Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação

Disciplina: Programação com Interfaces Gráficas

AP3 2^o semestre de 2022.

Nome -

Assinatura -

---

1. (10 pontos) Um CNPJ é um número com 14 dígitos, sendo que os dois dígitos menos significativos são calculados em função dos doze restantes e são chamados de dígitos verificadores, pois permitem validar o CNPJ.

   Alguns CNPJs válidos:

   - 14.358.805/0001-16
   - 72.181.240/0001-40
   - 91.655.845/0001-70
   - 72.060.999/0001-75

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

   $CNPJ = d_{12}d_{11}d_{10}d_9d_8d_7d_6d_5d_4d_3d_2d_1 - v_2v_1$

   $v_2 = 5 * d_{12} + 4 * d_{11} + 3 * d_{10} + 2 * d_9 + 9 * d_8 + 8 * d_7 + 7 * d_6 + 6 * d_5 + 5 * d_4 + 4 * d_3 + 3 * d_2 + 2 * d_1$

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

   $v_1 = 6 * d_{12} + 5 * d_{11} + 4 * d_{10} + 3 * d_9 + 2 * d_8 + 9 * d_7 + 8 * d_6 + 7 * d_5 + 6 * d_4 + 5 * d_3 + 4 * d_2 + 3 * d_1 + 2 * v_2$

   ```
   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 CNPJs, que podem conter pontos e/ou barra 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 CNPJ (sem os dígitos verificadores), os dois dígitos verificadores (dv) e retorna se o CNPJ 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 **randomCNPJ** para gerar um CNPJ aleatório, mas válido. Use a função $randrange(start : int, stop : int, step : int) \rightarrow int$ do módulo random.

(d) Implemente uma função **formatCNPJ** para formatar um CNPJ, inserindo pontos e uma barra para separar grupos 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 CNPJ

```
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 CNPJ usando os dígitos verificadores fornecidos (11 linhas).
#
#   @param cnpj CNPJ.
#   @param dv  dígitos verificadores.
#   @return uma tupla com:
#           - status: True se cnpj é válido, e False caso contrário
#           - os dígitos verificadores reais.
#
def areValidDigits ( cnpj: str, dv: int ) -> tuple[bool, str, int]:
    wv2 = list(range(5, 1, -1)) + list(range(9, 1, -1)) # pesos para cálculo de v2
    wv1 = [6] + wv2                                     # pesos para cálculo de v1

    .
    .    COMPLETE A FUNÇÃO !!!!!
    .

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

##
#   Valida um CNPJ (6 linhas).
#
#   @param cnpj 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),
#           - cnpj,
#           - os dígitos verificadores.
#
def validateCNPJ(cnpj: str) -> tuple[bool, str, int]:
    ....
    return ...
```

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

##
#  Formata um CNPJ inserindo pontos e uma barra entre grupos de dígitos (3 linhas).
#
#  @param cnpj string representando um CNPJ: "114447770001"
#  @return um CNPJ formatado: "11.444.777/0001"
#
def formatCNPJ(cnpj: str) -> str:
    ...
    return ...

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

    cnpj = input(
        "Digite o CNPJ a ser testado: xx.xxx.xxx/xxxx-dv ") \
        if (len(argv) < 2) else argv[1]

    try:
        status, cnpj, dv = validateCNPJ(cnpj)
    except ValueError:
        status = True
        cnpj = formatCNPJ("0" * 12)
        dv = 0
        print('Null CNPJ: {}'.format(cnpj))

    print("{}-{:02d} é um CNPJ {}válido".format(cnpj, dv, "" if status else "in"))
    print("CNPJ Aleatório: {}-{:02d}".format(*randomCNPJ()))
    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/cnpj.py
Digite o CNPJ a ser testado: xx.xxx.xxx/xxxx-dv 11444777000161
11.444.777/0001-61 é um CNPJ válido
CNPJ Aleatório: 52.936.608/4602-08
Digite o CNPJ a ser testado: xx.xxx.xxx/xxxx-dv 52.936.608/4602-08
52.936.608/4602-08 é um CNPJ válido
CNPJ Aleatório: 71.158.392/0884-07
Digite o CNPJ a ser testado: xx.xxx.xxx/xxxx-dv ^C
Bye, bye
```