



Paradigmas de linguagens de programação em python

Professores:

Sebastião Rogério feat. Kayo Monteiro

Grupo Whatsapp



PLP - Unifavip 2022.2

Grupo do WhatsApp



Agenda

01

Variáveis
Declaração Múltiplas

03

Listas, Tuplas e
Dicionários

02

Estruturas
Condicionais

04

Exercícios o/



01

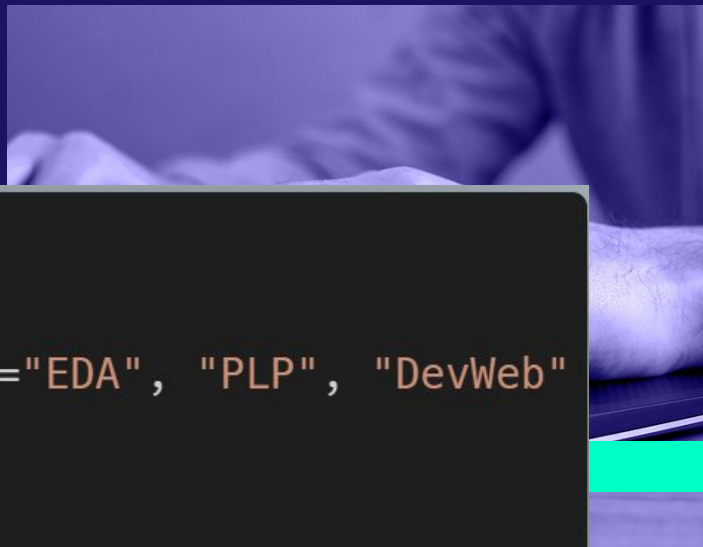
Declarações Múltiplas

Python – Variáveis

Em Python, é possível declarar várias variáveis de uma única vez:



```
disciplina1, disciplina2, disciplina3 = "EDA", "PLP", "DevWeb"  
print(disciplina1)  
print(disciplina2)  
print(disciplina3)
```



Python – Variáveis

Podemos também, atribuir o mesmo valor a mais de uma variável diferente:



```
disciplina1 = disciplina2 = disciplina3 = "PLP - Unifavip 2022"  
print(disciplina1)  
print(disciplina2)  
print(disciplina3)
```

Python – Variáveis

Podemos atribuir vários valores em **tipos** de variáveis diferentes de uma única vez? Esse código funciona?



```
nome, idade, estudante = "Bruce", 30, True  
print(nome)  
print(idade)  
print(estudante)
```

Python – Palavras Reservadas

<code>False</code>	<code>await</code>	<code>else</code>	<code>import</code>	<code>pass</code>
<code>None</code>	<code>break</code>	<code>except</code>	<code>in</code>	<code>raise</code>
<code>True</code>	<code>class</code>	<code>finally</code>	<code>is</code>	<code>return</code>
<code>and</code>	<code>continue</code>	<code>for</code>	<code>lambda</code>	<code>try</code>
<code>as</code>	<code>def</code>	<code>from</code>	<code>nonlocal</code>	<code>while</code>
<code>assert</code>	<code>del</code>	<code>global</code>	<code>not</code>	<code>with</code>
<code>async</code>	<code>elif</code>	<code>if</code>	<code>or</code>	<code>yield</code>




02

Estruturas Condicionais

Estruturas Condicionais

complexas

- Podemos comparar mais de uma condição dentro do **IF**:



```
periodo = int(input('digite qual e o seu periodo: '))
faculdade = "Unifavip"

if (faculdade == "Unifavip") and periodo > 1:
    print("Voce eh um estudante da %s e nao eh mais novato" %faculdade)
else:
    print("Voce nao eh estudante da %s ou eh um novato" %faculdade)
```


Estruturas Condicionais

Complexas – Prática

- Faça um programa que pergunte a **idade** do usuário e se ele está **acompanhado** de um adulto para determinar a entrada dele em um local. **Se** a idade for maior que 18 **ou** ele esteja acompanhado de um adulto a **entrada será liberada, se não a entrada é proibida**

Estruturas Condicionais

Complexas - ?Solução?



```
idade = int(input('digite qual sua idade: '))
acompanhado = input('voce esta acompanhado de um adulto Sim ou Nao: ')

if (acompanhado == "Sim") or idade > 18:
    print("Entrada liberada")
else:
    print("Entrada proibida")
```



03

Listas, Dicionarios e Tuplas

Coleções em Python

- Os 3 principais tipos de coleções de dados que permitem o armazenamento de informações são:
 - **Lista**
 - **Tupla**
 - **Dicionário**

Coleções em Python

- Os 3 principais tipos de coleções de dados que permitem o armazenamento de informações são:
 - **Lista** -> Ordenada e Mutável, que permite elementos duplicados.
 - **Tupla** -> Ordenada e Imutável, que permite elementos duplicados
 - **Dicionário** -> Não ordenados, mutáveis e indexáveis, que não permite duplicidade”;

Listas – List

- Os itens da lista são ordenados, alteráveis e permitem valores duplicados.
- Os itens da lista são indexados, o primeiro item tem índice [0], o segundo item tem índice [1] etc.
- Quando dizemos que as listas estão ordenadas, significa que os itens têm uma ordem definida, e essa ordem não será alterada. Se você adicionar novos itens a uma lista, os novos itens serão colocados no final da lista.

Listas – List

- Criando uma lista:



```
disciplinas = ["EDA", "PLP", "DevWeb", "EngSoft"]  
print(disciplinas)
```

Listas – List

- A lista é mutável, o que significa que podemos alterar, adicionar e remover itens em uma lista após ela ter sido criada.
- Como as listas são indexadas, as listas podem ter itens com o mesmo valor:



```
disciplinas = ["EDA", "PLP", "DevWeb", "EDA", "EngSoft"]  
print(disciplinas)
```

Listas – List

- Uma lista pode conter mais de um tipo de dados:



```
disciplinas = ["EDA", 2022, "DevWeb", True, "EngSoft"]  
print(disciplinas)
```

Qual será o valor printado do comando: **print(disciplinas[3])**

Listas – List

- Uma lista pode conter mais de um tipo de dados:



```
disciplinas = ["EDA", 2022, "DevWeb", True, "EngSoft"]  
print(disciplinas)
```

Qual será o valor printado do comando: **print(disciplinas[3])**

Listas – List [Métodos]

append() Adiciona um elemento no final da lista

clear() Remove todos os elementos da lista

copy() Retorna uma cópia da lista

count() Retorna o número de elementos com o valor especificado

extend() Adiciona os elementos de uma lista (ou qualquer iterável), ao final da lista atual

Listas – List [Métodos]

index() Retorna o índice do primeiro elemento com o valor especificado

insert() Adiciona um elemento na posição especificada

pop() Remove o elemento na posição especificada

remove() Remove o item com o valor especificado

reverse() Inverte a ordem da lista

sort() Classifica a lista

Listas – List [Prática]

- 0) Crie uma lista com três valores = maca, banana e cereja.
- 1) Imprima o segundo item da lista de frutas:
- 2) Altere o valor de "maca" para "kiwi", na lista de frutas;
- 3) Use o método append para adicionar "laranja" à lista de frutas;
- 4) Use o método de inserção para adicionar "limão" como o segundo item na lista de frutas;
- 5) Use o método remove para remover "banana" da lista de frutas.


Listas - List [Solução]



```
fruits = ["apple", "banana", "cherry"] #Resposta 0
print(fruits[1]) #Resposta 1
fruits[0] = "kiwi" #Resposta 2
fruits.append("orange") #Resposta 3
fruits.insert(1, "lemon") #Resposta 4
fruits.remove("banana") #Resposta 5
```



Listas – List

- É possível fazer a concatenação de listas:



```
list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]
```

```
list1.extend(list2)  
print(list1)
```




```
list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]
```

```
for x in list2:  
    list1.append(x)
```

```
print(list1)
```

Listas – List

- É possível fazer a concatenação de listas:



```
list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
  
list3 = list1 + list2  
print(list3)
```



Tuplas

Tuplas – Tuple

- É uma coleção de dados ordenada e imutável, que permite elementos duplicados. As tuplas podem ser visualizadas como listas:



```
disciplinas = ("EDA", "PLP", "DevWeb", "Cloud", "BD")  
print(disciplinas)
```

Tuplas – Tuple

- Itens de tupla são indexados, o primeiro item tem índice [0], o segundo item tem índice [1] etc.
- Quando dizemos que as tuplas estão **ordenadas**, significa que os itens têm uma ordem definida, e essa ordem não será alterada.
- As tuplas são **imutáveis**, o que significa que não podemos alterar, adicionar ou remover itens após a criação da tupla.
- Como as tuplas são indexadas, elas podem ter itens com o mesmo valor

Tuplas – Tuple

É possível criar uma tupla com apenas um item:




```
fruta = ("apple",)  
print(type(fruta))  
  
frutas = ("apple")  
print(type(frutas))
```

Tuplas – Tuple [Pratica]

Use um intervalo de índices para imprimir o terceiro, quarto e quinto item na tupla. frutas = ("maca", "banana", "cereja", "laranja", "morango", "melao", "manga")

Tuplas – Tuple [Pratica]

Use um intervalo de índices para imprimir o terceiro, quarto e quinto item na tupla. frutas = ("maca", "banana", "cereja", "laranja", "morango", "melao", "manga")



```
frutas = ("maca", "banana", "cereja",  
          "laranja", "morango", "melao", "manga")  
print(frutas[2:5])
```




Dicionários

Dicionários

Os dicionários são usados para armazenar valores de dados em pares **key:value**.

Um dicionário é uma coleção **ordenada***, mutável e que não permite duplicatas.

***A partir da versão 3.7 do Python, os dicionários são ordenados.**
No Python 3.6 e anteriores, os dicionários não são ordenados.

Dicionários

Exemplo:



```
carro = {  
    "marca": "Ford",  
    "modelo": "Mustang",  
    "ano": 1964  
}  
print(carro)
```



```
carro = {  
    "marca": "Ford",  
    "modelo": "Mustang",  
    "ano": 1964,  
    "ano": 2022  
}  
print(carro)
```

Dicionários

Podemos ter dicionários nos quais as chaves são associadas a listas ou mesmo a outros dicionários como valores:



```
carro = {  
    "marca": "Ford",  
    "modelo": "Mustang",  
    "ano": 1964,  
    "cores": ["vermelho", "preto", "branco"]  
}  
print(carro)
```



Resumo

Resumo – Coleção de Dados

Lista: inicializada com `[]`.

Tupla: inicializada com `()`.

Dicionário: inicializada com `{}`.

Possuem uma série de operações equivalentes para acesso, checagem de tamanho, etc

Referências

- <https://www.devmedia.com.br/python-estrutura-de-repeticao-while/38546#:~:text=A%20estrutura%20de%20repeti%C3%A7%C3%A3o%20%C3%A9,documenta%C3%A7%C3%A3o%20abordaremos%20o%20comando%20while.>
- <https://www.devmedia.com.br/estruturas-de-condicao-em-python/37158>
- <https://www.w3schools.com/python/>

To be...



CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.