UNIFAVIP
wyden

PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO EM PYTHON

Profº: Sebastião Rogério

Av. Adjar da Silva Casé, 800 Indianópolis

Caruaru – Pernambuco – PE CEP 55024-

740 | Telefone 0800 771-5001

Por Que Aprender Paradigmas De Programação?

Dominar diferentes paradigmas é como ter várias ferramentas em sua caixa de ferramentas. Cada uma ajuda a resolver problemas de maneiras diferentes e expande seu conhecimento em tecnologia. Os principais paradigmas se dividem em dois: Programação Imperativa e Programação Declarativa.

Programação Declarativa

Aqui, o foco é no que deve ser o resultado final, sem se preocupar com os detalhes do processo.

Programação Imperativa

Como um manual detalhado de instruções, o programador fornece o passo a passo para o computador seguir. Exemplos: Cobol, Fortran, Pascal.

Paradigma Procedural

O paradigma procedural é baseado na ideia de que um programa é composto por uma sequência de instruções, ou passos, que o computador deve seguir. O foco está no "como" as coisas devem ser feitas, através da definição de procedimentos ou funções que são chamadas para realizar tarefas específicas.

1. Uso intensivo de funções ou procedimentos

   Cada função ou procedimento contém uma série de comandos que manipula os dados para resolver problemas.

2. A execução segue um fluxo de controle sequencial (início, meio e fim)

   O programa é executado passo a passo, de cima para baixo.

3. Usa variáveis globais e locais para armazenar estados temporários

   As variáveis armazenam dados que podem ser modificados durante a execução do programa.

4. Exemplo de linguagens: C, Pascal, Fortran

   Essas linguagens são amplamente utilizadas em programação de sistemas e aplicações de alto desempenho.

Paradigma Orientado a Objetos (Poo)

A programação orientada a objetos (POO) organiza o código em torno de "objetos". Um objeto é uma entidade que combina dados (propriedades) e comportamentos (métodos).

O paradigma simula o mundo real, onde objetos interagem uns com os outros, e se concentra no "o quê" deve ser feito, abstraindo os detalhes de implementação.

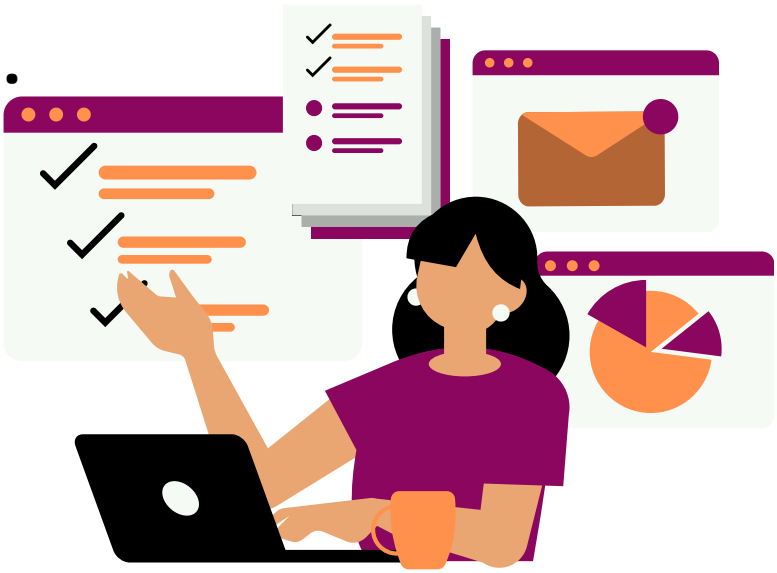
Encapsulamento
Os dados e os métodos relacionados são agrupados em classes.

Herança
Permite a criação de novas classes baseadas em classes existentes, promovendo o reuso de código

Polimorfismo
Objetos de diferentes tipos podem ser tratados de maneira uniforme.

Abstração
Simplificação de conceitos complexos, ocultando os detalhes internos e mostrando apenas o essencial.

Exemplo de linguagens
Java, C++, Python, C#.



Paradigma Funcional

A programação funcional baseia-se no uso de funções puras, que são funções matemáticas no sentido de que sempre produzem a mesma saída para uma dada entrada e não causam efeitos colaterais (como modificar o estado do programa). O paradigma funcional enfatiza o "o quê" fazer ao invés de "como" fazer, sendo declarativo.

Imutabilidade

Dados não podem ser alterados após a criação, promovendo um estilo de programação sem estados mutáveis.

Composição de funções

Funções podem ser combinadas para criar novas funções.

Funções de alta ordem

Funções que podem receber outras funções como parâmetros ou retornar funções.

Exemplo de linguagens

Haskell, Lisp, Scala, Erlang.

Paradigma Lógico

No paradigma lógico, a programação é baseada na lógica formal e na inferência. O programador define um conjunto de fatos e regras, e o sistema lógico tenta inferir respostas ou soluções com base nessas informações. O foco está em "o que" deve ser atingido, sem especificar "como".

Define regras lógicas para determinar as soluções
Usando proposições verdadeiras e a inferência.

Não imperativo
O programa é uma declaração de fatos e regras.

Usado em inteligência artificial, problemas de inferência e raciocínio automatizado
O paradigma lógico é útil para resolver problemas complexos que exigem raciocínio lógico.

Exemplo de linguagem
Prolog.

Paradigma Orientado A Eventos

A programação orientada a eventos é baseada em um fluxo de execução que responde a eventos. Um evento é uma ação, como um clique do mouse, pressionar uma tecla ou uma notificação de uma mudança de estado. O código não é executado sequencialmente, mas em resposta a esses eventos que ocorrem ao longo do tempo.

1. **Os programas esperam por eventos e os tratam assim que ocorrem**

   O código é executado apenas quando um evento específico ocorre.

2. **Usado em interfaces gráficas de usuário (GUIs), aplicações web, e sistemas interativos**

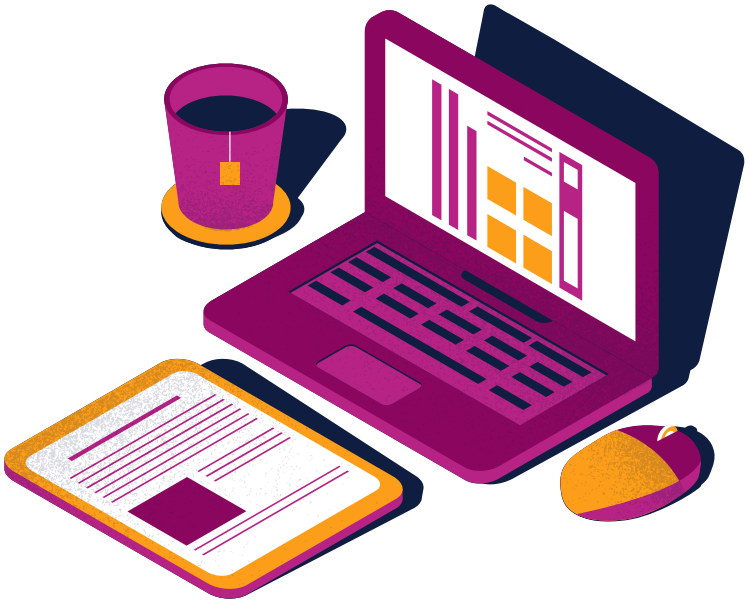
   O paradigma orientado a eventos é ideal para criar interfaces que respondem às ações do usuário.

3. **Requer um modelo de loop de eventos**

   Onde o código fica aguardando até que um evento ocorra para disparar a execução de uma função (chamada de event handler).

4. **Exemplo de linguagens**

   JavaScript, Visual Basic, C# com eventos de interfaces gráficas.



UNIFAVIP
wyden

Bons estudos!