Ariel Lisboa Lopes

Análise e Desenvolvimento de Sistemas

Atividade 1:

Criei uma variável de nome “time\_de\_futebol” que recebe um valor de escopo global através do input, que será adicionado pelo usuário. A função 1, recebe uma condicional IF/ELSE onde retornará ou não se o time é campeão mundial.

A função 2 altera em escopo local o valor da variável “time\_de\_futebol” e retorna que não é campeão mundial.

time\_de\_futebol = input('Qual é o seu time?: ')  # variavel de escopo global

def func1():         # primeira função

    if time\_de\_futebol == "Flamengo":        # condicioanais

        print(f'{time\_de\_futebol} é campeão mundial')

    else:

        print(f'{time\_de\_futebol} não é campeão mundial')

def func2():   # segunda função

    time\_de\_futebol = 'Palmeiras'     # escopo local / a variável muda de valor somente dentro dessa função

    print(f'{time\_de\_futebol} não é campeão mundial')

func1()

func2()

Atividade 2:

A variável “times\_de\_futebol” recebe um array com valores repetidos.

A variável ‘TimesNãoRepetidos’ receberá um novo valor a cada volta do laço FOR. A condicional IF determinará se o valor já existe ou não na array. E função “.append” adicionará o valor na variável ‘TimesNãoRepetidos’

times\_de\_futebol = ["Flamengo", "Fluminense", "Flamengo", "Santos", "Vasco", "Ceará", "Fluminense", "Real Madrid", "Bragantino"]

# lista com elementos duplicados

TimesNãoRepetidos = []      # nova lista que irá armazenar elementos únicos

for nome\_do\_time in times\_de\_futebol:        # loop for para percorrer a lista comnomes duplicados

    if nome\_do\_time not in TimesNãoRepetidos:  # condicional para determinar se o elemento será adicionada na nova lista

        TimesNãoRepetidos.append(nome\_do\_time) # o elemento só entra na nova lista sse não for repetido

print(TimesNãoRepetidos)

Atividade 3:

Tenho 3 programas.

No primeiro programa eu utilizo a classe ‘date’ do módulo ‘datetime’. A variável Data\_Hoje guarda o valor que será trazido pela função date.today.

No segundo programa eu importo o módulo secrets que é usado para gerar números aleatórios criptografados fortemente.

A variável ‘senha’ guardará o valor que será gerado através da função ‘.token\_hex’. Essa função retorna uma string hexadecimal aleatória cada byte será convertido em dois dígitos. O parâmetro passado foi ‘4’, então minha senha terá 8 caracteres aleatórios

No terceiro programa eu importo a classe randint do módulo random. Essa classe retorna valores inteiros de acordo com os parâmetros passados, porém me passará strings pois eu converti. Na última etapa do meu gerador de cpf, eu utilizo o laço FOR para decrementar meu contador e incrementar minha variável CPF\_novo. O cálculo usado encontra-se em: <https://campuscode.com.br/conteudos/o-calculo-do-digito-verificador-do-cpf-e-do-cnpj>

No final o print exibe informações geradas pelos 3 programas.

# Módulo 1

from datetime import date             ## importei a classe date do módulo datetime

Data\_Hoje = date.today()              ## usei o método today para retornar a data atual

# Módulo 2

import secrets                      ## o módulo secrets é usado para gerar números aleatórios criptograficamente forte

senha = secrets.token\_hex(4)        ## retorna uma string aleatória hexadecimal. Cada byte convertido em dois dígitos hexadecimais.

## como eu coloquei o parâmetro '4', me retornará uma sena de 8 caracteres aleatórios

# Módulo 3

from random import randint         ## importei a função randint do módulo random para retorna uma valor inteiro de acordo com os parâmetros passados abaixo

digitos\_aleatorios = str(randint(100000000, 999999999))   ## converti para string e passei parâmetros

CPF\_novo = digitos\_aleatorios

reverso = 10                     ## contador reverso

total = 0

                                 ## cálculos para gerar cpf

for v1 in range(19):

    if v1 > 8:                   ## primeiro índice vai de 0 a 9,

        v1 = v1 - 9              ## são os primeiros dígitos do cpf

    total += int(CPF\_novo[v1]) \* reverso  ## valor ttotal da multiplicação

    reverso = reverso - 1        ## decrementa o contador reverso

    if reverso < 2:

        reverso = 11

        d = 11 - (total % 11)

        if d > 9:               ## se o digito for maior que 9 o valor é 0

            d = 0

        CPF\_novo += str(d)      ## concatena o dígito gerando novo cpf

        total = 0               ## zera o total

print(f'Olá! Na data de hoje: {Data\_Hoje}, você gerou uma nova senha: {senha}! E também gerou um novo CPF: {CPF\_novo}')

Atividade 4:

Utilizei o módulo matplotlib para plotar os gráficos e a função ‘.loadtxt’ do móulo numpy para leitura dos dados em txt. ‘.plot’ irá plotar o gráfico de acordo com os parâmetros passados e o ‘.show’ mostra.

import matplotlib.pyplot as p     ## importei a função pyplot do módulo matplotlib e dei um 'apelido' curto para facilitar

import numpy as n                 ## importei numpy pra leitura de dados e apelidei de 'n'

informações\_vendas = n.loadtxt('dias\_e\_valores.txt')    ## loadtxt é uma função do numpy para ler arquivos .txt

p.plot(informações\_vendas[:,0], informações\_vendas[:,1])  ## plotar os dados

p.show()               ## mostrar o gráfico