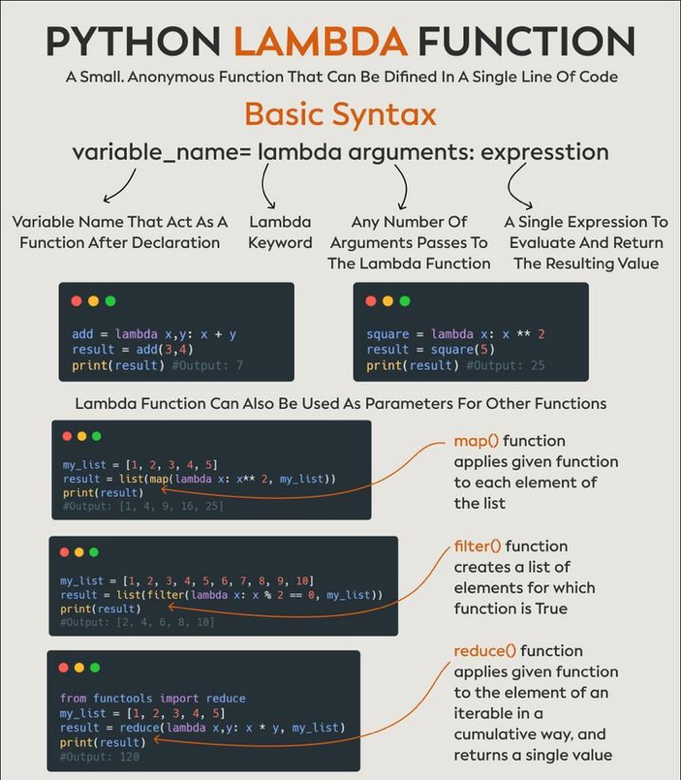
Resumo sobre a função lambda em python, destacando a sintaxe básica e exemplos de uso em conjunto com funções como map(), filter(), e reduce().



**1. Sintaxe Básica da Função Lambda**

**Estrutura:**

variable\_name = lambda arguments: expression

* **variable\_name**: Nome da variável que atuará como função após a declaração.
* **lambda**: Palavra-chave usada para definir a função lambda.
* **arguments**: Argumentos passados para a função, podendo ser múltiplos, separados por vírgulas.
* **expression**: A expressão que será avaliada e retornará o resultado. Pode ser uma expressão matemática, lógica, etc.

**Exemplos:**

1. **Soma de Dois Números**

add = lambda x, y: x + y

result = add(3, 4)

print(result) # Saída: 7

* + A função add é uma função lambda que soma dois números. A chamada add(3, 4) executa a soma e retorna 7.

1. **Quadrado de Um Número**

square = lambda x: x \*\* 2

result = square(5)

print(result) # Saída: 25

* + A função square eleva um número ao quadrado. square(5) retorna 25.

**2. Uso em Funções de Alta Ordem**

As funções lambda são frequentemente usadas como argumentos para outras funções, como map(), filter(), e reduce().

1. **map()**: Aplica uma função a cada elemento de um iterável (como uma lista) e retorna um iterador com os resultados.

my\_list = [1, 2, 3, 4, 5]

result = list(map(lambda x: x \*\* 2, my\_list))

print(result) # Saída: [1, 4, 9, 16, 25]

* + Neste exemplo, map() aplica a função lambda que eleva cada elemento ao quadrado.

1. **filter()**: Cria uma lista de elementos para os quais a função é verdadeira.

my\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

result = list(filter(lambda x: x % 2 == 0, my\_list))

print(result) # Saída: [2, 4, 6, 8, 10]

* + filter() aplica a função lambda que retorna True para números pares.

1. **reduce()**: Aplica a função de forma cumulativa ao iterável, reduzindo-o a um único valor.

from functools import reduce

my\_list = [1, 2, 3, 4, 5]

result = reduce(lambda x, y: x \* y, my\_list)

print(result) # Saída: 120

* + reduce() multiplica cumulativamente todos os elementos da lista, resultando em 120.

**3. Sugestões de Melhorias e Exemplos Adicionais**

Para enriquecer a aula, considere os seguintes pontos:

* **Complexidade Adicional**: Mostre como as lambdas podem ser usadas com mais de uma função ao mesmo tempo. Exemplo:

my\_list = [1, 2, 3, 4, 5]

result = list(map(lambda x: x \*\* 2, filter(lambda x: x % 2 != 0, my\_list)))

print(result) # Saída: [1, 9, 25]

* + Aqui, os números ímpares são filtrados antes de serem elevados ao quadrado.
* **Uso em Dicionários**: Como lambdas podem ser utilizadas para operações em dicionários.

my\_dict = {'a': 10, 'b': 20, 'c': 30}

result = {k: lambda v: v \*\* 2 for k, v in my\_dict.items()}

print(result) # Exemplo de saída: {'a': 100, 'b': 400, 'c': 900}

* **Funções Lambda em Condicionais**: Mostre como lambdas podem incluir condicionais para operações mais complexas.

max\_func = lambda a, b: a if a > b else b

print(max\_func(10, 20)) # Saída: 20

* **Usar com Sorting**: Lambdas também podem ser usadas para definir chaves de ordenação.

my\_list = [(1, 'one'), (2, 'two'), (3, 'three')]

sorted\_list = sorted(my\_list, key=lambda x: x[1])

print(sorted\_list) # Saída: [(1, 'one'), (3, 'three'), (2, 'two')]