

Script, Library, or Executable?
You can have it

ALL!

Luke Lee @durdan20

Python GUI and CLI applications for Oil and Gas

CLI
Library
GUI
Executable

1. CLI

```
import sys
```

```
lines = 0
```

```
words = 0
```

```
with open(sys.argv[1], 'rb') as file_obj:
```

```
    for line in file_obj:
```

```
        lines += 1
```

```
        words += len(line.split())
```

```
print(f'{lines} {words}')
```

Separation of concerns

Separation of concerns

```
def cli():  
    """  
    Parse sys.argv and return dictionary of arguments/configuration  
    """  
    pass  
  
def get_file_info(file_):  
    """  
    Return lines/words in file  
    """  
    pass  
  
if __name__ == '__main__':  
    file_ = cli()  
    print(get_file_info(file_))
```

```
file_info = collections.namedtuple('file_info', 'lines words')

def get_file_info(file_):
    lines = 0
    words = 0

    with open(file_, 'rb') as file_obj:
        for line in file_obj:
            lines += 1
            words += len(line.split())

    return file_info(lines, words)
```

```

def cli():
    parser = argparse.ArgumentParser(prog='pywc')

    parser.add_argument('arg', action='store',
                        help='File or directory to count lines/words for')

    parser.add_argument('-w', dest='count_words', action='store_true',
                        default=False, help='Show number of words in file(s)')

    parser.add_argument('-l', dest='count_lines', action='store_true',
                        default=False, help='Show number of lines in file(s)')

    # Using vars() to turn namespace object returned by parse_args() into a
    # dict.
    args = vars(parser.parse_args())

    # Mimic wc command by printing both of these if there are no other
    # arguments
    if not args['count_words'] and not args['count_lines']:
        args['count_words'] = True
        args['count_lines'] = True

    return args

```


2. Library

- `setup.py`
- `pywc/`
 - `api.py`
 - `__init__.py`
 - `__main__.py`

setup.py

```
from setuptools import setup, find_packages
```

```
setup(  
    name='pywc',  
    version='1.0.0',  
    packages=find_packages(),  
)
```

__main__.py

```
from .api import get_file_info

def cli():
    """
    Parse sys.argv and return dictionary of arguments
    """
    pass

def main():
    args = cli()
    file_info = get_file_info(args['file'])

    if args['count_lines']:
        print(f'    {file_info.lines}', end='')

    if args['count_words']:
        print(f'    {file_info.words}', end='')

    print(f'    {args['file']}')

if __name__ == '__main__':
    main()
```

CLI?

entry_points

```
from setuptools import setup

setup(
    name='pywc',
    version='1.0.0',
    packages=find_packages(),
    entry_points={
        'console_scripts': [
            'pywc = pywc.__main__:cli'
        ]
    },
)
```

python -m

python -m

http.server

json.tool

pdb

timeit

cProfile

unittest

doctest

tarfile

zipapp

zipfile

webbrowser

base64

calendar

sysconfig

python -m pywc -h
pywc -h

Gooney

3. GUI

```
from gooey import Gooney
```

```
@Gooney
```

```
def cli():
```

```
    parser = argparse.ArgumentParser(
```

```
        prog='pywc',
```

```
        description='Count lines/words in files or directory')
```

```
    parser.add_argument(
```


```
        'arg', action='store',
```

```
        help='File or directory to count lines/words for')
```

```
    ...
```

pywc

Count lines/words in files or directory



Required Arguments

arg

File or directory to count lines/words for

Optional Arguments

count_words

☐ Show number of words in file(s)

count_lines

☐ Show number of lines in file(s)

Cancel

Start

GUI as option (--gui)

```
def cli(allow_gui_option=True):  
    """  
    Add --gui option to call gui()  
    """  
    pass  
  
def gui():  
    # Goocy doesn't like functools.partial  
    def cli_only():  
        return cli(allow_gui_option=False)  
  
    return Goocy(cli_only, program_name='pywc',  
                 show_success_modal=False)
```

```
from setuptools import setup

setup(
    entry_points={
        'console_scripts': [
            'pywc = pywc.__main__:cli'
        ],
        'gui_scripts': [
            'pywcg = pywc.__main__:gui'
        ]
    },
)
```

**python -m pywc --gui
pywcg**

PyInstaller



4. Executable

- `setup.py`
- `cli.py`
- `gui.py`
- `pywc/`
 - `api.py`
 - `__init__.py`
 - `__main__.py`

cli.py

```
from pywc.__main__ import cli
```

```
if __name__ == '__main__':  
    cli()
```

gui.py

```
from pywc.__main__ import gui
```

```
if __name__ == '__main__':  
    gui()
```

pip install pyinstaller

pyinstaller cli.py --name pywc

```
pyinstaller gui.py --name pywcg -w
```


! **Takeaways** !

Separation of concerns

entry_points

__main__.py

Gooney

PyInstaller

Ready. Set. Launch.



- ✓ `pywc`
- ✓ `pywc --gui`
- ✓ `pywcg`
- ✓ `python -m pywc`
- ✓ `python -m pywc --gui`
- ✓ `pywc.exe`
- ✓ `pywcg.exe`
- ✓ `python cli.py`
- ✓ `python cli.py --gui`
- ✓ `python gui.py`

and these ... 🤗

- ✓ `pywcg --gui`
- ✓ `python gui.py --gui`
- ✓ `pywcg.exe --gui`

Luke Lee

@durdenden20

http://bit.ly/pyconde_pywc_refs