

A detailed painting of a window in a rustic setting. The window has green shutters and a small wind chime hanging from the top. A ginger and white cat is sitting on the windowsill, looking out. The window is surrounded by lush greenery, including ivy on the left and a large bush of yellow flowers on the right. The wall is made of light-colored stone or plaster. In the bottom right corner, there are pink and white flowers.

Nouveautés Python 3.13

Victor
Stinner

Victor Stinner



- **Développeur cœur** de Python depuis 2010
- Employé par **Red Hat** pour contribuer upstream (python.org) et downstream (RHEL, CentOS, Fedora)
- Utilisateur de **Fedora** et **vim**



Red Hat



Python 3.13

Message d'erreur : random.py



```
$ python random.py
Traceback (most recent call last):
  File "/home/me/random.py", line 1
    import random
  File "/home/me/random.py", line 3
    print(random.randint(5))
AttributeError: module 'random' has no
attribute 'randint' (consider renaming
'/home/me/random.py' since it has the same
name as the standard library module...)
```



Red Hat

Message d'erreur : numpy.py



```
$ python numpy.py
Traceback (most recent call last):
  File "/home/me/numpy.py", line 1
    import numpy as np
  File "/home/me/numpy.py", line 3
    np.array([1, 2, 3])
AttributeError: module 'numpy' has no
attribute 'array' (consider renaming
'/home/me/numpy.py' if it has the same
name as a third-party module you intended
to import)
```



Red Hat

Message d'erreur : keyword



```
>>> "Better messages!".split(max_split=1)
```

```
TypeError: split() got an unexpected  
keyword argument 'max_split'. Did you mean  
'maxsplit'?
```



Red Hat

Interpréteur interactif



- Historique **multiligne**
- Commandes `help`, `exit`, `quit` sans parenthèses
- Prompt et traceback **colorés**
- **F1** pour l'aide, F2 pour l'historique, F3 pour coller beaucoup de code
- Portable (**Windows**)



Red Hat

Ligne de commande random



```
# Lancé de dé [1; 6]
```

```
$ python -m random 6
```

```
4
```

```
# Nombre flottant [0; 3.14]
```

```
$ python -m random --float 3.14
```

```
1.8315066410262233
```

```
# Chifoumi (3 choix)
```

```
$ python -m random Pierre Feuille Ciseaux
```

```
Ciseaux
```



Red Hat

Free Threading



Free Threading



- Suppression du verrou global aka GIL
- Projet initié par **Sam Gross** à Meta
- Travail de longue haleine commencé il y a plusieurs années
- **PEP 703** acceptée dans Python 3.13
- Les threads scalent avec le nombre de CPUs
- **Expérimental** et **incomplet** dans 3.13 !



Red Hat

Free Threading



```
def fib(n):  
    if n < 2:  
        return n  
    else:  
        return fib(n - 1) + fib(n - 2)
```

```
def workload():  
    assert fib(37) == 24157817
```



Red Hat

Free Threading



```
from threading import Thread
threads = [Thread(target=workload)
           for _ in range(12)]
for thread in threads:
    thread.start()
for thread in threads:
    thread.join()
```

33.6 sec → 14.2 sec: **2.4x** plus rapide !

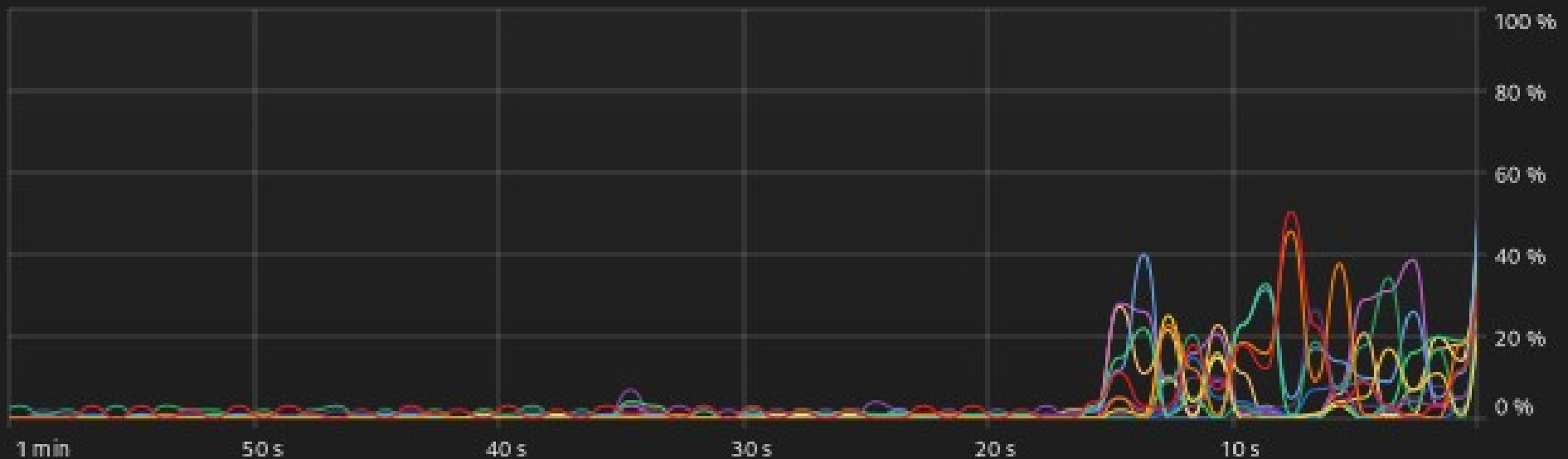














Red Hat

Python classique



▼ CPU



	CPU1	39,0 %		CPU4	25,0 %		CPU7	28,0 %		CPU10	24,0 %
	CPU2	26,5 %		CPU5	46,5 %		CPU8	28,0 %		CPU11	41,0 %
	CPU3	28,0 %		CPU6	60,4 %		CPU9	25,7 %		CPU12	53,9 %















Red Hat

Python Free Threading



▼ CPU



	CPU1	100,0 %		CPU4	99,0 %		CPU7	99,0 %		CPU10	99,0 %
	CPU2	100,0 %		CPU5	100,0 %		CPU8	100,0 %		CPU11	100,0 %
	CPU3	99,0 %		CPU6	100,0 %		CPU9	98,0 %		CPU12	100,0 %



Red Hat

Free Threading



- Python 3.13 : plus lent que Python classique avec un seul thread.
- Python 3.14 : réactiver les optimisations désactivées dans Python 3.13.



Red Hat



Compilation
à la volée

Bytecode adaptatif



- Tier 1
- PEP 659 par Mark Shannon (2021)
- Implémenté dans Python 3.11
- Modifie le bytecode "chaud" à l'exécution
- Ajoute des gardes et spécialise le bytecode



Red Hat

Bytecode adaptatif



```
>>> def sum(x,y): return x+y
...
>>> dis.dis(sum, adaptive=True)
1      RESUME                                0
      LOAD_FAST_LOAD_FAST                  1  (x,  y)
      BINARY_OP                            0  (+)
      RETURN_VALUE
```



Red Hat

Bytecode adaptatif



```
>>> for _ in range(1000): _ = sum(1,1)
...
>>> dis.dis(sum, adaptive=True)
1    RESUME_CHECK                0
      LOAD_FAST_LOAD_FAST        1  (x, y)
      BINARY_OP_ADD_INT          0  (+)
      RETURN_VALUE
```



Red Hat

Compilation à la volée



- Tier 2
- Conception "copy-and-patch" légère et rapide
- Backend LLVM pour compiler Python
- Expérimental et désactivé par défaut !



Red Hat

Compilation à la volée



- Représentation intermédiaire (IR) Tier 2 par **Mark Shannon** et **Guido van Rossum**
- Optimiseur Tier 2 par **Ken Jin**
- Publication "Copy-and-patch" par **Haoran Xu** et **Fredrik Kjolstad** (2021)
- PEP 744 par **Brandt Bucher** (2024)



Red Hat

Compilation à la volée



```
def eratosthene(limite):  
    L = [True] * limite  
    L[0] = False; L[1] = False  
    i = 2  
    while i * i < limite:  
        if L[i]:  
            j = i * i  
            while j < limite:  
                L[j] = False  
                j += i  
        i += 1  
    return L
```

limite=100: 3.16 us → 2.10 us: **1.5x** plus rapide !



Red Hat

Autres changements



Support des plateformes



- **WebAssembly** (WASM) passe en **Tier 2**
(2 core devs, bloque une release)
- **Android** (PEP 738) et Apple **iOS** (PEP 730)
passent en **Tier 3**
(1 core dev, ne bloque pas une release)
- Tiers : définis par la **PEP 11** où Tier 1 est le meilleur support



Red Hat

Divers changements



- multiprocessing tient compte des affinités CPU : ajout de `os.process_cpu_count()`.
- Sous Windows, `time.time()` et `time.monotonic()` ont une précision meilleure que 1 microseconde, au lieu de 15.6 ms.
- `doctest` compte les tests ignorés (*skipped*).



Red Hat

Dette technique



- Suppression de 19 modules, **PEP 594** (2019) : cgi, crypt, nntplib, xdrlib, etc.
- Suppression de **2to3** : son ancien parseur ne gère pas la grammaire Python 3.10.
- Suppression de 263 fonctions privées dans l'**API C**. Ajout de fonctions publiques pour les remplacer.



Red Hat

En résumé



Nouveautés Python 3.13



- Meilleurs messages d'erreur
- Interpréteur interactif plus complet
- **Free Threading** (expérimental)
- **Compilation à la volée** (expérimental)
- Réduction de la dette technique
- Beaucoup de petits changements :

<https://docs.python.org/3/whatsnew/3.13.html>



Red Hat

Questions ?



Droit d'auteur



- Dessins par l'artiste **Djamila Knopf**



Red Hat