Linguagem de programação

Python Orientado a Objetos

Prof.ª Elisa Antolli

- Unidade de Ensino: 03
- Competência da Unidade: Python Orientado a Objetos
- Resumo: Saber utilizar Python em P.O.O e modelos banco de dados na linguagem
- Palavras-chave: Algoritmos; Python; Banco de dados; Busca; Orientação a Objeto.
- Título da Teleaula: Python Orientado a Objetos
- Teleaula nº: 03

Contextualização

- Classes e Métodos em Python
- Bibliotecas e Módulos em Python
- Aplicação de banco de dados com Python

Linguagem de programação: conceitos de orientação a objetos

Conceitos de orientação a objetos

O que são objetos e o que as classes têm a ver com eles?

Uma classe é uma abstração que descreve entidades do mundo real e quando instanciadas dão origem a objetos com características similares.

Portanto, a classe é o modelo e o objeto é uma instância.

Conceitos de orientação a objetos

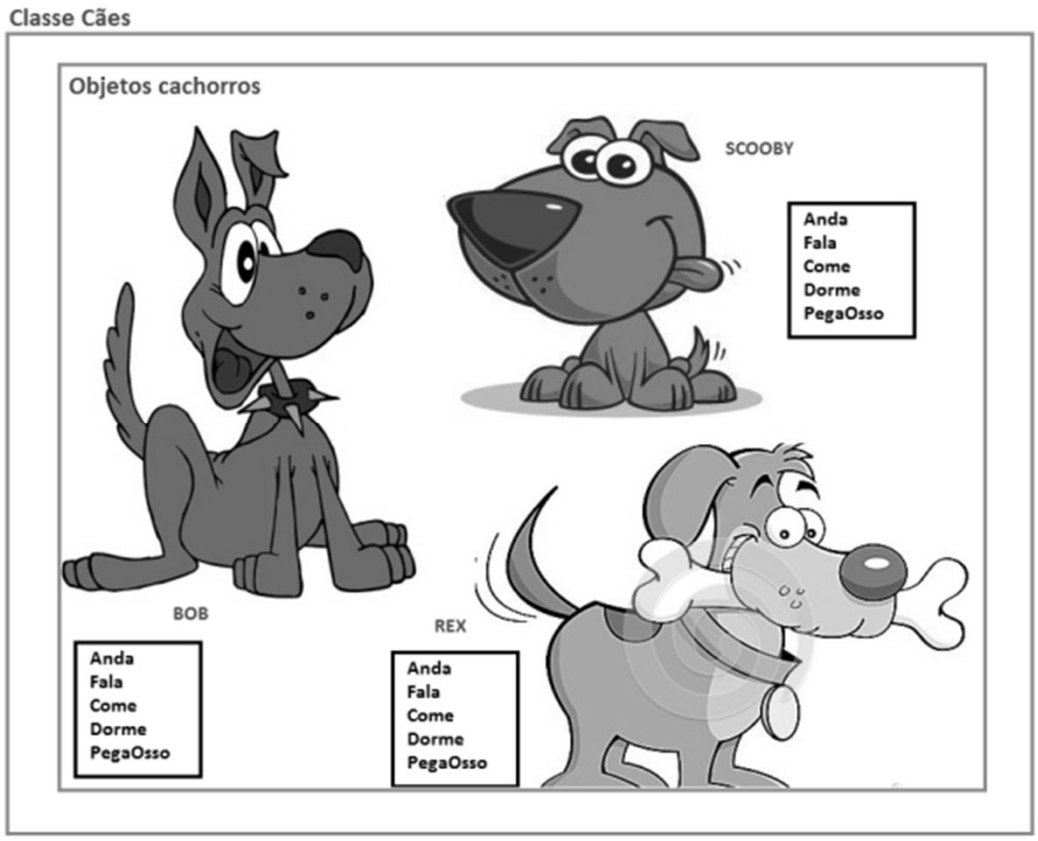
Abstração - Classes e objetos

Objetos são os componentes de um programa OO. Um programa que usa a tecnologia OO é basicamente uma coleção de objetos.

Uma classe é um modelo para um objeto. Podemos considerar uma classe uma forma de organizar os dados (de um objeto) e seus comportamentos (PSF, 2020a). A classe é o modelo e o objeto é uma instância.

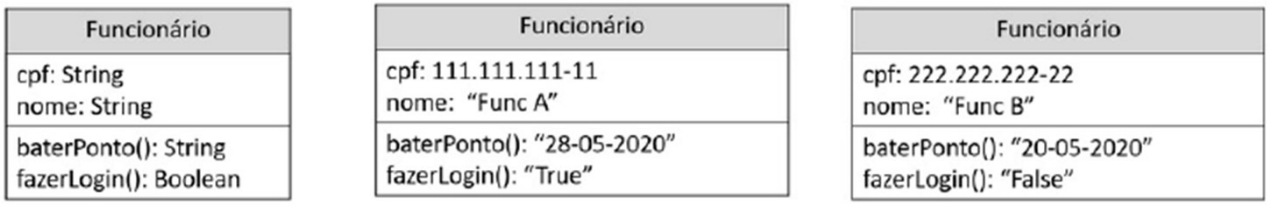
Entende-se por instância a existência física, em memória, do objeto.

Conceitos de orientação a objetos



Fonte: fabrica de software - senac.

Conceitos de orientação a objetos



Fonte: elaborado pelo autor.

Cada diagrama de classes é definido por três seções separadas: o próprio nome da classe, os dados e os comportamentos.

Temos a classe funcionário, são especificados o que um funcionário deve ter.

No nosso caso, como dados, ele deve ter um CPF e um nome e, como comportamento, ele deve bater ponto e fazer login. Esses dados estão "preenchidos", ou seja, foram instanciados e, portanto, são objetos.

Conceitos de orientação a objetos

Atributos

Os dados armazenados em um objeto representam o estado do

objeto. Na terminologia de programação OO, esses dados são chamados de atributos. Os atributos contêm as informações que diferenciam os vários objetos – os funcionários, neste caso. ***Atributos*** são portanto, as características de um objeto, essas características também são conhecidas como variáveis, utilizando o exemplo dos cães, temos alguns atributos, tais como: cor, peso, altura e nome.

```
public class Cachorro{
        public String nome;
        public float peso;
        public float altura;
        public String cor;
}
```

Conceitos de orientação a objetos

Métodos

Nas linguagens procedurais, o comportamento é definido por procedimentos, funções e sub-rotinas. Na terminologia de programação OO, esses comportamentos estão contidos nos métodos, aos quais você envia uma mensagem para invocá-los.

Métodos são portanto, as ações que os objetos podem exercer quando solicitados, onde podem interagir e se comunicarem com outros objetos, utilizando o exemplo dos cães, temos alguns exemplos: latir, correr, pular.

Implementando a classe "Cachorro" além dos Atributos, agora com Métodos:

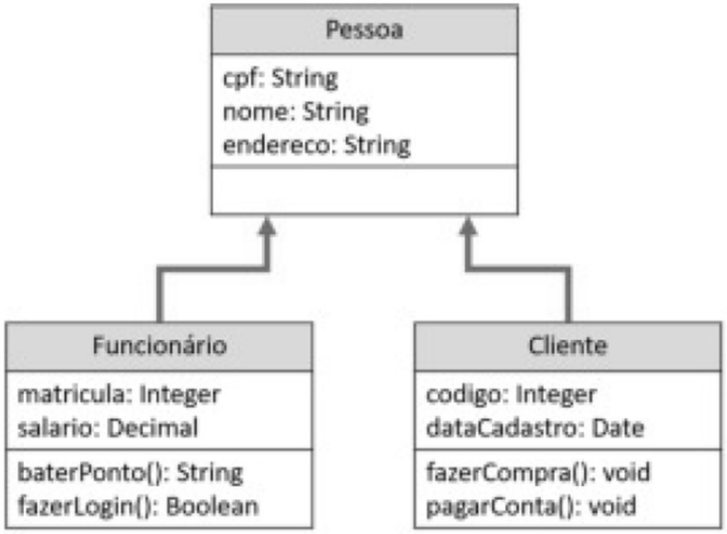
```
public class Cachorro{
        public String nome;   [...]
        public String cor;
  void pular {
       if (altura >= 80){
            System.out.println("Seu cachorro pula alto");
       }else{
            System.out.println("Seu cachorro pula baixo")
       }
    }
}
```

Conceitos de orientação a objetos

Herança

Por meio desse mecanismo, é possível fazer o reuso de código, criando soluções mais organizadas.

A herança permite que uma classe herde os atributos e métodos de outra classe. Observe, na Figura, que as classes funcionário e cliente herdam os atributos da classe pessoa. A classe pessoa pode ser chamada de classe-pai, classe-base, superclasse, ancestral; por sua vez, as classes derivadas são as classes filhas, subclasses.



Fonte: elaborado pelo autor.

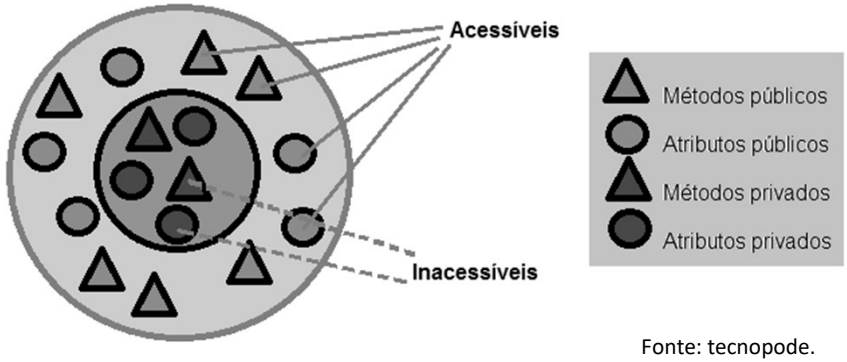
Conceitos de orientação a objetos

Encapsulamento

O ato de combinar os atributos e métodos na mesma entidade é, na linguagem OO, chamado de encapsulamento (Weisfeld, 2013), termo que também aparece na prática de tornar atributos privados, quando estes são encapsulados em métodos para guardar e acessar seus valores.

Isso se chama Ocultação de Informação e é muito importante na programação.

As Classes encapsulam dados e comportamento em cima desses dados.



Fonte: tecnopode.

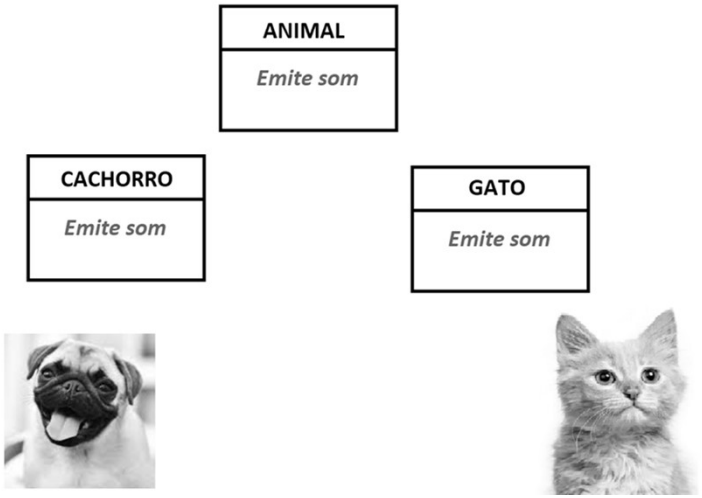
Conceitos de orientação a objetos

Poliformismo

Quando uma mensagem é enviada para um objeto, este deve ter um método definido para responder a essa mensagem. Em uma hierarquia de herança, todas as subclasses herdam as interfaces de sua superclasse.

No entanto, como toda subclasse é uma entidade separada, cada uma delas pode exigir uma resposta separada para a mesma mensagem.

Polimorfismo significa "muitas formas", é o termo definido em linguagens orientadas a objetos, que permite ao desenvolvedor usar o mesmo elemento de formas diferentes.



Fonte: elaborado pelo autor.

Classes e métodos em Python

Classes e métodos em Python

A Classes em Python

Utiliza-se a palavra reservada class para indicar a criação de uma classe, seguida do nome e dois pontos. No bloco identado devem ser implementados os atributos e métodos da classe.

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

Classes e métodos em Python

A Classes em Python

```python
class PrimeiraClasse:

    def imprimir_mensagem(self, nome): # Criando um método
        print(f"Olá {nome}, seja bem vindo!")
        objeto1 = PrimeiraClasse() # Instanciando um objeto do tipo PrimeiraClasse
        objeto1.imprimir_mensagem('João') #Invocando o método
```

Classes e métodos em Python

Construtor da classe __init__()

Atributos de instância, também chamadas de variáveis de instâncias. Esse tipo de atributo é capaz de receber um valor diferente para cada objeto.

Um atributo de instância é uma variável precedida com o parâmetro self, ou seja, a sintaxe para criar e utilizar é self.nome_atributo.

Ao instanciar um novo objeto, é possível determinar um estado

inicial para variáveis de instâncias por meio do método construtor da classe.

Em Python, o método construtor é chamado de ***__init__().***

Classes e métodos em Python

Construtor da classe __init__()

Criamos a classe Televisao, que possui um atributo de instância e três métodos, o primeiro dos quais é (__init__), aquele que é invocado quando o objeto é instanciado. Nesse método construtor, instanciamos o atributo volume com o valor 10, ou seja,

todo objeto do tipo Televisao será criado com volume = 10. Veja que o atributo recebe o prefixo self., que o identifica como variável de instância.

```
class Televisao:
    def __init__(self):
        self.volume = 10
    def aumentar_volume(self):
        self.volume += 1
    def diminuir_volume(self):
        self.volume -= 1

tv = Televisao()
print("Volume ao ligar a tv = ", tv.volume)
tv.aumentar_volume()
print("Volume atual = ", tv.volume)
```

Classes e métodos em Python

Variáveis e métodos privados

Em linguagens de programação OO, como Java e C#, as classes, os atributos e os métodos são acompanhados de modificadores de acesso, que podem ser: **public**, **private** e **protected**.

Em Python, **não** existem modificadores de acesso e todos os recursos são públicos. Para simbolizar que um atributo ou método é privado, por convenção, usa-se um sublinhado "_" antes do nome; por exemplo, _cpf, _calcular_desconto() (PSF, 2020a).

Classes e métodos em Python

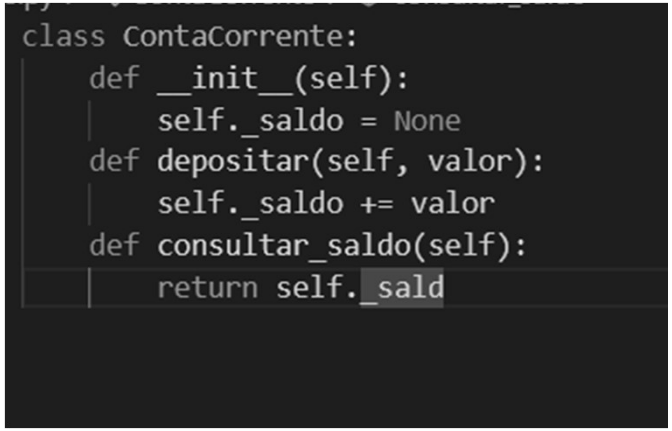
Variáveis e métodos privados

Conceitualmente, dado que um atributo é privado, ele só pode ser acessado por membros da própria classe. Portanto, ao declarar um atributo privado, precisamos de métodos que acessem e recuperam os valores ali guardados. Em Python, além de métodos para este fim, um atributo privado pode ser acessado por decorator.

Classes e métodos em Python

Variáveis e métodos privados

Implementamos a classe ContaCorrente, que possui dois atributos privados: _cpf e _saldo. Para guardar um valor no atributo cpf, deve-se chamar o método set_cpf, e, para recuperar seu valor, usa-se get_cpf.



Lembre-se: em Python, atributos e métodos privados são apenas uma convenção, pois, na prática, os recursos podem ser acessados de qualquer forma.

Classes e métodos em Python

HERANÇA EM PYTHON

Em Python, uma classe aceita múltiplas heranças, ou seja, herda recursos de diversas classes. A sintaxe para criar a herança é feita com parênteses após o nome da classe: *class NomeClasseFilha(NomeClassePai).*

Se for uma herança múltipla, cada superclasse deve ser separada por vírgula.

Classes e métodos em Python

HERANÇA EM PYTHON

```python
class Pessoa:
    def __init__(self):
        self.cpf = None
        self.nome = None
        self.endereco = None
```

```python
class Funcionario(Pessoa):
    def __init__(self):
        self.matricula = None
        self.salacio = None
    def bater_ponto(self):
        # código aqui
        pass
    def fazer_login(self):
        # código aqui
        pass
```

```python
f1 = Funcionario()
f1.nome = "Funcionário A"
print(f1.nome)

c1 = Cliente()
c1.cpf = "111.111.111-11"
print(c1.cpf)
```

Classes e métodos em Python

Métodos mágicos em Python

Quando uma classe é criada em Python, ela herda, mesmo que não declarado explicitamente, todos os recursos de uma classe-base chamada ***object***.

```
print(dir(Pessoa()))
```

Veja o resultado da função dir(), que retorna uma lista com os recursos de um objeto. A classe Pessoa, que explicitamente não tem nenhuma herança, possui uma série de recursos nos quais os nomes estão com underline (sublinhado). Todos eles são chamados de métodos mágicos e, com a herança, podem ser sobrescritos.

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
 '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__re
duce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'cp
f', 'endereco', 'nome']
```

Classes e métodos em Python

Método construtor na herança e sobrescrita

Na herança, quando adicionamos a função __init__(), a classe-filho

não herdará o construtor dos pais. Ou seja, o construtor da classe-filho sobrescreve (override) o da classe-pai. Para utilizar o construtor da classebase, é necessário invocá-lo explicitamente, dentro do construtor-filho, da seguinte forma:
ClassePai.__init__().

Herança múltipla

Python permite que uma classe-filha herde recursos de mais de

uma superclasse. Para isso, basta declarar cada classe a ser herdada

separada por vírgula.

Bibliotecas e módulos em Python

Bibliotecas e módulos em Python

Um módulo pode ser uma biblioteca de códigos, o qual possui diversas funções (matemáticas, sistema operacional. etc.) as quais possibilitam a reutilização de código de uma forma elegante e eficiente.

Bibliotecas e módulos em Python

Módulos e Bibliotecas em Python

Uma opção para organizar o código é implementar funções, contexto em que cada bloco passa a ser responsável por uma determinada funcionalidade. Outra forma é utilizar a orientação a objetos e criar classes que encapsulam as características e os comportamentos de um determinado objeto.

Conseguimos utilizar ambas as técnicas para melhorar o código, mas, ainda assim, estamos falando de toda a solução agrupada em um arquivo Python (.py).

Bibliotecas e módulos em Python

COMO UTILIZAR UM MÓDULO

Para utilizar um módulo é preciso importá-lo para o arquivo. Essa importação pode ser feita de maneiras distintas:

- import moduloXXText
- import moduloXX as apelido
- from moduloXX import itemA, itemB

A forma de importação também determina a sintaxe para utilizar a funcionalidade.

Bibliotecas e módulos em Python

Classificação dos módulos (bibliotecas)

Três categorias, cada uma das quais vamos estudar:

- Módulos built-in: embutidos no interpretador.
- Módulos de terceiros: criados por terceiros e disponibilizados via PyPI.
- Módulos próprios: criados pelo desenvolvedor

Bibliotecas e módulos em Python

- Módulos Built-in: Como estão embutidos no interpretador, esses módulos não precisam de nenhuma instalação adicional.

- Módulo random: Random é um módulo built-in usado para criar número aleatórios.

- Módulo os: OS é um módulo built-in usado para executar comandos no sistema operacional.

Bibliotecas e módulos em Python

- Módulo re: O módulo re (regular expression) fornece funções para busca de padrões em um texto. Uma expressão regular especifica um conjunto de strings que corresponde a ela. As funções neste módulo permitem verificar se uma determinada string corresponde a uma determinada expressão regular.

- Módulo datetime : Trabalhar com datas é um desafio nas mais diversas linguagens de programação. Em Python há um módulo built-in capaz de lidar com datas e horas. O módulo datetime fornece classes para manipular datas e horas.

Bibliotecas e módulos em Python

Módulos de terceiros

Programadores autônomos e empresas podem, com isso, criar uma solução em Python e disponibilizar em forma de biblioteca no repositório PyPI, o que permite que todos usufruam e contribuam para o crescimento da linguagem.

Para utilizar uma biblioteca do repositório PyPI, é preciso instalá-la.

Bibliotecas e módulos em Python

Módulos de terceiros

Para isso, abra um terminal no sistema operacional e digite:

pip install biblioteca [biblioteca é o nome do pacote que deseja instalar. Por exemplo: pip install numpy.]

Grande vantagem de usar bibliotecas é que elas encapsulam a complexidade de uma determinada tarefa, razão pela qual, com poucas linhas de códigos, conseguimos realizar tarefas complexas.

Bibliotecas e módulos em Python

Bibliotecas que têm sido amplamente utilizadas:

- Bibliotecas para tratamento de imagens
- Bibliotecas para visualização de dados
- Bibliotecas para tratamento de dados
- Bibliotecas para tratamento de textos
- Internet, rede e cloud
- Bibliotecas para acesso a bancos de dados
- Deep learning - Machine learning
- Biblioteca para jogos - PyGame

Bibliotecas e módulos em Python

Biblioteca requests

A biblioteca requests habilita funcionalidades do procotolo **HTTP**, como o get e o post. Dentre seus métodos, o get() é o responsável por capturar informação da internet. O método get() permite que você informe a URL de que deseja obter informação.

Sua sintaxe é: requests.get('https://XXXXXXX').

Para outros parâmetros dessa função, como autenticação, cabeçalhos, etc., consulte a documentação.

Exemplo:

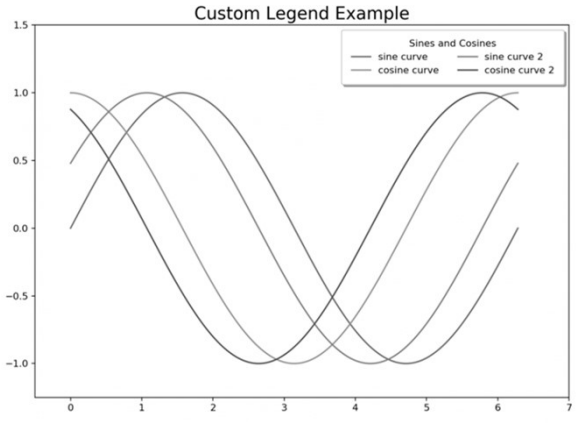
```
import requests

info = requests.get('https://api.github.com/events')
info.headers
```

Bibliotecas e módulos em Python

MATPLOTLIB

Matplotlib é uma biblioteca com funcionalidades para criar gráficos. A quantidade de funcionalidades embutidas em uma biblioteca traz um universo de possibilidades ao desenvolvedor. Consulte sempre a documentação e os fóruns para acompanhar as novidades.



Fonte: machinelearningplus.

Bibliotecas e módulos em Python

Módulos próprios

- Os códigos podem ser organizados em diversos arquivos com extensão .py, ou seja, em módulos.
- Cada módulo pode importar outros módulos, tanto os pertencentes ao mesmo projeto, como os built-in ou de terceiros.
- É dentro do módulo principal que vamos utilizar a funcionalidade de conexão para copiar um arquivo que está em um servidor para outro local.
- É importante ressaltar que, da forma pela qual fizemos a importação, ambos os arquivos .py precisam estar no mesmo nível de pasta. Se precisarmos usar o módulo utils em **vários projetos**, é interessante transformá-lo em uma biblioteca e disponibilizá-la via PyPI.

Aplicação de banco de dados com Python

Introdução a banco de dados

Os sistema de banco de dados podem ser divididos em duas categorias: banco de dados relacional e banco de dados NoSQL.

Na teoria base dos bancos de dados relacionais, os dados são persistidos em uma estrutura bidimensional, chamada de relação (que é uma tabela), que está baseada na teoria dos conjuntos pertencentes à matemática. Cada unidade de dados é conhecida como coluna, ao passo que cada unidade do grupo é conhecida como linha, tupla ou registro.

Introdução a banco de dados

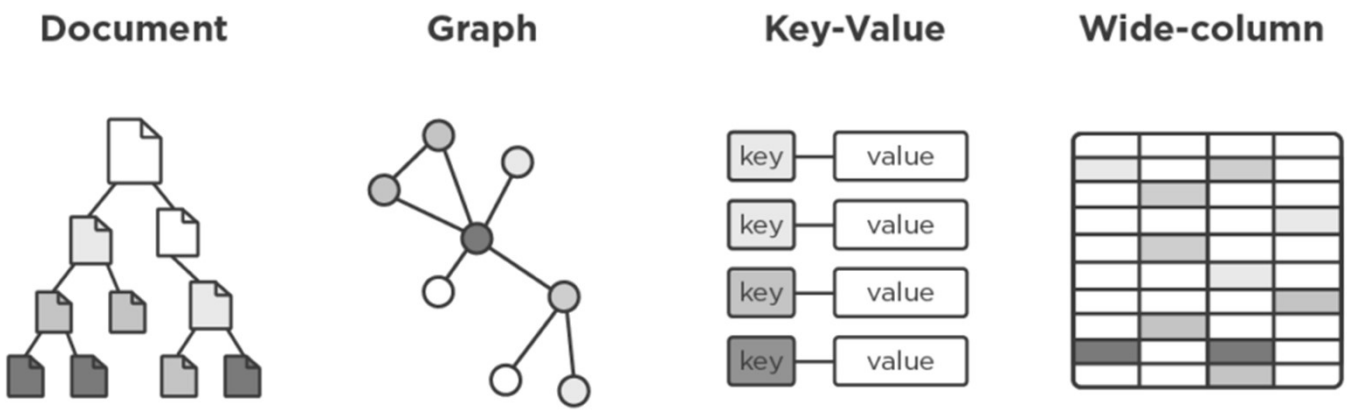
Exemplo	A	B	C	D	E
1	Id	Nome	Raça	Cor	Idade
2	1	Caramelo	Vira lata	Amarela	8
3	4	Xuxa	Dálmata	Branca	3
4	3	Lola	Golden Retriever	Amarela	4
5	2	Bolinha	Pinscher	Preta	12

Fonte: logap.

Introdução a banco de dados

NoSQL é usado para abordar a classe de bancos de dados que não seguem os princípios do sistema de gerenciamento de banco de dados relacional (RDBMS) e são projetados especificamente para lidar com a velocidade e a escala de aplicações.

Introdução a banco de dados



Fonte: logap.

Introdução a banco de dados

Linguagem de consulta estruturada - SQL

SQL, significa (structured query language), é a linguagem que permite aos usuários se comunicarem com banco de dados relacionais.

As instruções da linguagem SQL são divididas em três grupos: DDL, DML, DCL

Introdução a banco de dados

DDL (Data Definition Language). Fazem parte deste grupo as instruções destinadas a criar, deletar e modificar banco de dados e tabelas. Neste módulo vão aparecer comandos como *CREATE, o ALTER e o DROP.*

DML (Data Manipulation Language). Fazem parte deste grupo as instruções destinadas a recuperar, atualizar, adicionar ou excluir dados em um banco de dados. Neste módulo vão aparecer comandos como *INSERT, UPDATE e DELETE.*

DCL (Data Control Language). Fazem parte deste grupo as instruções destinadas a manter a segurança adequada para o banco de dados. Neste módulo vão aparecer comandos como *GRANT e REVOKE*.

Introdução a banco de dados

Conexão com banco de dados relacional

Para fazer a conexão e permitir que uma linguagem de programação se comunique com um banco de dados com a utilização da linguagem SQL, podemos usar as tecnologias **ODBC** (Open Database Connectivity) e **JDBC** (Java Database Connectivity).

A grande vantagem de utilizar as tecnologias ODBC ou JDBC está no fato de que uma aplicação pode acessar diferentes RDBMS sem precisar recompilar o código. Essa transparência é possível porque a comunicação direta com o RDBMS é feita por um driver. Um driver é um software específico responsável por traduzir as chamadas ODBC e JDBC para a linguagem do RDBMS

Introdução a banco de dados

Conexão de banco dados SQL em Python

Para se comunicar com um RDBMS em Python, podemos utilizar bibliotecas já disponíveis, com uso das quais, por meio do driver de um determinado fornecedor, será possível fazer a conexão e a execução de comandos SQL no banco.

Por exemplo, para se conectar com um banco de dados Oracle, podemos usar a biblioteca **cx-Oracle**, ou, para se conectar a um PostgreSQL, temos como opção o **psycopg2.**

Introdução a banco de dados

BANCO DE DADOS SQLITE

Essa tecnologia pode ser embutida em telefones celulares e computadores e vem incluída em inúmeros outros aplicativos que as pessoas usam todos os dias. Ao passo que a maioria dos bancos de dados SQL usa um servidor para rodar e gerenciar, o SQLite não possui um processo de servidor separado.

O SQLite lê e grava diretamente em arquivos de disco, ou seja, um banco de dados SQL completo com várias tabelas, índices, triggers e visualizações está contido em um único arquivo de disco.

O interpretador Python possui o módulo built-in **sqlite3**, que permite utilizar o mecanismo de banco de dados SQLite.

Introdução a banco de dados

Criando um banco de dados

O primeiro passo é importar o módulo **sqlite3**. Em razão da natureza do SQLite (ser um arquivo no disco rígido), ao nos conectarmos a um banco, o arquivo é imediatamente criado na pasta do projeto.

```
import sqlite3
conn = sqlite3.connect('aulaDB.db')
print(type(conn))
```

Introdução a banco de dados

Criando uma tabela

Agora que temos uma conexão com um banco de dados, vamos utilizar uma instrução DDL da linguagem SQL para criar a tabela fornecedor. O comando SQL que cria a tabela fornecedor está no código a seguir e foi guardado em uma variável chamada ***ddl_create***.

```
ddl_create = """
CREATE TABLE fornecedor (
id_fornecedor INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
nome_fornecedor TEXT NOT NULL,
cnpj VARCHAR(18) NOT NULL,
cidade TEXT,
estado VARCHAR(2) NOT NULL,
cep VARCHAR(9) NOT NULL,
data_cadastro DATE NOT NULL
);
"""
```

Introdução a banco de dados

Agora basta utilizar um mecanismo para que esse comando seja executado no banco. Esse mecanismo, segundo o PEP 249, deve estar implementado em um método chamado **execute()** de um objeto **cursor**.

Portanto, sempre que precisarmos executar um comando SQL no banco usando a linguagem Python, usaremos um cursor para construir essa ponte.

Além de criar uma tabela, também podemos excluí-la. A sintaxe para apagar uma tabela (e todos seus dados) é "*DROP TABLE table_name*".

Introdução a banco de dados

```
import sqlite3
conn = sqlite3.connect('aulaDB.db')

cursor = conn.cursor()
cursor.execute(ddl_create)
print(type(cursor))
print("Tabela criada!")
print("Descrição do cursor: ", cursor.description)
print("Linhas afetadas: ", cursor.rowcount)
cursor.close()
conn.close()
```

Introdução a banco de dados

CRUD - Create, Read, Update, Delete

CRUD é um acrônimo para as quatro operações de DML que podemos fazer em uma tabela no banco de dados. Podemos inserir informações (create), ler (read), atualizar (update) e apagar (delete).

Os passos necessários para efetuar uma das operações do CRUD são sempre os mesmos: (i) estabelecer a conexão com um banco; (ii) criar um cursor e executar o comando; (iii) gravar a operação; (iv) fechar o cursor e a conexão.

Introdução a banco de dados

Informações do banco de dados e das tabelas

Além das operações de CRUD, é importante sabermos extrair informações estruturais do banco de dados e das tabelas. Os comandos necessários para extrair essas informações podem mudar entre os bancos.

Introdução a banco de dados

CREATE: Uma maneira mais prática de inserir vários registros é passar uma lista de tuplas, na qual cada uma destas contém os dados a serem inseridos em uma linha. Nesse caso, teremos que usar o método executemany() do cursor.

READ: Recuperar os dados. Também precisamos estabelecer uma conexão e criar um objeto cursor para executar a instrução de seleção. Ao executar a seleção, podemos usar o método fetchall() para capturar todas as linhas, através de uma lista de tuplas.

UPDATE: Ao inserir um registro no banco, pode ser necessário alterar o valor de uma coluna, o que pode ser feito por meio da instrução SQL UPDATE

DELETE: Ao inserir um registro no banco, pode ser necessário removê-lo no futuro, o que pode ser feito por meio da instrução SQL DELETE.

Recapitulando

Recapitulando

- Classes e Métodos em Python
- Bibliotecas e Módulos em Python
- Aplicação de banco de dados com Python