

EP1 - 2017.1

Inspere - Engenharia - Design de Software - Python

Com base na correção dos trabalhos do EP1, seguem dicas de programação:

1. Minimizar uso de variáveis globais, só usar se for realmente necessário
2. Criar variáveis no início do programa ou função
3. Fazer o import de bibliotecas no início do programa
4. Usar e abusar dos comentários (#) para deixar claro os objetivos do seu código!
5. Organizar o programa criando funções
6. Verificar se o código pode ser menor e mais eficiente
7. Evitar [hard code](#)
8. Definir variáveis com nomes apropriados, evitar x, y, z, n... Etc.
9. Seguir estritamente o que foi solicitado na especificação! Mais atenção!
10. Conferir os resultados! Testar o programa, isso é fundamental.
11. Validar se o código estiver "repetitivo", neste caso tem alguma coisa errada! Use loop! Pensar em como organizar melhor o código.
12. Converter estruturas de dados diferentes tem custo alto: lista para set, set para lista, lista para string, etc! Procure manter os dados em um único formado.
13. Minimizar o uso de loops (for ou while). Consolidar o código.

Foi atribuído graduação de excelência (+) aos alunos que além da solução correta implementaram os seguintes recursos:

- Uso de funções (foi ensinado no dia da entrega do EP1 e usado por alguns alunos)
- Comentários (#) relevantes no código do programa
- Uso de regexp (não foi apresentado em aula, e foi bem utilizado em alguns programas)
- Saída em arquivo

Não houve penalização pela falta de otimização ou eficiência do código.

Duas soluções estão disponíveis no Github bem como um arquivo de dados extenso para testes: <https://github.com/danielscarvalho/Inspere-EP1-2017-1>

Teste o seu programa com o arquivo "nomes-big-mess.txt". Para aprender mais: Utilizando Python, tente responder para o arquivo de teste as seguintes questões:

1. Quais os primeiros nomes mais comuns? E os sobre nomes?
2. Quais os nomes femininos e masculinos?
3. Quais os nomes de origem japonesa?
4. Como identificar as pessoas que têm dois primeiros nomes? Ex.: **Kelly Cristina Silva, João Pedro Pereira, Ana Maria** de Carvalho Oliveira

Procure sempre usar boas práticas (*best practices*) de programação:

<https://pt.slideshare.net/carlosschults/10-boas-prticas-de-programao>

<http://www.devmedia.com.br/boas-praticas-de-programacao/31163>

[https://msdn.microsoft.com/en-us/library/aa260844\(v=vs.60\).aspx](https://msdn.microsoft.com/en-us/library/aa260844(v=vs.60).aspx)

[https://en.wikipedia.org/wiki/Best\\_coding\\_practices](https://en.wikipedia.org/wiki/Best_coding_practices)

*Zen of Python* por Tom Peters (em inglês):

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.

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

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

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

There should be one—and preferably only 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!

# THE EVOLUTION OF SOFTWARE ARCHITECTURE

---

## 1990's

SPAGHETTI-ORIENTED  
ARCHITECTURE  
(aka Copy & Paste)



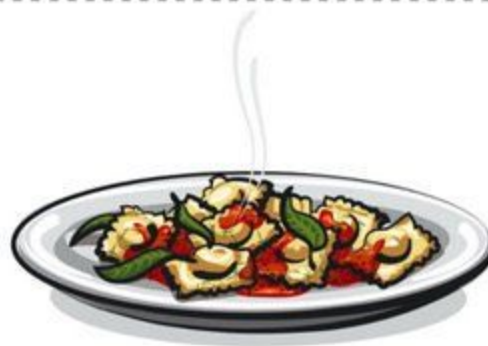
## 2000's

LASAGNA-ORIENTED  
ARCHITECTURE  
(aka Layered Monolith)



## 2010's

RAVIOLI-ORIENTED  
ARCHITECTURE  
(aka Microservices)



## WHAT'S NEXT?

PROBABLY PIZZA-ORIENTED ARCHITECTURE