

// laboratório codex 03 · mercado financeiro

python + ia + app

>_ Laboratório Codex 03

Simulador de carteira em Python, com IA

Um app de verdade, no navegador: você monta uma carteira, o app busca a Selic no Banco Central, desenha os gráficos, desconta o imposto e entrega o relatório em Excel.

// apostila do aluno III · o projeto em etapas

01 Dados do Banco Central

02 Cálculos da carteira

03 Gráficos

04 O app no navegador

05 Cenários e painel

06 Relatório em Excel

07 Monte Carlo

08 Apresentação



Prof. Caio Haffner

Mestre em Desenvolvimento Econômico · Unicamp

1 Como usar esta apostila

✖ LEIA TUDO, NA ORDEM

Leia esta apostila **desde a primeira palavra** e faça tudo o que ela manda, na ordem. Não pule direto para os exercícios: cada página prepara a seguinte. Se ainda não fez a **Apostila I** (inclusive a página "PARE. Leia antes de tudo"), comece por ela.

Esta é a **Apostila III**, que continua a I (instalação e primeiros passos) e a II (exercícios no terminal). Agora você vai construir **um projeto completo**: um **simulador de carteira de investimentos** que abre no navegador, busca a Selic no Banco Central, mostra gráficos e gera um relatório em Excel.

1. Dados

Selic e IPCA do Banco Central

2. Cálculos

Rendimento, IR e aportes

3. Gráficos

Evolução e alocação

4. App

A tela no navegador

5. Relatório

Painel e Excel

★ O PROJETO: SIMULADOR DE CARTEIRA

O usuário informa quanto tem, quanto aplica por mês, por quanto tempo e como divide o dinheiro entre Tesouro Selic, CDB, LCI e ações. O app mostra a evolução de cada aplicação, o valor final já descontado o imposto, o ganho acima da inflação e compara com a poupança.

Ao final, você deverá conseguir

Habilidade	Como saber que consegui	Ok
Buscar dados reais	O app traz a Selic e o IPCA do Banco Central, com plano B sem internet.	☐
Simular aplicações	Calculo mês a mês, com aportes, IR regressivo e isenções.	☐
Fazer gráficos	Mostro evolução e alocação em gráficos corretos.	☐
Construir um app	O usuário mexe nos controles e vê o resultado na hora.	☐
Gerar relatório	O app entrega um Excel com abas.	☐
Apresentar	Explico o app em 3 minutos, com um erro da IA.	☐

1**REGRA DE OURO**

Você é responsável pelo código que entrega, mesmo quando a IA o escreveu. **Não entregue código que você não consegue explicar.** No projeto, vale mais uma: **a cada etapa, o app precisa continuar funcionando.** Teste antes de seguir.

2 Antes de começar

Você precisa do que montou nas Apostilas I e II: VS Code, Python e Codex funcionando. Se faltar algo, volte à **Apostila I, seções 3 a 7**. A novidade são três **bibliotecas** que dão ao Python gráficos, telas e planilhas.

Instale as bibliotecas (uma vez só)

No terminal do VS Code, digite:

TERMINAL

```
python -m pip install streamlit matplotlib pandas openpyxl
```

Biblioteca	Para que serve
pandas	Trabalhar com tabelas (como uma planilha dentro do Python).
matplotlib	Fazer gráficos: linhas, barras, pizza, dispersão.
streamlit	Transformar o programa num aplicativo que abre no navegador.
openpyxl	Salvar e ler arquivos do Excel (.xlsx).

Para conferir, digite:

TERMINAL

```
python -c "import streamlit, matplotlib, pandas, openpyxl; print('tudo certo!')"
```



CHECKPOINT

O terminal mostrou `tudo certo`.



SE DER ERRO

Se aparecer "pip não é reconhecido" ou "python não é reconhecido", troque `python` por `py` (como na Apostila I). Se a instalação demorar, é normal: são bibliotecas grandes. Se travar, copie a mensagem e use o prompt universal de diagnóstico.

Crie a pasta do projeto

Crie a pasta `Laboratorio_Codex_MF3` em Documentos e abra no VS Code. Ao longo das etapas, ela vai ficar assim:

TERMINAL

```
Laboratorio_Codex_MF3/  
├─ dados/           ← arquivos de dados (CSV)  
├─ calculos.py      ← as contas (funções)  
├─ graficos.py      ← os gráficos  
├─ app.py           ← a tela (Streamlit)  
└─ relatorio.xlsx   ← gerado pelo app
```

i GUARDE VERSÕES

Ao terminar cada etapa com o app funcionando, copie a pasta inteira com o nome da etapa (por exemplo, `Laboratorio_Codex_MF3_etapa2`). Se algo quebrar depois, você volta.

3 Seu primeiro app

Antes do projeto, um teste rápido das ferramentas novas. Crie um arquivo `teste_app.py` com o código abaixo. Você pode digitar ou pedir ao Codex.

```
import streamlit as st  
import matplotlib.pyplot as plt  
  
st.title("Meu primeiro app")  
valor = st.number_input("Valor aplicado (R$)", value=1000.0)  
taxa = st.slider("Taxa ao mês (%)", 0.0, 2.0, 1.0)  
meses = st.slider("Meses", 1, 60, 12)  
  
saldos = [valor * (1 + taxa / 100) ** m for m in range(meses + 1)]  
fig, ax = plt.subplots()  
ax.plot(saldos)  
ax.set_xlabel("Mês")  
ax.set_ylabel("R$")  
st.pyplot(fig)  
st.metric("Valor final", f"R$ {saldos[-1]:,.2f}")
```

Para abrir o app, digite no terminal:

TERMINAL

```
python -m streamlit run teste_app.py
```



CHECKPOINT

O navegador abriu um app com o título "Meu primeiro app", dois controles deslizantes e um gráfico. Mexa nos controles: o gráfico muda na hora.

Complete a tela

TERMINAL

Valor aplicado: 1000 · Taxa ao mês: 1,00 · Meses: 12

Valor final: R\$ _____



ARMADILHA: O FORMATO DO NÚMERO

O app mostra o valor no formato americano (vírgula no milhar e ponto nos centavos). Peça ao Codex: *"mostre os valores no formato brasileiro, com ponto no milhar e vírgula nos centavos"*. Você vai usar isso em todo o projeto.



COMO O STREAMLIT FUNCIONA

Cada vez que você mexe num controle, o Streamlit roda o programa inteiro de novo, de cima para baixo. Por isso o gráfico se atualiza sozinho. Para fechar o app, volte ao terminal e aperte `Ctrl` + `C`.



LEMBRETES DAS APOSTILAS ANTERIORES

Peça sempre um **plano** antes do código · teste com contas feitas por você · peça o **diagnóstico antes da correção** · aceite vírgula ou ponto · só aprove o que entender.

4 Etapa 1 • Os dados de mercado

Toda simulação começa pelas taxas. Em vez de digitar a Selic, o app vai buscá-la no **Banco Central**. E, se a internet cair, usa um valor guardado: o **plano B**.

★ DE ONDE VÊM OS DADOS

API do SGS do Banco Central: <https://api.bcb.gov.br/dados/serie/bcdata.sgs.{código}/dados/ultimos/1?formato=json> · **432** meta da Selic · **13522** IPCA 12 meses.

PROMPT 1

Crie um arquivo `calculos.py` com uma função `buscar_taxas()` que:

1. busque no SGS do Banco Central a meta da Selic (série 432) e o IPCA de 12 meses (série 13522), usando só a biblioteca padrão (`urllib` e `json`);
2. devolva as duas taxas em % ao ano e a data de cada uma;
3. se não houver internet, use um plano B: Selic 13,75% e IPCA 4,22%, e avise que são valores de referência;
4. salve a última consulta em `dados/taxas.csv`.

Mostre como testar a função no terminal. Antes de escrever, apresente um plano.

☐ CHECKPOINT

Rodando a função, o terminal mostrou a Selic e o IPCA com as datas. Desligando o Wi-Fi, apareceu o aviso do plano B.

Taxa	Valor	Data	Fonte (BCB ou plano B)
Selic			
IPCA 12 meses			

5

Etapa 2 · Os cálculos da carteira

Agora as contas, mês a mês. É a Apostila II inteira trabalhando junta: taxas equivalentes, CDI, IR regressivo e isenções.

★ REGRAS DA SIMULAÇÃO

CDI = Selic - 0,10 ponto · taxa mensal equivalente = $(1 + \text{taxa anual})^{1/12} - 1$ · aportes no fim de cada mês · CDB e Tesouro Selic: IR regressivo sobre o rendimento no resgate · LCI e poupança (0,5% ao mês): isentas · ações: retorno anual informado pelo usuário, IR de 15% sobre o ganho no resgate (simplificação).

PROMPT 2

Em `calculos.py`, crie a função `simular(valor_inicial, aporte_mensal, meses, alocao, selic, ipca, pct_cdb, pct_lci, retorno_acoes)` que:

1. divida o valor inicial e cada aporte entre Tesouro Selic, CDB, LCI e ações pela alocação em % (a soma tem que dar 100);
2. calcule mês a mês o saldo de cada aplicação (taxas mensais equivalentes);
3. no fim, calcule o IR de cada uma pelas regras e o valor líquido;
4. calcule a poupança com o mesmo dinheiro, para comparar;
5. devolva uma tabela (DataFrame do pandas) com o saldo de cada mês e um resumo com bruto, IR e líquido de cada aplicação.

Explique as regras antes de escrever o código.

Teste: R\$ 10.000 iniciais, sem aportes, 12 meses, Selic 13,75%, 100% no Tesouro Selic.

Teste	Você espera	O app calculou
Valor bruto em 12 meses		
IR (20%)		
Valor líquido		
Poupança no mesmo período		

! CONFIRA COM A APOSTILA II

Esses números são os mesmos da atividade final da Apostila II. Se não baterem, alguma regra foi programada errado: use o diagnóstico antes da correção.

Etapa 3 · Os gráficos

Um bom gráfico responde a uma pergunta. Aqui são três: como o dinheiro cresce, como ele está dividido e qual aplicação rende mais.

PROMPT 3

Crie um arquivo graficos.py com três funções que recebem os resultados da simulação e devolvem gráficos do matplotlib (fig):

1. grafico_evolucao: linhas com o saldo de cada aplicação mês a mês, mais a poupança tracejada, com título, eixos e legenda;
2. grafico_alocacao: pizza com a divisão da carteira, em %;
3. grafico_liquido: barras com o valor líquido de cada aplicação, do maior para o menor.

Use valores em reais no formato brasileiro nos eixos.

Mostre como testar salvando os gráficos em imagens (png).



CHECKPOINT

Na pasta apareceram três imagens: evolução, alocação e líquido. Os títulos e as legendas estão em português.

- ☐ Todo gráfico tem título
- ☐ Eixos com nome e unidade (R\$, meses)
- ☐ Legenda quando há mais de uma linha
- ☐ Cores diferentes para cada aplicação, iguais nos três gráficos

7 Etapa 4 • O app no navegador

Hora de juntar tudo numa tela. Os controles ficam na **barra lateral**; os resultados, no centro.

PROMPT 4

Crie o arquivo `app.py` com Streamlit:

- barra lateral (`st.sidebar`) com: valor inicial, aporte mensal, prazo em meses (slider de 1 a 120), alocação em % de cada aplicação (sliders), % do CDI do CDB e da LCI e retorno esperado das ações;
 - avise em vermelho se a alocação não somar 100%;
 - no centro: as taxas usadas (com a data e a fonte), quatro indicadores com `st.metric` (total investido, valor líquido, ganho acima da inflação e diferença para a poupança) e os três gráficos da etapa 3.
- Use as funções de `calculos.py` e `graficos.py`, sem repetir contas no app.

TERMINAL

```
python -m streamlit run app.py
```



CHECKPOINT

O app abriu no navegador. Mudando o prazo de 12 para 24 meses, os gráficos e os indicadores mudaram na hora.

Teste no app	O que aconteceu
Alocação somando 90%	
Prazo de 3 meses (IR de 22,5%)	
Ações com retorno de -10%	

8

Etapa 5 · Cenários e painel completo

Investidor sério olha **cenários**, não só um número. Ações podem render muito ou nada. O painel vai mostrar isso lado a lado.

PROMPT 5

Melhore o app.py:

1. organize em abas (st.tabs): "Simulação", "Cenários" e "Tabela mês a mês";
 2. na aba Cenários, rode a simulação com as ações rendendo 0%, o retorno informado e o dobro dele, e mostre um gráfico comparando o valor líquido final;
 3. na aba Tabela, mostre a tabela mês a mês com st.dataframe, em reais.
- Mantenha os cálculos em calculos.py.



CHECKPOINT

As três abas funcionam. Na aba Cenários, dá para ver quanto a carteira depende das ações.



PARA PENSAR

Com 80% em ações, o cenário pessimista ainda vence a poupança? E com 20%? Use o app para responder e anote.

Etapa 6 • O relatório em Excel

O cliente quer levar a simulação para casa. O app vai gerar um **Excel com abas** e um botão para baixar.

PROMPT 6

Acrescente ao app um botão "Baixar relatório em Excel" (`st.download_button`) que gere um arquivo `.xlsx` com três abas:

1. "Premissas": todos os valores informados, as taxas e a fonte;
2. "Resumo": bruto, IR e líquido de cada aplicação e da poupança;
3. "Mês a mês": a tabela completa.

Use `pandas` com `ExcelWriter` (`openpyxl`). Formate os números com 2 casas.



CHECKPOINT

O botão baixou o arquivo. Abrindo no Excel, as três abas estão lá, com os números iguais aos do app.

- ☐ O Excel abre sem erro
- ☐ As abas têm nomes claros
- ☐ Os valores batem com o app
- ☐ A data e a fonte das taxas estão nas premissas

10 Desafio extra · Mil futuros possíveis

As ações não rendem um número fixo. A **simulação de Monte Carlo** sorteia milhares de caminhos possíveis e mostra a faixa de resultados.

PROMPT 7

desafio

Acrescente uma aba "Risco" ao app: simule 1.000 caminhos para as ações, sorteando o retorno de cada mês com média e volatilidade informadas pelo usuário (use o módulo random). Mostre um histograma dos valores finais da carteira e os resultados que separam os 10% piores e os 10% melhores casos. Explique a ideia de Monte Carlo antes de escrever o código.

11 Use o Codex como revisor do projeto

REVISAR A ORGANIZAÇÃO

Revise o meu projeto como um programador experiente revisaria o de um iniciante. Não faça alterações. Aponte: funções muito longas, código repetido, nomes pouco claros e cálculos misturados com a tela. Separe o que é obrigatório do que é opcional.

REVISAR O CONTEÚDO

Revise os cálculos do meu projeto como um professor de Mercado Financeiro. Não faça alterações. Confira o desconto do CDI (-0,10 ponto), a tabela do IR, as isenções da poupança e da LCI e a conversão de taxas e liste o que pode estar errado.



GRÁFICO BONITO NÃO É GRÁFICO CERTO

Confira pelo menos um número de cada gráfico com a calculadora ou com a Apostila II. A IA erra eixos, unidades e somas com frequência.

Entrega: apresente o seu projeto

Um projeto só existe quando alguém o vê funcionando. Prepare uma **apresentação de 3 minutos** e um registro para o seu portfólio.

Roteiro de 3 minutos

Tempo	O que mostrar
30 s	O problema que o app resolve e para quem.
1 min 30 s	O app funcionando: mexa nos controles e mostre quanto a carteira rende contra a poupança e o efeito do IR.
30 s	Um erro que a IA cometeu e como você descobriu.
30 s	O que você faria a seguir, se tivesse mais tempo.

O que entregar

- ☐ A pasta do projeto, com todos os arquivos .py e os dados
- ☐ O relatório em Excel gerado pelo app
- ☐ Três prints do app (telas principais)
- ☐ Um arquivo LEIAME.md explicando o que o app faz e como rodar (peça um rascunho ao Codex e revise)
- ☐ O diário de prompts: pelo menos 5 prompts importantes e o que aconteceu



PARA O SEU PORTFÓLIO

Um post no LinkedIn com um vídeo curto do app funcionando ("Construí em Python, com ajuda de IA, um app de simulação de carteira de investimentos") mostra a recrutadores algo que poucos estudantes têm: **uma ferramenta real feita por você.**

Reflexão individual

1. Qual etapa foi a mais difícil, e como você a resolveu?

2. Em que momento a IA errou, e como você percebeu?

3. O que você aprendeu de Mercado Financeiro construindo o app que não tinha aprendido só com a teoria?

13 Como você será avaliado

Critério	0	1	2	3
O app funciona	Não abre	Abre com erros	Funciona	Funciona e trata entradas inválidas
Conteúdo	Cálculos errados	Alguns certos	Quase todos certos	Todos certos e conferidos
Gráficos	Nenhum	Sem títulos ou eixos	Corretos	Corretos e fáceis de ler
Relatório	Não gera	Gera incompleto	Gera completo	Completo e organizado em abas
Uso do Codex	Não usou	Só copiou	Usou com compreensão	Usou de forma crítica
Apresentação	Não apresentou	Confusa	Clara	Clara, no tempo e com o erro da IA
Registro da auditoria humana (opcional, vale ponto extra)	☐			

Total máximo: 18 pontos, convertidos para a escala da disciplina.

14 Consulta rápida

Comandos

TERMINAL

```
python -m pip install streamlit
matplotlib pandas openpyxl
python -m streamlit run app.py    # abre
o app
Ctrl + C                          #
fecha o app
```

Streamlit

<code>st.title · st.header</code>	Títulos
<code>st.number_input · st.slider</code>	Entradas de número
<code>st.selectbox · st.file_uploader</code>	Escolha e upload
<code>st.metric · st.dataframe</code>	Indicador e tabela
<code>st.pyplot(fig)</code>	Mostrar um gráfico
<code>st.tabs · st.sidebar</code>	Abas e barra lateral
<code>st.download_button</code>	Botão para baixar o Excel

Glossário

Biblioteca

Código pronto que você instala para ganhar recursos (gráficos, telas).

pip

O instalador de bibliotecas do Python.

DataFrame

Tabela do pandas, com linhas e colunas.

App web

Programa que roda no navegador.

Aporte

Dinheiro novo colocado na aplicação.

IR regressivo

Imposto que diminui com o prazo da aplicação.

Alocação

Como o dinheiro se divide entre as aplicações.

Cenário

Hipótese sobre o futuro (ex.: ações rendem 12% ou 0%).

1 LEMBRE-SE

A IA é uma ferramenta. Conhecimento, julgamento, validação e responsabilidade continuam sendo de quem programa. Nada nesta apostila é recomendação profissional: é aprendizado.

Todo material produzido com IA precisa de auditoria humana. Sempre. A IA escreve rápido e com segurança, mas pode errar uma conta, inventar uma regra ou usar um dado desatualizado. E ela escreve o erro com a mesma confiança do acerto.

Esta apostila foi escrita com ajuda de IA e revisada pelo professor, conta por conta. Mesmo assim, algum erro pode ter escapado. Agora o auditor é você: encontre, comprove e relate.



REGRA DO LABORATÓRIO: IA SEMPRE PASSA POR AUDITORIA HUMANA

A IA não sabe quando errou: o texto errado sai tão convincente quanto o certo.

Quem assina o resultado é uma pessoa: você no trabalho, o analista no relatório, o professor na apostila.

Auditar não é desconfiar da ferramenta. É a etapa que transforma um rascunho da IA num material confiável.

Vale para tudo: código, contas, textos, gráficos e esta apostila.

O que procurar

Tipo de erro	Exemplo	Como conferir
Conta ou número	Um total que não fecha, uma porcentagem errada	Refaça na calculadora
Regra da disciplina	Fórmula trocada, conceito mal explicado	Compare com o livro ou o material da aula
Referência	"Veja a seção 9", quando o assunto está na 11	Vá até a seção indicada
Prompt	Um prompt que faz o Codex errar sempre do mesmo jeito	Rode o prompt duas vezes
Dado desatualizado	Selic, IPCA, regra de imposto	Confira no site oficial (Banco Central, IBGE, Receita Federal)
Texto	Frase confusa, erro de português	Leia o trecho em voz alta

Como auditar com o Codex

PROMPT DE AUDITORIA

use com cuidado

Vou colar um trecho da minha apostila de [DISCIPLINA].

Atue como revisor. Não reescreva o texto.

1. refaça cada conta e mostre o cálculo;
2. aponte afirmações que podem estar erradas ou desatualizadas;
3. para cada problema, diga como eu posso confirmar sozinho.

Trecho: [COLE AQUI]



A IA TAMBÉM ERRA NA AUDITORIA

O Codex pode apontar erros que não existem. Só relate o que você confirmou com uma conta, uma fonte oficial ou um teste.

Registro da auditoria

#	Seção e página	O que está escrito	O que deveria ser	Como confirmei
1				
2				
3				

Como enviar

- 1 Preencha a tabela acima.
- 2 Envie cada erro pelo formulário: **laboratorio-codex.pages.dev/auditoria**
- 3 Aguarde a confirmação do professor.



O QUE VOCÊ GANHA

Cada erro confirmado vale **+1 ponto extra** na avaliação (até 3 pontos), e o seu nome entra na **lista de revisores** da próxima versão da apostila. Se duas pessoas relatarem o mesmo erro, vale o primeiro envio.