1. Gerenciador de Tarefas (Interface Gráfica com Tkinter)

python

```python
import tkinter as tk
from tkinter import messagebox
from datetime import datetime

tarefas = []

def adicionar_tarefa():
    tarefa = entry_tarefa.get()
    if tarefa:
        tarefas.append({"tarefa": tarefa, "data": datetime.now()})
        listbox_tarefas.insert(tk.END, f"{tarefa} {datetime.now().strftime('%Y-
%m-%d %H:%M:%S')}")
        entry_tarefa.delete(0, tk.END)
    else:
        messagebox.showwarning("Aviso", "Digite uma tarefa.")

def concluir_tarefa():
    indice_selecionado = listbox_tarefas.curselection()
    if indice_selecionado:
        tarefa_concluida = tarefas.pop(indice_selecionado[0])
        listbox_tarefas.delete(indice_selecionado)
        messagebox.showinfo("Concluído", f"Tarefa '{tarefa_concluida['tarefa']}'
concluída!")

# Interface gráfica
root = tk.Tk()
root.title("Gerenciador de Tarefas")
label = tk.Label(root, text="Tarefa:")
label.pack(pady=5)
entry_tarefa = tk.Entry(root, width=30)
entry_tarefa.pack(pady=5)
botao_adicionar = tk.Button(root, text="Adicionar Tarefa",
command=adicionar_tarefa)
botao_adicionar.pack(pady=10)
listbox_tarefas = tk.Listbox(root, width=40, height=10)
listbox_tarefas.pack()
botao_concluir = tk.Button(root, text="Concluir Tarefa", command=concluir_tarefa)
botao_concluir.pack(pady=5)
root.mainloop()
```

Descrição: Programa para gerenciar tarefas usando interface gráfica Tkinter. Permite adicionar e concluir tarefas, exibindo data e hora de

inclusão

2. Estrutura Condicional – Verificação de Idade

python

```python
idade = 18
if idade < 18:
    print("Você é menor de idade.")
elif idade == 18:
    print("Você tem 18 anos.")
else:
    print("Você é maior de idade.")
```

Descrição: Exemplo simples de estrutura condicional para verificar a idade do usuário

3. Loop – Iteração em Lista de Frutas

python

```python
frutas = ["maçã", "banana", "laranja"]
for fruta in frutas:
    print(fruta)
```

Descrição: Exemplo de loop `for` para percorrer e imprimir cada elemento de uma lista

1

4. Função – Saudação Personalizada

python

```python
def saudacao(nome):
    return f"Olá, {nome}!"

mensagem = saudacao("João")
print(mensagem)
```

Descrição: Função que recebe um nome e retorna uma mensagem de saudação

personalizada

5. Listas – Acesso e Modificação

```python
numeros = [1, 2, 3, 4, 5]

terceiro_numero = numeros[2]
print(terceiro_numero)

numeros[1] = 10
print(numeros)
```

Descrição: Exemplo de como acessar e modificar elementos em uma lista

6. Compreensão de Listas – Quadrados dos Números

python

```python
numeros = [1, 2, 3, 4, 5]
quadrados = [x**2 for x in numeros]
print(quadrados)
```

Descrição: Uso de compreensão de listas para gerar uma nova lista com os quadrados dos números

7. Tuplas – Coordenadas de um Ponto

python

```python
coordenadas = (3, 4)
x, y = coordenadas
print(f"Coordenadas: x={x}, y={y}")
```

Descrição: Exemplo de criação e desempacotamento de tupla para representar

coordenadas

8. Enumerate – Índice e Valor em Loop

python

```python
frutas = ["maçã", "banana", "laranja"]
for indice, fruta in enumerate(frutas):
    print(f"Índice: {indice}, Fruta: {fruta}")
```

Descrição: Utilização da função `enumerate` para obter índice e valor durante a iteração

1

9. Conjuntos – Adição e Remoção de Elementos

python

```python
cores_primarias = {'vermelho', 'azul', 'amarelo'}
cores_primarias.add('verde')
print(cores_primarias)

cores_primarias.remove('azul')
print(cores_primarias)
```

Descrição: Exemplo de operações com conjuntos (sets) em

Python

python

```python
pessoa = {'nome': 'Ana', 'idade': 25, 'cidade': 'São Paulo'}
idade_da_pessoa = pessoa['idade']
print(idade_da_pessoa)

pessoa['cidade'] = 'Rio de Janeiro'
print(pessoa)
```