Python kaip pagalbinė priemonė

Dalius Dobravolskas

Tikslas

Mano tikslas parodyti:

1) kaip automatizuoti darbinius procesus

2) kad kartais Python'u atlikti užduotį paprasčiau

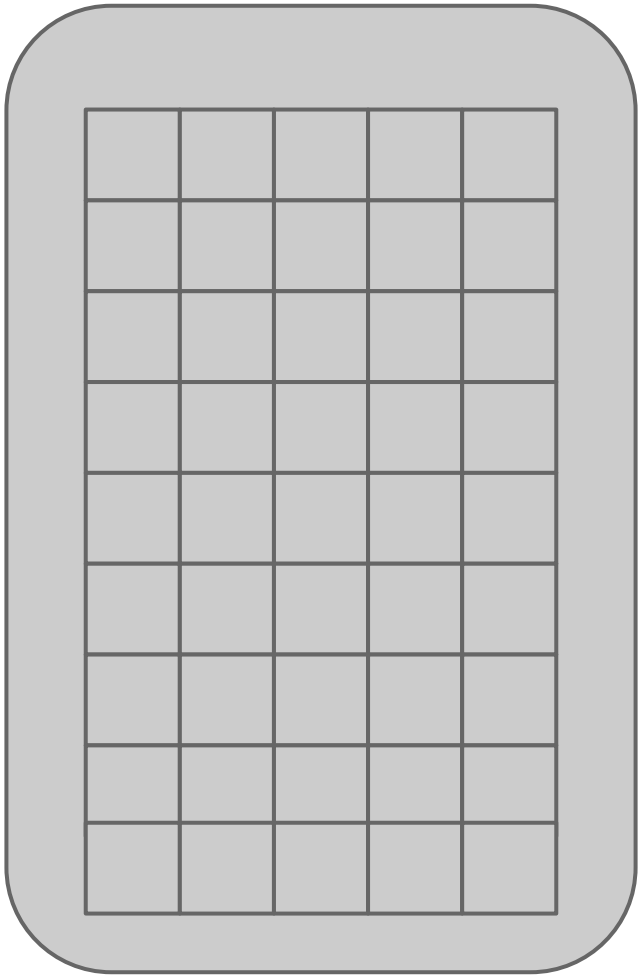
Ikonų problema

Daug skirtingų įrenginių

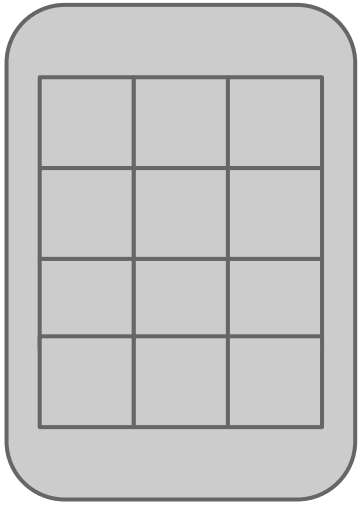
iOS sprendimas: normalios ir @2 ikonos

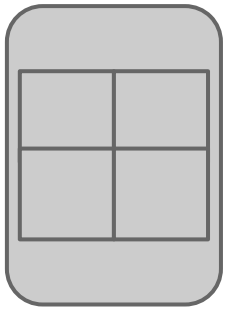
Android sprendimas: ldpi, mdpi, hdpi ir t.t. + small, normal, large, xlarge...

Ubuntu Touch sprendimas: tinklelis



Skirtingi dydžiai
Skirtingas DPI
Bent 10 skirtingų dydžių ikonų





subprocess

```
import subprocess

command = [
    'inkscape',
    '-f', ICON_NAME + '.svg',
    '-e', ICON_NAME + '64.png',
    '-w', '64']

subprocess.call(command)
```

elementtree

```
import xml.etree.ElementTree as ET

tree = ET.parse('help.svg')
root = tree.getroot()

for el in root.findall(".//*[@style]"):
    style = el.attrib["style"]
    style = style.replace("#b4b4b4",
                          "#b48080")
    el.attrib["style"] = style

tree.write('helpswitch.svg')
```

pycairo

```
import cairo

surface = cairo.ImageSurface
(cairo.FORMAT_ARGB32, width,
height)

...

surface.write_to_png (filename)
```

pycairo #2

```
ctx = cairo.Context (surface)
ctx.scale (width, height)
ctx.move_to (0.5, 0.4)

ctx.rel_curve_to ( 0.0, -0.1, 0.1,
                  -0.2, 0.2, -0.2)
ctx.rel_curve_to ( 0.1, 0.0, 0.2,
                   0.1, 0.2, 0.2)
ctx.rel_curve_to ( 0.0, 0.3, -0.4,
                   0.2, -0.4, 0.5)
ctx.fill()
```

subprocess #2

ImageMagick

subprocess.call(['convert', '-dispose', 'previous', '-delay', '5', '-loop', '0', 'heart*.png', filename], **shell=True**)

Problemos su ikonomis

Tarkime turite "Icon Publisher" produktą

Ikonų prisikaupė labai daug

Neaišku kurios dienos ikonos naudotos

Naudojamos pasenusios

Testuotojai naudoja vienas, programuotojai kitas

Sprendimas

Sukuriame paprastą puslapį, kur visi gali siųsti savo ikonas (SVG)

Parašome python'o skriptą/skriptus, kurie vykdomi tam tikru metu (Scheduled tasks/cron) ir pergeneruoja visas ikonas

Ikonų puslapis

```
<form enctype="multipart/form-data"
        method="post"
        action="/icon/upload.py">
    <input type="file" name="iconfile"/>
    <input type="submit" value="Upload"/>
</form>
```

Ikonų puslapis #2

```
import cgi
import os

form = cgi.FieldStorage()

if 'iconfile' in form:
    fileitem = form['iconfile']
    if fileitem.file:
        outpath = os.path.join(upload_dir,
            os.path.split( fileitem.filename)[1])

        fout = file (outpath, 'wb')
        while 1:
            chunk = fileitem.file.read(100000)
            if not chunk: break
            fout.write (chunk)
        fout.close()
```

Programinės įrangos išinstaliavimas Windows OS

```
with _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall") as key:
    try:
        index = 0
        while True:
            subkey_name = _winreg.EnumKey(key, index)
            with _winreg.OpenKey(key, subkey_name) as subkey:
                try:
                    name, _type = _winreg.QueryValueEx(subkey, "DisplayName")
                    if name == 'Icon Publisher':
                        command = ['start', '/wait', 'msiexec', '/x', subkey_name,
'/qn']
                        status = subprocess.call(command, shell=True)
                except WindowsError:
                    pass
            index += 1
    except WindowsError:
        pass
```

Problema Debesyse

Tarkime viršininkas nusprendė kraustytis į Debesis

Reikalauja naudoti node.js

Kaip atrodo node.js kodas

```
exports.users_delete = function(req, res) {
    ...
    db_client.eval(
        ...
        function(err, result) {
            ...
        });
}
```

Testuojame node.js

```
sukurti()
  tikriname ar sukūrė
  perskaityti sukurtą()
    tikriname ar gavoma gerą
    ištrinti sukurtą()
      tikriname trynimo rezultatus
      dar kartą perskaityti sukurtą()
        tikriname ar gavome, kad nebėra
```

Tada nebesistebi, kad node.js whitespace yra 2 tarpai

unittest

```
import unittest

class TestSomething(unittest.TestCase):

    def setUp(self):
        ...

    def tearDown(self):
        ...

    def test_something(self):
        ...
        self.assertEqual (..., ...)
        self.assertTrue (...)

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

virtualenv.py

http://www.virtualenv.org

virtualenv your_project
cd your_project

bin/pip install requests
bin/python

Scripts\pip.exe install requests
Scripts\python.exe

Python requests

python-requests.org

```
import requests
import json

r = requests.get('https://github.com/timeline.json')

js = json.loads(r.text)
len(js)
js[0].keys()
js[0]['url']
```

unittest #2

isrequest.py

```
def get(command, params, headers=None, username='test', password='test'):
    url = ROOT_URL + command
    resp = requests.get(url, auth = (username, password),
        headers = headers, params = params, verify = False)
    resp.json = json.loads(resp.content)
    return resp
```

test_something.py

```
import isrequest

def test_something(self):
    resp = isrequest.get('/icons', {'limit': 20})
    self.assertEqual (resp.status_code, 200)
    self.assertEqual (len(resp.json), 20)
```

unittest #3

python -m unittest discover

python -m unittest test_something

python -m unittest test_something.
TestSomething.test_something

Toliau?

Tiesiog pagalvokite:

a) kur tą patį darbą/veiksmą darote kelis kartus;

b) kur priemonės ne pačios patogiausios;

c) ir šiaip galvoti yra gerai.

Klausimai