



# 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.

# 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 > ...
1  # Dados do primeiro cachorro
2  nome_cachorro_1 = "Nelson"
3  comida_cachorro_1 = 3
4  sono_cachorro_1 = False
5
6  # Dados do segundo cachorro
7  nome_cachorro_2 = "Jeremias"
8  comida_cachorro_2 = 1
9  sono_cachorro_2 = True
10
11 # Funções para manipular os dados
12 def comer(quantidade_comida):
13     return quantidade_comida - 1
14
15 def dormir():
16     return True
17
18 # Usando as funções
19 comida_cachorro_1 = comer(comida_cachorro_1)
20 sono_cachorro_2 = dormir()
21
22 # Exibindo os resultados no terminal
23 print(f"{nome_cachorro_1} agora tem {comida_cachorro_1} unidades de comida.")
24 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
1  #Classe Cachorro (É como se fosse um molde de criação)
2  class Cachorro:
3      def __init__(self, nome, comida, sono):
4          self.nome = nome
5          self.comida = comida
6          self.sono = sono
7
8      def comer(self):
9          self.comida -= 1      -= = comida seja igual comida - 1
10
11     def dormir(self):
12         self.sono = False
13
14     # Criando (Objetos)
15     cachorro_1 = Cachorro("Nelson", 3, False)
16     cachorro_2 = Cachorro("Jeremias", 1, True)
17
18     # Usando métodos
19     cachorro_1.comer()
20     cachorro_2.dormir()
21
22     # Exibindo os resultados no terminal
23     print(f"{cachorro_1.nome} agora tem {cachorro_1.comida} unidades de comida.")
24     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

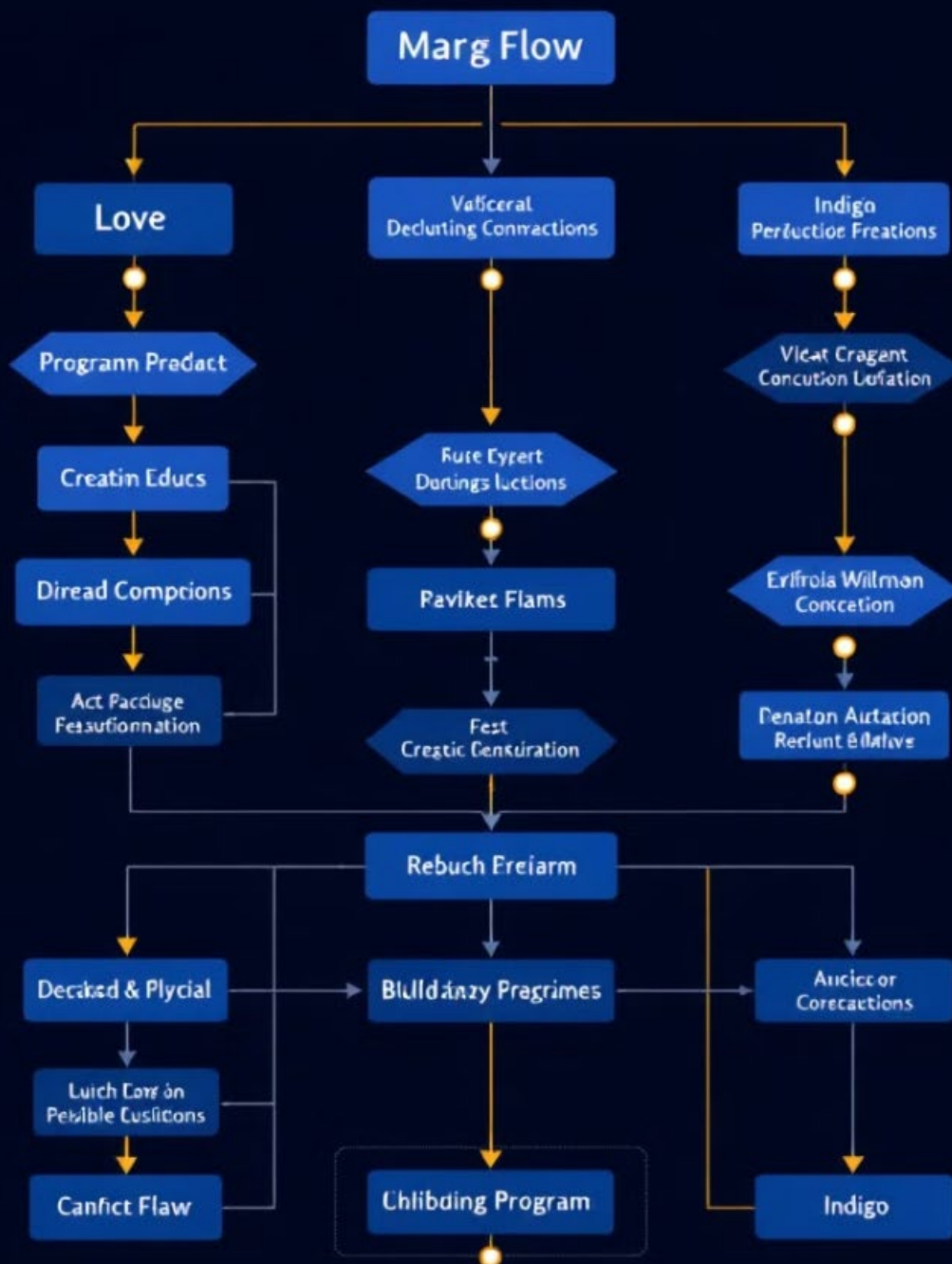
## Organização

Melhor organização dos dados.



# Program

FLOUR BAASIC SMALL . CHART



## 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.



```

1 123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555657585960616263646566676869707172737475767778798081828384858687888990919293949596979899100
3
11 the (pb is; that cerbes mett thp;
16 for (eloght inccr:(errater st, is laaef a cosap to fack the crstucter)
17 for il vetongv design
15   Tem deo Lictedy,
17   rrual ill rant. bmbi; a will code),
17   The lian ano and clubbv fhan = mcll:
19 }
18 sin = (lenen (l);
17   rant lecterions
16
16   wzicher is cae an cates;
16   she is weptes and for (lora)
16   thane uotal she to 17 (17 darkite and ir: "nibple (macte()), "3)

```



# 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 P00

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

Simple

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.

