

Building an executable with



PyInstaller

`#!/bin/env python`
`import as`
`import sys`

Binary code representation of the Python script:

```
011000110001001100011100  
011111110100100011110111010  
0111111100011110000111100011
```



Maureen M. Morton
June 9, 2020
Canton Python Meet-up



Brief bio



- Ph.D., Applied Mathematics, (cough) Michigan State University
- B.A., Russian Language & Literature / Mathematics, University of Kansas
- Several years experience teaching college mathematics
- A year as an aerospace reliability engineer (statistics, software development in Python, laser physics, materials/structural engineering) at N&R Engineering
- Transitioning to Professor of Applied Mathematics and Statistics at University of Mount Union
- Hobbies & Interests: Hiking, Writing, Drawing, Fuel Cell Technology, Paper Airplanes, Music, Non-intense board games, Uniting diverse communities



Files for tutorial are on GitHub <https://github.com/CantonPython/>

The screenshot shows a web browser window displaying the GitHub organization page for 'Canton Python'. The URL in the address bar is <https://github.com/CantonPython/>. The page header includes the GitHub logo, a search bar, and navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the header, there's a section for the 'Canton Python Meetup' with a location pin for 'Canton, Ohio' and a link to <https://www.meetup.com/Canton-Python/>. The main content area features tabs for 'Repositories' (14), 'Packages', 'People', and 'Projects'. A search bar allows users to 'Find a repository...', and dropdown menus let them filter by 'Type: All' and 'Language: All'. A green oval highlights the 'pyinstaller' repository, which is described as a 'PyInstaller tutorial'. It shows statistics: 1 star, 0 forks, 0 issues, 0 pull requests, and was updated 2 hours ago. To the right, a 'Top languages' section shows Python (blue dot) and JavaScript (yellow dot). Another section indicates that the organization has no public members.

Search or jump to... / Pull requests Issues Marketplace Explore

Canton Python

Canton Python Meetup

📍 Canton, Ohio <https://www.meetup.com/Canton-Python/>

Repositories 14 Packages People Projects

Find a repository... Type: All Language: All

pyinstaller
PyInstaller tutorial

Python 0 ⭐ 0 0 ! 0 0 Updated 2 hours ago

Top languages

Python JavaScript

boggle

People >

This organization has no public members.

What is



PyInstaller

#!/bin/env python
import as
import sys



?

<https://www.pyinstaller.org>

<https://pyinstaller.readthedocs.io/en/stable/>

- Bundles a Python application and dependencies into a single package
- Can run bundled app without installing Python interpreter or modules
- Supports Python 2.7 and 3.5+, and major packages: numpy, PyQt, Django, wxPython, etc.
- Windows, Mac OS X, GNU/Linux (but cannot cross-compile)

What is

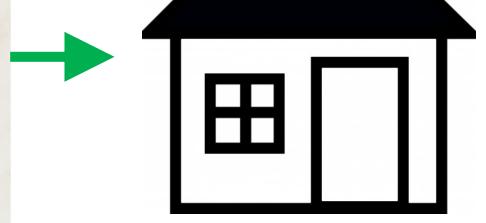


PyInstaller

?



data/image files
SUGAR



Install pie to kitchenless house,
no pantry, no cooking utensils



Your house, with kitchen, fully stocked
pantry, and all cooking utensils

Example program to bundle: CantonFavs

- App purpose: Find a recreational activity in Canton based on user preferences
- Used tkinter dependency to make GUI (Python 3.7)



[Click HERE to give your preferences and find an activity!](#)

CantonFavs

File

Find an activity to enjoy in Canton, Ohio!

What's your name?

What kind of activity do you want to do?

What price range do you want?

What season do you want?

Get Best Activity

Relevant details for bundling CantonFavs app

- Windows OS (I used Windows 10, 64-bit)
- Main script: `CantonFavs_GUI.py`
- Imports standard modules: tkinter, random (from numpy), pandas, sys, os
- Imports custom modules:
 - `data_and_images.py`: list data and image files as functions used by program for easy modification of file names
 - `cantonfavslib.py`: meat of the program
- Loads a data (text) file
- Loads image files
- Creates a (text) file

*Pyinstaller can handle
all these, BUT...*

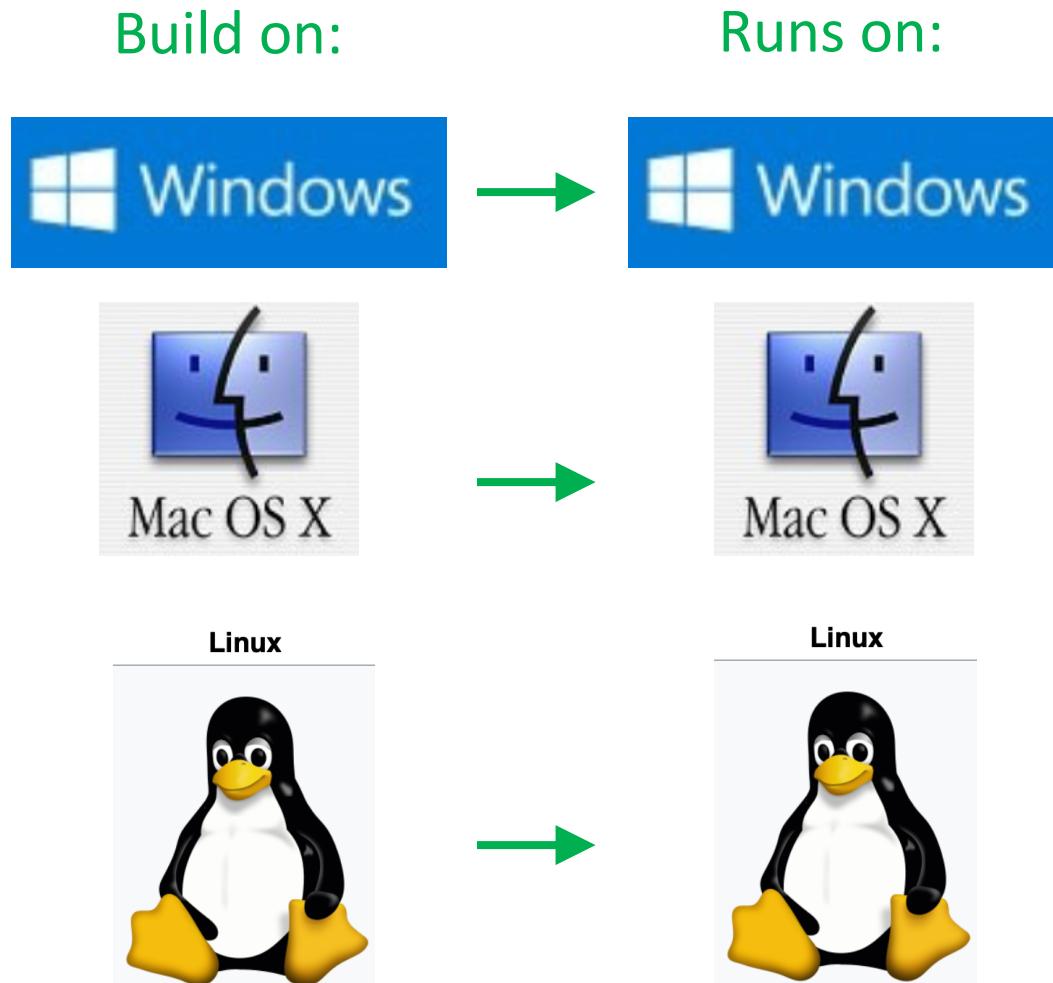
Relevant details for bundling CantonFavs app

- Windows OS (I used Windows 10, 64-bit)
- Main script: CantonFavs_GUI.py
- Imports standard modules: tkinter, random (from numpy), pandas, sys, os
- Imports custom modules:
 - data_and_images.py: list data and image files used by program for easy modification of file names
 - cantonfavslib.py: meat of the program
- Loads a data (text) file
- Loads image files
- Creates a (text) file

These (may) require
special attention

Operating system matters

- Build your executable on the same kind of system your customers will run it on
- No cross-compiling
- Version matters: e.g. I made exe on Windows 10, 64-bit
 - Works best on Windows 10, 64-bit
 - Made on Windows 7, didn't run on Windows 10
 - Made on Windows 10, did run on Windows 7, but not tested for full functionality
 - May need to watch 32-bit vs 64-bit too (not tested but according to online forums...)



Basic use of pyinstaller

- Basic command:

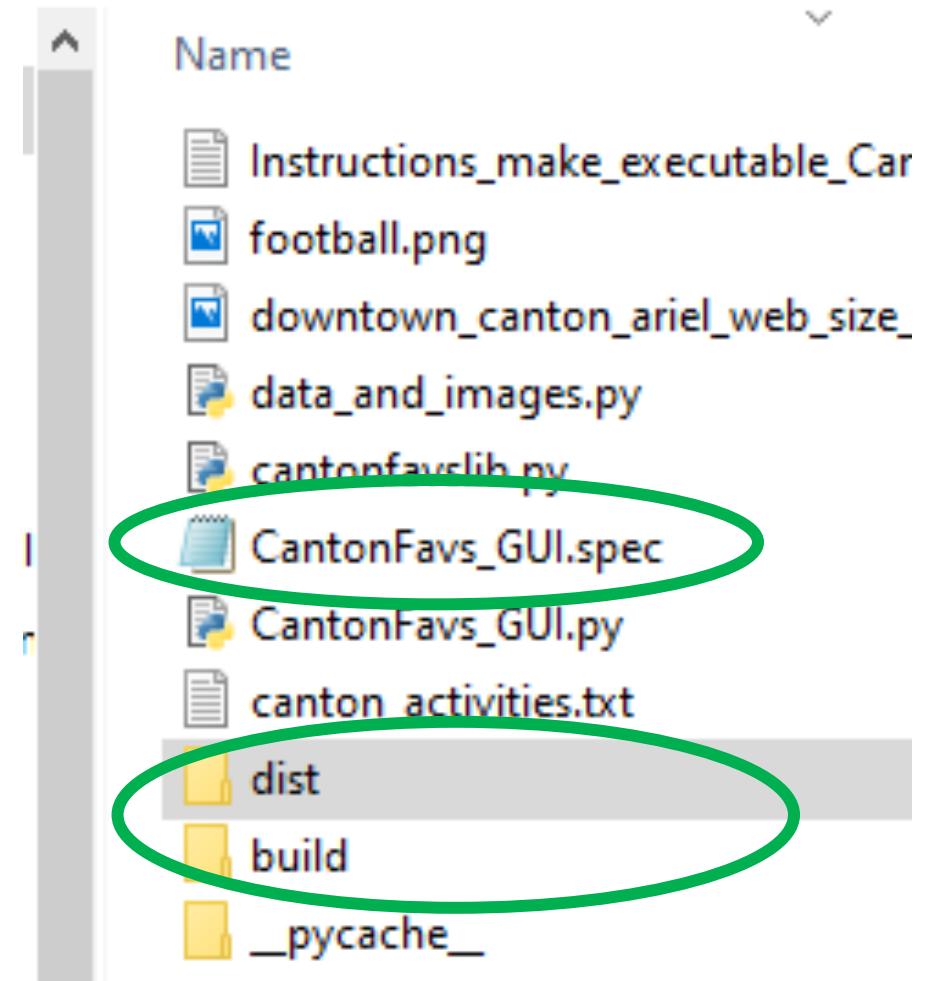
```
>> pyinstaller CantonFavs_GUI.py
```

- Basic output:

- **dist** folder: all contents must be distributed for executable to run
- **build** folder: includes useful info like warnings file
- **spec** file: can be used to adjust options for building, or repeat complicated build with simple command

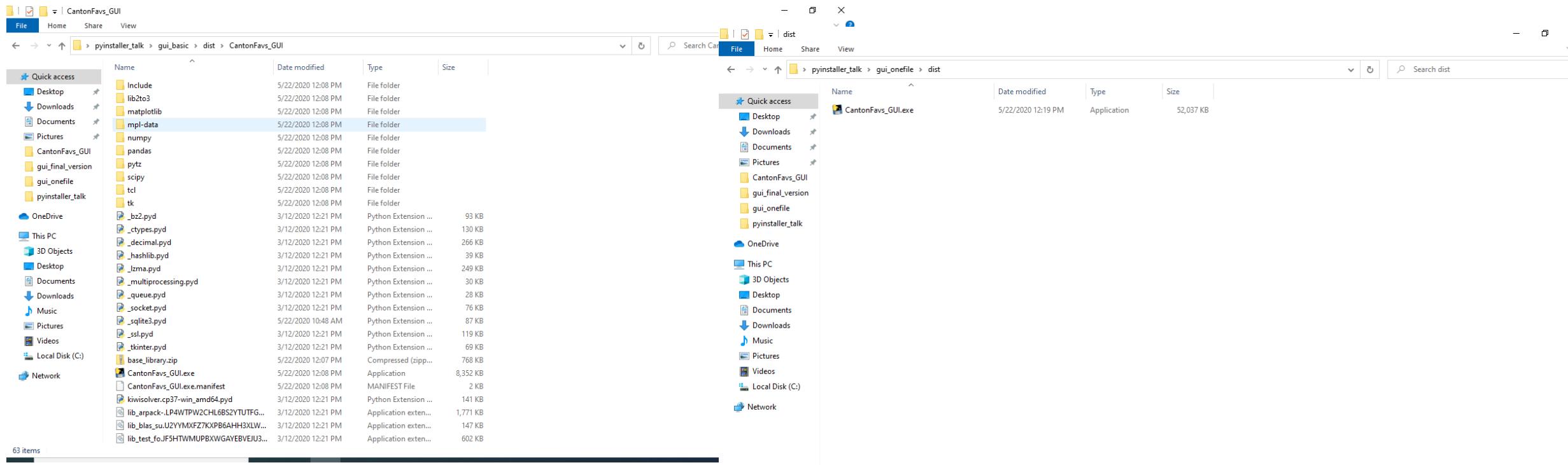
- But

- FAILS for our CantonFavs app
- gives a LOT of files in dist folder (bad for non-programmer user)



Simplifying distribution to one file

- Command: `>> pyinstaller --onefile CantonFavs_GUI.py`
- Nice for non-programmer; simplifies use on a different computer
- But it still FAILS to make our CantonFavs executable...



Possible issues: importing modules

- Look in build folder for: `warn-CantonFavs_GUI.txt`
- Lists missing modules that may or may not cause an issue...compare to what your app needs
- Possible fixes:
 - change your version(s) of the modules and/or pyinstaller
 - import specifically what you need in your python scripts e.g. “`import numpy.random.rand`” not just the basic “`import numpy`”
 - adjust `CantonFavs_GUI.spec` file: hiddenimports, hooks, etc. (...but we will look at spec file later)
- tkinter (hardest to fix), numpy, scipy have caused issues for me but ok for CantonFavs right now

/installer_talk > gui_onefile > build	
Name	
	<code>Analysis-00.toc</code>
	<code>base_library.zip</code>
	<code>CantonFavs_GUI.exe.manifest</code>
	<code>EXE-00.toc</code>
	<code>PKG-00.pkg</code>
	<code>PKG-00.toc</code>
	<code>PYZ-00.pyz</code>
	<code>PYZ-00.toc</code>
	<code>Tree-00.toc</code>
	<code>Tree-01.toc</code>
	<code>warn-CantonFavs_GUI.txt</code>
	<code>xref-CantonFavs_GUI.html</code>

Issue: Loading data and image files (same fix)

- Command with --add-data option:

```
>> pyinstaller --onefile --add-data downtown_canton_ariel_web_size_large.png;. --add-data football.png;. --add-data canton_activities.txt;. CantonFavs_GUI.py
```

- Create temporary folder for added files if you use --onefile:

use `sys._MEIPASS` (see `data_and_images.py` module)

- Makes it easier: make functions out of files added
- CantonFavs app should now work!

Different for Linux/Mac,
use colon instead, e.g.,
`canton_activities.txt:`

```
def resource_path(relative_path):
    """ Get absolute path to resource, works for dev and for PyInstaller """
    try:
        # PyInstaller creates a temp folder and stores path in _MEIPASS
        base_path = sys._MEIPASS
    except Exception:
        # For dev
        base_path = os.path.abspath(".")
    return os.path.join(base_path, relative_path)
```

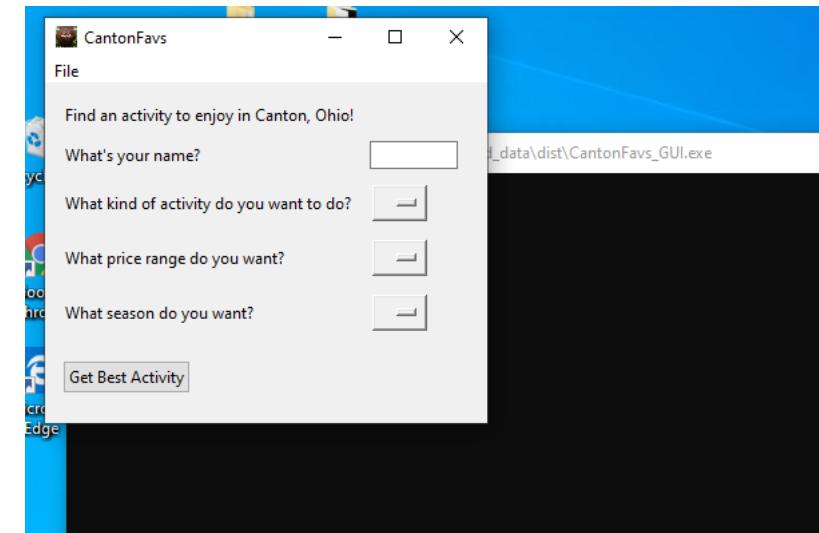
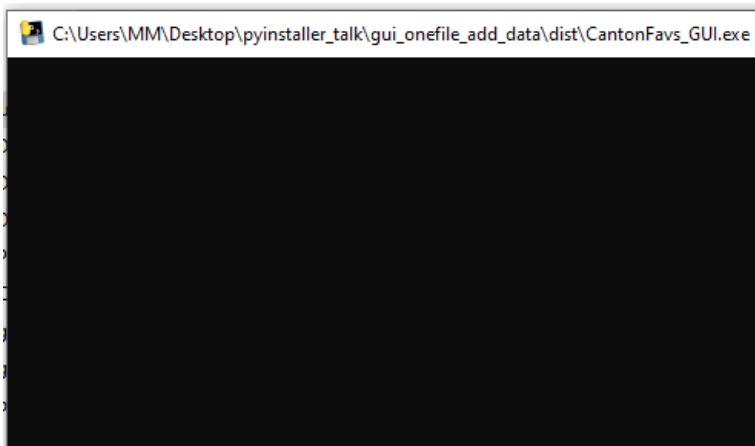
```
def UpperLeftPhoto():
    file_path = print_file('football.png')
    return file_path
```

What's that console window doing first?

- Use --windowed option to get rid of it:

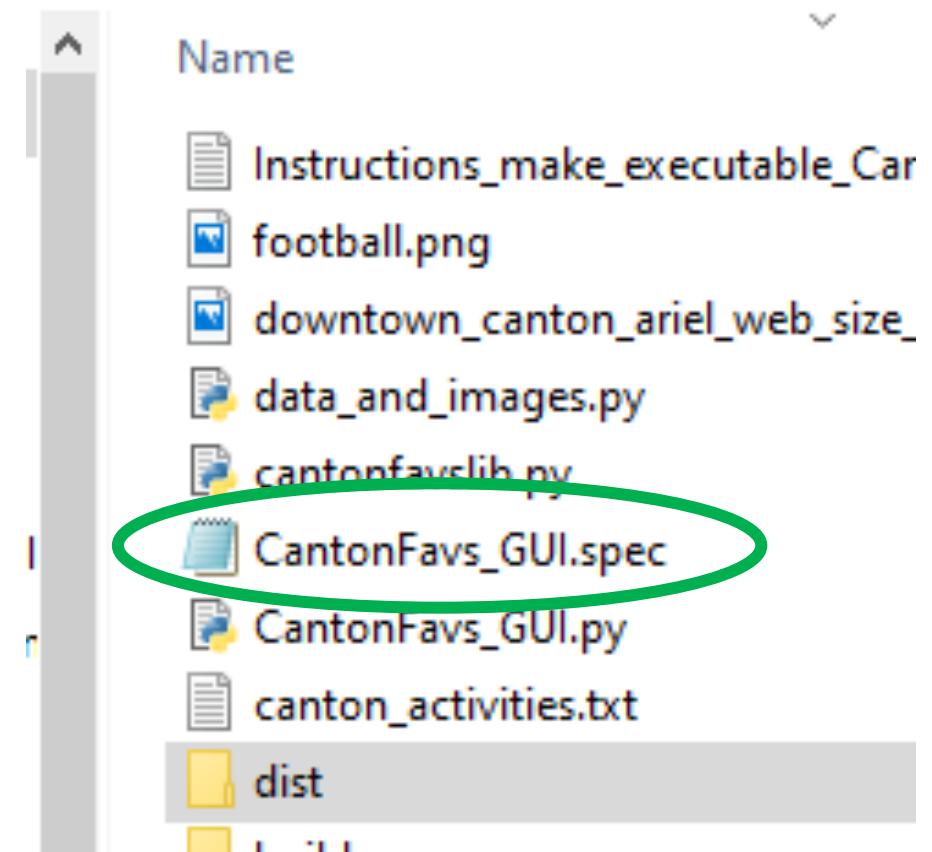
```
>>pyinstaller --windowed --onefile --add-data  
downtown_canton_ariel_web_size_large.png;. --add-data football.png;  
--add-data canton_activities.txt;. CantonFavs_GUI.py
```

- Pros: lets you know app is loading since splash screen has to wait to load too, sometimes writes warnings/errors if you can read quickly
- Cons: could confuse user, isn't "pretty"



Using the spec file: CantonFavs_GUI.spec

- Handy for complicated builds, especially repeating them!
- Nice, simple command: >> pyinstaller CantonFavs_GUI.spec
- For example:
 - Add/change data and image filenames
 - Keep/remove console
 - Change name of exe file
 - Fix imports



```
# -*- mode: python ; coding: utf-8 -*-
```

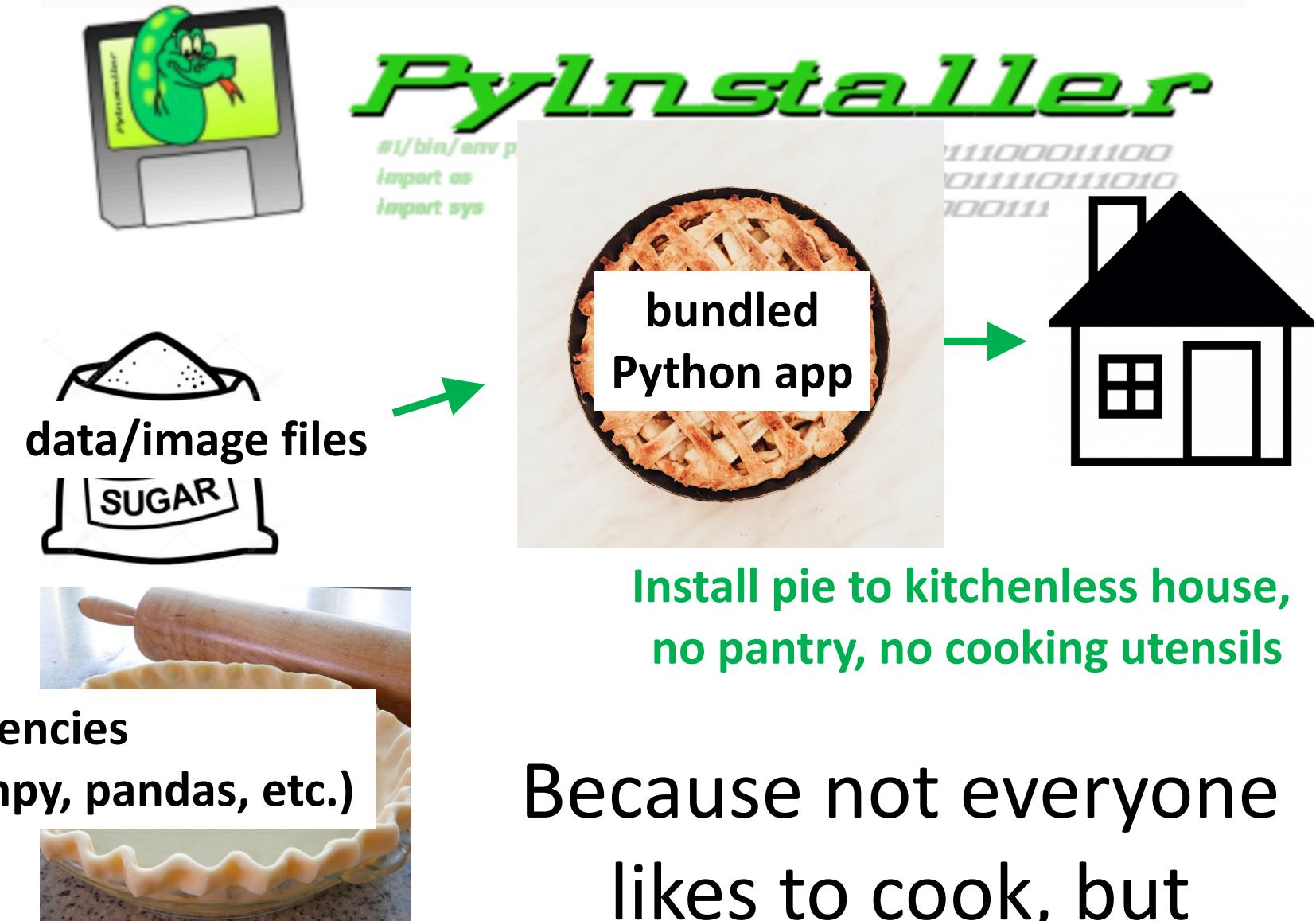
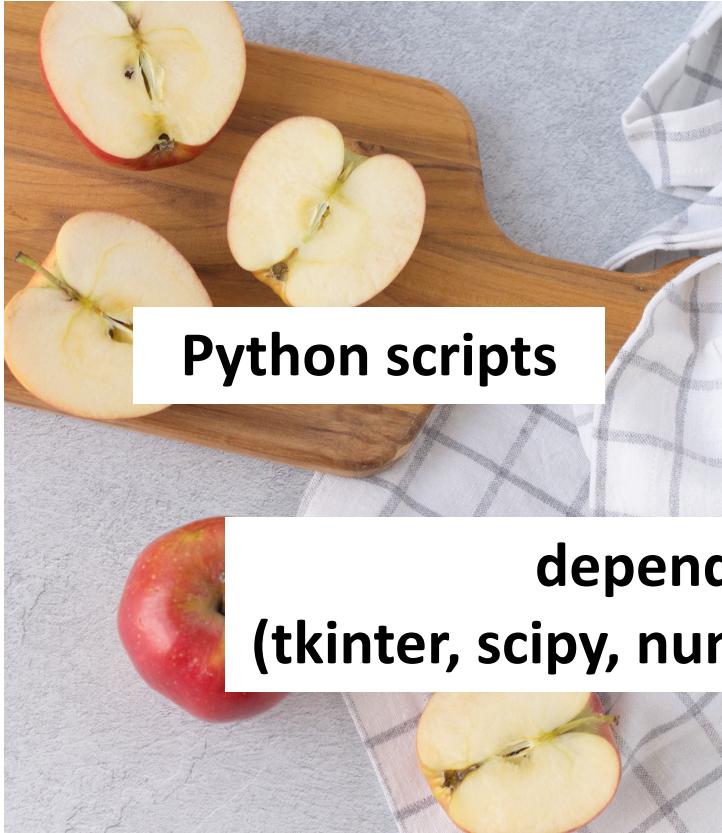
```
block_cipher = None
```

```
a = Analysis(['CantonFavs_GUI.py'],
             pathex=['C:\\\\Users\\\\MM\\\\Desktop\\\\pyinstaller_talk\\\\gui_version'],
             binaries=[],
             datas=[('football.png', '.'), ('downtown_canton_ariel_web_size__large.png', '.'), ('canton_activities.txt', '.')],
             hiddenimports=[],
             hookspath=[],
             runtime_hooks=[],
             excludes=[],
             win_no_prefer_redirects=False,
             win_private_assemblies=False,
             cipher=block_cipher,
             noarchive=False)
```

```
pyz = PYZ(a.pure, a.zipped_data,
           cipher=block_cipher)
```

```
exe = EXE(pyz,
          a.scripts,
          a.binaries,
          a.zipfiles,
          a.datas,
          [],
          name='CantonFavs_GUI',
          debug=False,
          bootloader_ignore_signals=False,
          strip=False,
          upx=True,
          upx_exclude=[],
          runtime_tmpdir=None,
          console=False )
```

In conclusion...



Your house, with kitchen, fully stocked pantry, and all cooking utensils



Have fun--try this on
your own now!



Questions? 😊