Nomes Com Agua Potável

Um dos princípios é que nomes de classes métodos, variáveis, módulos, pacotes, e etc, sejam significativos, que apartir de seus nomes consigamos reconhece-los como uma parte da nossa historia, e aonde eles se situam nela, para que ao longo do código consigamos entender com eles transitam dentro da historia ou código.

Se eu lhe perguntasse o que isso embaixo que dizer

```
d = ""
```

Agora se eu desse o código

```
import re

def verify(p):

  number = str(p.number)
  d = re(r"\D*", '', number)

  if len(d) != 11:
      return False
  elif re.match( f"{d[0]}{'{11}'}", str(d)):
      return False
  else:
      return True
```

É uma das coisas que podemos ver dentro desse código é que ainda a uma incógnita, eventualmente de vai ser uma string, que só tem dígitos numéricos, isso graças a expressão regular que esta substituindo tudo que não e um digito (”\D*”), por nada, ou seja number, talvez quando entre possa ou não ser um atributo do do tipo numero, como integer, long e etc. mas possa ser uma Sting ou outro tipo, mas para não ficar claro:

- O que é p
- O que é o atributo number de p
- O que number é do tipo de dado numero ou não
- O por que d só tem dígitos, qual é a importância disso
- Por que o numero 11
- por que o eu não posso ter o mesmo numero repetido onze vezes
- é o mais importante o que essa função verifica

Veja que são muitas perguntas, veja um coisa, primeiro podemos dizer que já entendemos que o atributo number pode não ser numérico, mas veja que para programação number sub se entende que seja um dado do tipo numero, é isso causa mal entendido. então é melhor utilizar nomes que possam causar mal entendido

vamos reescrever o código

```
def verify_cpf(person: People) -> bool:
```

```
    cpf = str(person.register_number_document)
    only_number_cpf = re.sub(r"\D*", '', cpf)
    TOTAL_NUMBER_CPF = 11

    if len(only_number_cpf) != TOTAL_NUMBER_CPF:
        return False
    elif re.match(
        f"{only_number_cpf[0]}{'{11}'}",
        only_number_cpf
    ):
        return False
    else:
        return True
```

Veja que eventualmente nos temos o mesmo código, e agora ficou muito mais claro, com nomes mais claros, também adicionamos, anotações de que tipo de parâmetros esperamos agora fica claro que espero um tipo dado como argumento que é da classe ou herda da classe People. entretanto só adicionamos uma linha que é a constante “TOTAL_NUMBER_LIMIT”, além de algumas quebras.

O código não aumentou muito, todavia não é o que ocorre é abaixo nos temos um código que melhora o de cima ainda mais:

```
def verify_cpf(register_number_document) -> bool:

  cpf = str(register_number_document)
  not_digit_regex = r"\D*"

  only_number_cpf = re.sub(not_digit_regex, '', cpf)
  TOTAL_NUMBER_CPF = 11

  number_repeated_all_regex = f"{only_number_cpf[0]}{'{11}'}"

  if len(only_number_cpf) != TOTAL_NUMBER_CPF:
      return False
  elif re.match(
      number_repeated_all_regex,
      only_number_cpf
  ):
      return False
  else:
      return True
```

Compreenda que agora, eu tenho as padrões das expressões regulares em variáveis e mesmo que eu não tenha conhecimento de expressões regulares, como construir uma expressão agora tendo um conhecimento unicamente dos métodos, consigo saber o que esta sendo realizado.

E perceba que agora eu também não tenho person, nos parâmetros agora eu recebo o registro direto.

Coisas que podem causar confusão

A primeira coisa que pode causar confusão na hora de nomear algo é que, existe uma dificuldade em nomear num primeiro momento, é o nome deve ser claro de forma que consigamos identificar o objetivo, o que temos que fugir é:

1. Evite utilizar nomes que estão ligados a alguma convenção dentro programação ou envolvendo tecnologia, principalmente se a variável não tem como objetivo evidenciar aquela características ou pode causar confusão

   como referencia algo como number, list, array essas palavras no nome podem causas confusão principalmente se você tem lá, “objects_list” e na verdade, não é um objeto, para usar esses nomes e necessário um certo cuidado

2. Outra coisa é a utilização de nomes semelhantes que podem causar muita confusão entre o que é o que principalmente quando eles são grande como