



Python

Pour quoi, pour qui et comment ?

Agenda

1. Anatomie des Applications Python non-web.
2. Les cibles typiques.
3. Le Cauchemar du portage.
4. Simplifier l'accès aux applications via WebAssembly.
5. Mise en pratique (hors slide)
6. Conclusion.



Pour quoi ?

Pour des applications classiques .

Anatomie des Applications python typiques

- Terminal
- Traitement de données distantes
- 2D
- 3D
- Calcul déporté sur processeurs “graphiques”
- Capteurs
- Accessibilité





Pour qui ?

**Une audience variée dans un
environnement très hétéroclite**

Des utilisateurs variés sur des machines qui le sont tout autant

- Windows : pas de vt100/420 ni de serveur X, pas toujours unicode
 - Système de types Unix : pas de vt420, pas toujours unicode
 - Mac : pas toujours de serveurs X
-
- Couches 3D hétéroclites : OpenGL / GLES / Vulkan / DirectX / Métal 1&2
 - Système kiosques : très hétérogènes et souvent très restreints
-
- Ultraportables/WebOS : difficultés pour lancer Python, se le procurer ou l'adapter aux besoins

- dans tous les cas des versions hétéroclites de Python et un support tierce partie souvent incomplet
- peut être une impossibilité d'installer ou d'exécuter une application avec les simples droits d'utilisateur
- possibilité d'un manque de confiance et/ou de formation de l'utilisateur final.
- et souvent des couches d'accessibilité négligées ou non standardisées.

Des concepteurs d'applications qui doivent satisfaire ces utilisateurs

- qui n'ont pas toujours accès aux plateformes à tester.
- qui n'ont pas toujours les moyens de se fournir la documentation ou les licences pour une plateforme.
- qui n'ont pas toujours la main d'œuvre requise pour un support cohérent de cette diversité.
- qui sont de moins en moins enclins à fournir le support pour les changements incessants imposés par l'industrie - avec toujours plus d'exigences.



Le Cauchemar du portage:

- Problème du portage natif : la méthode difficile est trop difficile ...
- Quelles sont les solutions ?



Problème du portage natif : la méthode difficile est trop difficile ...

- trouver et redistribuer des terminaux vt420 pour toutes les plateformes
- utiliser uniquement des dénominateurs graphiques communs ?
- un module universel pour les capteurs ?
- maintenir des binaires pour toutes les combinaisons hardware+software/os

Download and install instructions from 



itch.io

1. Extract the zip folder.

2. Run the exe.

Note: Your PC *may* warn you about a trojan virus. That is because this game is a packaged python project.



plyer

Platform	Android	iOS	Windows	OS X	Linux
Accelerometer	✓	✓		✓	✓
Audio recording	✓		✓	✓	
Barometer	✓	✓			
Battery	✓	✓	✓	✓	✓
Bluetooth	✓			✓	
Brightness	✓	✓			✓
Call	✓	✓			
Camera (taking picture)	✓	✓			
Compass	✓	✓			
CPU count			✓	✓	✓
Devicename	✓		✓	✓	✓
Email (open mail client)	✓	✓	✓	✓	✓
Flash	✓	✓			
GPS	✓	✓			
Gravity	✓	✓			
Gyroscope	✓	✓			
Humidity	✓				
IR Blaster	✓				
Keystore	✓	✓	✓	✓	✓
Light	✓				
Native file chooser	✓	✓	✓	✓	✓
Notifications	✓		✓	✓	✓
Orientation	✓				✓
Proximity	✓				
Screenshot			✓	✓	✓
SMS (send messages)	✓	✓			
Spatial Orientation	✓	✓			
Speech to text	✓				
Storage Path	✓	✓	✓	✓	✓
Temperature	✓				
Text to speech	✓	✓	✓	✓	✓
Unique ID	✓	✓	✓	✓	✓
Vibrator	✓	✓			
Wifi			✓	✓	✓

- Le problème ne date pas d'hier !

On peut trouver sur Wikipedia les trace de UNCOL (développé par SHARE)

Une sorte de “Langage universel orienté machine” qui est une représentation intermédiaire pour les compilateurs. C’est une idée introduite en 1958 mais plus un concept qu’une implémentation. L’idée était de rendre les compilateurs économiquement accessibles pour tout nouveau type d’architecture ainsi que tout nouveau langage avec un seul “backend” compilation par architecture et un seul frontend par langage transformant le problème NxM en N+M

... et puis, ZORK, une des plus anciennes fictions interactives, créée par la société INFOCOM.

ZORK et les titres suivants tournent sur la Z-Machine, une VM créée en 1979 pour réduire les coûts de portage vers les multiples machines d’un marché très fragmenté (Apple II, TRS-80, ZX, PC, Atari ST, Amiga...)



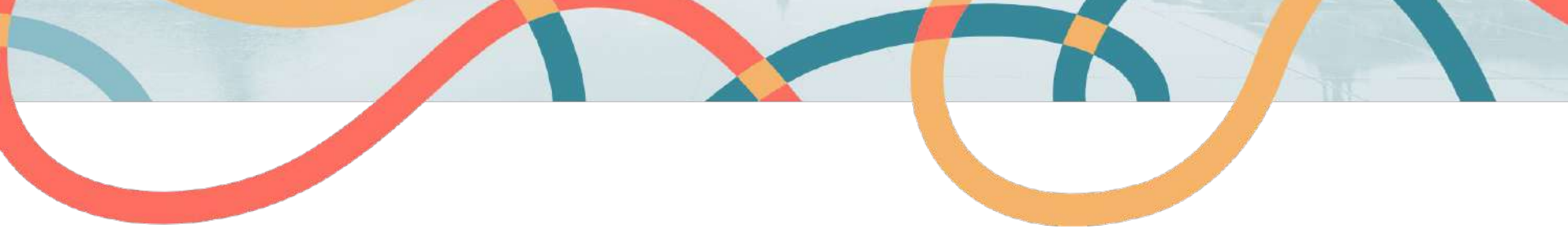
Le Cauchemar du portage: solutions ?

- S'écarter des solutions natives semblerait être la solution :
 - mais il faut un bon compromis encombrement/performance/sécurité.
 - on ne peut avoir les 3 en même temps donc quel argument de “vente” choisir ?
-
- Si l'idée ne date pas d'hier pourquoi n'a-t-on pas de solutions existantes ?



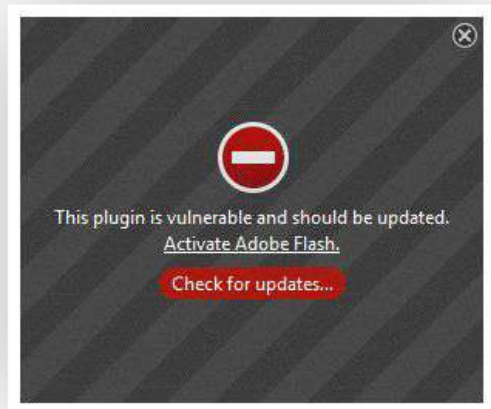
Pourtant au fil du temps, il y a eu de nombreuses machines virtuelles offrant un niveau d'abstraction pour des langages de haut niveau:

- perl : 1987 : - bon début - mais problème vers la fin.
- python : 1991 : pas de sécurité pour maximiser les performances
- ruby : assez similaire
- java : 1995 trop d'implémentations concurrentes ?
- llvm-ir : 2000: n'est pas utilisable par les humains, seulement les compilateurs
- parrotvm : toujours pas les compilateurs attendus.

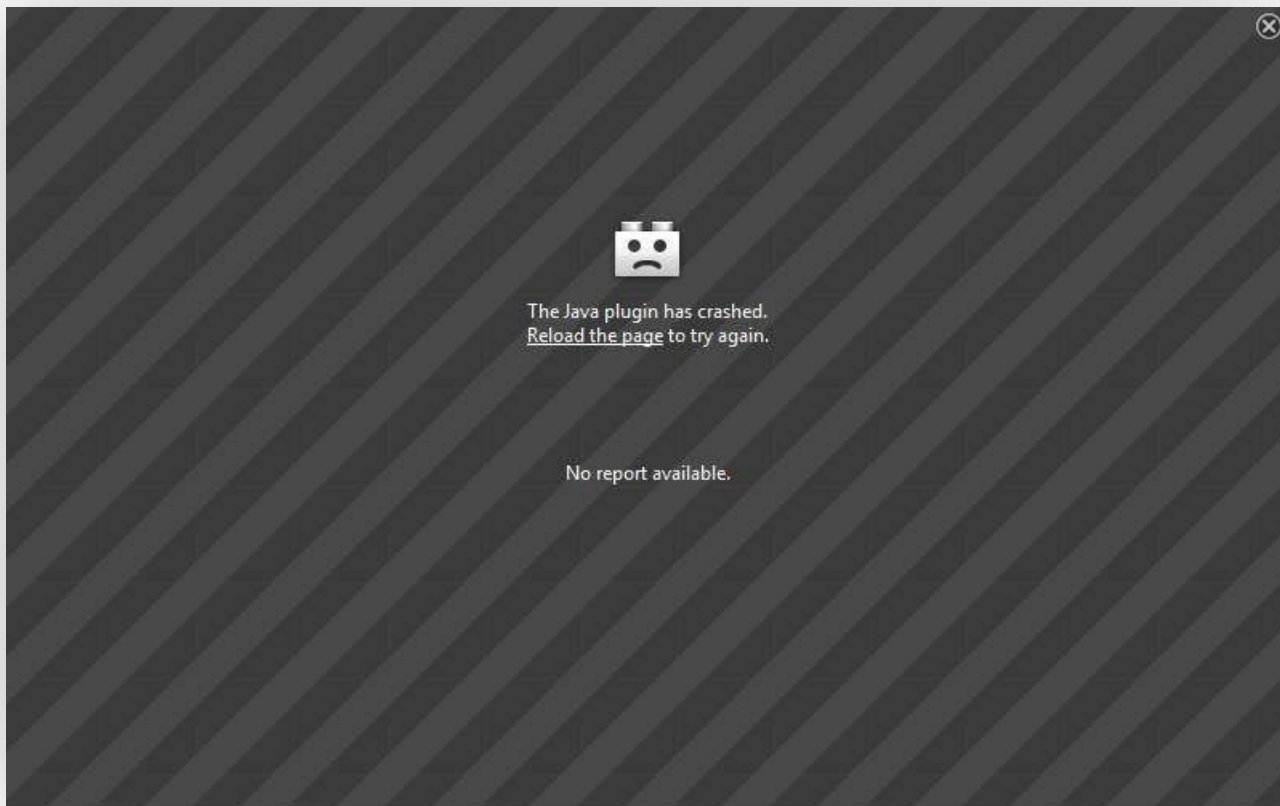


**et même des tentatives "universelles"
(en apparence) sur le web ...**

La plus connue ...



La plus ouverte ...



qui pour l'utilisateur se résumant souvent à :



This plug-in is not supported.

... et si l'on poursuit notre chronologie

- nodejs : 2009 : pas mal, le seul problème de taille c'est qu'il ne comprend que le javascript ...
- asm.js : 2013 : on prend le dénominateur commun de tout le monde : le C
 - Et on le fait cibler une vm universelle moins complexe que llvm-ir implémentée dans un interpréteur javascript
 - Ceci constitue un modèle viable et en preuve de concept, au lieu de Doom, il fait tourner un clone de Quake3* en réseau multiplayer et client-serveur.
- WebAssembly 2015 : C/C++/rust compilent sans problème pour une machine virtuelle très facile à compiler partout ou un compilateur C a vu le jour
- CloudABI (Ed Schouten) ** qui résume les points communs des OS à seulement une cinquantaine d'appels système donne indirectement naissance à WASI (WebAssembly System Interface)

La machine virtuelle sera partout grâce à W3C en 2019! Pourquoi WebAssembly a-t-il gagné ?

- Sécurité et simplicité
- Les performances ne sont pas mauvaises
- C'est juste un standard (MVP 1.0) que tout monde peut implémenter ou il veut, comme il veut. Le choix de l'implémentation de la machine virtuelle est laissé au vendeur.
- Et surtout les GAFAM en ont besoin sur le web !



Mais les PME/ETI en ont besoin aussi...

Quelle demande dans les PME ou ETI industrielles ?

La ... formation !

- Toutes les filières font face à des enjeux de formation
- Les outils de e-learning sont devenus la référence
- On parle de “LMS”, ou *learning management system*
- Les salariés sont capables de se former, en ligne
 - en présentiel
 - en distanciel
- L'évaluation est faite via les outils en ligne



Quelle demande dans les PME/ETI industrielles ?

Cas d'usage #1

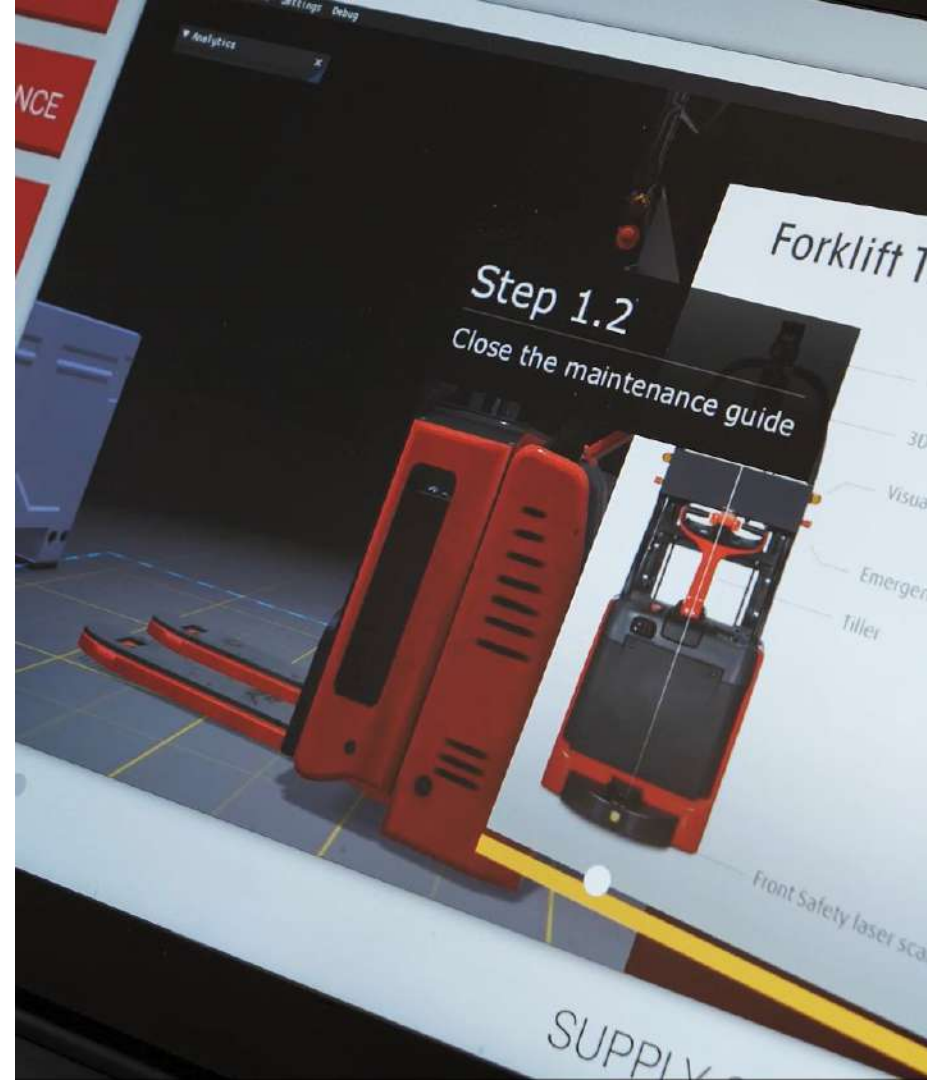
- Je surveille la production à distance
- J'ai besoin d'appréhender le fonctionnement et les contraintes mécaniques
- La 3D me serait utile pour visualiser certains dispositifs complexes sous plusieurs angles, mieux qu'une simple caméra



Quelle demande dans les PME/ETI industrielles ?

Cas d'usage #2

- J'interviens dans la maintenance sur des machines
- Je dois me former sur un nouveau site de production
- Mon métier fait appel à des connaissances mécaniques
- La 3D me serait bien utile pour comprendre certains dispositifs lors de mes sessions d'eLearning



Mais aussi...

La visualisation, la simulation, l'étude du facteur humain... (tout ça en Python 🧡)



La 3D est très utile pour représenter des notions complexes

Comment fonctionne un LMS ?

Un LMS est comme un wordpress, avec des outils pédagogiques.

- Le système sait agréger n'importe quel type de contenu HTML
- Généralement il sait lire des médias vidéo ou audio
- Pour de la 3D interactive, c'est plus compliqué...
 - Un LMS n'accepte généralement un seul type de fichier : un bon gros .ZIP

... ça tombe bien, c'est sous cette forme que Pygbag prépare ses livrables

Quelles alternatives pour amener de la 3D dans un LMS ?

- Unreal Engine *:
 - 30Go d'installation *
- Unity 3D *:
 - 5Go d'installation
- HTML5 avec du soft propriétaire
 - pas d'outil de création du contenu riche
 - pas de 3D ?
- Python
 - pygame-ce pour la 2D
 - Panda3D / HARFANG 3D pour la 3D
 - Avec HARFANG 3D, un éditeur 3D pour créer du contenu



Simplifier l'accès aux applications via WebAssembly.

- Et si on pouvait utiliser un Python complet (comme fait Kivy) ? Mais commun à toutes les applications d'un même type.
- Et si l'on pouvait utiliser la VM d'un navigateur déjà installée et entretenue par un vendeur ?

Le navigateur a un accès universel à presque tous les capteurs !

Sensor	Permission Policy Name
AbsoluteOrientationSensor	'accelerometer' , 'gyroscope' , and 'magnetometer'
Accelerometer	'accelerometer'
AmbientLightSensor	'ambient-light-sensor'
GravitySensor	'accelerometer'
Gyroscope	'gyroscope'
LinearAccelerationSensor	'accelerometer'
Magnetometer	'magnetometer'
RelativeOrientationSensor	'accelerometer' , and 'gyroscope'

Mise en pratique.

- stdin/stdout
- fibonacci
- Tetris
- Pong
- Harfang3D Cube

[Pleins de jeux à tester vous-même listés sur pygame-web.github.io]

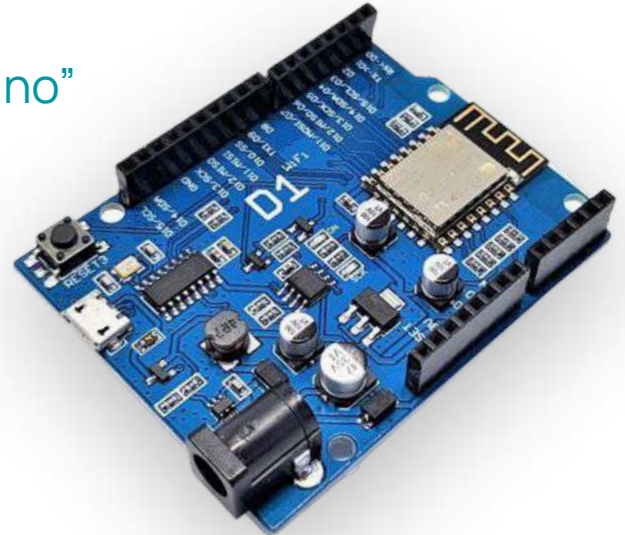
IoT

Microcontrôleurs très économiques sans liaison électrique avec poste de développement : risque de dégâts matériels minimum.

D1 “MINI”



D1 “arduino”



Interfaçage capteurs

l'IoT permet de compléter un pc de bureau ou un laptop et de rivaliser à moindre coût avec un périphérique mobile.

Il devient aussi possible de partager un matériel très simplement dans le labo via le réseau local ou le web.

Et cela sans changer d'api, ni brancher de fils !

Telemetrix-aio

Feature	Telemetrix	StandardFirmata
Entrée analogique (DAC)	X	X
Sortie analogique (PWM)	X	X
Entrée Numérique	X	X
Sortie Numérique	X	X
Servo Moteurs	X	X
Motor Pas à Pas	X	
Capteurs Temperature/Humidité (DHT)	X	
Télémètre ultrasonique HC-SR04	X	
bus SPI	X	
bus OneWire	X	
bus i2c	X	X
Client Python Asyncio Inclu	X	
Support For STM32 Boards (Black Pill)	X	
Extensions utilisateurs	X	
Debugger	X	
Documentation pour tout	X	

DEMOS

à retrouver + tard sur le lien en Forum AFPY

<https://discuss.afpy.org/>

Conclusion: Super ça marche ! Mais (car il y a toujours un mais ...)

Ne PAS faire la course à la nouveauté (64bits/threads/service workers/web gpu/simd-neon)

Ne PAS pousser le hardware.

Mettez au maximum votre app dans un module, et utilisez des imports relatifs.

Ne pas perdre son temps à obfusquer le code, annotez-le pour le compiler + tard avec mypyc.

Programmer de façon éco-responsable sur du vieux matériel pour “sentir” les performances.

Privilégier l'accessibilité. Surtout sur mobile.

Car le but est d'amener toutes les applications
à tout le monde, partout.

Questions

Paul Peny

<https://github.com/pmp-p>



François Guthertz

<https://github.com/astrofra>

