Empacotando aplicações com Briefcase

Live de Python # 280

Roteiro

1. Compartilhamento de Software

Terminei meu código, como envio pra alguém?

2. Briefcase

Uma introdução a biblioteca

3. Criando builds

Um exemplo usando o shell

4. Usando GUI

Builds usando Toga e Pygame

apoia.se/livedepython

pix.dunossauro@gmail.com

patreon.com/dunossauro

Ajude o projeto <3

Adriana Cavalcanti, Alan Costa, Alexandre Girardello, Alexandre Harano, Alexandre Lima, Alexandre Takahashi, Alexandre Villares, Alfredo Braga, Allan Kleitson, Alysson Oliveira, Andre Azevedo, Andre Makoski, Andre Paula, Apc 16, Arthur Santiago, Aslay Clevisson, Aurelio Costa, Belisa Arnhold, Bernarducs, Biancarosa, Brisa Nascimento, Bruno Barcellos, Bruno Batista, Bruno Freitas, Bruno Ramos, Bruno Russian, Brunu, Carlos Gonçalves, Celio Araujo, Christian Fischer, Cleiton Fonseca, Controlado, Curtos Treino, Daniel Aguiar, Daniel Bianchi, Daniel Brito, Daniel Souza, Daniel Wojcickoski, Danilo Boas, Danilo Silva, David Couto, David Kwast, Denis Bernardo, Dgeison, Diego Guimarães, Dino, Diogo Faria, Edgar, Eduardo Pizorno, Emerson Rafael, Érico Andrei, Everton Silva, Fabio Barros, Fabio Faria, Fabiokleis, Felipe Adeildo, Felipe Augusto, Felipe Corrêa, Fernanda Prado, Fernandocelmer, Fichele Marias, Francisco Aclima, Frederico Damian, Fulvio Murenu, Gabriel Lira, Gabriel Mizuno, Gabriel Paiva, Gabriel Simonetto, Geilton Cruz, Geisler Dias, Giovanna Teodoro, Giuliano Silva, Guibeira, Guilherme Felitti, Guilherme Ostrock, Guilherme Piccioni, Gustavo Suto, Haelmo Almeida, Harold Gautschi, Heitor Fernandes, Hellyson Ferreira, Helton, Helvio Rezende, Henri Alves, Henrique Andrade, Henrique Machado, Hiago Couto, Hideki, Igor Taconi, Ivan Santiago, Janael Pinheiro, Jean Melo, Jean Victor, Jeferson Vitalino, Jefferson Antunes, Jefferson Silva, Jerry Ubiratan, Jhonata Medeiros, Jlx, Joao Rocha, John Peace, Jonas Araujo, Jonatas Leon, Joney Sousa, Jorge Silva, Jose Barroso, Jose Edmario, Joseíto Júnior, Jose Mazolini, José Predo), Josir Gomes, Jrborba, Juan Felipe, Juliana Machado, Julio Franco, Julio Silva, Kaio Code, Kaio Peixoto, Leandro O., Leandro Pina, Leandro Vieira, Leonan Ferreira, Leonardo Adelmo, Leonardo Mello, Leonardo Nazareth, Lisandro Pires, Lucas Carderelli, Lucas Castro, Lucas Mello, Lucas Mendes, Lucas Nascimento, Lucas Schneider, Lucas Simon, Luciano Ratamero, Luciano Teixeira, Luis Ottoni, Luiz Carlos, Luiz Duarte, Luiz Martins, Luiz Paula, Luiz Perciliano, Mackilem Laan, Marcelo Araujo, Marcelo Fonseca, Marcelo Grimberg, Marcio Freitas, Marcio Junior, Marcos Almeida, Marcos Oliveira, Marina Passos, Marlon Rocha, Mateusamorim96, Mateus Lisboa, Matheus Vian, Mírian Batista, Mlevi Lsantos, Murilo Carvalho, Ocimar Zolin, Otávio Carneiro, Patrick Felipe, Pedro Gomes, Pedro Henrique, Peterson Santos, Phmmdev, Prof Santana, Pytonyc, Rafael Faccio, Rafael Romão, Raimundo Ramos, Ramayana Menezes, Renan, Renan Sebastião, Rene Pessoto, Renne Rocha, Ricardo Silva, Ricardo Viana, Rinaldo Magalhaes, Rodrigo Barretos, Rogério Nogueira, Rui Jr, Samanta Cicilia, Santhiago Cristiano, Sergio Nascimento, Sherlock Holmes, Shirakawa, Tenorio, Téo Calvo, Tharles Andrade, Thiago Araujo, Thiago Lucca, Thiago Paiva, Tiago, Tomás Tamantini, Valdir, Varlei Menconi, Vinicius Meneses, Vinicius Silva, Vinicius Souza, Vinicius Stein, Vladimir Lemos, Williamslews, Willian Lopes, Zeca Figueiredo, Zero! Studio

Obrigado você

Compartilhando

Soft
ware

Acabei meu código. E agora?

A função de um sistema, de maneira geral, é ser executado por algo/alguém. A forma como isso é distribuído depende da forma de execução.

Por exemplo:

- Uma biblioteca/framework
  - Alguém usa como dependência para outro código
  - Tem foco em desenvolvimento
- Direto do código fonte
  - Espera um ambiente preparado (que tenha o python)
- Uma aplicação/software
  - Pode ser usado por outros sistemas
  - Pode ser usado por pessoas não técnicas
    - Como dar isso a elas?

Acabei meu código. E agora?

A função de um sistema, de maneira geral, é ser executado por algo/alguém. A forma como isso é distribuído depende da forma de execução.

Por exemplo:

- Uma biblioteca/framework
  - Alguém usa como dependência para outro código
  - Tem foco em desenvolvimento
- Direto do código fonte
  - Espera um ambiente preparado (que tenha o python)
- **Uma aplicação/software**
  - Pode ser usado por outros sistemas
  - **Pode ser usado por pessoas não técnicas**
    - Como dar isso a elas?

Distribuindo **aplicações**

Pronto, segue software em anexo. (app.py)

Como roda saporra?

Formatos de distribuição Python

A distribuição tem alguns problemas graves.

Da parte de desenvolvimento:

- Qual a minha **plataforma** de desenvolvimento?
- Qual a minha **arquitetura** de desenvolvimento?

Da parte de quem usa:

- Qual a **plataforma** da pessoa que vai usar?
- Qual a **arquitetura** da pessoa que vai usar?
- Qual o **sistema operacional** da pessoa que vai usar?

Refazendo a conversa

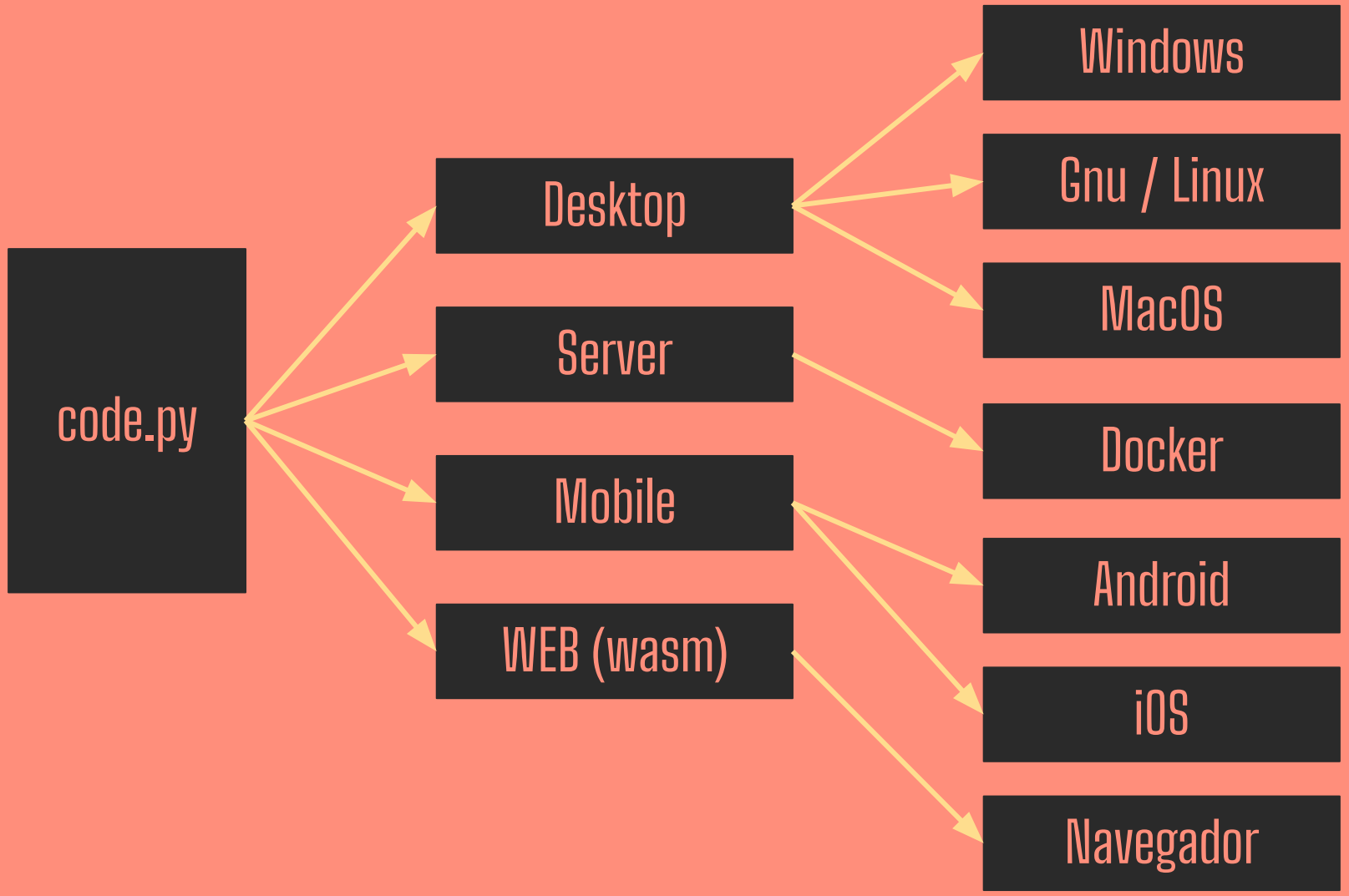
Quais os sistemas usados por aí?

Temos windows 11 e MacOS

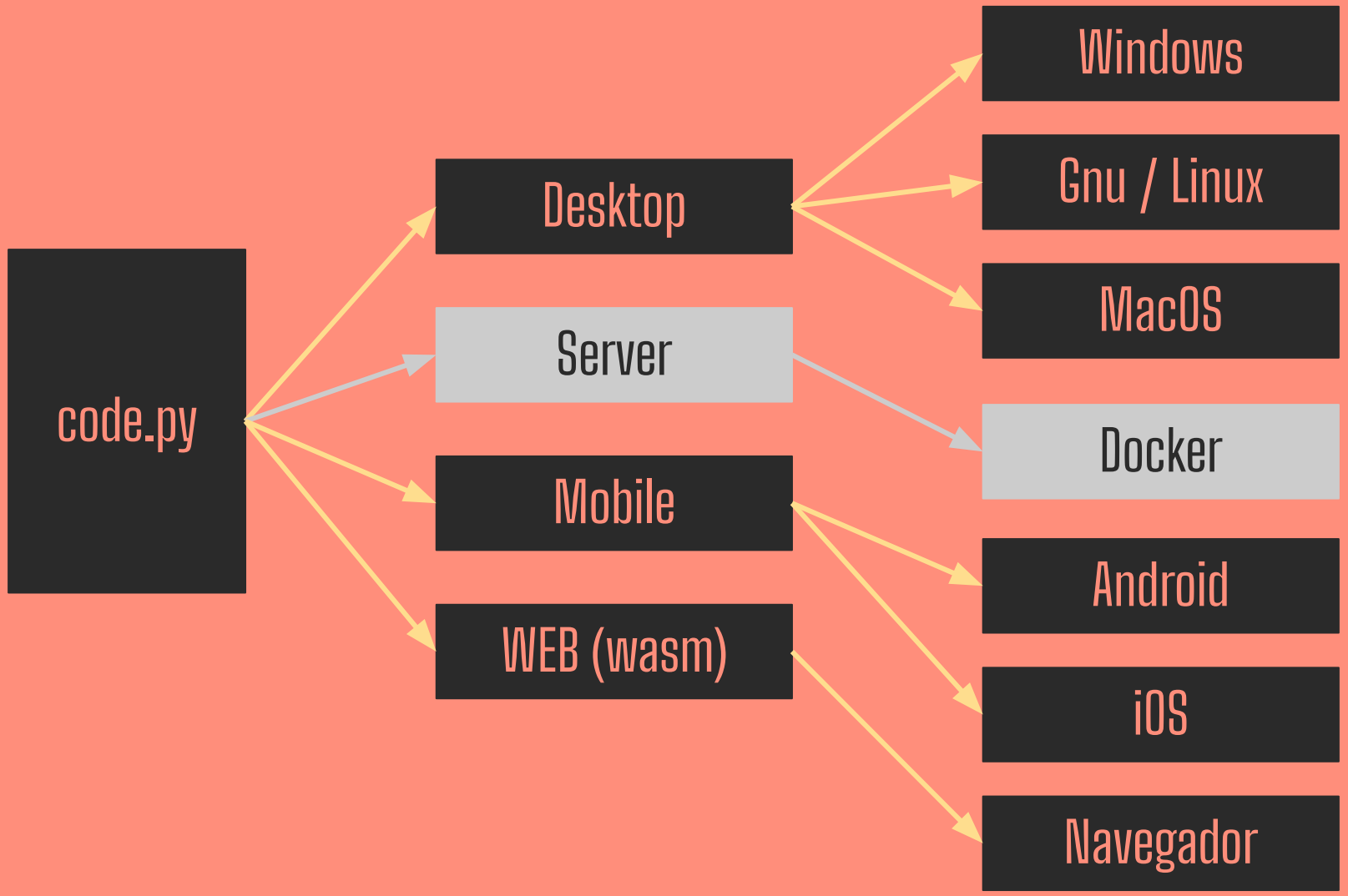
Seguem arquivos:

- app_windows**.msi**
- app_macos**.dmg**

Plataformas de saída possíveis



Aplicações para servidor não entram no papo hoje



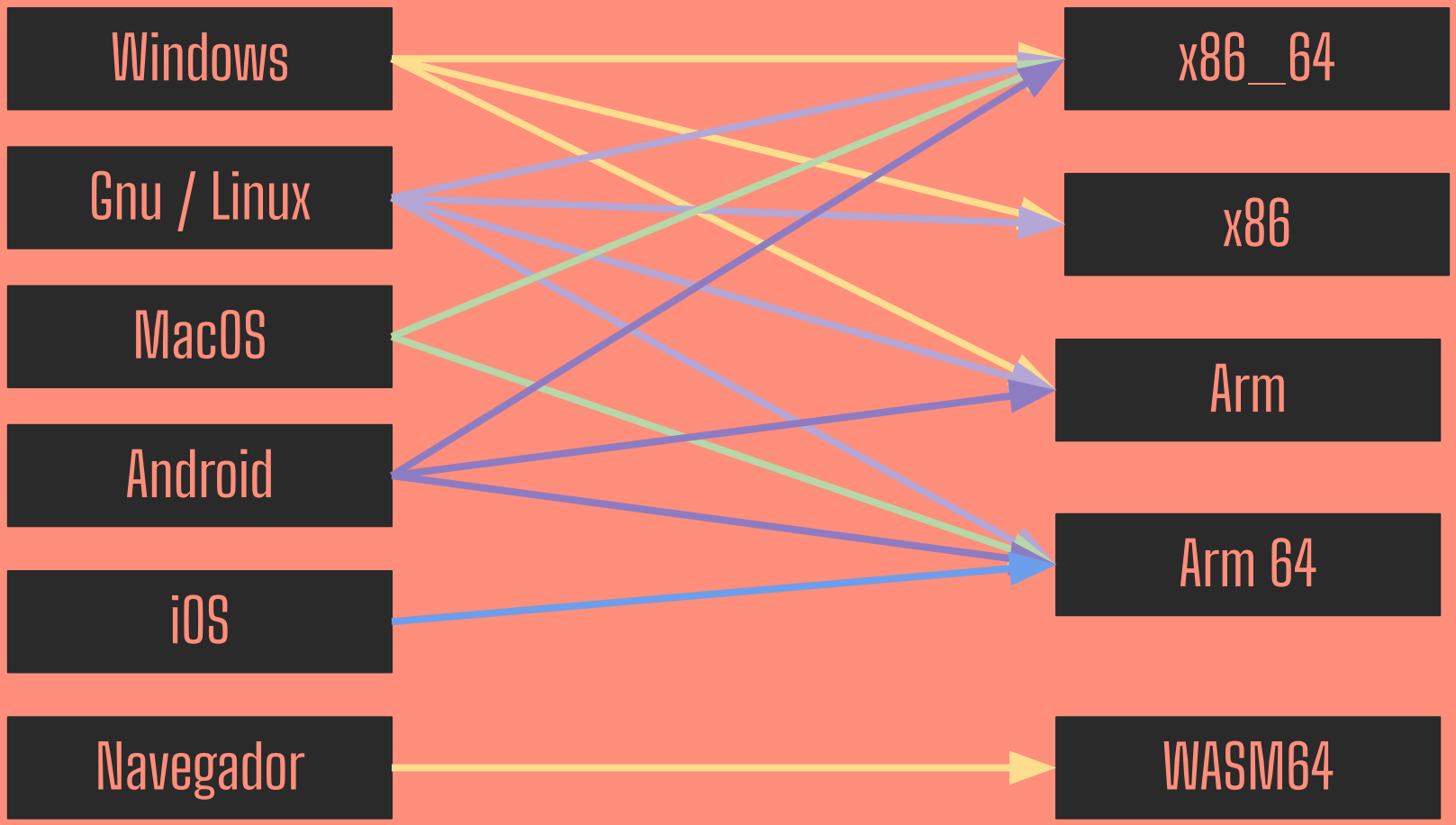
Distribuição de software

A ideia do nosso papo hoje é **empacotar** e distribuir **aplicações para** pessoas finais, **usuários finais**.

- Pessoas querem **usar a aplicação**
- Pessoas não precisam **saber python**
  - pip, pipx, venv, ...
- Pessoas não precisam de um **tutorial de instalação**

Precisamos de uma forma "uniforme" de distribuição.

Só que esse problema é complexo...



Tudo influencia o empacotamento

O seu software:

- É GUI? Essa biblioteca suporta a plataforma de destino?
- É Shell? O shell usa funções de shells específicos?
- O código é voltado para lidar com plataformas diferentes?

O python suporta a plataforma de saída? (ver PEP 11)

Existem diversas ferramentas de empacotamento

- **Pyinstaller** (tem live)
  - Cria um executável "portable"
  - Suporta desktops em geral (Win, Linux, mac, bsd, ...)
- **NSIS**
  - Cria uma distribuição com instalador
  - Suporta somente windows
- **Cx_freeze** (tem issue)
  - Cria uma distribuição com instalador
  - Suporta desktops (Linux, mac, windows)
  - Não suporta bsds
  - Ambiente complexo, mas completo
- **pyOxidizer** (tem live)
  - Cria um executável "portable"
  - Não é mais mantido

Brief case

O motivo de estarmos aqui!

Briefcase

Briefcase é uma das partes do projeto **Beeware**, que funciona de forma independente, para transformar aplicações python em instaladores nativos por plataforma.

- Licença: **BSD**
- Primeira release: **2015**
- Release atual: **0.3.22** (fev/2025)
- CLI - Interface de linha de comando
- **Multiplataforma** (windows, linux, mac)
- Suporte ao **pyproject.toml** (parcial)
- Suporte a **assinaturas**

```
pip install briefcase
# pipx é melhor pra isso xD
```

O nome Briefcase

briefcase

noun [C]

UK /ˈbriːf.keɪs/ US /ˈbriːf.keɪs/

Add to word list

a flat rectangular bag, used especially for carrying business documents

maleta, pasta, (para papéis, documentos, etc.)

O nome Briefcase

briefcase

noun [C]

UK /ˈbriːf.keɪs/ US /ˈbriːf.keɪs/

Add to word list

a flat rectangular bag, used especially for carrying business documents

maleta, pasta, (para papéis, documentos, etc.)

Plataformas suportadas

Briefcase suporta o empacotamento nativo para as seguintes plataformas:

- Windows
- Linux
- MacOS
- iOS
- Android
- Web

A construção vai depender das bibliotecas usadas

● Suportado e com CI | ○ Testado por mantenedores | △ Não testado regularmente

Target App Format		macOS		Windows			Linux			
		x86-64	arm64	x86	x86-64	arm64	x86	x86-64	arm	arm64
Android	Gradle project	●	●		●		△	●	△	△
iOS	Xcode project	●	●							
Linux	AppImage	△	△				△	△	△	△
	Flatpak						△	●	△	●
	System package	○	○				△	●	△	●
macOS	.app bundle	●	●							
	Xcode project	●	●							
Web	Static	●	●	△	●	△	△	●	△	●
Windows	Windows app				●					
	VS project				●					

https://briefcase.readthedocs.io/en/stable/reference/platforms/index.html

Por exemplo

Aplicações de Terminal (CLI, TUI, ...) são compatíveis com:

- Windows
- MacOS
- Linux

Aplicações GUI são compatíveis com:

- Windows, MacOS e Linux
- Android
- iOS
- Web

Bibliotecas suportadas oficialmente

- GUI
  - PySide
    - MacOS, Windows e Linux
  - PyGame
    - MacOS, Windows e Linux
  - Toga (beeware)
    - MacOS, Windows e Linux
    - Android
    - iOS
    - Web
    - TUI (textual)
- Terminal
  - MacOS, Windows e Linux

Bibliotecas suportadas via plugins

- Pygame-ce
  - Linux, Windows, MacOS, WEB
- PursuedPyBear
  - Linux, Windows e MacOS

Compilação cruzada?

Sim e não...

Qualquer plataforma com suporte a Docker pode fazer o build de sistemas disponíveis via docker (linux).

- Windows:
  - Nativo, Android, Linux (via docker), Web
- MacOS
  - Nativo, Android, Linux (docker), web, iOS
- Linux
  - Nativo, Linux (docker), Android, web, AppImage, Flatpak

Sobre o python!

Distribuição OFICIAL, mas alternativa:

- Não suporta pip
- Não suporta venv
- Não suporta tkinter
  - Não suporta TCL

Criando um app cli multiplataforma

BUILDS

Criando um projeto bobo!

```
poetry new exemplo
cd exemplo
poetry add cyclops
```

Criando um projeto bobo!

```python
# pkg/__main__.py
from cyclopts import App

app = App()

@app.default
def olar(nome):
    print(f'Olar {nome}')


app()
```

Usando com a aplicação pronta

O caso mais normal é já termos uma aplicação funcional. Pra isso podemos adicionar duas tabelas no nosso pyproject.toml:

```
[tool.briefcase]
project_name = "Exemplo"
bundle = "com.example"

[tool.briefcase.app.exemplo]
description = "Descrição curta..."
long_description = "Descrição longa..."
sources = ["exemplo"]  # ["src/exemplo"]
test_sources = ["tests"]
```

```toml
[tool.briefcase]
project_name = "Exemplo"
bundle = "com.example"
```

Tabela coms os metadados da distribuição

```toml
[tool.briefcase.app.exemplo]
description = "Descrição curta..."
long_description = "Descrição longa..."
sources = ["exemplo"]  # ["src/exemplo"]
test_sources = ["tests"]
```

Tabela coms os dados/meta da aplicação

Talvez seu pacote não tenha algumas coisas...

Como o briefcase precisa gerar um pacote, o seu pacote precisa de algumas coisas para termos um pacote válido:

- Um arquivo de **licença** e sua declaração no pyproject
  - license = {file = "LICENSE"}
  - ~ Ainda não suporta PEP 639
- Um arquivo de **changelog** *
  - Criado na raiz do projeto
- Um ponto de execução no pacote
  - Um arquivo **__main__.py**

* não necessários no windows

Executando pelo Briefcase

Agora podemos executar nosso projeto pelo briefcase no terminal.

Diversas coisas interessantes vão acontecer nesse momento!

Olhando o resultado no shell

A primeira passo do Briefcase é usar o **Cookiecutter**. Ele baixa um template para o seu sistema.

- Windows
- Linux
- Mac
- Android
- iOS

```
[exemplo] Generating application template…
Using app template: https://github.com/beeware/briefcase-linux-system-template.git...
```

Olhando o resultado no shell

Pacotes de plataforma (vamos ver isso depois)

[exemplo] Installing support package...
No support package required.

[exemplo] Installing application code...
Installing exemplo... done

Olhando o resultado no shell

pacotes no campo project.dependencies

[exemplo] Installing requirements...

...

Criando um pacote

Após executar e ver que tudo está funcionando bem, podemos criar um pacote. O formato do pacote vai depender do seu sistema operacional e da sua plataforma (/dist)

```
briefcase package
```

Targets e Linux

Um dos pontos fortes do briefcase é suportar compilação cruzada via **docker**. Com o docker instalado, usando a tag **--target**. Você pode gerar pacotes para todas as distribuições:

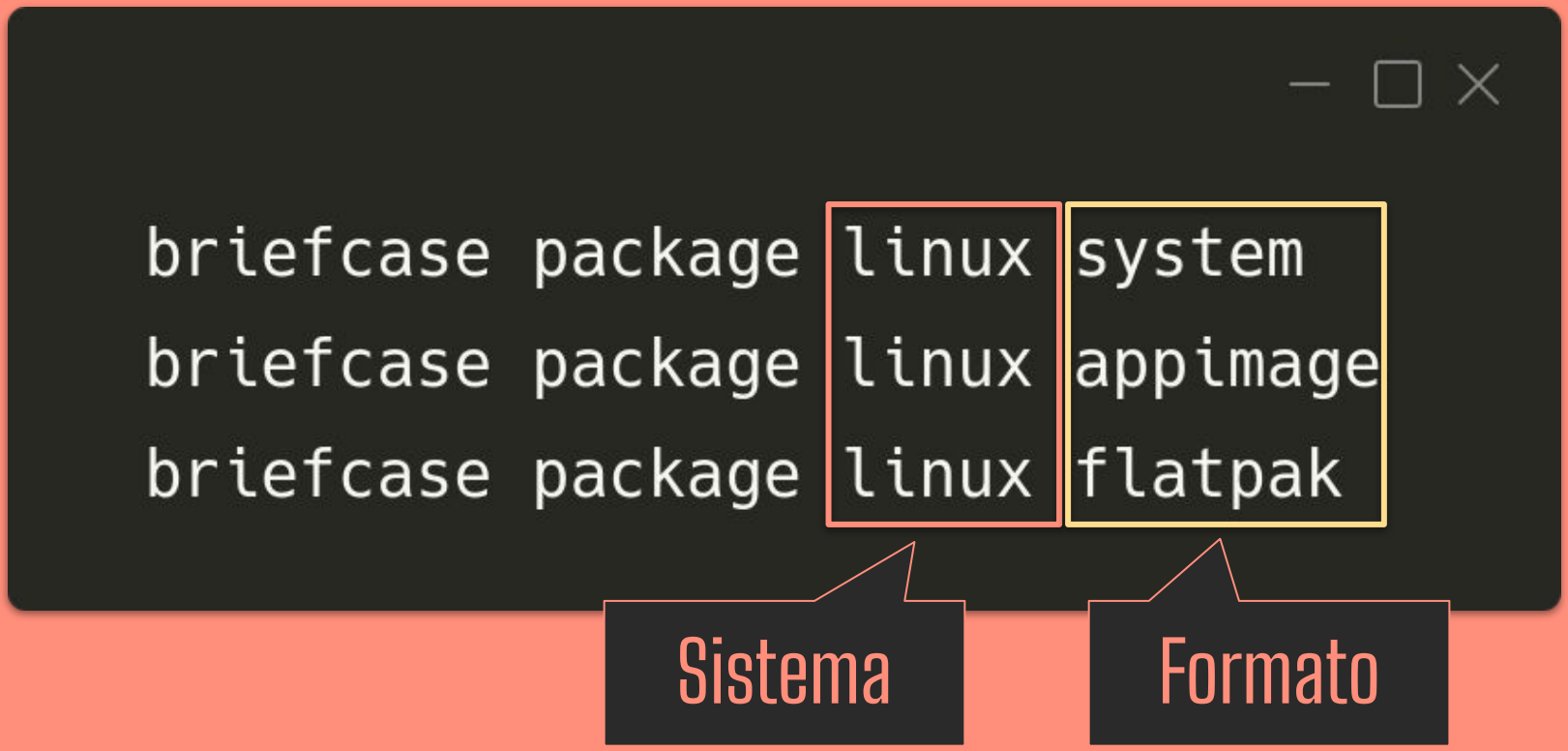
```
briefcase package linux --target ubuntu:latest
briefcase package linux --target fedora:40
briefcase package linux --target archlinux
briefcase package linux --target opensuse/tumbleweed
```

Formatos de build

O briefcase oferece formatos de build diferentes para cada tipo de projeto:

- Windows
  - app (default), VisualStudio
- Linux
  - system (default), appimage, flatpak
- Mac
  - app (default), Xcode
- Android
  - gradle
- iOS
  - Xcode

Exemplo usando meu SO



Integração contínua [github_actions.yaml]

Uma forma de conseguir fazer os builds do sistema que você não tem, é usar integração integração contínua (arquivo completo no repo).

```yaml
ci:
  name: Package
  runs-on: ${{ matrix.runs-on }}
  strategy:
    fail-fast: false
    matrix:
      target: ["AppImage", "Windows", "MacOs"]
      include:
        - target: "Windows"
          platform: "windows"
          output-format: "app"
          runs-on: "windows-latest"

        - target: "MacOs"
          platform: "macos"
          output-format: "app"
          runs-on: "macos-latest"

        - target: "AppImage"
          platform: "linux"
          output-format: "appimage"
          runs-on: "ubuntu-latest"
```

```yaml
- name: Install Briefcase
  run: |
    python -m pip install -U pip setuptools wheel
    python -m pip install briefcase

- name: Build App
  run: |
    briefcase build \
    ${{ matrix.platform || matrix.target }} \
    ${{ matrix.output-format }} \
    --test --no-input --log \
    ${{ matrix.briefcase-args }}

- name: Package App
  run: |
    briefcase package \
    ${{ matrix.platform || matrix.target }} \
    ${{ matrix.output-format }} \
    --update --adhoc-sign --no-input --log \
    ${{ matrix.briefcase-args }}
```

GUI

Usando
toga/pygame

Uma aplicação briefcase

Agora que colocamos uma aplicação simples e juá pronta pra rodar, vamos criar uma aplicação tendo o briefcase como Project Manager.

```
$ briefcase new

Let's build a new Briefcase app!
# [wizard...]
```

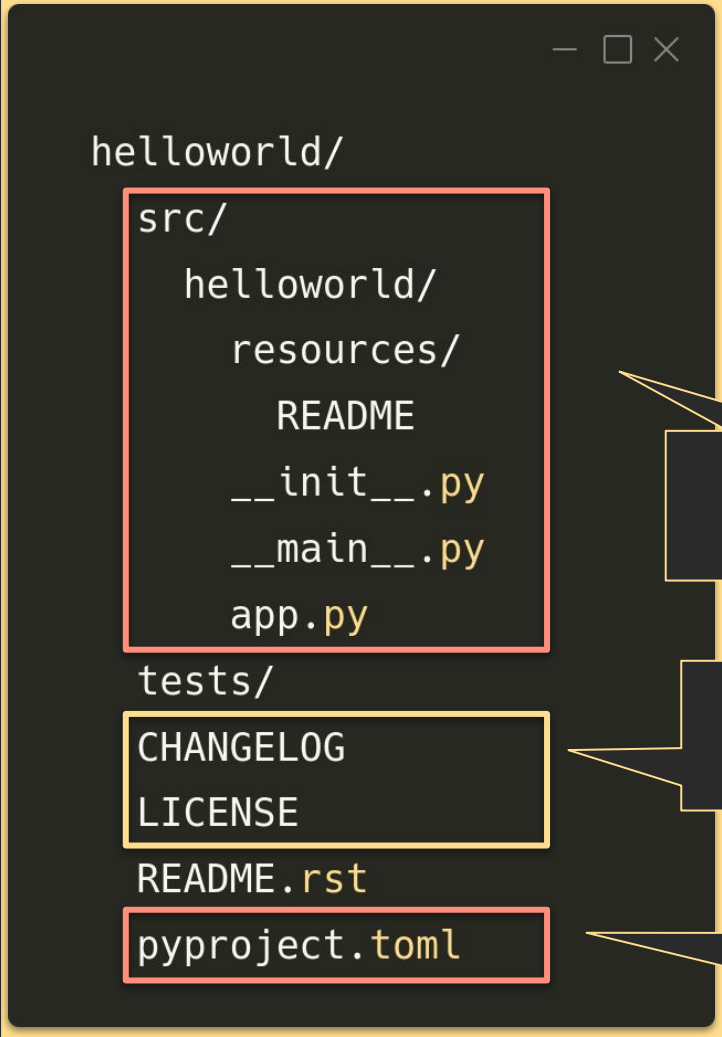
Vamos começar com o TOGA

Toga é uma biblioteca para interfaces gráficas **nativas**.

Mantida também pelo projeto beeware, com suporte para:

- Android
- iOS
- Mac
- linux (GTK)
- windows
- Web (via webassembly)
- ~ Terminal (Textual)

A estrutura do projeto



Quando criamos um projeto com o wizard, temos um resultado parecido com este.

projeto

Arquivos necessários

Configuração

Olhando o pyproject

O briefcase cria tabelas no pyproject diferentes para cada tipo de sistema. Algo como:

[tool.briefcase.app.helloworld.<sistema>]

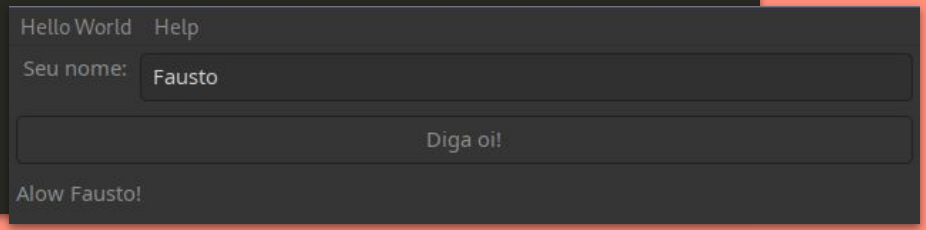
[tool.briefcase.app.helloworld.<sistema>.<formato>]

(vamos olhar o arquivo)

Código [app_toga.py]

```python
# widgets
name_label = toga.Label('Seu nome: ', style=Pack(padding=(0, 5)))
self.name_input = toga.TextInput(style=Pack(flex=1))
self.name_output = toga.Label('', style=Pack(padding=5, flex=1))
button = toga.Button(
    'Diga oi!', on_press=self.olar, style=Pack(padding=5)
)

# layout widgets
name_box = toga.Box(style=Pack(direction=ROW, padding=5))
name_box.add(name_label)
name_box.add(self.name_input)
main_box = toga.Box(style=Pack(direction=COLUMN))
main_box.add(name_box)
main_box.add(button)
main_box.add(self.name_output)
```



```
briefcase run web
briefcase run system
briefcase run android
```

```
briefcase package web
briefcase package system
briefcase package android
```

Pygame [app_pygame.py]

apoia.se/livedepython

pix.dunossauro@gmail.com

patreon.com/dunossauro

Ajude o projeto <3

Referências

1. https://briefcase.readthedocs.io/en/stable/index.html

2. https://cyclopts.readthedocs.io/en/latest/

3. https://docs.beeware.org/en/latest/

4. https://eu.louisvuitton.com/eng-e1/products/s-lock-briefcase-monogram-taurillon-leather-lg-g90-nvprod3750003v/M20835

5. https://www.geeksforgeeks.org/create-a-pong-game-in-python-pygame/

6. https://toga.readthedocs.io/en/stable/