# Console scripts

Or: How to execute your scripts in a terminal

# Setup tools

- “setup.py” defines requirements, packages, etc
- See previous talk

  https://github.com/HamPUG/meetings/tree/master/2018/2018-02-12/setuptools_pypi_mkdocs

- We are using the “console_scripts” entry point

  https://python-packaging.readthedocs.io/en/latest/command-line-scripts.html#the-console-scripts-entry-point

# Others do it, but how?

- Install Jupyter notebooks in a virtual env

```
pip install jupyter
```

- Content of <venv>/bin

```
activate           jupyter-bundlerextension  jupyter-trust
activate.csh       jupyter-console           pip
activate.fish      jupyter-kernel            pip3
activate_this.py   jupyter-kernelspec        pip3.7
easy_install       jupyter-migrate           pygmentize
easy_install-3.7   jupyter-nbconvert         python
iptest             jupyter-nbextension       python3
iptest3            jupyter-notebook          python3.7
```

# setup.py

- Add “entry_points” section:

```
entry_points={
    "console_scripts": [
        "msdp-hello=msdp.hello:sys_main",
    ]
}
```

- Format:

```
script_name=package.module:method
```

# Method

- Must return an int, to be used as exit code
- Calls the same method that “__main__” uses
- Example:

```
def sys_main():
    try:
        main()  # actual method doing the arg parsing/work
        return 0
    except Exception:
        return 1
```

# Full code

```
import argparse
import traceback

def main(args=None):
    parser = argparse.ArgumentParser()
    parser.add_argument("--text", help="the text to output", required=True)
    parsed = parser.parse_args(args=args)
    print(parsed.text)

def sys_main():
    try:
        main()
        return 0
    except Exception:
        print(traceback.format_exc())
        return 1

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

# ...and in action

- What happens when we install the project

# Thanks to...

Corey Sterling to figuring out console scripts.