Desenvolvimento de Sistemas

Aula 07 - Bibliotecas, APIs e Framework

Lorrany B A Marim

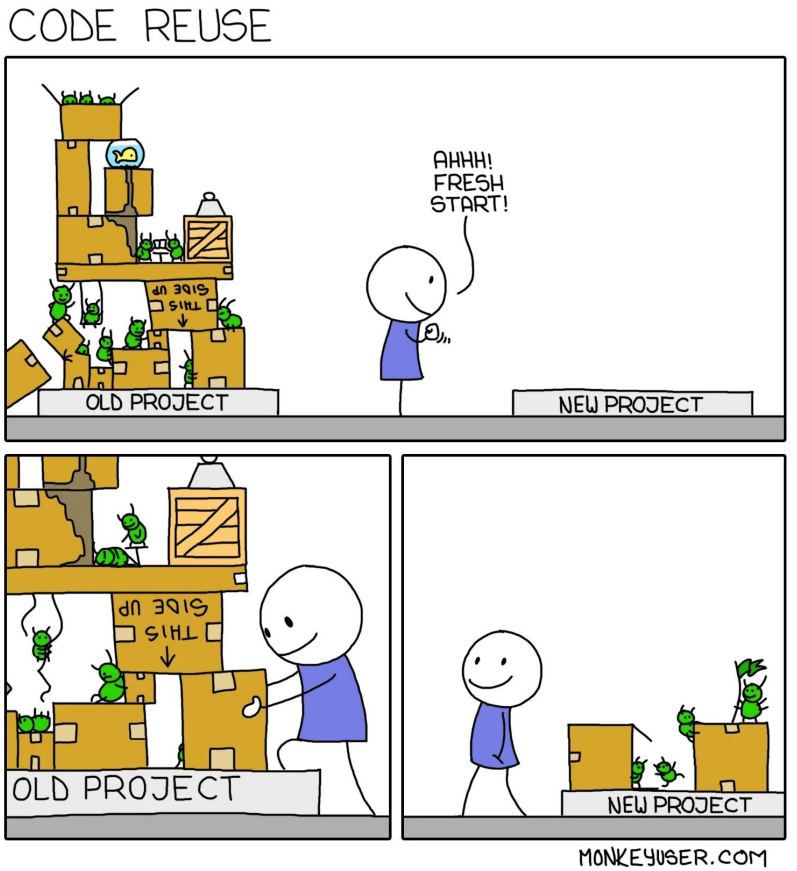
SENAI

O Conceito de Reutilização

Não reinvente a roda

- **Problema:** Escrever todo o código do zero é ineficiente e propenso a erros.

- **Solução:** Utilizamos código pronto criado por terceiros para acelerar o desenvolvimento.

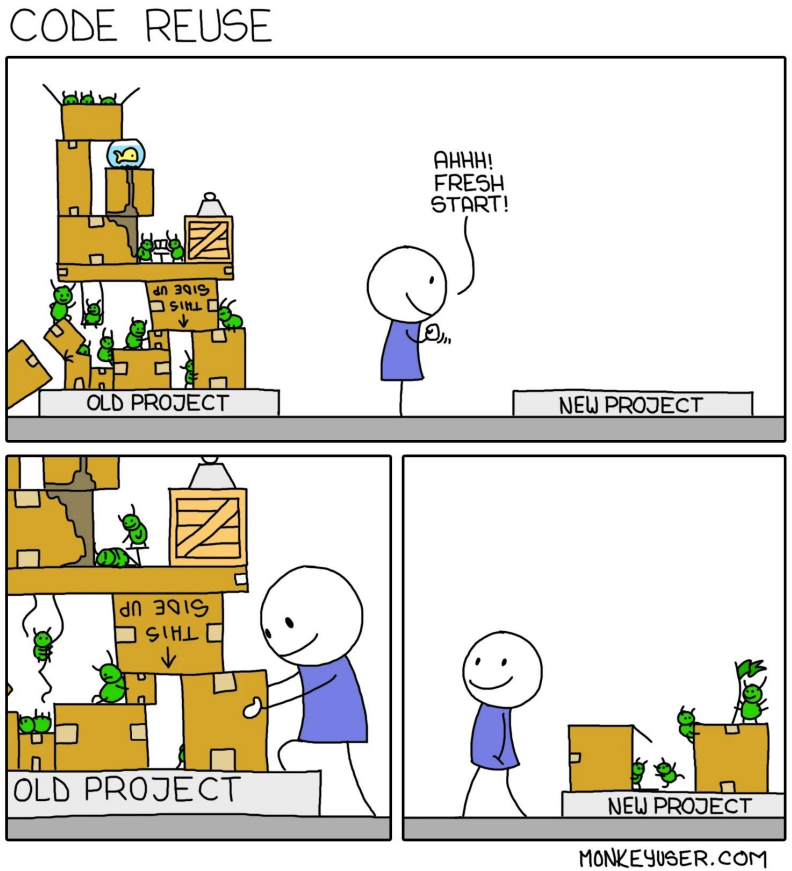


O Conceito de Reutilização

Não reinvente a roda

- **Formas de Reutilização:** As duas formas mais comuns são a **Composição de Objetos** (usar funcionalidades de caixas pretas) e a **Herança** (estender funcionalidades existentes).

- **Ferramentas:** Hoje vamos diferenciar três termos que costumam causar confusão: Bibliotecas, APIs e Frameworks.



O que é uma Biblioteca (Library)?

Um conjunto de ferramentas à sua disposição

- **Definição:** Uma biblioteca é um conjunto de classes e funções relacionadas e reutilizáveis, projetadas para fornecer funcionalidades úteis e de finalidade geral.

- **Como funciona:** Você, o programador, tem o controle. Seu código chama as funções da biblioteca quando precisa realizar uma tarefa específica.

O que é uma Biblioteca (Library)?

Um conjunto de ferramentas à sua disposição

- **Analogia:** É como uma caixa de ferramentas. Você decide quando pegar o martelo ou a chave de fenda.
- **Exemplos:**
  a. **Pandas** (Python). Você importa a biblioteca e chama funções para ler planilhas ou calcular médias.
  b. **Requests:** É a biblioteca para conversar com a internet. Seu código envia um pedido para um site ou API, e ela traz a resposta. Você não precisa entender de protocolos de rede complexos.
  c. **Matplotlib:** É a biblioteca de desenho. Você passa uma lista de números e ela gera gráficos, barras e diagramas profissionais.

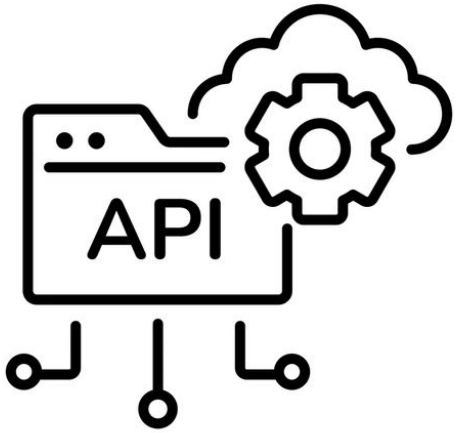
Código Python – Biblioteca

```python
import datetime

print("--- Cálculo de Idade Preciso (Uso de Biblioteca) ---")

nome = input("Digite seu nome: ")
dia = int(input("Dia do nascimento (dd): "))
mes = int(input("Mês do nascimento (mm): "))
ano = int(input("Ano do nascimento (aaaa): "))

# Uso da Biblioteca datetime para pegar a data atual
hoje = datetime.datetime.now()
nascimento = datetime.datetime(ano, mes, dia)

# Cálculo da diferença
diferenca = hoje - nascimento
anos = diferenca.days // 365
meses = (diferenca.days % 365) // 30

print(f"\nOlá {nome}, você tem exatamente {anos} anos e {meses} meses de idade.")
```

O que é uma API?

A interface de comunicação

- **Definição:** API (*Application Programming Interface*) é o contrato que define como dois sistemas se comunicam.
- **Interface Externa:** Enquanto a biblioteca é instalada "dentro" do seu projeto, o termo API é frequentemente usado para serviços externos que acessamos via rede (Web APIs).
- **Contrato:** A API define quais solicitações (requests) você pode fazer e quais respostas receberá, sem que você precise saber como o sistema funciona internamente.

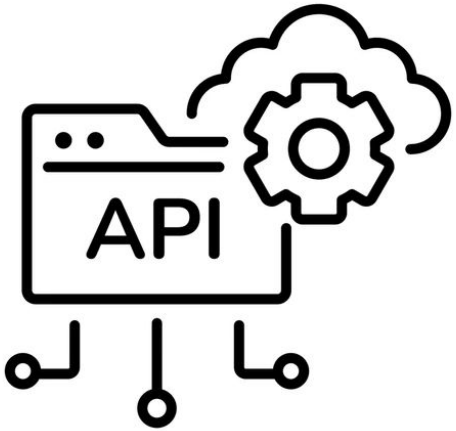


API Integration

O que é uma API?

A interface de comunicação

- **Separação:** A API externa não faz parte do seu código. Ela é um "serviço" rodando em outro lugar (outro servidor, outra empresa).
- **Comunicação:** Você não faz um "import". Você faz uma **requisição** (geralmente via rede/internet) enviando dados e esperando uma resposta.
- **Execução:** O processamento pesado acontece no servidor dono da API, não na sua máquina.



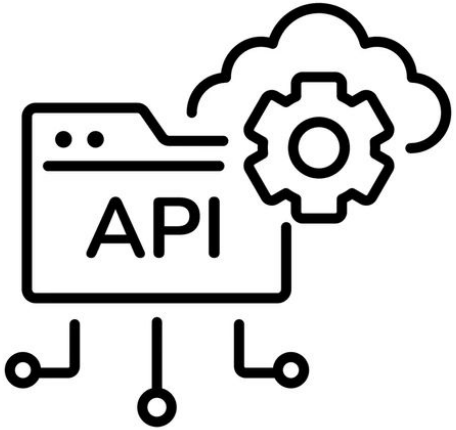
API Integration

O que é uma API?

- **Exemplo Real:**

  Google Maps API. Seu sistema envia um endereço para o Google, e a API devolve as coordenadas ou o mapa. O código do mapa não está no seu computador.

  YouTube Data API. Seu sistema envia o título de um vídeo ou o nome de um canal para o Google, e a API devolve o link do vídeo, o número de visualizações ou os comentários. O vídeo e o banco de dados do YouTube não estão no seu computador.



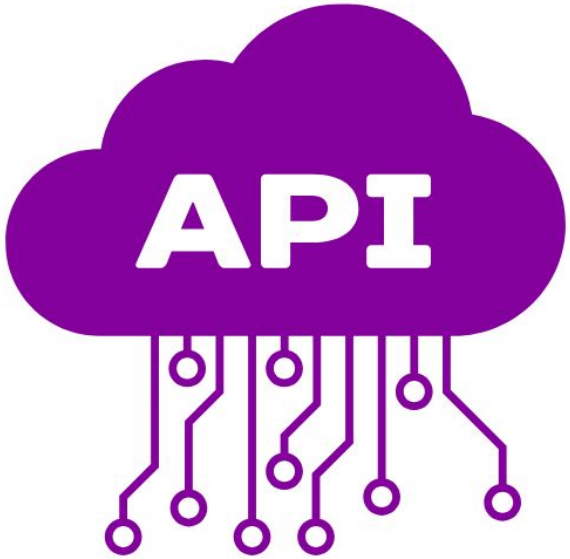
API Integration

O Problema da Comunicação

Sistemas diferentes, linguagens diferentes

- **O Desafio:** Um sistema escrito em Python precisa enviar dados para um sistema escrito em Java ou JavaScript. Eles não compartilham a mesma memória nem o mesmo formato de objetos.

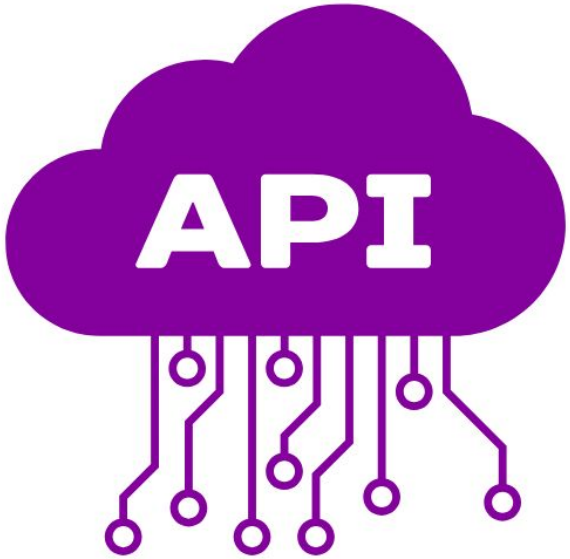
- **A Solução:** Precisamos de um formato de texto universal para trocar mensagens.



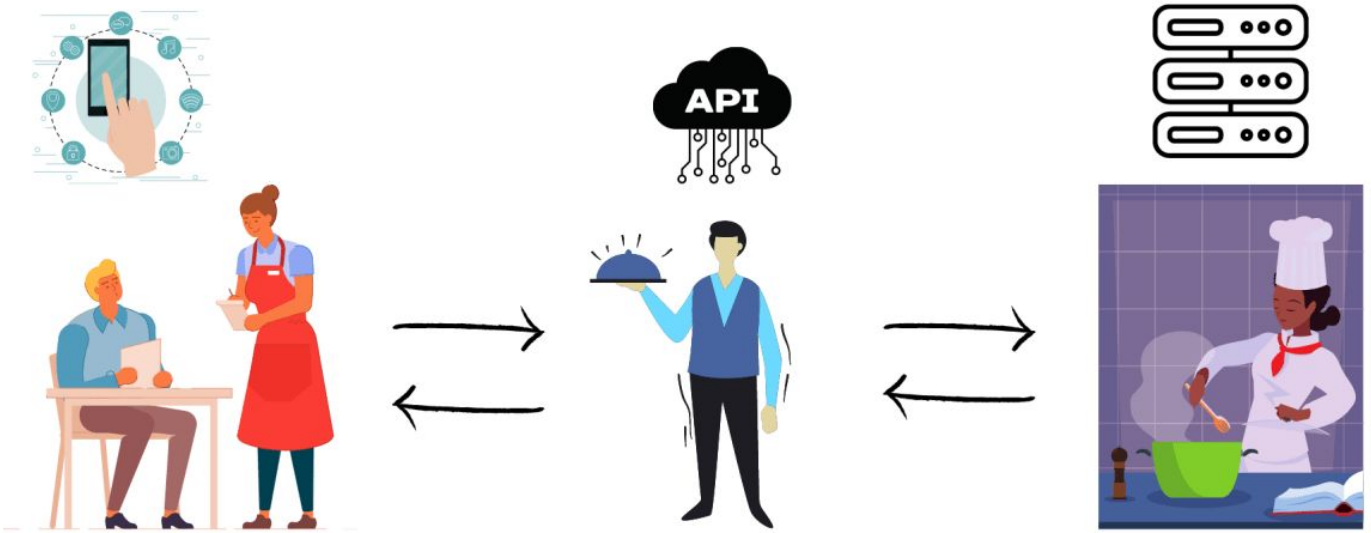
O Problema da Comunicação

- **Independência:** Protocolos como HTTP e formatos de mensagem não dependem da linguagem de programação. Você pode usar tecnologias diferentes para implementar cada parte do sistema.

- **Interface:** A API define a "assinatura" das operações disponíveis, ou seja, o que pode ser solicitado e o que será retornado.



What is API ?



Application/Customer

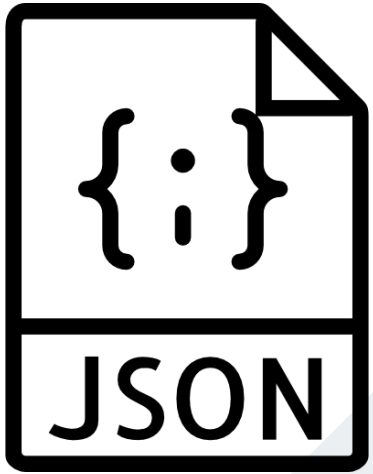
Waiter/API

Kitchen/Server

A “Linguagem” dos Dados

O formato padrão de mercado

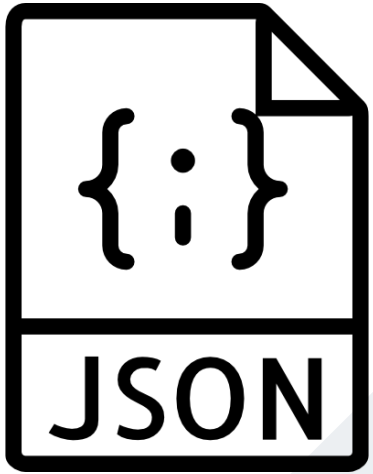
- **JSON (JavaScript Object Notation):** É o formato mais comum para APIs modernas. É leve, legível por humanos e fácil de processar por máquinas.
- **Estrutura:** Baseia-se em pares de chave/valor (semelhante aos dicionários em Python) e listas.
- **Interoperabilidade:** O JSON permite que dados complexos (como uma linha de banco de dados) sejam transformados em texto para viajar pela rede.



A “Linguagem” dos Dados

O formato padrão de mercado

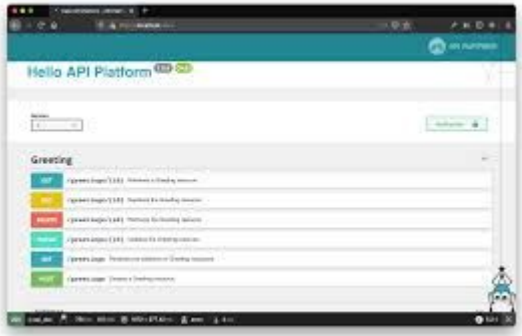
- **JSON (JavaScript Object Notation):** É o formato mais comum para APIs modernas. É leve, legível por humanos e fácil de processar por máquinas.
- **Estrutura:** Baseia-se em pares de chave/valor (semelhante aos dicionários em Python) e listas.
- **Interoperabilidade:** O JSON permite que dados complexos (como uma linha de banco de dados) sejam transformados em texto para viajar pela rede.



Plataformas para APIs

O que são:

- São softwares (muitos funcionam como IDEs especializadas) que permitem **enviar, receber e analisar dados** de uma API sem que você precise escrever um código completo em Python ou Java apenas para um teste rápido.
- Funcionam como o **Debugger** que você viu na Aula 06: permitem "espiar" o que está acontecendo por dentro da comunicação.



Plataformas para APIs

Plataformas mais usadas:

Assim como você conheceu o **VS Code** e o **IntelliJ** no material, no mundo das APIs as ferramentas padrão são:

- **Postman:** O "padrão da indústria". É uma plataforma completa para criar, testar e documentar APIs.
- **Insomnia:** Uma alternativa mais leve e elegante, focada na experiência do desenvolvedor.
- **Apidog:** Uma ferramenta moderna "tudo-em-um" que tem ganhado muito espaço por ser colaborativa.

Plataformas para APIs

Práticando com a API Via CEP:

A API **ViaCEP** é um serviço gratuito que recebe um CEP e devolve o endereço completo.

Passo 1: Configurar a Requisição

Abra sua plataforma (ex: Postman) e configure os campos:

- **Método:** Selecione GET (usado para buscar dados).
- **URL:** Digite https://viacep.com.br/ws/01001000/json/.

Passo 2: Enviar

Clique no botão **Send**. A plataforma enviará essa "pergunta" para o servidor da ViaCEP via internet.

Plataformas para APIs

Práticando com a API Via CEP:

Passo 3: Analisar o Retorno (JSON)

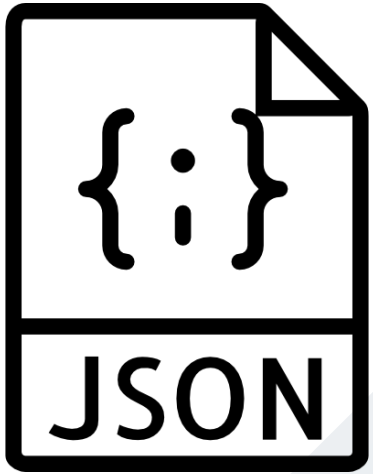
A plataforma receberá o retorno e mostrará na tela o status (como o 200 OK, que indica sucesso) e o corpo do dado.

Exemplo do que você verá na plataforma:


```
{
"cep": "01001-000",
"logradouro": "Praça da Sé",
"bairro": "Sé",
"localidade": "São Paulo",
 "uf": "SP"
}
```

Sintaxe Json

```json
{
  "nome": "João",
  "idade": 25,
  "estudante": true,
  "habilidades": ["JavaScript", "Python"],
  "endereco": {
    "rua": "Principal",
    "numero": 101
  }
}
```



Exemplo do Uso de API com Json

```python
# URL da API fictícia
api_url = "https://api.boletotech.com/v1/status"

# O JSON de envio (Payload)
payload = {
    "numero_boleto": "23793.38128 60083.035285 28000.063327 9 95230000010000"
}

try:
    # Enviando a requisição POST com o JSON
    response = requests.post(api_url, json=payload)

    # Verifica se a requisição foi bem-sucedida (status 200)
    if response.status_code == 200:
        # Como o código recebe o JSON: transformando em dicionário Python
        dados_retorno = response.json()

        status = dados_retorno.get("status")
        valor = dados_retorno.get("valor")

        print(f"Boleto processado com sucesso!")
        print(f"Status: {status.upper()}")
        print(f"Valor: R$ {valor}")
    else:
        print(f"Erro na API: {response.status_code}")

except Exception as e:
    print(f"Erro de conexão: {e}")
```

O código define a url (o caminho da api) e depois o payload (o que queremos saber que está dentro da api) então o envio ocorre e sua resposta é interpretada e mostrada para o usuário

Exemplo do Uso de API com Json

```
{
  "id_transacao": 98765,
  "status": "pago",
  "data_processamento": "2026-02-02T14:30:00Z",
  "valor": 150.00
}
```

a resposta da api será em json, porém, apenas a api define como será o padrão de suas respostas. Ao enviar o código do boleto a api retorna o id do boleto o status, data e valor, mas também poderia ter enviado o banco, que emitiu o boleto, para quem foi emitido, etc. Tudo isso depende do sistema da api e não de nós.

Código Python – Requisição para API

```python
import requests # É necessário instalar: pip install requests

print("--- Consulta de Endereço (Uso de API) ---")
cep = input("Digite o seu CEP (apenas números): ")

# Fazendo a requisição para a API externa
url = f"https://viacep.com.br/ws/{cep}/json/"
resposta = requests.get(url)

if resposta.status_code == 200:
    dados = resposta.json()

    if "erro" not in dados:
        print(f"\nLogradouro: {dados['logradouro']}")
        print(f"Bairro: {dados['bairro']}")
        print(f"Cidade: {dados['localidade']}")
        print(f"Estado: {dados['uf']}")
    else:
        print("CEP não encontrado.")
else:
    print("Erro na conexão com a API.")

```

O que é Framework?

O esqueleto da aplicação

- **Definição:** Um framework é um conjunto de classes que compõem um projeto reutilizável para uma categoria específica de software.

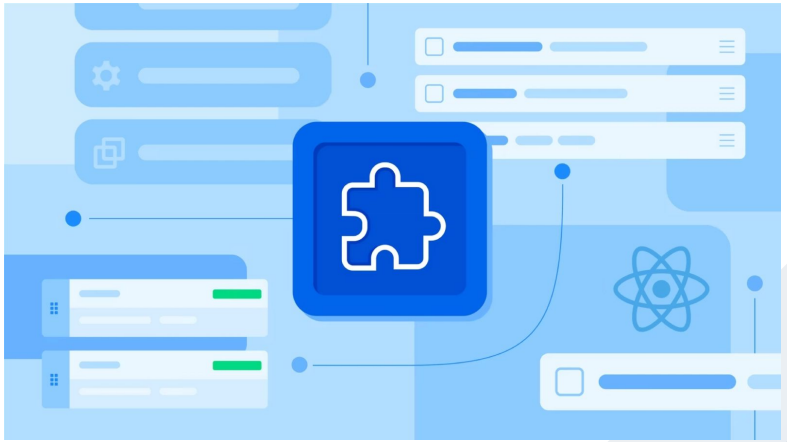
- **Diferença:** O framework dita a arquitetura da sua aplicação. Ele define a estrutura geral, o fluxo de controle e como as classes colaboram.

- **Objetivo:** Permitir que o desenvolvedor foque apenas nas regras de negócio específicas da aplicação, pois a infraestrutura já está pronta.

A Grande Diferença

Quem manda em quem?

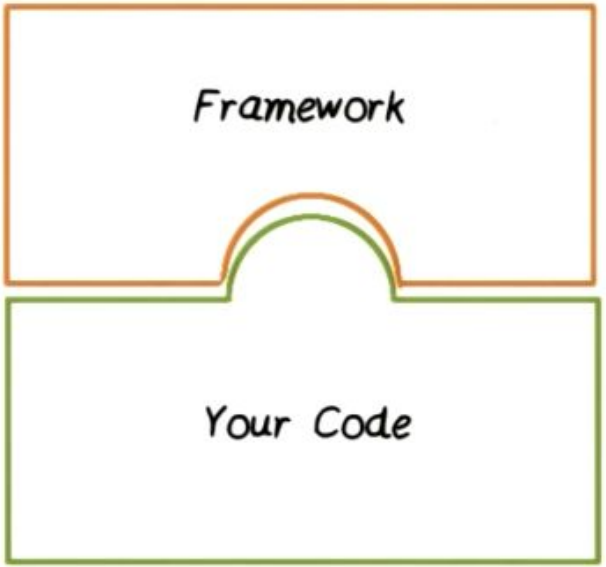
- **Biblioteca:** O seu código é o chefe. Você controla o fluxo principal e chama a biblioteca quando necessário.
- **Framework:** O framework é o chefe. Ele controla o fluxo principal da aplicação e chama o seu código nos momentos apropriados.
- **Resumo:** "Você chama a biblioteca; o framework chama você".



Inversão de Controle (IoC)

O Princípio de Hollywood

- **Conceito:** A característica que define um framework é a **Inversão de Controle** (IoC - *Inversion of Control*).
- **Princípio de Hollywood:** "Não nos chame, nós chamaremos você".
- **Na Prática:** Você escreve o código (ex: uma função de validação ou um controlador), mas quem decide *quando* executar esse código é o framework (ex: quando um usuário clica em um botão ou acessa uma URL).



Código Python – Framework Tkinter

```python
import tkinter as tk
from datetime import datetime

def calcular_contagem():
    try:
        nome = entry_nome.get()
        # Pega a data de nascimento e ajusta para o próximo aniversário
        nasc_str = entry_data.get() # formato dd/mm/aaaa
        hora_str = entry_hora.get() # formato HH:MM

        nascimento = datetime.strptime(f"{nasc_str} {hora_str}", "%d/%m/%Y %H:%M")
        hoje = datetime.now()

        # Define o próximo aniversário para este ano
        proximo_niver = nascimento.replace(year=hoje.year)

        # Se o aniversário já passou este ano, calcula para o ano que vem
        if proximo_niver < hoje:
            proximo_niver = proximo_niver.replace(year=hoje.year + 1)

        falta = proximo_niver - hoje

        meses = falta.days // 30
        dias = falta.days % 30
        horas = falta.seconds // 3600
```

Parte I

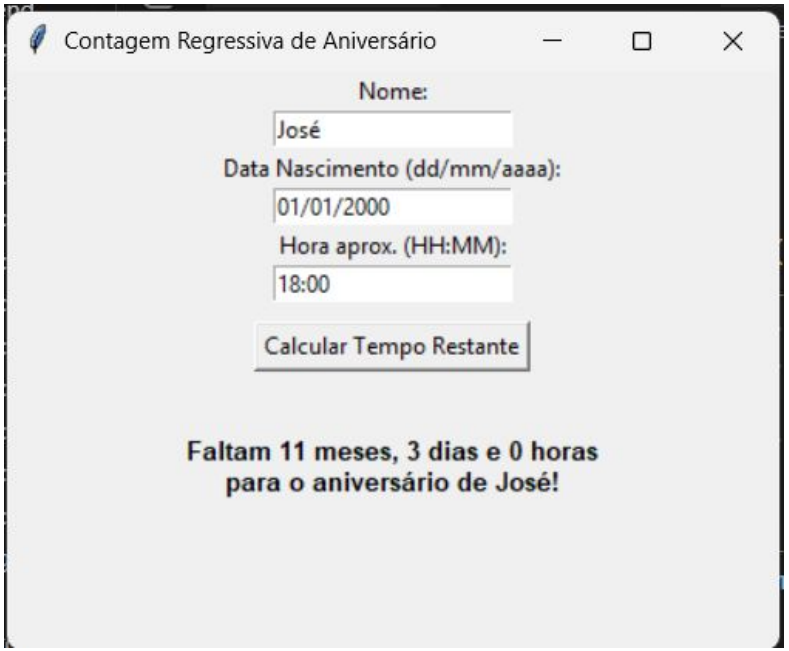
Código Python – Framework Tkinter

```
26
27          label_resultado.config(text=f"Faltam {meses} meses, {dias} dias e {horas} horas\npara o aniversário de {nome}!")
28      except ValueError:
29          label_resultado.config(text="Erro: Use os formatos dd/mm/aaaa e HH:MM", fg="red")
30
31  # Inicialização do Framework
32  root = tk.Tk()
33  root.title("Contagem Regressiva de Aniversário")
34  root.geometry("400x300")
35
36  # Elementos da Interface (Widgets)
37  tk.Label(root, text="Nome:").pack()
38  entry_nome = tk.Entry(root)
39  entry_nome.pack()
40
41  tk.Label(root, text="Data Nascimento (dd/mm/aaaa):").pack()
42  entry_data = tk.Entry(root)
43  entry_data.pack()
44
45  tk.Label(root, text="Hora aprox. (HH:MM):").pack()
46  entry_hora = tk.Entry(root)
47  entry_hora.pack()
48
49  # O Framework gerencia o clique do botão
50  btn = tk.Button(root, text="Calcular Tempo Restante", command=calcular_contagem)
51  btn.pack(pady=10)
52
53  label_resultado = tk.Label(root, text="", font=("Arial", 10, "bold"))
54  label_resultado.pack(pady=20)
55
56  # O controle passa a ser do Framework aqui
57  root.mainloop()
```

Parte II

SENAI

Código Python – Framework Tkinter



Resultado

Instalação vs. Acesso

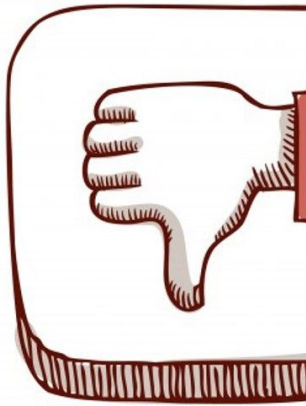
Como começamos a usar?

- **Biblioteca:** Exige instalação e gerenciamento de pacotes.
- Usamos ferramentas como o `pip` (em Python) para baixar os arquivos para nossa máquina.
- Precisamos nos preocupar com a versão instalada e se ela é compatível com nosso sistema operacional.
- **API Externa:** Exige autenticação e conectividade.
- Não instalamos o código do "Google Maps". Nós obtemos uma chave de acesso (API Key).
- Precisamos ter acesso à internet para alcançar o serviço (endpoint).

Vantagens e Desvantagens

Escolhendo a ferramenta certa

- **Bibliotecas:**
  - ***Vantagem:*** Flexibilidade total. Fácil de trocar se necessário.
  - ***Desvantagem:*** Você precisa escrever todo o código de conexão ("cola").
- **Frameworks:**
  - ***Vantagem:*** Desenvolvimento rápido. Muita coisa vem pronta (segurança, conexão com banco de dados).
  - ***Desvantagem:*** Curva de aprendizado alta. Você fica "amarrado" à arquitetura dele. Se o framework mudar, você precisa reescrever seu código.



Exemplos do Mercado

Tecnologias que usam IoC

- **Django (Python):** Um framework Web completo. Você define os "Modelos" e as "Visualizações", e o Django gerencia toda a recepção de requisições e acesso ao banco de dados.
- **Spring Boot (Java):** Um framework que usa Injeção de Dependência (uma forma de IoC) para gerenciar os objetos da aplicação. O Spring inicializa e conecta os componentes para você.
- **React (JavaScript):** Embora se autodenomine uma "biblioteca", possui características de framework. Ele gerencia o ciclo de vida dos componentes e "chama" seu método render quando os dados mudam (IoC).

Exemplos Comparativos

Calculando uma Rota

- **Cenário A (Biblioteca Interna):** Você baixa uma biblioteca de grafos (ex: `networkx`). Você tem o mapa da cidade salvo no seu banco de dados.
  - **Ação:** Seu computador processa todos os caminhos.
  - **Pró:** Funciona offline.
  - **Contra:** Você precisa manter o mapa atualizado e ter muita memória RAM.
- **Cenário B (API Externa):** Você envia o ponto A e B para o Google Maps API.
  - **Ação:** O Google calcula a rota nos supercomputadores deles e te devolve apenas o desenho do caminho.
  - **Pró:** Mapa sempre atualizado, trânsito em tempo real.
  - **Contra:** Paga-se por uso, não funciona sem internet.

Prática - Bibliotecas, APIs e Framework

O Projeto: World Guessing Game (Ranking Edition)

A lógica: O programa busca um país aleatório via API. O aluno deve adivinhar o nome do país com base em uma dica (como a capital ou o continente). No final, o sistema calcula a pontuação, aplica um bônus matemático e salva o recorde no banco de dados.

Como os 4 Pilares se encaixam:

1. **API (Rest Countries):**
   - **URL:** https://restcountries.com/v3.1/all
   - **Uso:** O código baixa uma lista de países e escolhe um aleatoriamente. Ele extrai o nome (resposta), a capital (dica) e a população.
2. **Biblioteca (math):**
   - **Uso:** Calcular a pontuação final. Exemplo: pontos = math.floor(1000 / tempo_gasto). Se o país tiver uma população pequena, o aluno ganha um multiplicador de dificuldade usando math.log.
3. **Biblioteca (datetime):**
   - **Uso:** Cronometrar quanto tempo o aluno levou para responder (diferença entre o momento em que a dica apareceu e o momento do palpite).
4. **Banco de Dados (mysql.connector):**
   - **Uso:** Uma tabela ranking com colunas: jogador, pontuacao, data_hora.
5. **Framework (tkinter):**
   - **Uso:** Uma tela com a dica, um campo de entrada para o palpite e uma tabela (Treeview) para mostrar os 5 melhores jogadores do banco.

Fontes Utilizadas

- *Downey, A. B. "Pense em Python".*
- *Gamma, E. et al. "Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos".*
- *Danjou, J. "Python Levado a Sério".*
- *Valente, M. T. "Engenharia de Software Moderna".*


Obrigado!
SENAI