Coverage for **validador_cpf.py**: 100%

26 statements | 26 run | 0 missing | 0 excluded


```
"""
Módulo responsável por validar um CPF.

O CPF é composto por 11 dígitos, sendo os dois últimos dígitos verificadores.
Para validar um CPF, é necessário calcular os dígitos verificadores e compará-los
com os dígitos verificadores do CPF. Se os dígitos verificadores forem iguais,
o CPF é válido.
"""

import re


class LimpezaCPF:
    """ Classe responsável por limpar o CPF, removendo caracteres não numéricos. """

    @staticmethod
    def limpeza(cpf: str) -> str:
        """
        Limpa o CPF removendo caracteres não numéricos.

        Args:
            cpf (str): O CPF a ser limpo.

        Returns:
            str: O CPF limpo, contendo apenas dígitos.
        """
        return re.sub(r"[^0-9]", "", cpf)


class ValidadorCPF:
    """ Classe responsável pela validação do CPF. """

    def __init__(self, cpf: str):
        """
        Inicializa o validador de CPF.

        Args:
            cpf (str): O CPF a ser validado.
        """
        self.cpf = LimpezaCPF.limpeza(cpf)

    @staticmethod
    def _calcular_digito(digitos: str, peso_inicial: int) -> int:
        """
        Calcula um dígito verificador baseado nos dígitos do CPF e um peso inicial.

        Args:
            digitos (str): Parte do CPF usada para calcular o dígito verificador.
```

```
49              peso_inicial (int): Peso inicial usado no cálculo.
50
51          Returns:
52              int: O dígito verificador calculado.
53          """
54          digito_calculado = sum(int(digito) * peso for digito, peso
55                                 in zip(digitos, range(peso_inicial, 1, -1)))
56          return (digito_calculado * 10) % 11
57
58      @staticmethod
59      def _obter_digito_verificador(digitos_cpf: str, peso: int) -> int:
60          """
61          Obtém o dígito verificador correto para uma parte do CPF.
62
63          Args:
64              digitos_cpf (str): Parte do CPF usada para obter o dígito verificador.
65              peso (int): Peso inicial usado para calcular o dígito verificador.
66
67          Returns:
68              int: O dígito verificador.
69          """
70          digito_verificador = ValidadorCPF._calcular_digito(digitos_cpf, peso)
71          return digito_verificador if digito_verificador < 10 else 0
72
73      @staticmethod
74      def _validar_digitos(cpf: str) -> bool:
75          """
76          Valida os dígitos verificadores do CPF.
77
78          Returns:
79              bool: Verdadeiro se os dígitos verificadores são válidos, falso caso contrário.
80          """
81          if len(cpf) != 11 or cpf[0] * len(cpf) == cpf:
82              return False
83
84          cpf_sem_digitos_verificadores = cpf[:9]
85          digito_verificador_1 = ValidadorCPF._obter_digito_verificador(
86              cpf_sem_digitos_verificadores, 10)
87          digito_verificador_2 = ValidadorCPF._obter_digito_verificador(
88              cpf_sem_digitos_verificadores + str(digito_verificador_1), 11)
89
90          return int(cpf[9]) == digito_verificador_1 and int(cpf[10]) == digito_verificador_2
91
92      def validacao_cpf(self) -> bool:
93          """
94          Verifica se o CPF é válido.
95
96          Returns:
97              bool: Verdadeiro se o CPF é válido, falso caso contrário.
98          """
99          return ValidadorCPF._validar_digitos(self.cpf)
```