La Recette du jour:

Pier-Francesco Rocci - école de photométrie - 09/06/2024

Ingrédients:

1. *Qu'est-ce que Python?*
2. *Une histoire de Python*
3. *Pourquoi Python ?*
4. *Zen de Python*
5. *Petite Comparaison avec C++*
6. *Avantages*
7. *Désavantages*
8. *Le succès en Astronomie*
9. *Message finale*

Qu'est-ce que Python?

*Langage de programmation **interprété**, de haut niveau.*

*Usage **“généraliste”***

***Simple**, **lisible** et **facile** à apprendre.*

Une histoire de Python

Créateur : Guido Van Rossum

Versions : 1991 (0.9.0) → 2000 (2.0) → 2008 (3.0)

Python Software Foundation (https://www.python.org/psf-landing/)

Communauté Open-Source

Pourquoi Python?

Lisibilité et Simplicité: *Syntaxe claire et concise.*

Productivité: *Rapidité de développement.*

Vaste Écosystème: *Nombreuses bibliothèques et frameworks.*

Polyvalence: *Utilisé pour le web, l'IA, sciences des données, l'astronomie, etc…*

Zen de Python

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts. Errors should never pass silently.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

In the face of ambiguity, refuse the temptation to guess.

There should be one – obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea – let's do more of those!

Préfère :

la beauté à la laideur,
l'explicite à l'implicite,
le simple au complexe
et le complexe au compliqué,
le déroulé à l'imbriqué,
l'aéré au compact.

Prends en compte la lisibilité.
Les cas particuliers ne le sont jamais assez pour violer les règles.
Mais, à la pureté, privilégie l'aspect pratique.
Ne passe pas les erreurs sous silence,
... ou bâillonne-les explicitement.
Face à l'ambiguïté, à deviner ne te laisse pas aller.
Sache qu'il ne devrait [y] avoir qu'une et une seule façon de procéder,
même si, de prime abord, elle n'est pas évidente, à moins d'être Néerlandais.
Mieux vaut maintenant que jamais.
Cependant jamais est souvent mieux qu'immédiatement.
Si l'implémentation s'explique difficilement, c'est une mauvaise idée.
Si l'implémentation s'explique aisément, c'est peut-être une bonne idée.
Les espaces de nommage ! Sacrée bonne idée ! Faisons plus de trucs comme ça.

Petite Comparaison avec C++

```
#include<iostream>
using namespace std;

int main()
{
    cout<<"Hello World"<<endl;
    return 0;
}
```

```
print("Hello World")
```

Fonction factorielle en C	Fonction factorielle en Python

```c
int factorielle(int n) {
    if (n < 2) {
        return 1;
    } else {
        return n * factorielle(n - 1);
    }
}
```

```python
def factorielle(n):
    if n < 2:
        return 1
    else:
        return n * factorielle(n - 1)
```

Avantages

Facilité d'apprentissage et Polyvalence.

Bibliothèques très riches.

Communauté Active = Support et ressources.

Pas de Compilation

Typage Dynamique / Gestion automatique de la mémoire

Désavantages

Performance: Plus lent que les langages compilés.

Gestion de la mémoire: moins de contrôle sur la gestion de la mémoire.

Versioning *(solution: environnements virtuelles)*

Le succès en Astronomie

Astropy (https://www.astropy.org/)

Ccdproc (https://ccdproc.readthedocs.io/en/latest/)

Photutils (https://photutils.readthedocs.io/en/stable/)

Astroquery (https://astroquery.readthedocs.io/en/latest/)

Astroplan (https://astroplan.readthedocs.io/en/latest/index.html)

TO BE CONTINUED…