

# Oficina de Programação Básica em Python



**PET - Programa de Educação Tutorial**

# Oficina de Python - Aula 4

## Conteúdo Aula 3:

- LISTAS
- DICIONÁRIOS



# LISTAS

- **SINTAXE DE CRIAÇÃO**
- **INDEXAÇÃO E FATIAÇÃO**
- **MÉTODOS DE MANIPULAÇÃO**

# SINTAXE DE CRIAÇÃO

---

Em Python, as listas são estruturas de dados versáteis que podem armazenar uma coleção ordenada de elementos. Você pode criar listas de diferentes tipos de dados, incluindo números, strings e até outras listas. Aqui estão algumas informações sobre a criação de listas em Python, juntamente com exemplos de código resumidos:



```
1 listas = []
2
3 lista_mista = [1, 'hello', 3.14, True]
4
5 lista1 = [1, 2, 3]
6 lista2 = ['a', 'b', 'c']
7 lista_concatenada = lista1 + lista2
```

# INDEXAÇÃO E FATIAMENTO

**Indexação em Listas:** A indexação permite acessar elementos individuais em uma lista. Os índices começam em 0, e é possível usar índices negativos para contar a partir do final.

**Fatiamento em Listas:** O fatiamento extrai porções de uma lista usando a notação início:fim. O intervalo inclui o elemento no índice de início, mas exclui o elemento no índice de fim. Pode-se omitir índices para começar do início ou ir até o final, e é possível incluir um passo para pular elementos.

```
1  lista = ['a', 'b', 'c', 'd', 'e']
2  primeiro_elemento = lista[0]  # 'a'
3  terceiro_elemento = lista[2]  # 'c'
4
5  ultimo_elemento = lista[-1]  # 'e'
6
7  sublista = lista[1:4]  # ['b', 'c', 'd']
8
9  inicio_omissao = lista[:3]  # ['a', 'b', 'c']
10 fim_omissao = lista[2:]  # ['c', 'd', 'e']
11
12 com_passo = lista[::2]  # ['a', 'c', 'e']
13
14 invertida = lista[::-1]  # ['e', 'd', 'c', 'b', 'a']
15
```

# Exercícios

- **Crie uma função lambda que receba uma string e inverta e retorne ela invertida. ( dica: palavras se comportam como listas )**

# MÉTODOS DE MANIPULAÇÃO

**Indexação em Listas:** A indexação permite acessar elementos individuais em uma lista. Os índices começam em 0, e é possível usar índices negativos para contar a partir do final.

**Fatiamento em Listas:** O fatiamento extrai porções de uma lista usando a notação início:fim. O intervalo inclui o elemento no índice de início, mas exclui o elemento no índice de fim. Pode-se omitir índices para começar do início ou ir até o final, e é possível incluir um passo para pular elementos.

```
1  lista = [1, 2, 3]
2  lista.append(4)
3  # Resultado: lista é agora [1, 2, 3, 4]
4
5  listanova = [5,6]
6
7  lista.extend(listanova)
8  # Resultado: lista é agora [1, 2, 3, 4, 5, 6]
9
10 lista.insert(1, 5)
11 # Resultado: lista é agora [1, 5, 2, 3, 4, 5, 6]
12
13 lista.remove(5)
14 # Resultado: lista é agora [1, 2, 3, 4, 5, 6]
15
16 elemento = lista.pop(0)
17 # Resultado: lista é agora [2, 3, 4, 5, 6]
18
19 indice = lista.index(3)
20 # indice: 1 (primeira ocorrencia do 3)
21
```

# Exercícios

- **Crie um script que rode um loop, que dentro desse loop, tenha uma pergunta pro usuário, se ele quer adicionar, remover ou sair, caso escolha adicionar, você deve pedir um número para o usuário e adicionar este número ao final da lista, caso peça para remover, você deve pedir uma posição para remover o item que estiver nela, caso escolha sair, pare de rodar o script ( lembre-se de exibir a lista )**



# ITERANDO SOBRE LISTAS

---

Iterar sobre listas é uma tarefa comum em programação, e Python oferece várias maneiras de realizar iterações eficientes. Aqui estão algumas abordagens comuns:



```
1 lista = [1, 2, 3, 4, 5]
2 for elemento in lista:
3     print(elemento)
4
5 for indice, valor in enumerate(lista):
6     print(f"Indice: {indice}, Valor: {valor}")
7
8 indice = 0
9 while indice < len(lista):
10     print(lista[indice])
11     indice += 1
```

# Exercícios

- **Escreva uma função em Python que recebe uma lista de números inteiros e retorna a soma dos elementos pares presentes na lista. Utilize uma estrutura de iteração para percorrer a lista.**
- **Crie uma função que recebe uma lista de palavras e uma palavra-alvo. A função deve retornar uma lista contendo todas as palavras da lista original que contêm a palavra-alvo. Utilize uma estrutura de iteração para percorrer a lista.**

# DICIONÁRIOS

- CRIAÇÃO E MANIPULAÇÃO
- ITERAÇÃO
- MÉTODOS

# CRIAÇÃO E MANIPULAÇÃO

Os dicionários em Python são estruturas de dados que mapeiam chaves a valores. A sintaxe básica é usando chaves {} e pares chave-valor separados por dois-pontos :.

```
1 meu_dicionario = {'chave1': 'valor1', 'chave2': 'valor2', 'chave3': 'valor3'}
2
3 valor_da_chave2 = meu_dicionario['chave2']
4
5 # Adição
6 meu_dicionario['chave4'] = 'valor4'
7
8 # Remoção
9 del meu_dicionario['chave1']
10
11 # Atualização
12 meu_dicionario['chave2'] = 'novo_valor'
13
14
```

# ITERAÇÃO

---

Utilização de loops (for) para percorrer chaves, valores ou ambos:

Loops podem ser utilizados para percorrer as chaves, os valores ou ambos em um dicionário.

```
1  # Iteração sobre chaves
2  for chave in meu_dicionario:
3      print(chave)
4
5  # Iteração sobre valores
6  for valor in meu_dicionario.values():
7      print(valor)
8
9  # Iteração sobre chaves e valores
10 for chave, valor in meu_dicionario.items():
11     print(f'Chave: {chave}, Valor: {valor}')
```

# MÉTODOS

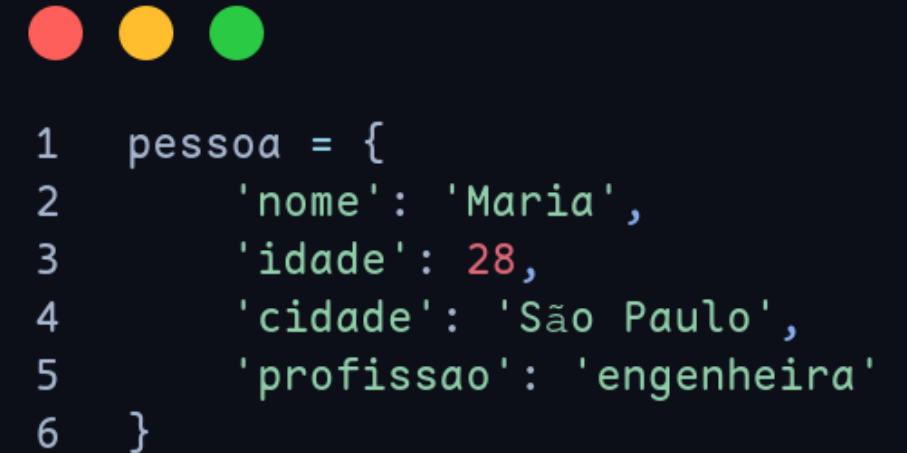
---

Existem métodos que facilitam a manipulação de dicionários, como `keys()`, `values()`, `items()`, `get()`, entre outros.

```
1  # Obter todas as chaves  
2  chaves = meu_dicionario.keys()  
3  
4  # Obter todos os valores  
5  valores = meu_dicionario.values()  
6  
7  # Obter pares chave-valor como tuplas  
8  itens = meu_dicionario.items()  
9  
10 # Obter valor associado a uma chave ou valor padrão se a chave não existir  
11 valor = meu_dicionario.get('chave1', 'valor_padrão')
```

# Exercícios

- Considere o seguinte dicionário:
  - a) Imprima o nome da pessoa.
  - b) Adicione a chave 'email' com o valor 'maria@email.com' ao dicionário.
  - c) Atualize a idade para 29.



```
1  pessoa = {  
2      'nome': 'Maria',  
3      'idade': 28,  
4      'cidade': 'São Paulo',  
5      'profissao': 'engenheira'  
6  }
```

# Exercícios

- Considere o seguinte dicionário de notas de alunos:
  - a) Crie um loop para imprimir os nomes dos alunos.
  - b) Crie um loop para imprimir os nomes dos alunos e suas respectivas notas.



```
1 notas_alunos = {  
2     'João': 85,  
3     'Maria': 92,  
4     'Pedro': 78,  
5     'Ana': 95  
6 }  
7
```



# Exercícios

- Considere dois dicionários:
  - a) Crie um novo dicionário que seja a união de `dicionario1` e `dicionario2`.
  - b) Remova a chave `'b'` do novo dicionário.
  - c) Calcule a soma dos valores no novo dicionário.



```
1 dicionario1 = {'a': 1, 'b': 2, 'c': 3}
2 dicionario2 = {'c': 10, 'd': 4, 'e': 5}
```