Tutorial de Python usando Dialética Artificial e DSL

oZumbiAnalitico

Conceitos : Sócrates, Método Socrático, Maiêutica, Ironia Socrática.

Paradigma [Dialética Artificial] É um paradigma de estudo centrado em uma estrutura de perguntas (Dialética) para ser usado como roteiro para estudo assistido com inteligência artificial.

Paradigma [DSL] É um paradigma de desenvolvimento de software que busca otimizar a construção de aplicativos construindo prompts de inteligência artificial usando uma linguagem semiformal (Declarative Semiformal Language).

A linguagem DSL se situa entre a linguagem natural humana e uma linguagem de alto nível, contendo símbolos para operações bem definidas e instruções em linguagem natural para serem interpretadas por inteligências artificiais. Você pode entender mais sobre a linguagem aqui (https://github.com/EYO-07/DSL). O paradigma DSL faz parte do que é chamado de engenharia de prompt, que busca otimizar o uso de inteligências artificias, em específico, para desenvolvimento de softwares.

Conceitos : Computador, Hardware, Software, Linguagens de Programação, Linguagens de Alto Nível, Paradigma Imperativo, Paradigma de Orientação a Objeto, Bibliotecas.

Faz parte do paradigma da dialética artificial registrar perguntas e conceitos obscuros e interagir com a inteligência artificial para obter as respostas (Diferente do diálogo socrático, você é aquele que faz as perguntas, mas você também aquele que busca se libertar da ignorância). Este texto é apenas um roteiro, com sugestões de perguntas, de conceitos e definições que possivelmente não estão disponíveis antes da criação deste documento.

Conceitos : Código Fonte, Interpretador, Compilador, Editores de Texto, Editores de Código, IDE, Biblioteca Padrão, Bibliotecas Externas, Pip.

```
* O que é Python ?
** O que é linguagem de programação ?
** Quais as vantagens e desvantagens em usar Python ?
** Quem criou Python ?
* Como usar Python ?
** Como instalar Python ?
** O que é possível fazer em Python ?
*** O que é possível fazer com a biblioteca padrão de Python ?
*** Como instalar bibliotecas externas em Python ?
**** O que é a ferramenta PIP ?
**** Como usar a ferramenta PIP ?
**** Quais os riscos de usar o PIP ?
**** Como fazer update do PIP ?
**** Como saber quais bibliotecas externas foram instaladas pelo PIP ?
**** Como remover bibliotecas usando o PIP ?
** O que são scripts em Python ?
** Como executar scripts em Python ?
```

```
** Como editar scripts em Python ?
** Quais erros um programador Python pode cometer ? Quais os mais comuns ?
** Erros de Lógica versus Erros de Sintaxe ?
```

Mesmo usando lógicas DSL para criar os programas, ainda é necessário verificar e corrigir erros dos códigos retornados pelas inteligências artificiais, além de fazer os ajustes necessários para que o programa tenha o comportamento desejado. Ter conhecimento sobre os erros comuns e sobre as regras de sintaxe da linguagem alvo do DSL é indispensável, o DSL não foi desenhado para não-programadores, mas para facilitar a vida dos programadores. O uso do paradigma DSL dá algumas vantagens ao processo, pois a lógica DSL é um modelo compacto da lógica do seu programa alvo, facilitando o processo de desenvolvimento, manutenção e a consciência do funcionamento do programa. A lógica DSL também é uma forma de documentação, apesar de ser mais criptica que a linguagem natural é uma maneira de enxergar o funcionamento do programa.

```
* Como é a sintaxe da linguagem Python ?
** Sintaxe para Criação de Variáveis ?
>> Me dê exemplos de criação de variáveis em python
*** Quais os tipos de variáveis em Python ?
** Sintaxe para Criação de Funções ?
>> Me dê exemplos de criação de funções em python
** Sintaxe para Criação de Estruturas Condicionais ?
>> Me dê exemplos de criação de estruturas condicionais em python
** Sintaxe para Criação de Laços de Repetição ?
>> Me dê exemplos de criação de laços de repetição em python
*** Quais os tipos de laços de repetição ?
*** O que é List Comprehension ?
>>> Me dê exemplos de list comprehension em python
** Sintaxe para Criação de Classes e uso de Objetos ?
*** O que é Orientação a Objeto ?
*** O que é uma Classe em Python ?
*** O que é um Objeto em Python ?
*** O que é um método e o que é um atributo ?
*** O que é herança em orientação a objeto ?
**** O que é polimorfismo em orientação a objeto ?
**** Relação entre polimorfismo e herança ?
**** O que é 'override' de função e sua relação com herança ?
*** Quais as sintaxe para definir uma classe ?
*** Quais as sintaxe para definir atributos e métodos ?
*** Quais as sintaxe para criar um objeto ?
*** Quais as sintaxe para usar um método de um objeto ?
>> Me dê exemplos de classes e objetos em python
>> Me dê exemplos de herança e polimofismo de classes em python
** Sintaxe para Importação de Bibliotecas ?
> Me mostre erros de sintaxe como exercício (não mencione os tipos de erro)!
> Me mostre um exemplo de um arquivo completo com erros de sintaxe como exercício
(não mencione os tipos de erro)!
```

O símbolo > faz parte da Dialética Artificial e denota uma sugestão de prompt, enquanto o símbolo * denota perguntas que são primariamente para a consciência humana, porém podem e devem ser perguntadas para a inteligência artificial.

```
* O que é um bloco de código ?
* O que é um identificador ? O que é escopo ?
** Escopo Global versus Escopo Local ?
* O que são argumentos de função ?
* Quais tipos mais comuns de blocos e estruturas em uma linguagem de programação ?
```

```
* O que é programação orientada a eventos ?
** O que é um evento ?
>> Exemplos de Eventos
** O que é um callback ou handle de evento ?
** Como o Laço de Eventos implementa a lógica da programação orientada a eventos ?
>> Exemplos de Tipos de Programa Orientado a Eventos
```

Abaixo temos um exemplo de uma lógica DSL para uma aplicação orientada a eventos (de interface gráfica) que instrui a inteligência artificial a construir um painel flutuante que fica no lado direito da tela e o usuário pode colocar atalhos de programas clicando com o botão direito. Você pode acessar o código python gerado em (github.com/EYO-07/DSL), porém para que a inteligência artificial possa gerar os códigos a partir da lógica será necessário alimentar a inteligência artificial com textos sobre o DSL ou colocar no prompt o texto contido no diretório do github (DSL_AI_Prompt.txt). Caso esteja testando o programa, aperte a tecla escape para sair, clique no botão direito para adicionar atalhos e aperte o botão delete para removê-los.

```
[ Side Panel ] { Qt, Python, Windows Desktop Side Panel in Right Part of the Screen,
Translucent 60 percent }
I := Imports
A := Main Application Object
C := Main Window Creation
1. /QtSidePanel.pyw || I | A | C | Show Window | Start Event Loop
-> A || $ (QApplication) App
-> C || $ (QMainWindow) MainWindow | Set title "Side Panel App" | Set Black Background |
Set White Text | $ (IconListView) list_view || list_view occupy the client area
{ The Height of Application must be 65 percent of screen height }
2. IconListView : QListView {} || ... | Right Mouse Button Handle () | Keydown Handle ()
-> IconListView : QListView {} || ... | Right Mouse Button Handle () ||
Open a Dialog to Search for any files or applications |
% Selected a File or Application with Dialog ||
Add an Icon Image to this list view | Adjust the Height to the Listview making more compact
-> Keydown Handle () || % escape pressed || % Confirmation Dialog || exit properly the program
-> Keydown Handle () || % escape pressed | % delete pressed || remove selected itens from listview
3. Double Click on Item of Listview ~ Execute the App or File of the Item
```

Uma lógica DSL possui um cabeçalho (no exemplo: [Nome ou Descrição]{ Contexto }), possui definições de nomes curtos (X := Descrição), e diretivas chamadas de caminhos lógicos que se iniciam com números ou setas. A lógica DSL mais elementar é uma lógica que possui apenas cabeçalho e uma declaração pura, escrita em linguagem informal, descrevendo o que se quer do programa. Porém, para ter maior controle sobre o resultado da inteligência artificial, símbolos e notações são usados em combinação com descrições informais para refinar o resultado.

Tipos de Caminhos Lógicos :

1. Declaração Pura : Descrição sem símbolos ou notações descrevendo um comportamento do programa.
2. Declaração End Point : Caminho Lógico que termina em uma descrição informal.
3. Armação : É um caminho lógico que tem como objetivo estruturar o programa e geralmente não termina em uma descrição informal.
4. Caminho Aberto : É um caminho terminado com um símbolo de interrogação (?) e que pode ter dois significados, para o ser humano indica que a definição desta parte da lógica está sendo postergada, para a inteligência artificial significa que esta parte deve ser sugerida.

Exercício : Construa o código do programa **Side Panel** a partir da lógica. Use a lógica acima junto com o texto contido em DSL_AI_Prompt.txt para que a I.A. tenha consciência do código. Não se esqueça de pedir o passo a passo de como fazer o programa funcionar e instalar as dependências usando a ferramenta pip de controle de repositórios (tome cuidado para não errar a digitação da instalação do pyqt, sempre tire as dúvidas com a I.A.).

O programa **Side Panel** se criado corretamente deverá ser um painel semitransparente de ícones do lado direito da tela. Deverá ter os eventos de clique com botão direito, e de pressionamento de teclado. Para adicionar atalhos clique com o botão direito e ele deverá abrir um diálogo de busca de arquivos, para remover pressione a tecla delete, e para sair pressione a tecla escape. Observe que você pode usar a descrição deste parágrafo como entrada do prompt, então porque utilizar o paradigma DSL? Controle e reprodutibilidade, você terá mais controle da estrutura do código, permitindo uma engenharia mais planejada de sua construção, e o resultado da I.A. será mais constante e previsível que comparado com uma descrição livre.

A linguagem DSL busca aproveitar o espaço ao máximo para compactar as instruções. Por isso os símbolos | e || que são usados para explicitar estruturação hierárquica na mesma linha (blocos). $A||B$ significa que B pertence a estrutura (bloco) de A, ou está subordinado a A, ou pertencem ao escopo de A, enquanto $A|B$ significa que A e B pertencem a uma mesma estrutura (bloco) e estão no mesmo nível, sendo B executado depois de A. Os símbolos %, &, (), $ são respectivamente para denotar condicional, laço, função e variável, enquanto os outros símbolos como a enumeração, as setas e $\sim$ tem como finalidade anotação ou clareza. Na notação de variável usamos a forma [$ (descrição de tipo) descrição da variável], para explicitar o tipo da variável além de sua descrição informal. E temos X : Y { } para denotar a classe X derivada da classe Y.

Exercícios : Faça pequenas modificações do programa pela Lógica DSL para tomar consciência da sintaxe DSL. Tente fazer as mesmas modificações no código python gerado do programa.

1. Mude o Título do Programa
2. Peça uma imagem como Background
3. Adicione uma escuta de evento de teclado enter para executar ou abrir um item selecionado. Use como base a diretiva do botão delete.
4. Coloque as instruções de posição transparência dentro da diretiva C de criação da janela principal (Main Window Creation).
5. Mude a posição da janela e a transparência.

```
* O que é Regexp ?
** Para que o Regexp pode ser utilizado ?
** Regexp de Metacaracteres !
*** Sintaxe Regexp para dígitos, caracteres de Palavra, espaço em Branco ?
*** Sintaxe Regexp para qualquer caractere, para início de linha, para fim de linha ?
*** Sintaxe para não dígito ?
*** Sintaxe para alfanumérico ?
*** Sintaxe para não espaço ?
** Regexp de Operadores de Repetições !
*** Sintaxe de operador para zero ou mais vezes ?
*** Sintaxe de operador para uma ou mais vezes ?
*** Sintaxe de operador de zero ou uma vez ?
*** Sintaxe de operador de exatamente n vezes ?
*** Sintaxe de operador de pelo menos n vezes ?
*** Sintaxe de operador de n até m vezes ?
** Regexp de Conjuntos !
```

```
*** Sintaxe de Regexp para conjunto de caracteres ?
*** Sintaxe de Regexp para complementar de conjunto de caracteres ?
*** Sintaxe de Intervalo de Letras ?
*** Sintaxe de Intervalo Numérico ?
** Regexp de Âncoras !
*** Sintaxe Regexp de Início da String ou da Linha ?
*** Sintaxe Regexp de Fim da String ou da Linha ?
*** Sintaxe Regexp de palavra isolada ?
*** Sintaxe Regexp de string dentro de string ?
** O que é um grupo ? Qual a diferença entre grupo e conjunto ?
** O que é um grupo com captura e sem captura ?
```

Conceitos : GUI, Widget, Label, Button, CheckBox, RadioButton, LineEdit, MultiLineEdit, ComboBox, SpinBox, Slider, ProgressBar, Frame, Tabs, ScrollArea, ContextMenu.

```
* O que é Interface Gráfica de Usuário - GUI ?
* O que é widget no contexto de GUI ?
* O que é um menu de contexto ?
```

Abaixo temos uma versão mais refinada da lógica DSL para o programa **Side Panel**. Adicionamos salvamento de estado através da criação de um arquivo, adicionamos também um menu de contexto com duas opções, uma para procurar arquivos, a outra para filtrar os arquivos visíveis com base em uma expressão regular (regexp).

```
[ Side Panel ] { PyQt5 }
I := Imports
G := Global Variables
A := Main Application Object
C := Main Window Creation
1. /QtSidePanel.pyw || I | A | C | Show Window | Start Event Loop
-> A || $ (QApplication) App
-> C || $ (QMainWindow) MainWindow | Set Black Background | Set White Text |
 Set Oppaccity 60 percent | Set Window as a Side Panel at Right Part of the Desktop Screen |
 Center Vertically to the Screen |
 The Height of Application must be 65 percent of screen height |
 The width must be the size of icon plus some padding | $ (IconListView) list_view
-> ... $ (IconListView) list_view || list_view occupy all the client area
2. IconListView : QListView {} || $ (list of strings) visible_items_file_path |
 $ (list of strings) items_file_path | $ (string) filter | init () |
 Keydown Handle () | Context Menu Event Handle () | Update Visible Icons () |
 Process Action () | Search for File or App () | Apply Filter () |
 Save () | Load ()
-> Save () ||
 save items_file_path on current executable directory as lines of the file "QtSidePanel.sav"
-> Load () || % QtSidePanel.sav exists || load items_file_path with the lines of this file
-> Load () || % QtSidePanel.sav exists | % else || create an empty file
-> ... init () || ... | $* filter || empty string
-> ... init () || ... | $* filter | Load ()* | Update Visible Icons ()*
3. Context Menu Event Handle () || Create Menu | Add Action "Select File or Application"
 | Add Action "Apply Filter" | Process Action ()*
4. Search for File or App () || Open a Dialog to Search for any files or applications
 | % Selected a File or Application with Dialog ||
 Add full path to items_file_path | Update Visible Icons ()*
5. Apply Filter () || Open a Input Dialog with display text "Apply Filter" |
```

```
% If the text is a valid regexp || update filter | Update Visible Icons ()*
6. Process Action () || $ action | % action is "Select File or Application" |
 % action is "Apply Filter"
-> Process Action () || $ action | % action is "Select File or Application" ||
 Search for File or App ()*
-> Process Action () || $ action | % action is "Select File or Application" |
 % action is "Apply Filter" || Apply Filter ()*
7. Update Visible Icons () || % filter not empty string || create regexp object |
 & element in visible_items_file_path || remove elements that don't match
-> ... | & element in visible_items_file_path |
 update the list_view with icons of visible_items_file_path
8. Keydown Handle () || % escape pressed || % Confirmation Dialog || Save ()* |
 exit properly the program
-> Keydown Handle () || % escape pressed | % delete pressed ||
 remove selected item file path of items_file_path | Update Visible Icons ()*
9. Double Click on Item of Listview ~ Execute the App or File of the Item
```

Exercícios :

1. Tente construir o programa a partir da lógica.
2. Coloque vários atalhos e pratique regexp na opção de filtragem.
3. Modifique a lógica DSL para que ao apertar um botão (por exemplo, F5) o programa desempenhe a mesma função do menu de contexto para aplicar filtro. Teste a construção e execução do programa.
4. Modifique a lógica DSL para adicionar um item do menu de contexto com a finalidade de usar um input de caminho de arquivo para adicionar um novo atalho. Teste a construção e execução do programa.
5. Modifique a lógica DSL para adicionar um item no menu de contexto que irá criar um pop-up com slider (pergunte a I.A. como criar um pop-up com slider) para controlar a transparência da janela. Teste a construção e execução do programa.
6. Modifique a lógica DSL para adicionar um item no menu de contexto para modificar a cor do background.

Se você usar a mesma conversa dos programas anteriores a inteligência artificial irá manter o progresso e apenas mudar as partes necessárias, é um bom exercício testar a construção do programa em uma nova conversa ou mesmo com outra inteligência artificial. Você irá perceber que terá que fazer ajustes e corrigir erros do programa, e que irá haver variações e inconstâncias.