35  
... }  
>>> a['idade']  
35
```

# Objetos (Dicionários)

```
>>> a = {  
...     'nome': 'Rafael',  
...     8: 'abc'  
... }  
>>> a[8]  
'abc'
```

**Diferente do  
JavaScript, Python  
pode ter chaves que  
não são apenas  
textos, como é o caso  
do 8!!!**

# Objetos (Dicionários)

```
>>> a = {  
...     '8': 'Rafael',  
...     8: 'abc'  
... }  
>>> a[8]  
'abc'  
>>> a['8']  
'Rafael'
```

**Diferente do  
JavaScript, Python  
pode ter chaves que  
não são apenas  
textos, como é o caso  
do 8!!!**



# Objetos (Dicionários)

## Alterando o valor de uma chave

```
>>> a = { 'nome': 'Rafael', 'idade': 35 }  
>>> a['idade'] = 21  
>>> a  
{'nome': 'Rafael', 'idade': 21}
```

# Objetos (Dicionários)

## Checando a existência de uma chave

```
>>> a = { 'nome': 'Rafael', 'idade': 35 }
```

```
>>> 'idade' in a
```

```
True
```

```
>>> 'email' in a
```

```
False
```

# Objetos (Dicionários)

## Removendo uma chave

```
>>> a = { 'nome': 'Rafael', 'idade': 35 }  
>>> del a['idade']  
>>> a  
{'nome': 'Rafael'}
```

# Objetos (Dicionários)

**Assim como ocorre com listas, tentar acessar ou excluir uma chave que não existe**

```
>>> a = { 'nome': 'Rafael', 'idade': 35 }
```

```
>>> a['email']
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
KeyError: 'email'
```

# Objetos (Dicionários)

**Assim como ocorre com listas, tentar acessar ou excluir uma chave que não existe**

```
>>> a = { 'nome': 'Rafael', 'idade': 35 }
```

```
>>> del a['email']
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

KeyError: 'email'

# Objetos (Dicionários)

Assim como ocorre com listas, é possível utilizar o `for` para percorrer os elementos de um dicionário

```
>>> a = { 'nome': 'Rafael', 'idade': 35 }  
>>> for x in a:  
...     print(x)  
...  
nome  
idade
```

# Objetos (Dicionários)

No caso de dicionários, é possível percorrer os valores, além das chaves

```
>>> a = { 'nome': 'Rafael', 'idade': 35 }  
>>> for x in a.values():  
...     print(x)  
...  
Rafael  
35
```

# Objetos (Dicionários)

**Para mais informações sobre listas, sequências,  
tuplas, conjuntos e dicionários:**

**[docs.python.org/3/tutorial/datastructures.html](https://docs.python.org/3/tutorial/datastructures.html)**




# Acesso à Base de Dados

**A forma de acessar bases de dados depende da empresa que criou a base de dados em questão**

**Por exemplo, o acesso a uma base MySQL é diferente do acesso a uma base SQL Server**

# Acesso à Base de Dados

Para o exemplo, utilizaremos uma base de dados MySQL chamada workshop, com uma tabela chamada pessoa



```
CREATE TABLE pessoa (  
    id int NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    nome varchar(100) NOT NULL,  
    email varchar(100) NOT NULL,  
    nascimento datetime NOT NULL  
)
```

# Acesso à Base de Dados

Para mais informações sobre o uso de MySQL em Python:

[dev.mysql.com/doc/connector-python/en/connector-python-versions.html](https://dev.mysql.com/doc/connector-python/en/connector-python-versions.html)

[dev.mysql.com/doc/connector-python/en/connector-python-installation.html](https://dev.mysql.com/doc/connector-python/en/connector-python-installation.html)

[dev.mysql.com/doc/connector-python/en/connector-python-examples.html](https://dev.mysql.com/doc/connector-python/en/connector-python-examples.html)

[dev.mysql.com/doc/connector-python/en/connector-python-tutorials.html](https://dev.mysql.com/doc/connector-python/en/connector-python-tutorials.html)

[dev.mysql.com/doc/connector-python/en/connector-python-connecting.html](https://dev.mysql.com/doc/connector-python/en/connector-python-connecting.html)

# Acesso à Base de Dados

**Por se tratar de uma biblioteca de terceiros, que não é padrão do Python, é necessário instalar um pacote manualmente**

**Para isso, utilizaremos o gerenciador de pacotes do Python, chamado pip**

```
pip install mysql-connector-python
```

# Acesso à Base de Dados

**Se esse comando não funcionar, porque o Linux/Windows não reconheceu o comando pip, basta executar o comando pip através do python**

```
python -m pip install mysql-connector-python
```

# Acesso à Base de Dados

Se tudo falhar, existem outras opções de instalação para cada sistema operacional:

[dev.mysql.com/doc/connector-python/en/connector-python-installation.html](https://dev.mysql.com/doc/connector-python/en/connector-python-installation.html)



# Acesso à Base de Dados

As informações de acesso ao MySQL de testes, para esse workshop:

**host:**

**teste-condominio.c1hpz28jfdku.us-east-2.rds.amazonaws.com**

**user:**

**root**

**password:**

**6#vWHD\_\$**

**database:**

**workshop**

# Acesso à Base de Dados

As informações de acesso ao MySQL de testes, para esse workshop:

**host:**

**teste-condominio.c1hpz28jfdku.us-east-2.rds.amazonaws.com**

**user:**

**root**

**password:**

**6#vWHD\_\$**

**database:**

**workshop**

**Essas informações são  
válidas para hoje! ...**

**Esse servidor não  
estará no ar por muito  
mais tempo... 😄**



# Acesso à Base de Dados

## Abrindo e fechando a conexão com a base de dados

```
import mysql.connector
conn = mysql.connector.connect(host='...',
                               user='root',
                               password='6#vWHD_$',
                               database='workshop')

# Tudo que você quiser fazer deve estar aqui, entre a abertura e o
# fechamento da conexão
conn.close()
```

# Acesso à Base de Dados

## Listando registros

```
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute('SELECT id, nome, email, nascimento FROM pessoa')
for (id, nome, email, nascimento) in cursor:
    print(f'{id} {nome} {email} {nascimento}')
cursor.close()

conn.close()
```

# Acesso à Base de Dados

## Listando registros

```
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute('SELECT id, nome, email, nascimento FROM pessoa')
for (id, nome, email, nascimento) in cursor:
    print(f'{id} {nome} {email} {nascimento}')
cursor.close()

conn.close()
```

# Acesso à Base de Dados

## Listando registros

```
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute('SELECT id, nome, email, nascimento FROM pessoa')
for (id, nome, email, nascimento) in cursor:
    print(f'{id} {nome} {email} {nascimento:%d/%m/%Y}')
cursor.close()

conn.close()
```

# Acesso à Base de Dados

## Listando registros

```
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute('SELECT id, nome, email, nascimento FROM pessoa')
for (id, nome, email, nascimento) in cursor:
    print(f'{id} {nome} {email} {nascimento:%d/%m/%Y}')
```

```
cursor.close()
```

Para mais formatos de data:

[docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior](https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior)

```
conn.close()
```

# Acesso à Base de Dados

## Inserindo novos registros

```
from datetime import datetime
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute(
    'INSERT INTO pessoa (nome, email, nascimento) VALUES (%s, %s, %s)',
    ('Novo', 'novo@espm.br', datetime(1990, 10, 20))
)
conn.commit()

conn.close()
```

# Acesso à Base de Dados

## Inserindo novos registros

```
from datetime import datetime
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute(
    'INSERT INTO pessoa (nome, email, nascimento) VALUES (%s, %s, %s)',
    ('Novo', 'novo@espm.br', datetime(1990, 10, 20))
)
conn.commit()
conn.close()
```

**Um valor deve ser  
passado para cada  
parâmetro %s, em  
ordem!**

# Acesso à Base de Dados

## Inserindo novos registros

```
from datetime import datetime
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute(
    'INSERT INTO pessoa (nome, email, nascimento) VALUES (%s, %s, %s)',
    ('Novo', 'novo@espm.br', datetime(1990, 10, 20))
)
conn.commit()
conn.close()
```

**Se o commit() não for executado, as alterações serão desfeitas!!!**



# Acesso à Base de Dados

## Inserindo novos registros

```
from datetime import datetime
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute(
    'INSERT INTO pessoa (nome, email, nascimento) VALUES (%s, %s, %s)',
    ('Novo', 'novo@espm.br', datetime.strptime('20/10/1990', '%d/%m/%Y'))
)
conn.commit()

conn.close()
```

# Acesso à Base de Dados

## Inserindo novos registros

```
from datetime import datetime
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute(
    'INSERT INTO pessoa (nome, email, nascimento) VALUES (%s, %s, %s)',
    ('Novo', 'novo@espm.br', datetime.strptime('20/10/1990', '%d/%m/%Y'))
)
conn.commit()

conn.close()
```

# Acesso à Base de Dados

## Inserindo novos registros

```
from datetime import datetime
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute(
    'INSERT INTO pessoa (nome, email, nascimento) VALUES (%s, %s, %s)',
    ('Novo', 'novo@espm.br', datetime.strptime('20/10/1990', '%d/%m/%Y'))
)
conn.commit()
```

Para mais formatos de data:

[docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior](https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior)

# Acesso à Base de Dados

Obtendo o id gerado do último registro inserido

```
from datetime import datetime
import mysql.connector
conn = mysql.connector.connect(...)

cursor = conn.cursor()
cursor.execute('INSERT ...')

cursor.execute('SELECT last_insert_id()')
registro = cursor.fetchone()
id = registro[0]

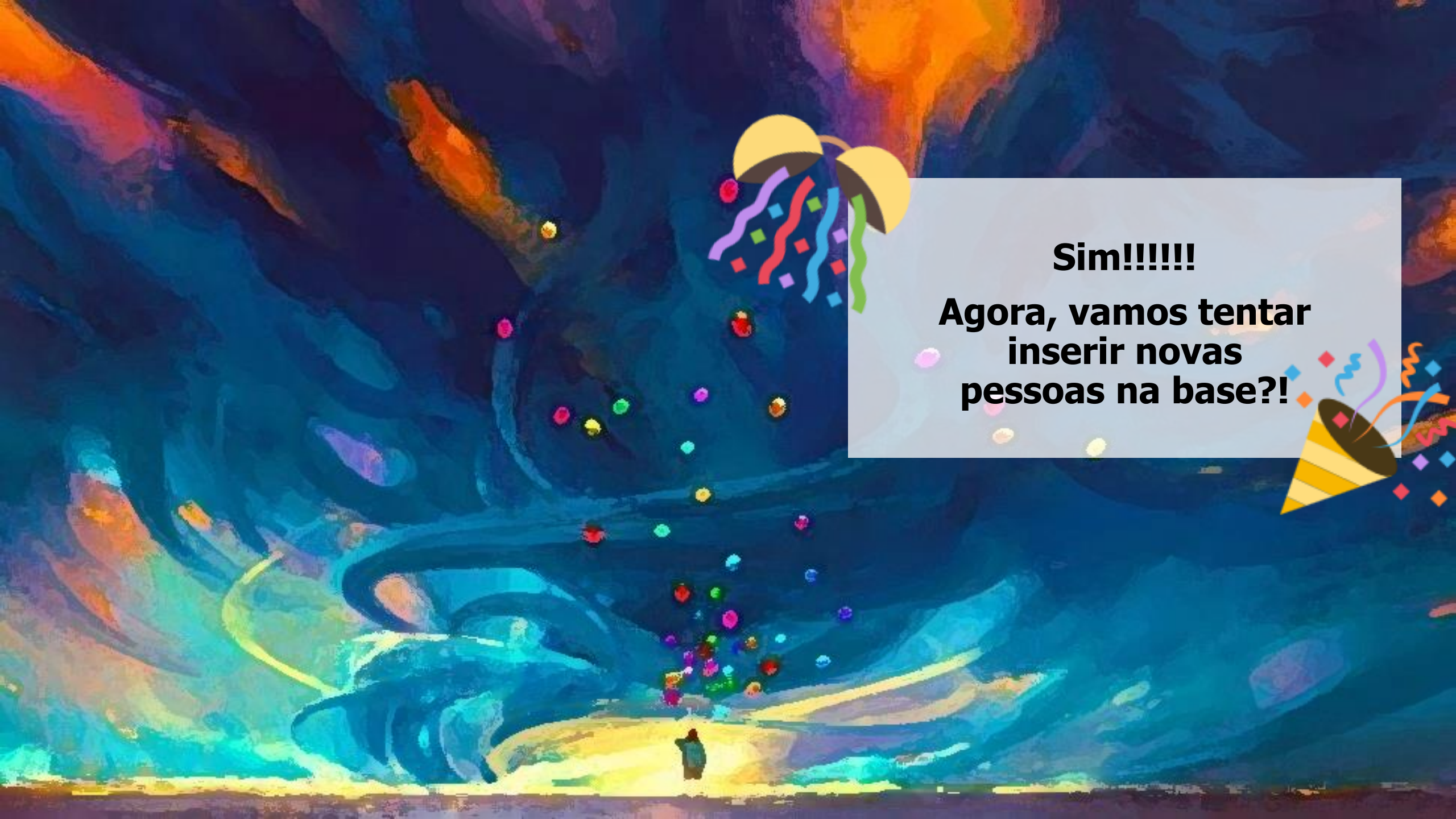
conn.commit()

conn.close()
```



**Sério... Até no último  
dia esse slide?!**





**Sim!!!!!!**

**Agora, vamos tentar  
inserir novas  
pessoas na base?!**

# Procurando Ajuda

[\*\*www.python.org/doc\*\*](http://www.python.org/doc)

[\*\*docs.python.org/3/reference\*\*](http://docs.python.org/3/reference)

[\*\*docs.python.org/release/3.7.3/reference\*\*](http://docs.python.org/release/3.7.3/reference)

[\*\*docs.python.org/3/library\*\*](http://docs.python.org/3/library)

[\*\*docs.python.org/release/3.7.3/library\*\*](http://docs.python.org/release/3.7.3/library)

[\*\*docs.python.org/3/tutorial/introduction.html\*\*](http://docs.python.org/3/tutorial/introduction.html)

[\*\*stackoverflow.com\*\*](http://stackoverflow.com)

[\*\*pt.stackoverflow.com\*\*](http://pt.stackoverflow.com)





**Ficamos por aqui!**