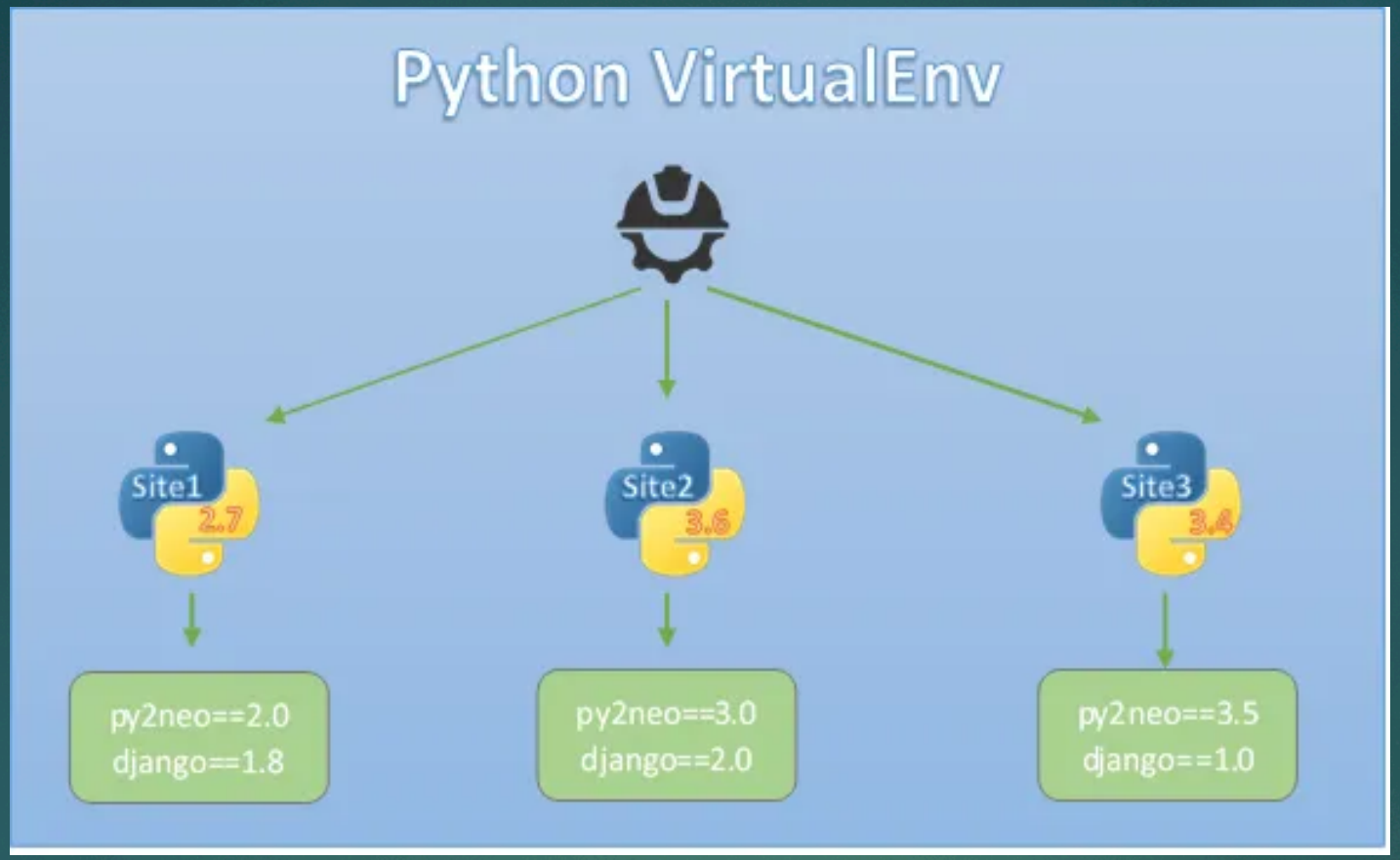
Virtualenv

CREATING ISOLATED ‘VIRTUAL’ PYTHON ENVIRONMENTS

Pourquoi une venv?

- **Problème :**
  - Version python
  - Conflit librairies
  - Versions modules



- Versions v1.8 et v2.0 se trouve dans le même répertoire avec le même nom
- Projets sont stockés en fonction de leur nom : pas de différence entre versions
- Solution : environnements virtuels

- Solutions :
  - Virtualbox, docker, sandbox
- Solution principale :
  - Création des environnements virtuel pour python

Vocabulaire

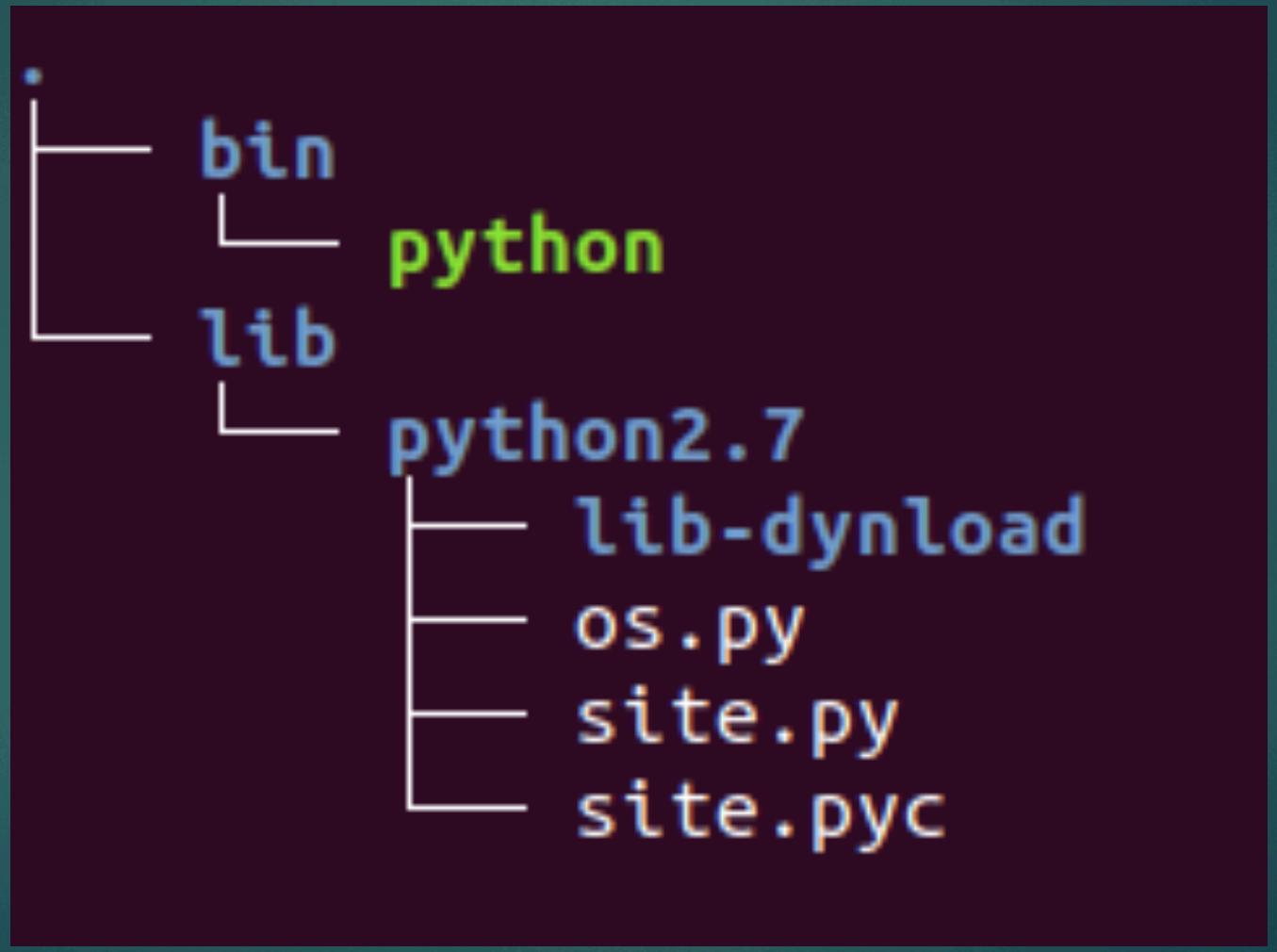
- Sys.prefix: Location du folder principale de Python; PYTHONHOME
- PYTHONPATH: Tout les dossiers directement accessibles dans python
- PATH: Tout les dossiers directement accessibles; Ordre important

Demo

PYTHON2.7 VS PYTHON3.7

Créer sa propre venv

Structure minimale



Comment ça marche?

- Emplacement du binaire important
  - Doit etre dans prefix/bin
  - Landmark doit exister: prefix/lib/pythonversion/os.py
- Environment validé? PYTHONPATH en découle

Example PYTHONPATH

```
>>> for elem in sys.path:
...     print(elem)
...

/home/ben/Desktop/Virt/lib/python2.7
/home/ben/Desktop/Virt/lib/python2.7/plat-x86_64-linux-gnu
/home/ben/Desktop/Virt/lib/python2.7/lib-tk
/home/ben/Desktop/Virt/lib/python2.7/lib-old
/usr/lib/python2.7/lib-dynload
```

Et maintenant

- Pas de librairie standard
  - Python non fonctional
  - Pas de site.py => pas de site-packages
- Copie de tout les fichiers importants (Mais pas toute la stdlib)
- Site.py différent!
- Orig-prefix.txt

```
def virtual_install_main_packages():
    f = open(os.path.join(os.path.dirname(__file__), "orig-prefix.txt"))
    sys.real_prefix = f.read().strip()
    f.close()
    pos = 2
    hardcoded_relative_dirs = []
    if sys.path[0] == "":
        pos += 1
    if _is_pypy:
        if sys.version_info > (3, 2):
            cpyver = "%d" % sys.version_info[0]
        elif sys.pypy_version_info >= (1, 5):
            cpyver = "%d.%d" % sys.version_info[:2]
        else:
            cpyver = "%d.%d.%d" % sys.version_info[:3]
        paths = [os.path.join(sys.real_prefix, "lib_pypy"), os.path.join(sys.real_prefix, "lib-python", cpyver)]
        if sys.pypy_version_info < (1, 9):
            paths.insert(1, os.path.join(sys.real_prefix, "lib-python", "modified-%s" % cpyver))
        hardcoded_relative_dirs = paths[:]  # for the special 'darwin' case below
        #
        # This is hardcoded in the Python executable, but relative to sys.prefix:
        for path in paths[:]:
            plat_path = os.path.join(path, "plat-%s" % sys.platform)
            if os.path.exists(plat_path):
                paths.append(plat_path)
    elif sys.platform == "win32":
        paths = [os.path.join(sys.real_prefix, "Lib"), os.path.join(sys.real_prefix, "DLLs")]
```

Librarie Standard dans PYTHONPATH

```
PREFIX:
C:\Users\Jeremy\Desktop\Demo\Demo1

PATH:
C:\Users\Jeremy\Desktop\Demo
C:\Users\Jeremy\Desktop\Demo\Demo1\Scripts\python37.zip
C:\Users\Jeremy\Desktop\Demo\Demo1\DLLs
C:\Users\Jeremy\Desktop\Demo\Demo1\lib
C:\Users\Jeremy\Desktop\Demo\Demo1\Scripts
c:\users\jeremy\appdata\local\programs\python\python37-32\Lib
c:\users\jeremy\appdata\local\programs\python\python37-32\DLLs
C:\Users\Jeremy\Desktop\Demo\Demo1
C:\Users\Jeremy\Desktop\Demo\Demo1\lib\site-packages

REAL PREFIX:
c:\users\jeremy\appdata\local\programs\python\python37-32
```

Fichier 'activate'

- Seulement un luxe
- Utilisation du bon binaire important
- Modifie le PROMPT
- Rajoute path_to_venv/Scripts dans le PATH
  - Ordre important, premier python.exe utilisé

Isolation

- Moins isolé qu'ons pourrait penser
- A toujours accès à real_prefix/Scripts
- Modules non-installé dans site-packages
  - Virtualenv
  - Bottle
  - Flask
  - Pip
  - Dans PATH mais pas PYTHONPATH

Conclusion

- Facile d'utilisation
- Répond à une demande réelle
- Bon rélfexe "Pro"
- Canevas blanc