

// laboratório codex 03 · macroeconomia

python + ia + app

>_ Laboratório Codex 03

Simulador macro em Python, com IA

Curvas IS e LM que se movem, choques de oferta e de demanda, estagflação na tela e o crescimento de Solow: a Macroeconomia num app feito por você.

// apostila do aluno III · o projeto em etapas

01 Modelos em funções

02 IS-LM e políticas

03 Oferta e demanda

04 O app em abas

05 Crescimento de Solow

06 Relatório em Excel

07 Economia aberta

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 macroeconômico**: o multiplicador, o IS-LM com políticas que deslocam as curvas, a oferta e a demanda agregadas com choques e o crescimento de Solow, tudo num app interativo.

1. Modelos

Funções testadas

2. Gráficos

IS-LM e políticas

3. Choques

Oferta e demanda agregadas

4. App

Abas interativas

5. Relatório

Solow e Excel



O PROJETO: SIMULADOR MACROECONÔMICO

Cada aba é um modelo da Apostila II em gráfico interativo: a renda de equilíbrio do multiplicador, as curvas IS e LM que se movem com a política fiscal e a monetária, os choques de demanda e de oferta no modelo de oferta e demanda agregadas e a trajetória do capital no modelo de Solow, com um relatório em Excel.

Ao final, você deverá conseguir

Habilidade	Como saber que consegui	Ok
Programar modelos	Funções testadas com os números da Apostila II.	☐
Desenhar o IS-LM	Curvas, equilíbrio e deslocamentos.	☐
Simular choques	Oferta e demanda agregadas, preços e hiato.	☐
Mostrar o crescimento	A trajetória de Solow até o estado estacionário.	☐
Comparar políticas	Fiscal x monetária, com juros e investimento.	☐
Explicar	O app escreve o que cada gráfico mostra.	☐

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_MACR03` em Documentos e abra no VS Code. Ao longo das etapas, ela vai ficar assim:

TERMINAL

```
Laboratorio_Codex_MACR03/  
├─ 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_MACR03_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 modelos em funções

Os modelos da Apostila II reunidos num arquivo e testados.

PROMPT 1

Crie calculos.py com funções para: renda de equilíbrio e multiplicador, inflação acumulada e juro real (Fisher), equilíbrio IS-LM (com política fiscal e monetária), oferta e demanda agregadas lineares e o estado estacionário de Solow. Escreva testes com assert usando os números da Apostila II.

Teste	Esperado	A função devolveu
Renda (C_0 200, c 0,8, I 300, G 400, T 250)	3.500 ($k = 5$)	
IS-LM (Exercício 4 da Apostila II)	r 2,5%, Y 3.250	
Juro real (Selic 15%, inflação 4,5%)	10,05%	
Solow (s 0,25, α 0,3, $n + \delta$ 0,06)	$k^* \approx 7,68$, $y^* \approx 1,84$	

5

Etapa 2 · O IS-LM e as políticas em gráfico

O gráfico mais famoso da Macroeconomia: duas curvas, um cruzamento, e o que acontece quando o governo ou o Banco Central agem.

PROMPT 2

Crie graficos.py com grafico_islm: as curvas IS e LM (juros no eixo vertical, renda no horizontal), o equilíbrio marcado e, se houver política, as curvas novas tracejadas, o novo equilíbrio e uma seta mostrando o deslocamento.



CHECKPOINT

Com $G + 40$, a IS vai para a direita e o equilíbrio passa de (3.250; 2,5%) para (3.350; 3,5%).

6

Etapa 3 · Choques de oferta e de demanda

Com a oferta e a demanda agregadas, o app mostra o que cada choque faz com os preços e com o hiato, inclusive a temida **estagflação**.



demanda agregada: $Y = 4.000 - 750 \cdot P$ · **oferta agregada**
(curto prazo): $Y = 2.500 + 750 \cdot P$

Produto potencial: $3.250 \cdot P$: nível de preços (índice)

PROMPT 3

Em `calculos.py`, crie `equilibrio_oa_da(...)` e, em `graficos.py`, `grafico_oa_da`: as duas retas, o equilíbrio, o produto potencial como linha vertical e, para um choque informado (de demanda ou de oferta, em R\$ bilhões), as retas novas, o novo equilíbrio, a variação dos preços e o hiato do produto.

Choque	Preços (P)	Renda (Y)	Hiato
Nenhum			
Demanda + 300			
Oferta – 300			



PARA PENSAR

Nos dois choques os preços sobem para 1,2. Por que um é "bom" para o emprego e o outro é "ruim"? O que o Banco Central deveria fazer em cada caso?

Cada modelo numa aba, com controles que movem os gráficos.

PROMPT 4

Crie app.py com Streamlit e abas:

1. "Multiplicador": sliders para C_0 , c , I , G e T ; a renda e o multiplicador;
2. "IS-LM": os parâmetros na barra lateral, sliders de política (ΔG e $\Delta M/P$), o gráfico e uma tabela antes × depois (juros, renda e investimento);
3. "Oferta e demanda": o tipo e o tamanho do choque, o gráfico e o hiato.

TERMINAL

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

Teste no app	Resultado
c de 0,8 para 0,9: renda e multiplicador	
$\Delta M/P = 100$: juros e renda	
Choque de oferta – 300: hiato	

Etapa 5 · O crescimento de Solow

Uma aba de longo prazo: o capital por trabalhador caminhando até o estado estacionário.

PROMPT 5

Acrescente a aba "Crescimento": sliders para s , α , n e δ ; o gráfico da trajetória de k por 200 anos (com k^* como linha horizontal) e o gráfico do consumo de estado estacionário para poupanças de 0,10 a 0,50, com a regra de ouro marcada.

Teste no app	Resultado
$s = 0,25$: k^* e y^*	
$s = 0,30$: k^* e consumo	
Poupança da regra de ouro	

Etapa 6 · O relatório em Excel

O app gera um relatório dos experimentos, uma aba por modelo.

PROMPT 6

Acrescente um botão "Baixar relatório" com uma aba por modelo (Multiplicador, IS-LM, Oferta e demanda, Crescimento), com os parâmetros usados, as tabelas e a frase de explicação.



CHECKPOINT

O Excel tem 4 abas com os parâmetros que estavam na tela.

10 Desafio extra · Economia aberta e com impostos

Na vida real, parte da renda vira imposto e parte do gasto vira importação. O multiplicador fica **bem menor**.

PROMPT 7

desafio

Acrescente à aba Multiplicador a tributação proporcional (t) e a propensão a importar (m):

$k = 1 \div (1 - c \cdot (1 - t) + m)$. Mostre os dois multiplicadores lado a lado e o efeito de $G + 100$

em cada um. Com $c = 0,8$, $t = 0,25$ e $m = 0,1$, o multiplicador deve dar 2.

Antes de escrever o código, explique por que t e m reduzem o multiplicador.

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 Macroeconomia.

Não faça alterações. Confira as equações das curvas, o sinal de cada política, os eixos dos gráficos e as hipóteses 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 as curvas IS e LM se movendo com a política fiscal e a monetária.
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 macroeconômica") 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 Macroeconomia 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.

Demanda agregada

Quanto o país quer comprar a cada nível de preços.

Oferta agregada

Quanto as empresas produzem a cada nível de preços.

Estagflação

Inflação subindo com a produção caindo.

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.