**Nome:** Pedro Beresford Rocha

**RA:** 23015967

-------------------------------------------------------------------------------------------------------------------

**Diário de Bordo - Projeto 2 - APPC-P**  
***Todo o desenvolvimento será publicado no meu repositório público do GitHub:***[*https://github.com/xSh4doow/SI-PUCCAMP/tree/main/Python%20-%201%20Semestre/APPC-P/Trabalho%202*](https://github.com/xSh4doow/SI-PUCCAMP/tree/main/Python%20-%201%20Semestre/APPC-P/Trabalho%202)  
  
**Dia 24/05** - Criação do Dataset.  
**Dia 25/05** - Envio da Fase 1, Início do código base.  
**Dia 26/05** - Continuação do Código Base.  
**Dia 05/06** - Finalização do Código Base e Entrega Fase 2.

-------------------------------------------------------------------------------------------------------------------

**Descrição da proposta de implementação:**

O código apresentado é um sistema seletor de carreira que ajuda os usuários a identificar quais profissões são mais adequadas para eles com base em suas habilidades. O programa funciona da seguinte maneira:

1. Importa o módulo **time** e os dados necessários dos módulos **fase1\_dataset** **dataset** e **habilidades**.
2. Define a função **show\_Habilidades()** que imprime as habilidades disponíveis.
3. Define a função **get\_Habilidades(datas)** que solicita ao usuário que selecione as habilidades com as quais ele mais se relaciona. A entrada é separada por vírgulas e verificada se todas as habilidades fornecidas estão presentes no conjunto de dados.
4. Define a função **calc\_prob(ds, profissao, habilidade)** que calcula a probabilidade de uma determinada habilidade ser relevante para uma profissão específica. É usado um cálculo simples de porcentagem com base no número de ocorrências da habilidade em relação ao total de habilidades da profissão.
5. Define a função **match\_Habilidades(u\_input, ds)** que recebe as habilidades do usuário e o conjunto de dados e encontra as profissões mais adequadas para essas habilidades. O cálculo de probabilidade é feito para cada habilidade em relação a cada profissão, e as profissões são classificadas com base na soma das probabilidades das habilidades correspondentes.
6. Define a função **format\_print(dic)** que formata e imprime a resposta final com base nas profissões encontradas e suas respectivas probabilidades.
7. Define a função **continuar()** que permite ao usuário decidir se deseja continuar no sistema ou sair.
8. Define a função principal **main()** que inicia o programa. Ela imprime uma mensagem de boas-vindas e solicita ao usuário que selecione suas habilidades. Em seguida, encontra as profissões correspondentes e imprime os resultados. Após isso, pede ao usuário se ele deseja continuar no sistema.
9. A função **main()** é chamada se o módulo for executado diretamente.

-------------------------------------------------------------------------------------------------------------------

**Algoritmo (Portugol):**

// Início do Código

função mostrar\_Habilidades() para cada hab em habilidades faça imprimir '\033[32m' + hab + '\033[0;0m'

função obter\_Habilidades(datas)

enquanto Verdadeiro faça

user\_hab = input('Selecione as habilidades que você mais se relaciona! Separe por vírgulas \n').split(', ')

se todos(hab não está em datas para hab em user\_hab) então

retorne user\_hab

senão imprimir '\033[31m' + 'A(s) habilidade(s) escrita(s) não está(ão) no nosso DataSet, tente novamente!' + '\033[0;0m'

função calcular\_probabilidade(ds, profissao, habilidade)

para cada prof em ds faça

se prof['PROFISSÃO'] é igual a profissao então

vals = prof['HABILIDADES'].valores()

hab = prof['HABILIDADES'].obter(habilidade)

calc = arredondar((hab / soma(vals)) \* 100, 1)

retorne calc

função encontrar\_Habilidades(u\_input, ds)

dic = {}

para cada prof em ds faça

para cada hab em u\_input faça

se hab está em prof['HABILIDADES'] então

val = calcular\_probabilidade(ds, prof['PROFISSÃO'], hab)

se prof['PROFISSÃO'] não está em dic então

dic.adicionar({prof['PROFISSÃO']: val})

senão \_val = dic.obter(prof['PROFISSÃO']) \_nval = val + \_val dic.adicionar({prof['PROFISSÃO']: \_nval})

retorne dic.ordenado(por=(função item: item[1]), reverso=Verdadeiro)

função formatar\_impressão(dic)

imprimir 'De acordo com o que você inseriu, calculamos que:'

para cada prof em dic faça

imprimir 'Você tem ' + "\033[36m" + string(dic.obter(prof)) + "\033[0;0m" + '% de chance de se dar bem como: ' + "\033[35m" + prof + "\033[0;0m"

imprimir '-----------------------------------------------------------'

função continuar()

esperar(1.5)

enquanto Verdadeiro faça

imprimir 'Deseja continuar no sistema?'

resp = input('Sim ou Não?\n')

se resp.lower() é igual a 'sim' então

imprimir 'Ok, redirecionando para o menu principal!'

esperar(1.5)

parar

senão se resp.lower() é igual a 'nao' então

imprimir 'Tudo bem, obrigado por usar nosso sistema!'

esperar(1)

sair

senão imprimir 'Resposta Inválida, Tente Novamente!'

função principal()

imprimir '\033[34m''''\n-----------------------------------------------------------\nOlá! Seja bem vindo(a) ao Sistema Seletor de Carreira!\nPara continuar-mos, selecione as habilidades com as quais\nvocê mais se relaciona:\n-----------------------------------------------------------\nHABILIDADES:' + '\033[0;0m'

mostrar\_Habilidades()

u = obter\_Habilidades(habilidades)

d = encontrar\_Habilidades(u, dataset)

formatar\_impressão(d)

continuar()

se **name** é igual a '**main**' então principal()