Sumário

[1.Python 2](#_Toc151194224)

[1.1 Propriedades 2](#_Toc151194225)

[1.1.1 Funções 2](#_Toc151194226)

[1.2 Strings 2](#_Toc151194227)

[1.3 Lista [ ] 2](#_Toc151194228)

[1.4 Tupla ( ) 2](#_Toc151194229)

[1.5 IF - ELIF – ELSE 2](#_Toc151194230)

[1.6 While 2](#_Toc151194231)

[1.7 For 2](#_Toc151194232)

[1.8 Funções 2](#_Toc151194233)

[1.8.1 Definir uma Função 2](#_Toc151194234)

[1.8.2 Escopo da Função 2](#_Toc151194235)

[1.8.3 return 3](#_Toc151194236)

[1.8.4 \*args (Argumentos não nomeados) 4](#_Toc151194237)

[2. SQL 5](#_Toc151194238)

[2.1 Propriedades 5](#_Toc151194239)

[2.1.1 Tipos Primitivos 5](#_Toc151194240)

[2.2 Create Database 6](#_Toc151194241)

[2.3 Create Table 6](#_Toc151194242)

[2.4 Insert Into 7](#_Toc151194243)

[2.5 Alter Table 8](#_Toc151194244)

[2.5.1 Renomear tabela 8](#_Toc151194245)

[2.5.2 Adicionar uma coluna na tabela 8](#_Toc151194246)

[2.6 Select 9](#_Toc151194247)

# 1.Python

## 1.1 Propriedades

### 1.1.1 Funções

## 1.2 Strings

## 1.3 Lista [ ]

## 1.4 Tupla ( )

## 1.5 IF - ELIF – ELSE

## 1.6 While

## 1.7 For

## 1.8 Funções

### 1.8.1 Definir uma Função

### 1.8.2 Escopo da Função

Escopo significa o local onde aquele código pode atingir.

Existe o escopo global e local.

O escopo global é o escopo onde todo o código é alcançável.

O escopo local é o escopo onde apenas nomes do mesmo local

podem ser alcançados.

x = 1

def escopo1():

    x = 2

    def escopo2():

        x = 3

        print(x)

    print(x)

    escopo2()

print(x)

escopo1()

print(x)

Saída

1

2

3

1

O valor da variável fora da função não altera o valor da variável dentro da função, e vice-versa. É possível alterar variáveis de escopo externo em escopo interno usando a propriedade global.

def escopo1():

    global x

    x = 2

    def escopo2():

        global x

        x = 3

        print(x)

    print(x)

    escopo2()

print(x)

escopo1()

print(x)

Saída

1

2

3

3

### 1.8.3 return

Retorna os valores das funções

def soma(*x*, *y*):

    if *x* > 10:

        return [10, 20]

    return *x* + *y*

soma1 = soma(2, 2)

soma2 = soma(3, 3)

print(soma1)

print(soma2)

print(soma(11, 55))

saída

4

6

[10, 20]

### 1.8.4 \*args (Argumentos não nomeados)

args - Argumentos não nomeados

\* - \*args (empacotamento e desempacotamento)

Lembre-te de desempacotamento:

x, y, \*resto = 1, 2, 3, 4

print(x, y, resto)

def soma(\**args*):

    total = 0

    for numero in *args*:

        total += numero

    return total

print(soma(1, 2, 3))

saída

6

# 2. SQL

## 2.1 Propriedades

### 2.1.1 Tipos Primitivos

Em SQL, os tipos primitivos (ou tipos de dados básicos) são usados para representar valores simples, como números inteiros, números de ponto flutuante, caracteres, datas e assim por diante. Os tipos primitivos podem variar um pouco de um sistema de gerenciamento de banco de dados (SGBD) para outro, mas geralmente incluem os seguintes:

1. INTEGER ou INT: Representa números inteiros. Dependendo do SGBD, pode haver variações, como TINYINT, SMALLINT, MEDIUMINT, BIGINT, entre outros, para armazenar inteiros de diferentes tamanhos.

2. DECIMAL ou NUMERIC: Usado para armazenar números de ponto flutuante com precisão fixa. Pode especificar o número de dígitos na parte inteira e decimal.

3. FLOAT ou REAL: Representa números de ponto flutuante com precisão de ponto flutuante. Pode ser usado para valores aproximados.

4. CHAR: Armazena cadeias de caracteres com tamanho fixo. Você precisa especificar o tamanho máximo.

5. VARCHAR: Armazena cadeias de caracteres com tamanho variável. A alocação de espaço é feita com base no tamanho real dos dados.

6. DATE: Armazena datas, sem incluir informações de tempo.

7. TIME: Armazena horários, sem incluir informações de data.

8. DATETIME: Armazena valores de data e hora.

9. TIMESTAMP: Armazena uma marca de data e hora, geralmente usada para rastrear a última modificação de um registro.

10. BOOLEAN ou BIT: Armazena valores lógicos, como verdadeiro (TRUE) ou falso (FALSE).

11. BINARY: Armazena dados binários de tamanho fixo.

12. VARBINARY: Armazena dados binários de tamanho variável.

13. BLOB: Armazena objetos binários grandes, como imagens ou arquivos.

14. CLOB: Armazena objetos de caracteres grandes, como texto longo.

Lembre-se de que a disponibilidade de tipos de dados e a sintaxe exata podem variar entre os SGBDs, como MySQL, PostgreSQL, SQL Server, Oracle, entre outros. Certifique-se de consultar a documentação do seu SGBD específico para obter informações detalhadas sobre os tipos de dados suportados.

## 2.2 Create Database

create database meu\_banco character set utf8 collate utf8\_general\_ci;

Explicando o código: Os comandos character set utf8 e collate utf8\_general\_ci são usados para colocar o banco no padrão da língua portuguesa. Para usar o banco criado você pode usar o seguinte comando: use meu\_banco;

## 2.3 Create Table

create table clientes (

id int not null auto\_increment,

nome varchar(30) not null,

nascimento date,

sexo enum('M','F'),

peso decimal(5,2),

altura decimal(3,2),

nacionalidade varchar(20) default 'Brasil',

primary key (id) )

default charset utf8;

Explicando o código:

not null - Campo obrigatório, não aceita vazio

enum('M','F') - Recebe apenas M ou F

decimal(5,2) - 5 casas com 2 casas após a virgula. Ex:102,85

default 'Brasil' - Se não digitar nada, o padrão é Brasil

## 2.4 Insert Into

Inserindo Dados na Tabela

insert into clientes

(id, nome, nascimento, sexo, peso, altura, nacionalidade)

values

('1', 'Jose', '19993-03-13', 'M', '65', '1.75', ‘Brasil’);

Como o id é auto incrementável, ele pode ser retirado. Como o Brasil foi definido como padrão, ele pode ser substituído por default.

insert into clientes

(nome, nascimento, sexo, peso, altura, nacionalidade)

values

('Jose', '1993-03-13', 'M', '65', '1.75', default);

Se os dados então sendo inseridos na mesma ordem dos campos, os campos podem ser omitidos.

insert into clientes values

(default, 'Jose', '19993-03-13', 'M', '65', '1.75', default);

Para mostrar os dados de cada pessoa, use o comando:

select \* from clientes;

Pode inserir diversos dados de uma vez da seguinte forma:

insert into clientes values

(default, 'John', '2000-08-30', 'M', '100.3', '1.85', 'EUA'),

(default, 'Clara', '2004-01-3', 'F', '59', '1.69', 'Brasil'),

(default, 'João', '1995-12-2', 'M', '80', '1.78', 'Espanha');

select \* from clientes;

## 2.5 Alter Table

### 2.5.1 Renomear tabela

alter table clientes rename to pessoas;

retornando

alter table pessoas rename to clientes;

### 2.5.2 Adicionar uma coluna na tabela

alter table clientes add column proficao varchar(10);

adiciona uma coluna no final da tabela.

O parâmetro column é opcional.

alter table clientes add column proficao varchar(10) after nome;

Adiciona uma coluna em um lugar especifico da tabela.

alter table clientes add column proficao varchar(10) first;

Adiciona uma coluna no início da tabela.

alter table clientes drop column proficao;

Apaga uma coluna da tabela.

alter table clientes modify column proficao varchar(20) not null default'';

Modifica o tipo primitivo e os parâmetros em uma coluna da tabela.

alter table clientes change column proficao prof varchar(20);

Modifica o nome e todas as especificações da coluna da tabela.

describe pessoas;

ou

desc pessoas;

mostra tabela.

create table if not exists cursos(

nome varchar(30) not null unique,

descricao text,

carga int unsigned,

totaulas int unsigned,

ano year default '2016'

) default charset=utf8;

alter table cursos add column idcurso int first;

alter table cursos add primary key(idcurso);

Adiciona uma chave primaria na tabela.

select \* from pessoas;

## 2.6 Update - Delete – Truncate

## 2.7 Select