Apostila 2: Python - Funções

1. Funções

Em Python, funções, são blocos de código reutilizáveis que podem ser chamados em diferentes partes do programa.

```
# Definição de uma função simples
def saudacao():
 print("Olá, bem-vindo ao curso de Python!")

# Chamando a função
saudacao()
```

2. Funções com Parâmetros e Argumentos

Podemos passar informações para uma função através de

parâmetros.

```
def soma(a, b):
 return a + b

resultado = soma(5, 3)
print(resultado) # 8
```

3. Função com Retorno de Valor

Uma função pode retornar um valor utilizando `return`.

```
def quadrado(numero):
 return numero ** 2

print(quadrado(4)) # 16
```

4. Função com Parâmetro Padrão

Se um parâmetro não for informado, ele assume um valor

padrão. def saudacao(nome="Aluno"):

```
 print(f"Olá, {nome}!")

saudacao() # Olá, Aluno!
saudacao("Maria") # Olá, Maria!
```

5. Função com Argumentos e Parâmetros Variáveis

Podemos definir funções que aceitam uma quantidade variável de

argumentos.

```
def soma(*numeros):
 return sum(numeros)

print(soma(1, 2, 3, 4)) # 10
```

6. Escopo de Variável em uma Função

O escopo determina onde uma variável pode ser acessada.

```
def funcao():
 local = "Sou uma variável local"
 print(local)

funcao()
# print(local) # Isso geraria um erro, pois 'local' não existe fora da função
```

7. Função Lambda (Função Anônima)

Funções lambda são funções curtas e anônimas.

```
quadrado = lambda x: x ** 2
print(quadrado(5)) # 25
```

8. Criar Módulos e Pacotes

Podemos dividir nosso código em módulos para facilitar a manutenção.

```
# Criando um módulo chamado "operacoes.py"
def soma(a, b):
 return a + b
```
No arquivo principal:

```
import operacoes
print(operacoes.soma(2, 3)) # 5
```

9. Modularização e Importação de Módulos

Podemos importar módulos inteiros ou apenas partes deles.

```
from operacoes import soma
print(soma(4, 5)) # 9
```

10. Manipulação de Listas

```
cores = ["Azul", "Amarelo", "Branco"]
cores.append("Verde")
print(cores)
```

11. Manipulação de Tuplas

```
dados = ("Andre", "andre@gmail.com")
print(dados[0]) # Andre
```

12. Manipulação de Dicionários

```
pessoa = {"Nome": "Andre", "Idade": 26}
print(pessoa["Nome"]) # Andre
```

13. Manipulação de Conjuntos

```
numeros = {1, 2, 3, 4, 5, 5}
print(numeros) # {1, 2, 3, 4, 5}
```

14. Função `map`

```
def quadrado(x):
 return x ** 2
valores = [1, 2, 3, 4]
resultado = list(map(quadrado, valores))
print(resultado) # [1, 4, 9, 16]
```

15. Função `filter`

```
def par(x):
 return x % 2 == 0

numeros = [1, 2, 3, 4, 5, 6]
pares = list(filter(par, numeros))
print(pares) # [2, 4, 6]
```

16. Função `reduce`

```
from functools import reduce

def soma(a, b):
 return a + b

valores = [1, 2, 3, 4, 5]
resultado = reduce(soma, valores)
print(resultado) # 15
```

17. Função `input` e `print`

```
nome = input("Digite seu nome: ")
print(f"Olá, {nome}!")
```

Exercícios

1. Crie uma função que receba dois números e retorne o maior deles. 2. Escreva uma função que calcule a fatorial de um número. 3. Utilize `map` para elevar ao quadrado todos os elementos de uma lista. 4. Utilize `filter` para filtrar os números ímpares de uma lista. 5. Utilize `reduce` para somar todos os valores de uma lista.
6. Crie um dicionário com informações sobre um produto e imprima seus valores.
7. Solicite ao usuário um número e informe se ele é par ou ímpar.