aula

June 24, 2021

1 Curso Python

- Linguagem de alto nível > Blocos de código em Python devem ter dois pontos (:) e o conteúdo do bloco deve estar identado. ## Comandos Utilitários #### ***dir()*** | exibe todos os tipos de atributos e funções/métodos de um tipo de dado ou variável.

Exemplo:

```
dir("Fabio")
```

help() **| exibe a documetação de como utilizar os atributos/propriedades e funções/métodos disponíveis.** Exemplo:

```
help("Fabio".lower)
```

1.0.1 *type()* | exibe o tipo de dado de uma variável.

Exemplo:

```
type(7.0)
```

1.1 Operadores aritméticos

Subtração: 7 - 4 = 3

Adição: 7 + 4 = 11

Multiplicação: 7 * 4 = 28

Divisão: 7 / 4 = 1.75 # Aqui o resultado é um Float e se precisar do resultado em Inteiro deveira fazer *casting*. Mas o Python nós dá a solução Pythonica.

Módulo: 7 % 4 = 3

Formas Phytonicas:

Divisão Phytônica: 7 // 4 = 1

Exponenciação: 3 ** 3 = 9

Dica:

1000000000 | 1bi | Escrever números de forma mais legível de forma Phytonica: 1_000_000_000 o resultado final será o mesmo.

1.2 Tipos de dados

Integer => 7

Float => 7.0

Complex => 7j (basta adicionar o **j** ao lado do número)

Boolean => True | False

String => Texto

2 Estrutura condicional

if | elif | else

```
if idade < 16:
    exit()
elif idade < 18:
    "Acesso limitado"
else
    "Acesso total"
```

```
# esse exemplo é o in_array do PHP
if 7 in range(10):
    print("encontrado")
else:
    print("Não encontrado")
```

3 Operadores lógicos

3.1 Operadores binários

and | or | is

```
if ativo and logado:
    "Bem vindo"
elif ativo or logado:
    "Acesso negado"

if ativo is True:
    "Acesso autorizado"
```

3.2 Operadores unários

not

```
if not ativo:
    "Acesso negado"
```

4 Estrutura de repetição

loop for

```
for coluna in tabela:
    print(coluna)
```

loop while

```
num = 1
while num < 10:
    print(num)
    num = num + 1
```

[ ]:
```
tabela = range(10, 20)
for v in enumerate(tabela):
    print(v)
```

5 Collection | Coleções

lista vetor/matriz são a mesma coisa que *array*. Em Python é dinâmico e podem receber qualquer tipo de dado como valor. `lista = [] ou lista = list() lista = list(range(10))`

Adicionar 1:1 elemento na lista => lista.append()

Adicinar 1:N elementos na lista => lista.extend([])

Adicionar 1:1 elemento em uma posição especifica da lista => lista.insert(posicao, valor)

Inverter a ordem => lista.reverse()

Inverter a ordem => lista[::-1]

Ordernar a lista => lista.sort()

Copiar uma lista => lista.copy()

Tamanho de uma lista => len(lista)

Remover último elemento da lista => lista.pop() # Remove e retorna o elemento removido.

Remover elemento pela posição => lista.pop(2) # Se não houver o índice informado dá error

Limpar a lista => lista.clear()

Criar lista com indice => list(enumerate(range(inicio,fim)))

Encontrar o índice através do valor => lista.index(valor) # retorna a primeira ocorrencia # Se não existe gerar error

Encontrar o índice através do valor e posição => lista.index(valor, posicao)

Encontrar o índice através do valor e intervalo (posição) => lista.index(valor, pos_ini, pos_fim)

Retornar a partir de uma posição inicial => lista[1:]

Retornar dos elementos => lista[::]

tupla

dicionário

map

conjunto

5.1 Informações úteis

range(1,10) -> utilizado para gerar sequência númerica.

Exemplos: range(valor_final)

range(inicio, fim)

range(inicio, fim, passo) O passo é o valor de incremento ou decremento.

enumerate() -> Retorna (index, value)

Separa a string pelo espaço como default => string.split()

Para definir o caracter => string.split(';')

Juntando elementos com separador => "-".join(lista)

[26]:
```
lista = [2,4,6]
lista[:3]
```

[26]:
```
[2, 4, 6]
```

[ ]:

[ ]: