Programação com Python - TP2

Professor: Adalberto Oliveira

Nome do Aluno(a): Alberto Fontenelle Pluecker

Respostas das Questões

Questão 1

Desenvolva uma função que apresente um menu inicial com as opções listadas abaixo e que tenha como retorno a opção escolhida:

1. Criar um registro
2. Consultar um registro pelo ID
3. Listar os registros
4. Modificar um registro
5. Apagar um registro
6. Sair.

```
def select_operation() -> int | None:
    print("Menu:")
    print("1. Criar um registro")
    print("2. Consultar um registro pelo ID")
    print("3. Listar os registros")
    print("4. Modificar um registro")
    print("5. Apagar um registro")
    print("6. Sair")
    opcao = int(input("Escolha uma opção: "))
    if opcao not in [n for n in range(1,7)]:
      print("Opção invádlida - digite um número entre 1 e 6")
      return None

    return opcao

select_operation()
```

```
Menu:
```

```
1. Criar um registro
2. Consultar um registro pelo ID
3. Listar os registros
4. Modificar um registro
5. Apagar um registro
6. Sair
Escolha uma opção: 1
1
```

Questão 2

Desenvolva uma função que solicite o nome completo que será salvo no registro e devolva com as iniciais em maiúscula.

```
def input_name() -> str | None:
  import re

  nome = input("Digite o nome completo: ")
  if re.match(r'^[A-Za-zÀ-ÖØ-öø-ÿ\s\.\'-çÇ]+$', nome):
    return nome.title()

  print("Nome inválido")

input_name()
```

```
Digite o nome completo: alberto pluecker
'Alberto Pluecker'
```

Double-click (or enter) to edit

Questão 3

Desenvolva uma função que solicite a data de nascimento e devolva no formato dd-mm-aaaa, verificando se é uma data válida. Utilize como critério o valor de mês entre 1 e 12, e se o dia corresponde a um dia válido no mês. Para isso, garanta que casos com o ano bissexto ou dias 31 ocorram de forma correta.

```
def solicitar_data_nascimento():
  import datetime
  data_str = input("Digite a data de nascimento (dd-mm-aaaa): ")
  try:
    return datetime.datetime.strptime(data_str, "%d-%m-%Y").strftime("%d-%m-%Y")
  except:
```

```
  except:
    print("Data inválida. Tente novamente, no formato dd-mm-aaaa")
    return False
```

Questão 4

Desenvolva uma função que receba como parâmetro de entrada o dado CPF no formato XXXXXXXXXXX e verifique se esse é um valor válido, tendo como saída um valor booleano de Verdadeiro ou Falso.

```
def validar_cpf(cpf: str, check_formatting = False) -> bool:

    if check_formatting:
      import re
      if not re.match(r'\d{3}\.\d{3}\.\d{3}-\d{2}', cpf):
          return False

    numbers = [int(digit) for digit in cpf if digit.isdigit()]

    if len(numbers) != 11 or len(set(numbers)) == 1:
        return False

    sum_of_products = sum(a*b for a, b in zip(numbers[0:9], range(10, 1, -1)))
    if numbers[9] != ((sum_of_products * 10 % 11) % 10):
        return False

    sum_of_products = sum(a*b for a, b in zip(numbers[0:10], range(11, 1, -1)))
    if numbers[10] != ((sum_of_products * 10 % 11) % 10):
        return False

    return True

assert(validar_cpf("14528532743") == True)
assert(validar_cpf("99999999999") == False)
assert(validar_cpf("3") == False)
assert(validar_cpf("999999999991") == False)
```

Questão 5

Desenvolva uma função que solicite o CPF e retorne o valor no formato XXX.XXX.XXX-XX, caso seja um valor válido, ou Falso, caso seja inválido.

```
def formatar_cpf(cpf_str: str) -> str | bool:
    if validar_cpf(cpf_str):
      return f"{cpf_str[:3]}.{cpf_str[3:6]}.{cpf_str[6:9]}-{cpf_str[9:]}"
    return False

assert(formatar_cpf("14528532743") == "145.285.327-43")
assert(validar_cpf("999999999991") == False)
```

```
def get_cpf() -> str | bool:
  cpf = input("Digite o CPF: ")
  return formatar_cpf(cpf)

get_cpf()
```

```
Digite o CPF: 0
False
```

Questão 6

Desenvolva uma função que solicite ao usuário os dados cadastrais listados abaixo e que tenha como retorno uma lista contendo esses valores na ordem apresentada:

```
Nome completo
Data de nascimento
CPF
Endereço de email
Obs.: A função deve utilizar um laço que garanta que todos os valores sejam válidos e uti
```

```
def get_email() -> str | None:
  import re
  EMAIL_EXPR = re.compile("^[\w\-\.]+@([\w-]+\.)+[\w-]{2,}$")

  email = input("Digite seu email: ")
  if re.match(EMAIL_EXPR, email):
    return email

def solicitar_dados_cadastrais():
    while True:
        nome = input_name()
        data_nascimento = solicitar_data_nascimento()
        cpf = get_cpf()
        email = get_email()
```

```
        email = get_email()
        dados_validos = nome and data_nascimento and cpf and email
        if dados_validos:
            return [nome, data_nascimento, cpf, email]
        else:
            print("Todos os campos são obrigatórios. Tente novamente.")

# Exemplo de uso da função
dados_cadastrais = solicitar_dados_cadastrais()
print("Dados cadastrais inseridos:", dados_cadastrais)
```

```
Digite o nome completo: Alberto Pluecker
Digite a data de nascimento (dd-mm-aaaa): 28-10-1996
Digite o CPF: 14528532743
Digite seu email: alberto@mail.com
Dados cadastrais inseridos: ['Alberto Pluecker', '28-10-1996', '145.285.327-4
```

Double-click (or enter) to edit

Questão 7

Desenvolva uma função que receba como parâmetro um vetor de cadastro e um valor de ID e imprima o valor do registro desejado a partir da lista recebida por parâmetro.

```
def imprimir_registro_por_id(vetor_cadastro: list[str], id: int):
    if id >= 0 and id < len(vetor_cadastro):
        print("Registro encontrado:")
        print(vetor_cadastro[id])
        return vetor_cadastro[id]

    print("Registro não encontrado.")

# Exemplo de uso da função
cadastro = [
    ["João Silva", "01-01-1990", "12345678901", "joao@example.com"],
    ["Maria Oliveira", "15-05-1985", "98765432109", "maria@example.com"]
]

imprimir_registro_por_id(cadastro, 0)
imprimir_registro_por_id(cadastro, 1)
imprimir_registro_por_id(cadastro, 2)
```

```
Registro encontrado:
['João Silva', '01-01-1990', '12345678901', 'joao@example.com']
Registro encontrado:
['Maria Oliveira', '15-05-1985', '98765432109', 'maria@example.com']
Registro não encontrado
```

```
Registro não encontrado.
```

Questão 8

Desenvolva uma função que imprima, de forma ordenada, todos os registros existentes em uma lista passada por parâmetro.

```
def imprimir_registros_ordenados(lista_registros: list[str]):
    if lista_registros:
        print("Registros:")
        for i, registro in enumerate(sorted(lista_registros)):
            print(f"ID: {i}, Registro: {registro}")
    else:
        print("Nenhum registro encontrado.")

# Exemplo de uso da função
registros = [
    ["Maria Oliveira", "15-05-1985", "98765432109", "maria@example.com"],
    ["João Silva", "01-01-1990", "12345678901", "joao@example.com"]
]

imprimir_registros_ordenados(registros)
```

```
Registros:
ID: 0, Registro: ['João Silva', '01-01-1990', '12345678901', 'joao@example.co
ID: 1, Registro: ['Maria Oliveira', '15-05-1985', '98765432109', 'maria@examp
```

Questão 9

Desenvolva uma função que receba como parâmetros uma lista de registros e um número ID e realize a remoção desse registro.

```
def remover_registro_por_id(lista_registros: list[str], id: int):
    if id >= 0 and id < len(lista_registros):
        del lista_registros[id]
        print("Registro removido com sucesso.")
    else:
        print("ID de registro inválido.")

# Exemplo de uso da função
registros = [
    ["Maria Oliveira", "15-05-1985", "98765432109", "maria@example.com"],
    ["João Silva", "01-01-1990", "12345678901", "joao@example.com"]
```

```
]

print("Registros antes da remoção:")
imprimir_registros_ordenados(registros)

id_para_remover = 1
remover_registro_por_id(registros, id_para_remover)

print("\nRegistros após a remoção:")
imprimir_registros_ordenados(registros)
```

```
Registros antes da remoção:
Registros:
ID: 0, Registro: ['João Silva', '01-01-1990', '12345678901', 'joao@example.co
ID: 1, Registro: ['Maria Oliveira', '15-05-1985', '98765432109', 'maria@examp
Registro removido com sucesso.

Registros após a remoção:
Registros:
ID: 0, Registro: ['Maria Oliveira', '15-05-1985', '98765432109', 'maria@examp
```

Questão 10

Desenvolva uma função que solicite ao usuário qual registro deve ser modificado e qual dos valores deverão ser atualizados. O retorno deverá ser o ID do registro que será modificado e os respectivos valores dos campos. Sugestão: os valores que permanecerão inalterados podem ser definidos como None.

```
def solicitar_modificacao_registro(lista_registros: list[str]):
    while True:
        imprimir_registros_ordenados(lista_registros)
        id_registro = int(input("Digite o ID do registro que deseja modificar: ")
        if id_registro >= 0 and id_registro < len(lista_registros):
            registro = lista_registros[id_registro]
            print("Registro selecionado:", registro)
            campos_modificados = {}
            for i, campo in enumerate(registro):
                novo_valor = input(f"Digite um novo valor para '{campo}' ou press
                if novo_valor:
                    campos_modificados[i] = novo_valor
                else:
                    campos_modificados[i] = None
            return id_registro, campos_modificados
        else:
            print("ID de registro inválido. Tente novamente.")
```

```
# Exemplo de uso da função
registros = [
    ["Maria Oliveira", "15-05-1985", "98765432109", "maria@example.com"],
    ["João Silva", "01-01-1990", "12345678901", "joao@example.com"]
]

id_modificar, campos_modificados = solicitar_modificacao_registro(registros)
print("\nID do registro a ser modificado:", id_modificar)
print("Campos modificados:", campos_modificados)
```

```
Registros:
ID: 0, Registro: ['João Silva', '01-01-1990', '12345678901', 'joao@example.cor
ID: 1, Registro: ['Maria Oliveira', '15-05-1985', '98765432109', 'maria@examp
Digite o ID do registro que deseja modificar: 0
Registro selecionado: ['Maria Oliveira', '15-05-1985', '98765432109', 'maria@
Digite um novo valor para 'Maria Oliveira' ou pressione Enter para manter o va
Digite um novo valor para '15-05-1985' ou pressione Enter para manter o valor
Digite um novo valor para '98765432109' ou pressione Enter para manter o valo
Digite um novo valor para 'maria@example.com' ou pressione Enter para manter

ID do registro a ser modificado: 0
Campos modificados: {0: 'Ronaldo Gaúcho', 1: None, 2: None, 3: None}
```

Questão 11

Desenvolva uma função que receba um registro em formato de lista e modifique um ou mais valores dessa lista. Esse valor (ou valores) deverá ser passado por parâmetro para a função. O retorno deverá ser o registro com o valor modificado.

```
def modificar_registro(registro, **kwargs):
    for chave, valor in kwargs.items():
        if chave.isdigit():  # Check if the key is a string representation of an
            indice = int(chave)
            if 0 <= indice < len(registro):
                registro[indice] = valor
            else:
                print(f"Índice inválido: {indice}. O registro possui {len(registr
        else:
            print(f"Chave inválida: {chave}. Deve ser um número inteiro.")
    return registro

# Exemplo de uso da função
registro = ["Maria Oliveira", "15-05-1985", "98765432109", "maria@example.com"]
print("Registro antes da modificação:", registro)

registro_modificado = modificar_registro(registro, **{'0': "João Silva", '3': "jo
```

```
print("Registro modificado:", registro_modificado)
```

```
Registro antes da modificação: ['Maria Oliveira', '15-05-1985', '98765432109',
Registro modificado: ['João Silva', '15-05-1985', '98765432109', 'joao@exampl
```

Questão 12

Desenvolva uma função que modifique um registro dentro de uma lista de registros (banco de dados). Ela deve receber como parâmetros a lista com todos os registros (banco de dados), a posição que deverá ser modificada e uma lista com o novo valor do registro que deverá ser alterado. O retorno deve ser a lista de registros, com o valor modificado.

```
def modificar_registro_na_lista(banco_de_dados: list[list[str]], posicao: int, no
    if posicao >= 0 and posicao < len(banco_de_dados):
        banco_de_dados[posicao] = novo_registro
        return banco_de_dados
    else:
        print("Posição inválida.")
        return banco_de_dados

# Exemplo de uso da função
banco_de_dados = [
    ["Maria Oliveira", "15-05-1985", "98765432109", "maria@example.com"],
    ["JPablo Marçal", "01-01-1990", "12345678901", "joao@example.com"]
]

novo_registro = ["Geraldo Suarez", "20-10-1978", "87654321098", "jose@example.com
posicao_modificar = 1

banco_de_dados_modificado = modificar_registro_na_lista(banco_de_dados, posicao_m
print("Banco de dados modificado:", banco_de_dados_modificado)
```

```
Banco de dados modificado: [['Maria Oliveira', '15-05-1985', '98765432109', '
```

Questão 13

Desenvolva um script que seja capaz de realizar as seguintes ações:

```
Inicialize um vetor para salvar vetores de registros.
Exiba uma lista de opções para o usuário, utilizando a função desenvolvida na questão 1.
Execute o programa de forma indefinida, até que a opção de sair seja escolhida.
```

```
# Importando as funções previamente desenvolvidas
from datetime import datetime

def main():
    registros = []
    while True:
        opt = select_operation()
        match str(opt):
          case "1":
              registros.append(solicitar_dados_cadastrais())
          case "2":
              id = int(input("Digite o ID do registro que deseja consultar: "))
              imprimir_registro_por_id(registros, id)
          case "3":
              imprimir_registros_ordenados(registros)
          case "4":
              modificar_registro(registros)
          case "5":
              id = int(input("Digite o ID do registro que deseja apagar: "))
              remover_registro_por_id(registros, id)
          case "6":
              print("Encerrando o programa.")
              break

if __name__ == "__main__":
    main()
```

```
Menu:
1. Criar um registro
2. Consultar um registro pelo ID
3. Listar os registros
4. Modificar um registro
5. Apagar um registro
6. Sair
Escolha uma opção: 6
Encerrando o programa.
```