CLASSES E OBJETOS

Classes Abstratas, Métodos Mágicos, Duck Typing, Criando Exceções com Classes.

Classes Abstratas: As classes abstratas são as que não permitem realizar qualquer tipo de instância. São classes feitas especialmente para serem modelos para suas classes derivadas. As classes derivadas, via de regra, deverão **sobrescrever os métodos** para realizar a implementação dos mesmos.

Vamos criar a classe abstrata Animal. Ela será a classe que não poderá ser instanciada.

Por convenção temos que sempre importar os métodos da classe Built-in ABC

```
from abc import ABC, abstractmethod
```

CLASSES E OBJETOS

```
class Animal(ABC):
    def __init__(self,tipo,especie):
        self._tipo = tipo
        self.especie = especie
```

Para tornar uma classe abstrata de definer um método padrão para as subclasses ou classes filhas utlizamos o método decorador @abstractmethod

Criando um método abstrato padrão.

```
@abstractmethod
    def raca(self):
        pass
```

Tentando instanciar classe abstrata.

```
an1 = Animal("Mamifero","Baleias")
print(an1.especie)
```

CLASSES E OBJETOS

Criando a classe filha chamada cachorro.

```
from classesabstratas import Animal

class Cachorro(Animal):
    def __init__(self,tipo,especie,cor):
        super().__init__(tipo,especie)
        self.cor = cor
```

É obrigatório sobrepor o método raca pois o mesmo herda de uma classe abstrata.

```
    def raca(self,raca):
        return self.raca
```

CLASSES E OBJETOS

Criando o arquivo main e importando a classe e instanciando os objetos.

```
from classefilhaabstrata import Cachorro

cachorro1 = Cachorro("Mamifero","Canino","Preta")
cachorro1.raca("BullDog")

print(cachorro1.cor)
print(cachorro1.especie)
```

JCAVI
TREINAMENTOS EM TI

EXERCÍCIO DE FIXAÇÃO

Crie uma classe abstrata com um método abstrato. Tente instanciar algum objeto nessa classe.

Crie uma classe filha da classe abstrata sobrescrevendo o método abstrato. E instancie objetos.

CLASSES E OBJETOS

Métodos Mágicos:

Métodos mágicos são todas as funções que usamos o dunder __ (Duplo Underline)
Todas as classes herdam por padrão da classe object.

Alguns Métodos Mágicos que já vimos:

__init__ constrói uma classe. (Método Construtor)
__del__ deleta um objeto instanciado.
__dict__ Exibe os valores e atributos de uma instancia em dicionário.

Outros métodos mágicos utilizados.

__repr__ representa objetos de uma classe

__len__ Retorna o comprimento do objeto (se for um container)

CLASSES E OBJETOS

```
class Pessoa():
    def __init__(self,nome,sobrenome,rg):
        self.nome = nome
        self.sobrenome = sobrenome
        self.rg = rg

Criar o metodo repr
    def __repr__(self):
        return self.nome

Criar o metodo len

    def __len__(self):
        return (self.rg)
```

CLASSES E OBJETOS

Instanciando os objetos

p1 = Pessoa("João","Cavichiolli",443222)

Testar o método de representação
print(p1)

Testar o método len

print(len(p1))

Listar Metodos Magicos

print(dir())

print(dir(__builtins__))

CLASSES E OBJETOS

Duck Typing: O que é ?

Duck typing é um estilo de codificação de linguagens dinamicamente tipadas onde o tipo de uma variável nã
o importa, contanto que seu comportamento seja o desejado.
O nome "tipagem de pato" vem da expressão "se anda como pato, nada como um pato e faz quackcomo um
pato, então provavelmente é um pato".
Se um objeto ou tipo de dados são similares provavelmente eles terão os mesmo métodos.

Vamos criar exemplos.

```
lista = [1,42,3,35]
lista2 = ["Joao","Pedro","Juca"]
dic = {"A": "Abobora"}

num1 = 45
nome = "Gilberto"
```

CLASSES E OBJETOS

Vamos analisar os métodos para cada tipo de objetos ou coleções de dados. Onde as funções
E métodos se aplicando para tipos parecidos de dados.

```
print(len(lista))
print(len(nome))
print(len(dic))
print(len(num1))
print(sum(lista))
print(sum(lista2))
```

CLASSES E OBJETOS

Criando classes como exceções. Podemos utilizar classes para criar exceções. Utilizando os mesmos conceitos que aprendemos de try,except e raise.

Para criar uma exceções herdamos da classe excepction.

```
class ErroProg(Exception):
    pass

def teste():
    raise ErroProg("Mensagem de Erro")

def prog():
    try:
        teste()
    except ErroProg as msg :
        print(msg)

prog()
```

TRABALHANDO COM BANCO DE DADOS

No Python trabalhamos com banco de dado no modelo de CRUD.

CRUD (acrónimo do inglês **Create, Read, Update and Delete**) são as quatro operações básicas (criação, consulta, atualização e destruição de dados) utilizadas em bases de dados relacionais (RDBMS) fornecidas aos utilizadores do sistema.

Vamos utilizar o sqllite3 que é nativo nas bibliotecas do python3.

Vamos utilizar também o conceito de cursores de banco de dados.

Um **cursor** de banco de dados é uma estrutura de controle que permite percorrer sobre os registros em um banco de dados. Os cursores facilitam o processamento subsequente em conjunto com a iteração, tal como recuperação, adição e remoção de registros de banco de dados. A característica de iteração do cursor de banco de dados faz os cursores semelhantes ao conceito de um iterador em python.

TRABALHANDO COM BANCO DE DADOS

Utilizando o Banco de Dados. Caso ele não exista será criado.

conn = sqlite3.connect('cadastro.db’)

Criando o Cursor e abrindo a conexão com o banco de dados

cursor = conn.cursor()

Como já criamos o banco de dados o segundo passo é criamos a nossa tabela do banco de dados.

Para vamos utilizar o cursor.execute()

Cursor Execute cria ações em linguagem sql para comunicação direta com o banco de dados

TRABALHANDO COM BANCO DE DADOS

```
cursor.execute( """
create table cadastro_clientes (
id integer NOT NULL  PRIMARY KEY AUTOINCREMENT,
nome varchar(100) not null,
sobrenome varchar(100) not null,
cpf varchar(11) not null
);
""" )
```

Fechando a Conexão com Banco de Dados

```
conn.close()
```

TRABALHANDO COM BANCO DE DADOS

Trabalhando com Inserts no Banco de dados.

Vamos criar um novo arquivo python para trabalhar os inserts.

```
import sqlite3

conn = sqlite3.connect('cadastro.db')

cursor = conn.cursor()

cursor.execute("""
INSERT INTO cadastro_clientes (nome,sobrenome,cpf)
VALUES ('João', 'Cavichiolli','7773554444') """)

cursor.execute("""
INSERT INTO cadastro_clientes (nome,sobrenome,cpf)
VALUES ('Valéria', 'Cunha','553333') """)
```

TRABALHANDO COM BANCO DE DADOS

Gravar as alterações no banco de dados.

```
conn.commit()
```

Trabalhando com SELECTS no banco de dados. Além de usarmos o cursos vamos usar a função fetchall() para buscar os registros.

```
cursor.execute(""" select * from cadastro_clientes""")
```

Trazendo os registros com for

```
for regs in cursor.fetchall():
    print(regs)
```

Encerrando a conexão com o banco de dados.

```
conn.close()
```

TRABALHANDO COM BANCO DE DADOS

Trabalhando os inserts com Inputs de Usuário.

```
import sqlite3

conn = sqlite3.connect('cadastro.db')

cursor = conn.cursor()
```

Passando as informações de Input

```
i_nome = input('Nome: ')
i_sobrenome = input('Sobrenome: ')
i_cpf = input('CPF: ’)
```

TRABALHANDO COM BANCO DE DADOS

Fazendo o insert no banco. Utilizando o método ?.? Nos valores do banco de dados

```
cursor.execute(""INSERT INTO cadastro_clientes (nome, sobrenome, cpf) VALUES (?,?,?)
""", (i_nome, i_sobrenome, i_cpf))
```

Executando o Select Para Testes de Inserção de Registos

```
cursor.execute(""" select * from cadastro_clientes """)

for reg in cursor.fetchall():
    print (reg)
```

Gravar as configurações no Banco.

```
conn.commit()
conn.close()
```

TRABALHANDO COM BANCO DE DADOS

Trabalhando Updates em Banco de Dados

```
import sqlite3

conn = sqlite3.connect('cadastro.db')

cursor = conn.cursor()

Passando as informações de Input
u_id = input("Insira o Id de Cadastro: ")
u_nome = input(' Insira o Novo Nome: ')
u_cpf = input(' Insira o Novo CPF: ’)

cursor.execute("""
UPDATE cadastro_clientes
SET nome = ?, cpf = ?
WHERE id = ?
""", (u_nome, u_cpf, u_id))
```

TRABALHANDO COM BANCO DE DADOS

Gravando Banco de Dados

```
conn.commit()
```

Executando a Leitura da Alteração com o Select

```
cursor.execute(""" select * from cadastro_clientes
where id = ? """,(u_id,))

for reg in cursor.fetchall():
    print (reg)
```

Encerrando a Conexão com Banco de dados

```
conn.close()
```

TRABALHANDO COM BANCO DE DADOS

Trabalhando Delete com Banco de Dados.

```
import sqlite3

conn = sqlite3.connect('cadastro.db')

cursor = conn.cursor()

Passando as informações de Input
u_id = input("Insira o Id de Cadastro Para Excluir: ")

cursor.execute("""
delete from cadastro_clientes
WHERE id = ?
""", (u_id,))
```

TRABALHANDO COM BANCO DE DADOS

Gravando Banco de Dados
conn.commit()

Validando com o Select se o registro foi excluído

```
cursor.execute(""" select * from cadastro_clientes
where id = ? """,(u_id,))

for reg in cursor.fetchall():
    print (reg)
```

Fechando conexão com Banco de dados
conn.close()

JCAVI
TREINAMENTOS EM TI

EXERCICIO FIXAÇÃO

Importe o módulo do sqlite3 e crie um banco de dados e uma tabela para armazenar informações.
Execute as operações CRUD dentro dessa tabela criada.

Utilize as opções de Insert, Update,Delete e não esqueça de utilizar o Select para validar as informações e os registros do banco.