NOTE:

The slides for this talk are meant to be presented in a terminal emulator, not from HTML/PDF

The sources for the talk are available in my "slides" repo currently hosted at Github:

http://git.io/9InuYA

Terminals, Command Lines, and Text User Interfaces

Petr Viktorin
encukou.cz

Hello! I'm **Petr Viktorin**

I work at Red Hat
on the FreeIPA project
(LDAP & Kerberos)

but this talk is not about that

I also help organize **Python meetups**

In **Brno** last Thursday of any month?
Join us at PyVo!
→ **python.cz**

but this talk is not about that either

The Console

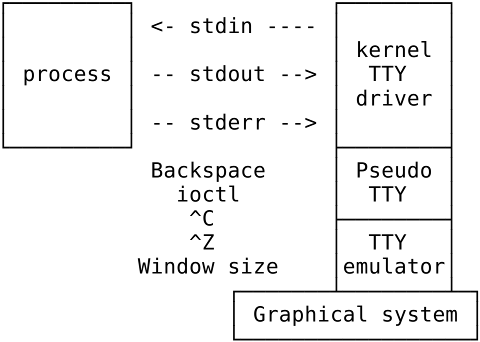
Terminal

TTY

UNIX and terminals

```
 ┌──────────┐                 ┌─────────┐
 │          │ <- stdin ----   │         │
 │          │                 │ kernel  │
 │ process  │ -- stdout -->   │  TTY    │
 │          │                 │ driver  │
 │          │ -- stderr -->   │         │
 └──────────┘                 └─────────┘
                                 ^   |
                                 |   |
                                 |   |
                                 |   |
                                 |   v
                              ┌──────────┐
                              │ terminal │
                              └──────────┘
```

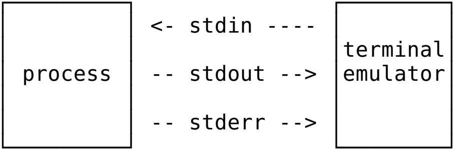
Terminal emulators



```
$ stty -a

speed 38400 baud; rows 45; columns 266;
line = 0; intr = ^C; quit = ^\; erase =
^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start =
^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min
= 1; time = 0; -parenb -parodd cs8 -hupcl
-cstopb cread -clocal -crtscts -ignbrk
-brkint -ignpar -parmrk -inpck -istrip
-inlcr -igncr icrnl ixon ixoff -iuclc
-ixany -imaxbel iutf8 opost -olcuc -ocrnl
onlcr -onocr -onlret -ofill -ofdel nl0
cr0 tab0 bs0 vt0 ff0 isig icanon iexten
echo echoe echok -echonl -noflsh -xcase
-tostop -echoprt echoctl echoke
```

Features



Character sets
Backspace
Underline
"Input mode"
Bold
^C
Mouse
Blinking
^Z
Unicode
Window size
Beep
More Colors!
Cursor movement

Ugh.

Character sets

Backspace

Underline

"Input mode"

Bold

^C

Mouse

Blinking

^Z

Unicode

Window size

Beep

More Colors!

Cursor movement

Why do I use this?

Why do I use this?

TUIs are **simple**

```
print('Hello world!')
raw_input('What is your name?')
```

Why do I use this?

TUIs are simple, **universal**

```
pip install pyshinylib
```

Why do I use this?

TUIs are simple, universal, **scriptable**

```
git diff | pep8 --diff
```

Why do I use this?

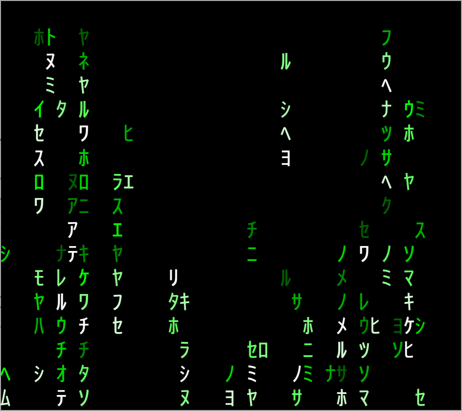
TUIs are simple, universal, scriptable, **lightweight**

This talk: ~1MB

Why do I use this?

TUIs are simple, universal, scriptable, lightweight, **cool**

At least to **you**



Types of TUIs

- Line-based
  cat, grep, git
  simple, universal
  scriptable, lightweight

- Full-screen
  less, vim, mc
  lightweight?, cool

How to build a text user interface

- Decide on the type
  Line-based? Full-screen TUI?
  Traditional GUI? Web UI?
- Parse command-line arguments

- Handle input

- Output the output

Command-line arguments

- Positional arguments
```
prog data.txt
```
- Options
```
prog -v
prog --verbose
prog --file data.txt
prog --file=data.txt
prog -f data.txt
```
- Subcommands
```
git log
git -c color=always log --oneline
```

```
import argparse
replaces: optparse
```

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument(
    '-v', '--verbose',
    action='store_true',
    help='Print more text')
parser.add_argument(
    'filename',
    help='File to process')

args = parser.parse_args(
    ['-v', 'data.txt'])
print([args.verbose, args.filename])
→ [True, 'data.txt']
```

Documentation

you can never write enough

- --help option
  cheatsheet
- man page
  tersely documents everything
- website, tutorial
  explains, teaches

```
$ program.py --help
usage: program.py [-h] [-v] filename

positional arguments:
filename        File to process

optional arguments:
-h, --help      show this help
                message and exit
-v, --verbose   Print more text

$ █
```

```
pip install docopt
```

```
from docopt import docopt

doc = """Read a file
Usage: script.py [options] <filename>

Options:
  --help, -h       Show help
  --verbose, -v    Print more text
"""

args = docopt(doc, ['-v', 'd.txt'])
print(args)
```

```
{'--help': False,
 '--verbose': True,
 '<filename>': 'd.txt'}
```

Top-level structure of a program

```python
import sys

def main(argv):
    args = parse_args(argv)
    return process(args)

if __name__ == '__main__':
    sys.exit(main(sys.argv))
```

pip install **termcolor**

Simple colorization

Outputs **ANSI** escape sequences

```python
from termcolor import colored

text = colored('Hello, World!', 'red')
print(text)

    Hello, World!

print(repr(text))

    '\x1b[31mHello, World!\x1b[0m'
```

import **colorama**

Colorization wrapper for Windows

```
import colorama
colorama.init()
```

import **blessings**

Output colorization, etc.

```python
from blessings import Terminal

term = Terminal()

print term.red('Hi there!')

    Hello, World!
```

Pipe friendliness

Default to color only for terminals

Provide a way to force color on or off

clint

colors, indents, columns,
progress bars, argument handling,
interactive prompting,
English-language join()

TODO: unit tests, docs, py3k

import **termios**

Posix calls for tty I/O control

---

import **tty**

setcbreak()
setraw()

```python
import sys, tty
from termios import tcgetattr
from termios import tcsetattr

old = None
try:
  if sys.stdin.isatty():
    old = tcgetattr(sys.stdin)
    tty.setcbreak(sys.stdin)

  print(sys.stdin.read(1))

finally:
  if old is not None:
    f = termios.TCSAFLUSH
    tcsetattr(sys.stdin, f, old)
```

import **curses**

“Full-screen” TUI library

Write to a buffer, let curses display it

Virtual “windows”

curses curses

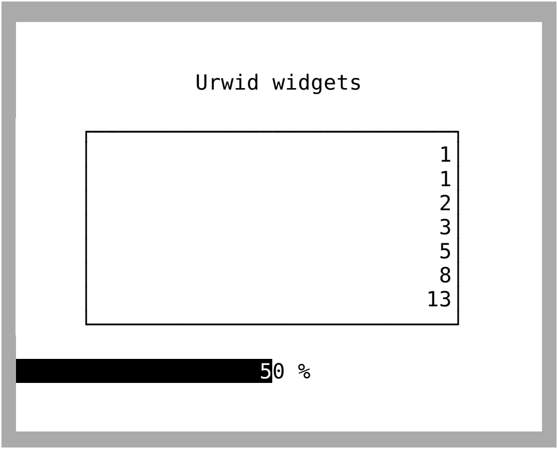
80s era C library, very thin Python wrapper

No docstrings (Use man pages!)

import **urwid**

Widget-based TUI library

Multiple backends (curses, terminal, web)



```
                Urwid widgets

     ┌──────────────────────────────┐
     │                             1│
     │                             1│
     │                             2│
     │                             3│
     │                             5│
     │                             8│
     │                            13│
     └──────────────────────────────┘

                   50 %
```

```
┌─────────────────────┐
│        abcd         │
│        efgh         │
└─────────────────────┘
```

```python
pile = urwid.Pile([
  urwid.Text('abcd', align='center'),
  urwid.Text('efgh', align='center'),
])

widget = urwid.LineBox(pile)
```

Mouse input

```
urwid.MainLoop(handle_mouse=False)
```

Keep it simple!

questions?