Recuperação - Validador de CPF

Resumo

O algoritmo de validação do CPF calcula o primeiro dígito verificador a partir dos 9 primeiros dígitos do CPF, e em seguida, calcula o segundo dígito verificador a partir dos 9 (nove) primeiros dígitos do CPF, mais o primeiro dígito, obtido na primeira parte.

Exemplo

Vamos usar como exemplo o CPF fictício, mas válido : 111.444.777-35.

Cálculo dos Dígitos

Cálculo do Primeiro Dígito

O primeiro passo é calcular o primeiro dígito verificador, e para isso, separamos os primeiros 9 dígitos do CPF (111.444.777) e multiplicamos cada um dos números, da direita para a esquerda por números crescentes a partir do número 2, como no exemplo abaixo :

1	1	1	4	4	4	7	7	7
*	*	*	*	*	*	*	*	*
10	9	8	7	6	5	4	3	2
=	=	=	=	=	=	=	=	=
10	9	8	28	24	20	28	21	14

Multiplicamos cada digito do CPF pelo respectivo número e somamos cada um dos resultados : 10 + 9 + 8 + 28 + 24 + 20 + 28 + 21 + 14 = 162
Pegamos o resultado obtido 162 e dividimos por 11. Consideramos como quociente apenas o valor inteiro.

162 / 11 = 14 com resto 8

- Se o resto da divisão for menor que 2, então o dígito é igual a 0 (Zero).
- Se o resto da divisão for maior ou igual a 2, então o dígito verificador é igual a 11 menos o resto da divisão (11 - resto).

No nosso exemplo temos que o resto é 8 então faremos 11 - 8 = 3

Logo o primeiro dígito verificador é 3. Então podemos escrever o CPF com os dois dígitos calculados : 111.444.777-3X

Cálculo do Segundo Dígito

Para calcular o segundo dígito vamos usar o primeiro digito já calculado. Vamos montar a mesma tabela de multiplicação usada no cálculo do primeiro dígito. Só que desta vez usaremos na segunda linha os valores 11, 10, 9, 8, 7, 6, 5, 4, 3, 2 já que estamos incluindo mais um digito no cálculo(o primeiro dígito calculado):

1	1	1	4	4	4	7	7	7	3
*	*	*	*	*	*	*	*	*	*
11	10	9	8	7	6	5	4	3	2
=	=	=	=	=	=	=	=	=	=
11	10	9	32	28	24	35	28	21	6

Novamente efetuamos a soma dos resultados da multiplicação : 11 + 10 + 9 + 32 + 28 + 24 + 35 + 28 + 21 + 6 = 204
Dividimos o total do somatório por 11 e consideramos o resto da divisão.

204 / 11 = 18 e resto 6
Após obter o resto da divisão, precisamos aplicar a mesma regra que utilizamos para obter o primeiro dígito:

- Se o resto da divisão for menor que 2, então o dígito é igual a 0 (Zero).
- Se o resto da divisão for maior ou igual a 2, então o dígito é igual a 11 menos o resto da divisão (11 - resto).

11 - 6 = 5 logo 5 é o nosso segundo dígito verificador.

Logo o nosso CPF fictício será igual a : 111.444.777-35.

Passo 1 - Primeiro Código

Crie um algoritmo em Python que realize a validação do CPF de acordo com a forma que foi descrita acima.

Algumas dicas para o desenvolvimento :

- foque na resolução do problema;
- divida o grande problema em problemas menores, depois junte tudo;
- use uma declaração de variável para o CPF, não se preocupe em receber do usuário agora;

Para esse projeto, crie uma pasta separada para todos os arquivos. Mais tarde isso vai ser importante pela quantidade de arquivos criados.

Passo 2 - Funções

Após criado o algoritmo de Validação do CPF, adapte para que o sistema utilize funções e chamadas de funções para realizar essa validação.

PS: repare que a maneira de realizar a validação do primeiro e do segundo dígito é praticamente a mesma, com apenas duas diferenças:

- o multiplicador inicia em 10 no primeiro dígito, enquanto no segundo ele inicia em 11;
- para o primeiro dígito é realizado o cálculo com os **9 primeiros números** do CPF, enquanto no segundo dígito é utilizado os **10 primeiros números**, incluindo no cálculo o primeiro dígito verificador;

Passo 3 – Input e Exception

Nessa etapa, adapte seu algoritmo para que um número indeterminado de CPFs a serem testados sejam digitados pelo usuário. Ofereça uma opção para que o usuário encerre o programa.

Faça os testes necessários para validar esse CPF digitado:

- caracteres válidos: números, ponto e traço;
- caracteres inválidos: todo o resto;

Faça as adaptações necessárias para utilizar as cláusulas try...except (https://docs.python.org/pt-br/3/tutorial/errors.html#handling-exceptions) e validar o CPF digitado de modo que o programa nunca seja interrompido por um erro de execução;

Passo 4 - Arquivos

Nessa etapa, adapte seu algoritmo para salvar os resultados da validação do CPF em dois arquivos diferentes:

- resultado.txt: arquivo com todos os CPFs válidos testados;
- erros.log: arquivo de log onde terá todos os CPFs inválidos e uma mensagem de erro explicando o porquê dele ser rejeitado;

PS: os arquivos **não devem** ser limpos a cada execução do programa e cada CPF verificado deve ser colocado em uma linha individual, junto com a mensagem.

Passo 5 – Classes e Módulos

Nessa etapa:

- crie uma classe Cpf que irá realizar o gerenciamento do CPF;
    - se for o caso, atribua o CPF diretamente à declaração do objeto para agilizar o desenvolvimento, depois faça o uso do input para receber do usuário na versão final;
    - adicione atributos e métodos à classe criada de acordo com as necessidades, com a condição que ela tenha:
        - pelo menos **uma** implementação de @classmethods e @staticmethods (um de cada);

        - pelo menos **um** método mágico, excluindo o __init__ (logo, pelo menos 2 métodos mágicos);
- crie uma classe Arquivo para salvar os arquivos resultado.txt e erros.log;
    - adicione atributos e métodos à classe criada de acordo com as necessidades, com a condição que ela tenha :
        - pelo menos **uma** implementação de @classmethods e @staticmethods (um de cada);
        - pelo menos **um** método mágico, excluindo o __init__ (logo, pelo menos 2 métodos mágicos);

O código deve ser dividido em 2 módulos. Um para ficar com as classes do programa e outro para servir de executor do código.

Considerações Finais

Não será necessário usar tkinter para o desenvolvimento. A interface do programa será toda pelo terminal. Use os.system('clear') para limpar o terminal e deixar organizado.

O programa deve aceitar diversos CPFs consecutivos, salvando os válidos e inválidos nos arquivos previamente descritos. Em uma nova execução, o programa não deve apagar os CPFs já verificados dos arquivos.

Siga o passo a passos acima em ordem. Isso vai facilitar no desenvolvimento, já que ele segue uma ordem lógica.

Para a formatação do código e padrões, use a PEP8 – Style Guide for Python Code (https://peps.python.org/pep-0008).