

[<-- vissza](#)

Tartalomjegyzék

- 1 Körültekintés a nagyvilágban python szemmel
 - ◆ 1.1 Miért is jó a python
 - ◇ 1.1.1 Lassabb
 - ◇ 1.1.2 Hozzáférés
 - ◇ 1.1.3 JIT
 - ◆ 1.2 Példák
 - ◇ 1.2.1 Projekt kezel? rendszer: Trac
 - ◇ 1.2.2 Build rendszerek
 - ◇ 1.2.3 GUI megvalósítások
 - ◇ 1.2.4 Játékok
 - ◇ 1.2.5 3D
 - ◇ 1.2.6 Web framework
 - ◇ 1.2.7 Web alkalmazás szerver
 - ◇ 1.2.8 Web CMS
 - ◇ 1.2.9 Egyéb példák

Körültekintés a nagyvilágban python szemmel

Miért is jó a python

A python alapvet? célja a könny? és széleskör? felhasználhatósága. Megalkotása során nem volt cél az optimalizált program futás vagy a kereskedelmi szoftver kívánalmainak teljesítése.

Így aztán önmagában **nem alkalmas bonyolult számításokra vagy ellen?rzött csatornákon elérhet? alkalmazások kialakítására.**

Ha ezek nem, akkor mire igen: **rengeteg felhasználása van**

Lassabb

Habár lassabban fut a Java vagy C nyelvénél mégis els? esetben 3-5 míg második esetben akár **8-10 szer kevesebb kóddal** valósítható meg ugyanaz a program. Ebb?l következtetve inkább "<http://wiki.math.bme.huragasztó>"<http://wiki.math.bme.hu> nyelvnek mondják a python-t. Azaz hogy az id?igényes feladatokat ellátó részeket C ill C++ nyelven míg a közöttük kialakított infrastruktúrát python nyelven valósítják meg.

Ebb?l adódóan:

- automatizált test, build rendszerek írására tökéletes
- saját cégen belüli eszközök
- egyéni segéd programok
- stb.

Hozzáférés

Python nyelvben nincs adat védelem. Azaz ha látunk egy komponenst, akkor minden elemét látjuk és módosíthatjuk is. :) Szoktál mondani: "http://wiki.math.bme.hu python feln?tt embereknek találták ki" http://wiki.math.bme.hu Ez ugye biztonságkritikus rendszereknél nem jöhet szóba, persze vannak technikák amikkel megfelel? interfészek mögé tudjuk rejteni alkalmazásunkat. Ilyen technika pl. az internet.

Így aztán internetes alkalmazások írására tökéletes. Erre több nagyszer? példát is találunk:

- [Django](#)
- [Plone](#)
- [Zope](#)
- stb.

JIT

Továbbá JIT nyelv azaz "http://wiki.math.bme.hu Just In Time" http://wiki.math.bme.hu tehát tökéletes még ismeretlen elemeken m?köd? programok írására. Mint ahogy a JavaScript a webes világ DOM-on való munkára a python a [Maya](#) nev? 3D kép és animáció készít? script nyelve.

egy kis önfényezés

Példák

Projekt kezel? rendszer: [Trac](#)

A webalkalmazások egyik ékes példája. Célja hogy a szoftver fejlesztést megkönnyítse azzal hogy egy feladat kezel? rendszert adjon a fejleszt?k alá, kiegészítve mindenféle csatolható alkalmazással. (Java-s nagy testvére a [Jira](#))

Build rendszerek

- [waf](#)
- [scons](#)

A épít? rendszerek alapvet?en azt a célt próbálják megvalósítani, hogy a fejleszt? vállárol levegyék a terhet az automatizálható build (configuráció, fordítás), és deploy (telepítés) feladatokat elvégzésével.

GUI megvalósítások

- [wxPython](#) - GUI toolkit
 - ◆ open-source
 - ◆ "http://wiki.math.bme.hu platform független" http://wiki.math.bme.hu (32-bit Microsoft Windows, legtöbb Unix vagy unix-szer? rendszer, és Macintosh OS X.
- [Tkinter](#) - python kapcsoló a Tk GUI-hoz

Játékok

- [pygame](#)
- [Kivy](#)

Mindkett? animációk-, rajzok-, képek-, ablakozó rendszerek-, események kezelését valósítják meg. A python kód hordozhatóságára építve a platformfüggetlenség az egyik f? céljuk. Utóbbi már a hordozható

Érintőképernyővel rendelkező eszközöket is képes kezelni, mint pl. iPad, Android tablet stb.

Egyszerű ablak megjelenítése egy gombbal a közepén.

```
from kivy.app import App
from kivy.uix.button import Button

class TestApp(App):
    def build(self):
        return Button(text='Hello World')

TestApp().run()
```

3D

OpenGL támogatás a PyOpenGL keresztül.

- Így elérhető az OpenGL által nyújtotta lehetőségek python alatt is.
- Kompatibilis más GUI rendszerekkel mint a wxPython, pyGTK stb.

Web framework

- Django

Nagyon egyszerűen és tisztán használható webalkalmazás keretrendszer, sok-sok hasznos kiegészítővel.

- webapp és Appengine

Egyszerű webalkalmazás:

```
from google.appengine.ext import webapp
from google.appengine.ext.webapp.util import run_wsgi_app

class MainPage(webapp.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write('Hello, webapp World!')

application = webapp.WSGIApplication(
    [('/', MainPage)],
    debug=True)

def main():
    run_wsgi_app(application)

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

Server konfiguráció:

```
application: helloworld
version: 1
runtime: python
api_version: 1

handlers:
- url: /*
  script: helloworld.py
```

Játékok

Web alkalmazás szerver

Zope Egy komplex de jól használható alkalmazás szerver. Webes kezelőfelülettel és sok hasznos eszközzel.

Mit is takar egy alkalmazáserver kifejezés:

- olyan szerver ami gyűjtésként foglalja össze a webalkalmazásunkhoz felhasznált szolgáltatásokat
 - ◆ webalkalmazások
 - ◆ adatbázis szerver
 - ◆ stb.

Kialakítása olyan hogy segítse a részegységek közötti kommunikációt.

Web CMS

Plone A Plone a Zope-ra épülő tartalom kezelő rendszer (CMS). A CMS egy wiki-nek kell elképzelni, csak a Plone kicsit komplexebb, saját weblap:

- saját kinézettel
- saját funkciókkal
- saját adatmodellel
- stb.

Egyéb példák

Python success