Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação

Disciplina: Programação com Interfaces Gráficas

AP1 2^o semestre de 2022.

Nome -

Assinatura -

---

1. (10 pontos) Um CPF é um número com 11 dígitos, sendo que os dois dígitos menos significativos são calculados em função dos nove restantes e são chamados de dígitos verificadores, pois permitem validar o CPF.

   Alguns CPFs válidos:

   - 757.331.783-29
   - 841.315.629-79
   - 635.111.771-20
   - 348.536.715-01

   O processo de validação é bastante simples e está descrito a seguir:

   $CPF = d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 - v_2 v_1$

   $v_2 = 10 * d_9 + 9 * d_8 + 8 * d_7 + 7 * d_6 + 6 * d_5 + 5 * d_4 + 4 * d_3 + 3 * d_2 + 2 * d_1$

   ```
   resto = v2 mod 11
   Se resto ≤ 1 então v2 = 0
        senão v2 = 11 - resto
   ```

   $v_1 = 11 * d_9 + 10 * d_8 + 9 * d_7 + 8 * d_6 + 7 * d_5 + 6 * d_4 + 5 * d_3 + 4 * d_2 + 3 * d_1 + 2 * v_2$

   ```
   resto = v1 mod 11
   Se resto ≤ 1 então v1 = 0
        senão v1 = 11 - resto
   ```

   Complete o esqueleto do Código 1 que valida, gera e formata CPFs, que podem conter pontos e/ou hífen.

   (a) Escreva uma função **dotProd**, com uma única linha de código, que retorna a soma dos produtos dos elementos correspondentes de duas listas, ou seja, o produto escalar ou interno de v_1 e v_2:

   $$v_1[1] * v_2[1] + v_1[2] * v_2[2] + ... + v_1[n] * v_2[n]$$

   (b) Usando a função dotProd, faça uma outra função **areValidDigits** que recebe um CPF (sem os dígitos verificadores), os dois dígitos verificadores (dv) e retorna se o CPF é válido ou não.

A ideia é receber uma string com n dígitos e transformá-la em uma lista d de inteiros (dígito 1 na posição n, dígito 2 na posição $n-1$, ..., dígito n na posição 1). Depois, fazer o dotProd desta lista com uma lista com os pesos de cada dígito.

(c) Implemente uma função **randomCPF** para gerar um CPF aleatório, mas válido. Use a função $randrange(start : int, stop : int, step : int) \rightarrow int$ do módulo random.

(d) Implemente uma função **formatCPF** para formatar um CPF, inserindo pontos para separar triplas de dígitos.

(e) Transforme o Código 1 em uma classe.

(f) Usando o unittest, escreva testes unitários para todas as funções criadas.

Código 1: Autenticador de CPF

```
import sys
from operator import mul
from typing import List
import random
import re

##
#   Produto escalar de v1 e v2 (1 linha).
#
def dotProd(v1: List[int], v2: List[int]) -> int:
    ....
    return ...

##
#   Valida um CPF usando os dígitos verificadores fornecidos (11 linhas).
#
#   @param cpf CPF.
#   @param dv  dígitos verificadores.
#   @return uma tupla com:
#           - status: True se cpf é válido, e False caso contrário
#           - os dígitos verificadores reais.
#
def areValidDigits ( cpf: str, dv: int ) -> tuple[bool, str, int]:
    wv2 = list(range(10, 1, -1))    # pesos para cálculo de v2
    wv1 = [11] + wv2                # pesos para cálculo de v1

    .
    .    COMPLETE A FUNÇÃO !!!!!
    .

    DV = v2 * 10 + v1
    return (DV == dv), DV

##
#   Valida um CPF (6 linhas).
#
#   @param cpf uma string incluindo pontos e traços.
#   @throw lança um exceção ValueError se a string fornecida for inválida.
#   @return uma tupla com:
#           - status: True (válido) ou False (inválido),
#           - cpf,
#           - os dígitos verificadores.
#
def validateCPF(cpf: str) -> tuple[bool, str, int]:
    ....
    return ...
```

```
##
#   Gera um CPF aleatório (3 linhas).
#
#   @return uma tupla (cpf, dv).
#
def randomCPF() -> tuple[str, int]:
    ....
    return ...

##
#  Formata um CPF inserindo pontos entre triplas de dígitos (3 linhas).
#
#  @param cpf string representando um CPF: "111444777"
#  @return um CPF formatado: "111.444.777"
#
def formatCPF(cpf: str) -> str:
    ...
    return ...

##
#   Programa principal para testagem.
#
def main(argv=None):
    if argv is None:
        argv = sys.argv

    cpf = input(
        "Digite o CPF a ser testado: xxx.xxx.xxx-dv ") \
        if len(argv) < 2 else argv[1]

    try:
        status, cpf, dv = validateCPF(cpf)
    except ValueError:
        status = True
        cpf = formatCPF("0" * 9)
        dv = 0
        print('Null CPF: {}'.format(cpf))

    print("{}-{:02d} é um CPF {}válido".format(cpf, dv, "" if status else "in"))
    print("CPF Aleatório: {}-{:02d}".format(*randomCPF()))
    main('a')


if __name__ == "__main__":
    try:
        sys.exit(main())
    except (KeyboardInterrupt, EOFError):
        sys.exit("\nBye, bye")
```

Eis uma execução do programa:

```
roma: ~/cederj/AP/2022-2$ gabarito/cpf.py
Digite o CPF a ser testado: xxx.xxx.xxx-dv 11144477735
111.444.777-35 é um CPF válido
CPF Aleatório: 191.547.591-02
Digite o CPF a ser testado: xxx.xxx.xxx-dv 191.547.591-02
191.547.591-02 é um CPF válido
CPF Aleatório: 479.090.875-51
Digite o CPF a ser testado: xxx.xxx.xxx-dv ^C
Bye, bye
```