Programação Estruturada vs. POO

Aula prática com cachorros! Vamos explorar duas abordagens de programação. Entenda como resolver problemas de maneiras diferentes. Descubra por que a escolha da abordagem é crucial.

FIESP SESI SENAI IRS
SENAI
ESCOLA SENAI "A. JACOB LAFER"

Programação Estruturada

Código organizado em variáveis e funções separadas. Dados e comportamentos são distintos. Funções operam sobre dados passados como parâmetros. A execução é sequencial e as variáveis são independentes.

Dados Separados

Dados e funções são entidades distintas.

Execução Sequencial

O código é executado linha por linha.

estruturada.py > ...

```python
# Dados do primeiro cachorro
nome_cachorro_1 = "Nelson"
comida_cachorro_1 = 3
sono_cachorro_1 = False

# Dados do segundo cachorro
nome_cachorro_2 = "Jeremias"
comida_cachorro_2 = 1
sono_cachorro_2 = True

# Funções para manipular os dados
def comer(quantidade_comida):
    return quantidade_comida - 1

def dormir():
    return True

# Usando as funções
comida_cachorro_1 = comer(comida_cachorro_1)
sono_cachorro_2 = dormir()

# Exibindo os resultados no terminal
print(f"{nome_cachorro_1} agora tem {comida_cachorro_1} unidades de comida.")
print(f"{nome_cachorro_2} está com sono? {sono_cachorro_2}")
```

Programação Orientada a Objetos

Dados e comportamentos unidos em uma classe. Objetos mantêm seu próprio estado. Métodos operam diretamente sobre os dados do objeto. Reutilização através de instâncias.

União de Dados

Dados e funções formam um objeto.

Reutilização

Objetos podem ser instanciados várias vezes.

poo.py > Cachorro

```python
#Classe Cachorro (É como se fosse um molde de criação)
class Cachorro:
    def __init__(self, nome, comida, sono):
        self.nome = nome
        self.comida = comida
        self.sono = sono

    def comer(self):
        self.comida -= 1

    def dormir(self):
        self.sono = False

# Criando (Objetos)
cachorro_1 = Cachorro("Nelson", 3, False)
cachorro_2 = Cachorro("Jeremias", 1, True)

# Usando métodos
cachorro_1.comer()
cachorro_2.dormir()

# Exibindo os resultados no terminal
print(f"{cachorro_1.nome} agora tem {cachorro_1.comida} unidades de comida.")
print(f"{cachorro_2.nome} está com sono? {cachorro_2.sono}")
```

Comparação Prática

POO facilita a criação de múltiplos cachorros. A estruturada exige repetição de código. POO organiza melhor os dados e comportamentos. Veja como a POO simplifica a criação de objetos.

1

Criação Fácil

POO simplifica a criação de objetos.

2

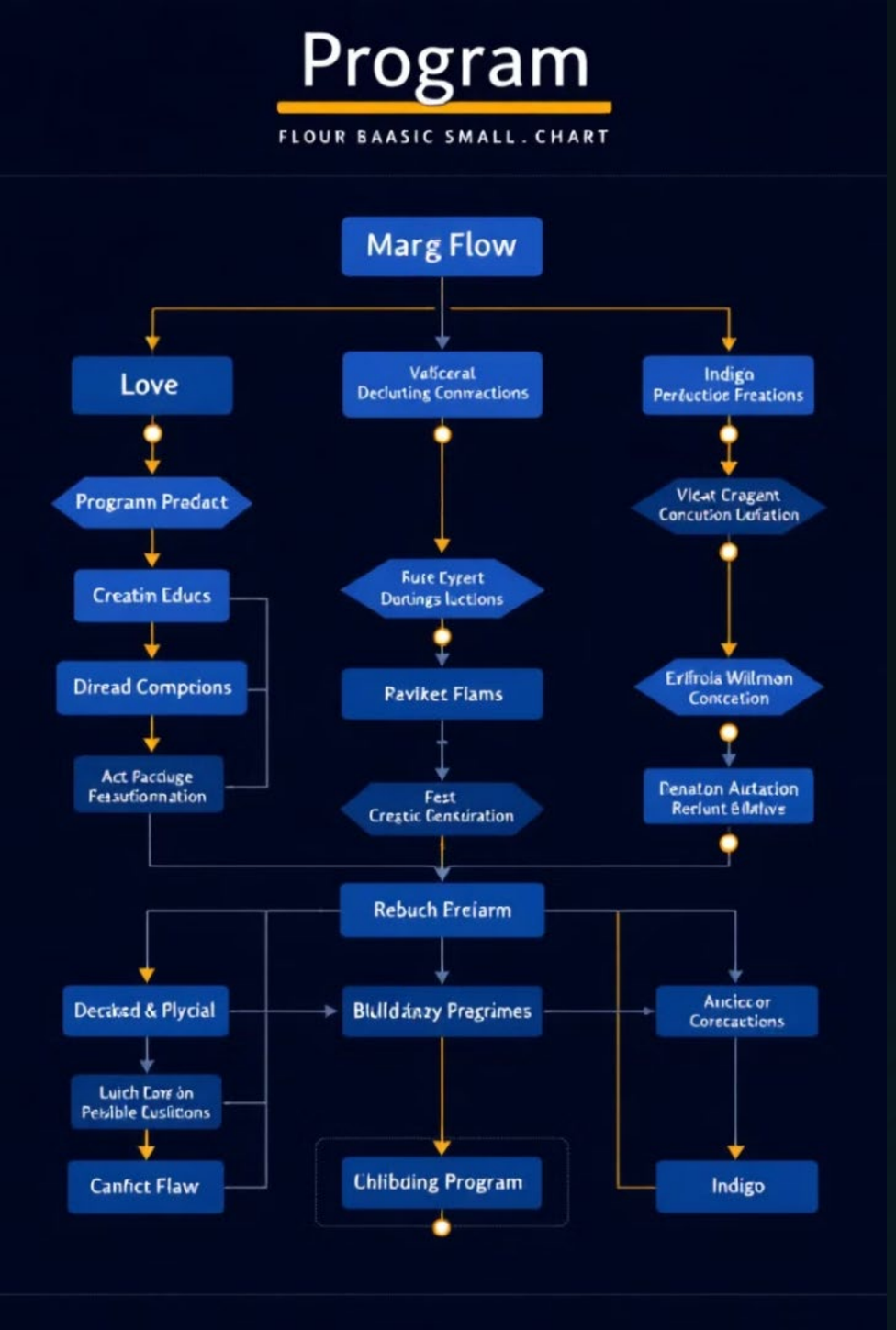
Menos Código

Evita repetição de código.

3

Organização

Melhor organização dos dados.



Vantagens da Estruturada

Simplicidade para problemas pequenos. Fácil de entender para iniciantes. Ideal para scripts simples e programas pequenos. Processamento sequencial de dados.

Simples

Fácil para iniciantes.

Scripts

Ideal para scripts simples.

Desvantagens da Estruturada

Difícil de manter com o crescimento do código. Duplicação de código. Difícil de organizar dados relacionados. Não é ideal para sistemas complexos.

1

Manutenção

Difícil de manter a longo prazo.

2

Duplicação

Repetição de código comum.

Vantagens da POO

Melhor organização do código. Facilidade de manutenção. Reutilização de código. Representa melhor o mundo real. Ideal para sistemas complexos.

1 **Organização**

Código bem estruturado.

2 **Reutilização**

Código reutilizável.

Desvantagens da POO

Curva de aprendizado inicial maior. Pode ser complexo demais para problemas simples. Requer mais planejamento inicial. Nem sempre é a melhor escolha.

Complexidade

Pode ser complexo para iniciantes.

Planejamento

Requer mais planejamento inicial.

Quando Usar Estruturada ?

Scripts simples. Programas pequenos. Processamento sequencial de dados. Ideal para tarefas rápidas e fáceis. Quando a simplicidade é fundamental.

1 Simples

2 Pequeno

3 Rápido

Quando Usar POO ?

Sistemas complexos. Códigos que precisam ser mantidos por muito tempo. Representar entidades do mundo real. Projetos que vão crescer com o tempo.