from fpdf import FPDF import os

--- CONTEÚDO DO README ---

titulo = "Otimizador de Portfólio & Backtest (Markowitz Pro)"

texto_intro = """ Ferramenta de engenharia financeira em Python que une a Teoria Moderna do Portfólio (Markowitz) com validação prática via Backtest.

O sistema calcula a alocação ideal (Max Sharpe) em dois cenários (Livre vs. Restrito) e gera relatórios automáticos comparando a performance da IA contra uma carteira manual e o índice Nasdaq-100. """

funcionalidades = ["1. Otimização de Média-Variância: Encontra a fronteira eficiente usando PyPortfolioOpt.", "2. Matriz de Covariância Robusta: Aplica Ledoit-Wolf Shrinkage para reduzir ruídos estatísticos.", "3. Cenários Duplos:", " - Sem Restrições (0-100%): A 'aposta' matemática pura.", " - Com Restrições (Compliance): Respeita limites mínimos e máximos (ex: 1% a 30%).", "4. Automação Inteligente: O backtest lê automaticamente as configurações usadas no otimizador.", "5. Relatórios Excel: Gera dashboards com gráficos nativos via XlsxWriter."]

instalacao = ["1. Criar e Ativar Ambiente Virtual:", " Windows: python -m venv .venv e depois ..venv\Scripts\Activate", " Linux/Mac: python3 -m venv .venv e depois source .venv/bin/activate", "", "2. Instalar Bibliotecas:", " pip install numpy pandas yfinance matplotlib PyPortfolioOpt xlsxwriter openpyxl scikit-learn"]

uso = ["Passo 1: Configurar Ativos", "Edite o arquivo 'data/raw/assets.csv' e insira os tickers (ex: AAPL, AMZN).", "", "Passo 2: Otimizar (markowitz_optimizer.py)", "Calcula a carteira ideal. Edite MIN_ALOCACAO e MAX_ALOCACAO neste arquivo se desejar.", "Comando: python markowitz_optimizer.py", "", "Passo 3: Validar (compare_strategies.py)", "Realiza o backtest dos últimos 12 meses.", "Comando: python compare_strategies.py"]

grafico_final = """ Entendendo o Resultado Final:

- Linha Cinza (Carteira Inicial): Desempenho da sua lista original.
- Linha Azul (Alocação Sem Restrições): Carteira com potencial teórico máximo (mais volátil e sem restrições de alocação).
- Linha Verde (Alocação Com Restrições): A carteira sugerida deve obedecer critérios de alocação de portfólio.
- Linha Laranja (Benchmark): Referência de mercado (Nasdaq-100). """

--- CLASSE PARA GERAR O PDF ---

class PDF(FPDF): def header(self): self.set_font('Arial', 'B', 16) # Título Centralizado self.cell(0, 10, self.title_doc, 0, 1, 'C') self.ln(10)

```
def chapter_title(self, label):
    self.set_font('Arial', 'B', 12)
    self.set_fill_color(200, 220, 255) # Azul claro
    self.cell(0, 6, f"{label}", 0, 1, 'L', 1)
```

```
        self.ln(4)

    def chapter_body(self, body):
        self.set_font('Arial', '', 11)
        # Resolve problemas de acentuação convertendo para latin-1
        txt = body.encode('latin-1', 'replace').decode('latin-1')
        self.multi_cell(0, 6, txt)
        self.ln()

    def chapter_list(self, items):
        self.set_font('Arial', '', 11)
        for item in items:
            txt = item.encode('latin-1', 'replace').decode('latin-1')
            self.multi_cell(0, 6, txt)
        self.ln()
```

--- EXECUÇÃO ---

pdf = PDF() pdf.title_doc = titulo # Passa o título para a classe pdf.add_page()

Seção 1: Intro

pdf.chapter_body(texto_intro)

Seção 2: Funcionalidades

pdf.chapter_title("Funcionalidades Principais") pdf.chapter_list(funcionalidades)

Seção 3: Instalação

pdf.chapter_title("Instalação e Dependências") pdf.chapter_list(instalacao)

Seção 4: Uso

pdf.chapter_title("Como Utilizar") pdf.chapter_list(uso)

Seção 5: Gráfico

pdf.chapter_title("Análise dos Resultados") pdf.chapter_body(grafico_final)

Salvar

output_path = os.path.join(os.getcwd(), "README_Markowitz.pdf") pdf.output(output_path)

print(f"Sucesso! PDF gerado em: {output_path}")