SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL
CENTRO DE FORMAÇÃO PROFISSIONAL FIDÉLIS REIS
CURSO DE DESENVOLVIMENTO DE SISTEMAS

TESTE DE SISTEMAS

RODRIGO BORGES

EXERCÍCIOS DE TESTE DE SISTEMAS

UBERABA - MG
2023

SUMÁRIO

1. INTRODUÇÃO

A atividade envolve a criação de um programa que simula operações bancárias, como depósito, saque e verificação de saldo. Os exercícios são divididos em três níveis: fácil, médio e difícil, com cada nível contendo vários cenários de teste.

2. DESENVOLVIMENTO

2.1. Exercício 1: Verificação de Número Primo

Neste exercício, o objetivo é desenvolver um programa que determine se um número fornecido pelo usuário é primo, sendo este aquele que é divisível apenas por 1 e por ele mesmo. O programa deve verificar se o número informado atende a essa condição.

Exemplo do exercício:

```
'''
Exercício 1: Verificação de Número Primo
Descrição: Escreva um programa que verifica se um número fornecido pelo usuário é primo ou não.
Teste de Sistema:

Cenário 1: Número Primo
Entrada: Número: 17
Saída Esperada: Resultado: O número 17 é primo.

Cenário 2: Número Não Primo
Entrada: Número: 10
Saída Esperada: Resultado: O número 10 não é primo.
'''

# CÓDIGO LIMPO

def is_prime(number):
    if number <= 1:
        return False
```

```
    if number <= 3:
        return True
    if number % 2 == 0 or number % 3 == 0:
        return False
    i = 5
    while i * i <= number:
        if number % i == 0 or number % (i + 2) == 0:
            return False
        i += 6
    return True

try:
    num = int(input("Digite um número inteiro positivo: "))
    if is_prime(num):
        print(f"Resultado: O número {num} É primo.")
    else:
        print(f"Resultado: O número {num} NÃO é primo.")
except ValueError:
    print("[ERRO] Por favor, insira um número inteiro válido.")

# CÓDIGO COMENTADO

# Função para verificar se um número é primo
def is_prime(number): # Esta linha define uma função chamada is_prime que recebe um número como argumento.
    if number <= 1: # Verifica se o número é menor ou igual a 1.
        return False # Se for, a função retorna False (F), pois números menores ou iguais a 1 não são primos.
    if number <= 3: # Esta condição verifica se o número é menor ou igual a 3.
        return True #  Se for, a função retorna True (V), pois os números 2 e 3 são primos.
    if number % 2 == 0 or number % 3 == 0: # Verifica se o número é divisível por 2 ou por 3 (ou seja, se o resto da divisão por 2 ou por 3 é igual a zero).
        return False # Se qualquer uma dessas condições for verdadeira, a função retorna False, pois isso indica que o número não é primo.
    i = 5 # Uma variável i é inicializada com o valor 5. Esta variável será usada em um loop para verificar se o número é divisível por outros números maiores do que 3.
```

```
    while i * i <= number: # Este loop continuará enquanto o valor de i multiplicado por ele mesmo for < ou = ao número que estamos verificando. Isso é feito para otimizar o processo de verificação de primos.
        if number % i == 0 or number % (i + 2) == 0: # Dentro do loop, este verifica se o número é divisível por i ou "i + 2". A verificação começa com i = 5 e i + 2 = 7, e depois se move para valores subsequentes de i e "i + 2".
            return False # Se o número for divisível por qualquer um deles, a função retorna False (F), indicando que o número não é primo.
        i += 6 # Após cada iteração do loop, i é incrementado em 6. Isso é feito para verificar se o número é divisível por números que não são múltiplos de 2 ou 3, economizando assim verificações desnecessárias.
    return True # Se o loop terminar sem encontrar nenhum divisor para o número, a função retorna True (V), indicando que o número é primo.

# Solicitar entrada do usuário
try: # Estrutura de controle "bloco try-except", para definir um bloco de código onde exceções (erros) podem ocorre
    num = int(input("Digite um número inteiro positivo: ")) # O programa tenta obter um número inteiro positivo da entrada do usuário e o armazena na variável num.
    if is_prime(num): # Verifica se o número num é primo usando a função is_prime que foi definida anteriormente.
        print(f"Resultado: O número {num} É primo.") # Se for primo, esta imprime indicando que o número É primo.
    else: # Se o número não for primo, o código entra no bloco else.
        print(f"Resultado: O número {num} NÃO é primo.") # Imprime uma mensagem indicando que o número NÃO é primo.
except ValueError: # Este bloco except lida com exceções do tipo ValueError, que podem ocorrer se o usuário inserir algo que não possa ser convertido em um número inteiro válido.
    print("[ERRO] Por favor, insira um número inteiro válido.") # Se uma exceção ValueError for lançada (por exemplo, se o usuário inserir texto em vez de um número), esta linha imprimirá uma mensagem de erro solicitando que o usuário insira um número inteiro válido.
```

2.2. Exercício 2: Contagem de Vogais

Neste exercício, deve haver a criação de um programa que conta quantas vogais estão presentes em uma frase fornecida pelo usuário, sendo as vogais as letras "a", "e", "i",

"o" e "u", em letras minúsculas e maiúsculas. O programa deve fazer a contagem e retornar o resultado.

Exemplo do exercício:

```
'''
Exercício 2: Contagem de Vogais
Descrição: Crie um programa que conta o número de vogais em uma frase fornecida pelo usuário.
Teste de Sistema:

Cenário 1: Frase com Vogais
Entrada: Frase: "Olá, mundo!"
Saída Esperada: Resultado: A frase possui 5 vogais.

Cenário 2: Frase sem Vogais
Entrada: Frase: "Nenhuma vogal aqui"
Saída Esperada: Resultado: A frase não possui vogais.
'''

# CÓDIGO LIMPO

def contar_vogais(frase):
    vogais = "AEIOUaeiou"
    count = 0
    for letra in frase:
        if letra in vogais:
            count += 1
    return count

frase = input("Digite uma frase: ")
numero_de_vogais = contar_vogais(frase)

if numero_de_vogais > 0:
    print(f"Resultado: A frase possui {numero_de_vogais} vogais.")
else:
    print("Resultado: A frase não possui vogais.")
```

```
# CÓDIGO COMENTADO

def contar_vogais(frase): # Definindo uma função (responsável por contar o número de vogais na frase fornecida) chamada contar_vogais que aceita uma frase como argumento.
    vogais = "AEIOUaeiou" # Uma string que contém todas as vogais maiúsculas e minúsculas.
    count = 0 # Inicializa uma variável chamada "count" para contar o número de vogais na frase.
    for letra in frase: # Inicia um loop que percorre cada letra na frase.
        if letra in vogais: # Verifica se a letra atual está na lista de vogais.
            count += 1 # Se a letra for uma vogal, incrementa o contador "count" em 1.
    return count # Retorna o valor final do contador "count" como resultado da função.

frase = input("Digite uma frase: ") # Solicita ao usuário que insira uma frase e armazena-a na variável "frase".
numero_de_vogais = contar_vogais(frase) # Chama função "contar_vogais" que está acima, com a frase inserida pelo usuário e armazena o resultado em "numero_de_vogais".
if numero_de_vogais > 0: # Verifica se o número de vogais é > do que zero (ou seja, se há pelo menos uma vogal na frase).
    print(f"Resultado: A frase possui {numero_de_vogais} vogais.") # Se houver vogais, imprime quantas vogais foram encontradas.
else:
    print("Resultado: A frase não possui vogais.") # Se não houver vogais, imprime que NÃO há vogais.
```

2.3. Exercício 3: Soma dos Números Pares

O desafio é escrever um programa que calcule a soma dos números pares em uma lista de números fornecida pelo usuário. O programa deve identificar quais números na lista são pares e somá-los para fornecer o resultado.

Exemplo do exercício:

```
'''
Exercício 3: Soma dos Números Pares
```

```
Descrição: Escreva um programa que calcula a soma dos números pares em uma lista fornecida pelo usuário.
Teste de Sistema:

Cenário 1: Lista com Números Pares
Entrada: Lista: [2, 4, 6, 8, 10]
Saída Esperada: Resultado: A soma dos números pares é 30.

Cenário 2: Lista sem Números Pares
Entrada: Lista: [1, 3, 5, 7, 9]
Saída Esperada: Resultado: A lista não contém números pares.
'''

# Solicita ao usuário que insira a lista de números, separados por vírgulas e entre colchetes.
entrada = input("Digite uma lista de números (ex: [2, 4, 6, 8, 10]): ")

# Tenta converter a entrada em uma lista de números
try:
    lista = eval(entrada)
except NameError:
    lista = []

# Inicializa uma variável para a soma dos números pares
soma_pares = 0

# Percorre a lista e adiciona os números pares à soma
for numero in lista:
    if numero % 2 == 0:
        soma_pares += numero

# Verifica se a soma_pares é maior que zero para determinar se a lista contém números pares
if soma_pares > 0:
    print(f"Resultado: A soma dos números pares é {soma_pares}.")
else:
    print("Resultado: A lista não contém números pares.")
```

2.4. Exercício 4: Contagem de Palavras em uma Frase

Neste exercício, deve ser criado um programa que conta o número de palavras em uma frase fornecida pelo usuário, sendo a palavra uma sequência de caracteres separados por espaços. O programa deve contar essas sequências e apresentar o total.

Exemplo do exercício:

```
'''
Exercício 4: Contagem de Palavras em uma Frase
Descrição: Crie um programa que conta o número de palavras em uma frase fornecida pelo usuário.
Teste de Sistema:

Cenário 1: Frase com Palavras
Entrada: Frase: "O rato roeu a roupa do rei de Roma"
Saída Esperada: Resultado: A frase possui 8 palavras.

Cenário 2: Frase sem Palavras
Entrada: Frase: colocar nada, deixra vazio
Saída Esperada: Resultado: A frase não possui palavras.
'''

# CÓDIGO LIMPO

frase = input("Digite uma frase: ") # Solicita ao usuário que insira a frase
frase = frase.strip() # Remove espaços em branco extras no início e no final da frase
if not frase: # Verifica se a frase está vazia
    print("Resultado: A frase não possui palavras.")
else:
    # Divide a frase em palavras usando espaços em branco como delimitadores
    palavras = frase.split()
    numero_de_palavras = len(palavras)
    print(f"Resultado: A frase possui {numero_de_palavras} palavras.")
```

2.5. Exercício 5: Média de uma Lista de Números

Neste exercício, o objetivo é escrever um programa que calcule a média dos números em uma lista fornecida pelo usuário. O programa deve somar todos os números na lista e dividir pelo total de números para calcular a média.

Exemplo do exercício:

```
'''
Exercício 5: Média de uma Lista de Números
Descrição: Escreva um programa que calcula a média de uma lista de números fornecida pelo usuário.
Teste de Sistema:

Cenário 1: Lista com Números
Entrada: Lista: [5, 10, 15, 20]
Saída Esperada: Resultado: A média dos números é 12.5.

Cenário 2: Lista Vazia
Entrada: Lista: []
Saída Esperada: Resultado: A lista está vazia, não é possível calcular a média.
'''

entrada = input("Digite uma lista de números (ex: [5, 10, 15, 20]): ") # Solicita ao usuário que insira a lista de números, separados por vírgulas e entre colchetes.
# Tenta converter a entrada em uma lista de números
try:
    lista = eval(entrada)
except NameError:
    lista = []
# Verifica se a lista está vazia
if not lista:
    print("Resultado: A lista está vazia, não é possível calcular a média.")
else:
    # Calcula a média dos números na lista
    media = sum(lista) / len(lista)
    print(f"Resultado: A média dos números é {media}.")
```

2.6. Exercício 6: Verificação de Palíndromo

Neste exercício, o objetivo é criar um programa que determina se uma palavra fornecida pelo usuário é um palíndromo (palavra ou frase que é lida da mesma forma de trás para frente). O programa deve verificar se a palavra atende a esse critério.

Exemplo do exercício:

```
'''
Exercício 6: Verificação de Palíndromo
Descrição: Crie um programa que verifica se uma palavra fornecida pelo usuário é um palíndromo (ou seja, lida da mesma forma de trás para frente).
Teste de Sistema:

Cenário 1: Palavra Palíndroma
Entrada: Palavra: "radar"
Saída Esperada: Resultado: A palavra "radar" é um palíndromo.

Cenário 2: Palavra Não Palíndroma
Entrada: Palavra: "hello"
Saída Esperada: Resultado: A palavra "hello" não é um palíndromo.
'''
palavra = input("Digite uma palavra: ") # Solicita ao usuário que insira uma palavra
palavra = palavra.strip() # Remove espaços em branco extras no início e no final da palavra
palavra_invertida = palavra[::-1] # Inverte a palavra
# Verifica se a palavra é um palíndromo
if palavra == palavra_invertida:
    print(f"Resultado: A palavra \"{palavra}\" é um palíndromo.")
else:
    print(f"Resultado: A palavra \"{palavra}\" não é um palíndromo.")
```

2.7. Exercício 7: Busca Binária

Neste exercício, deve ser implementado um programa que realiza uma busca binária em uma lista de números inteiros. A busca binária é um método eficiente para

encontrar um número em uma lista ordenada, e o programa deve indicar se o número foi encontrado e em qual posição.

Exemplo do exercício:

```
'''
Exercício 7: Busca Binária
Descrição: Implemente um programa que realiza uma busca binária em uma lista ordenada de números inteiros.
Teste de Sistema:

Cenário 1: Número Encontrado
Entrada: Lista: [1, 3, 5, 7, 9, 11, 13, 15]
Buscar: 7
Saída Esperada: Resultado: O número 7 foi encontrado na posição 3.

Cenário 2: Número Não Encontrado
Entrada:
Lista: [2, 4, 6, 8, 10]
Buscar: 3
Saída Esperada: Resultado: O número 3 não foi encontrado na lista.
'''

def busca_binaria(lista, elemento):
    esquerda, direita = 0, len(lista) - 1

    while esquerda <= direita:
        meio = (esquerda + direita) // 2
        if lista[meio] == elemento:
            return meio
        elif lista[meio] < elemento:
            esquerda = meio + 1
        else:
            direita = meio - 1

    return -1
```

```
# Solicita ao usuário que insira a lista ordenada de números
entrada_lista = input("Digite a lista ordenada de números (ex: 1, 3, 5, 7, 9, 11, 13, 15): ")

# Tenta converter a entrada em uma lista de números
try:
    lista = [int(x) for x in entrada_lista.split(',')]
except ValueError:
    lista = []

elemento = int(input("Digite o número a ser buscado: ")) # Solicita ao usuário que insira o número a ser buscado

posicao = busca_binaria(lista, elemento) # Realiza a busca binária

# Exibe o resultado
if posicao != -1:
    print(f"Resultado: O número {elemento} foi encontrado na posição {posicao}.")
else:
    print(f"Resultado: O número {elemento} não foi encontrado na lista.")
```

2.8. Exercício 8: Ordenação de Lista

O desafio deste exercício é criar um programa que ordene uma lista de números inteiros em ordem crescente. O programa deve reorganizar os elementos da lista de forma que fiquem em ordem ascendente.

Exemplo do exercício:

```
'''
Exercício 8: Ordenação de Lista
Descrição: Crie um programa que ordena uma lista de números inteiros em ordem crescente.
Teste de Sistema:

Cenário 1: Lista Desordenada
Entrada: Lista: [9, 2, 5, 1, 7, 3]
Saída Esperada: Resultado: Lista ordenada: [1, 2, 3, 5, 7, 9]
```

```
Cenário 2: Lista Já Ordenada
Entrada: Lista: [1, 2, 3, 4, 5]
Saída Esperada: Resultado: Lista já está ordenada: [1, 2, 3, 4, 5]
'''

# Solicita ao usuário que insira a lista de números
entrada_lista = input("Digite a lista de números (ex: 9, 2, 5, 1, 7, 3): ")

# Tenta converter a entrada em uma lista de números
try:
    lista = [int(x) for x in entrada_lista.split(',')]
except ValueError:
    lista = []

# Verifica se a lista está vazia
if not lista:
    print("Resultado: A lista está vazia.")
else:
    # Ordena a lista em ordem crescente
    lista_ordenada = sorted(lista)

    # Verifica se a lista já está ordenada
    if lista == lista_ordenada:
        print(f"Resultado: Lista já está ordenada: {lista}")
    else:
        print(f"Resultado: Lista ordenada: {lista_ordenada}")
```

2.9. Exercício 9: Calculadora de Fatorial

Neste exercício, o programa calcula o número fatorial (o fatorial de um número é o produto de todos os inteiros positivos de 1 até esse número) de um número fornecido pelo usuário. O programa deve calcular e retornar o resultado.

Exemplo do exercício:

```
'''
```

```
Exercício 9: Calculadora de Fatorial
Descrição: Escreva um programa que calcula o fatorial de um número fornecido pelo usuário.
Teste de Sistema:

Cenário 1: Fatorial de 5
Entrada: Calcular o fatorial de 5
Saída Esperada: Resultado: O fatorial de 5 é 120.

Cenário 2: Fatorial de 0
Entrada: Calcular o fatorial de 0
Saída Esperada: Resultado: O fatorial de 0 é 1.
'''

# Solicita ao usuário que insira o número para o qual deseja calcular o fatorial
numero = int(input("Calcular o fatorial de: "))

# Verifica se o número é negativo
if numero < 0:
    print("Resultado: O fatorial de um número negativo não é definido.")
else:
    fatorial = 1
    for i in range(1, numero + 1):
        fatorial *= i
    print(f"Resultado: O fatorial de {numero} é {fatorial}.")
```

2.10. Exercício 10: Validador de Senha

Desenvolvimento de um programa que verifica se uma senha atende a critérios específicos, como ter pelo menos 8 caracteres, conter pelo menos uma letra maiúscula, uma letra minúscula, pelo menos um número e conter pelo menos um caractere especial (implementado pelo autor). O programa deve determinar se a senha é válida ou não.

Exemplo do exercício:

```
'''
Exercício 10: Validador de Senha
```

```
Descrição: Crie um programa que verifica se uma senha atende aos seguintes critérios:
Pelo menos 8 caracteres de comprimento.
Contém pelo menos uma letra maiúscula e uma letra minúscula.
Contém pelo menos um caractere especial.
Pelo menos um número.

Teste de Sistema:

Cenário 1: Senha Válida
Entrada: Senha: Abcdefg1
Saída Esperada: Resultado: A senha é válida.

Cenário 2: Senha Inválida (Falta de Número)
Entrada: Senha: Abcdefgh
Saída Esperada: Resultado: A senha deve conter pelo menos um número.
'''

import re

# Solicita ao usuário que insira a senha
senha = input("Digite a senha: ")

# Verifica o comprimento da senha
if len(senha) < 8:
    print("Resultado: A senha deve ter pelo menos 8 caracteres.")
else:
    # Verifica se a senha contém pelo menos uma letra maiúscula e uma letra minúscula
    if not (any(c.islower() for c in senha) and any(c.isupper() for c in senha)):
            print("Resultado: A senha deve conter pelo menos uma letra maiúscula e uma letra
minúscula.")
    # Verifica se a senha contém pelo menos um caractere especial
    elif not re.search(r'[!@#$%^&*()_+{}[\]:;<>,.?~\\]', senha):
        print("Resultado: A senha deve conter pelo menos um caractere especial.")
    # Verifica se a senha contém pelo menos um número
    elif not any(c.isdigit() for c in senha):
        print("Resultado: A senha deve conter pelo menos um número.")
    else:
```

```
    print("Resultado: A senha é válida.")
```

2.11. Exercício 11: Jogo da Forca

Neste exercício, deve ser implementado o clássico jogo da forca. O computador escolherá uma palavra aleatória, e o jogador tentará adivinhar as letras que compõem a palavra. O jogador tem um número limitado de tentativas para adivinhar a palavra.

```
'''
Exercício 11: Jogo da Forca
Descrição: Implemente o jogo da forca em que o computador escolhe uma palavra aleatória e o jogador tenta adivinhar as letras.
Teste de Sistema:

Cenário 1: Palavra Adivinhada Corretamente
Entrada:
Palavra escolhida pelo computador: "elefante"
Letras escolhidas pelo jogador: e, l, f, a, n, t
Saída Esperada: Resultado: Parabéns, você adivinhou a palavra!

Cenário 2: Tentativas Esgotadas
Entrada:
Palavra escolhida pelo computador: "banana"
Letras escolhidas pelo jogador: x, y, z, m, n, b
Saída Esperada: Resultado: Você esgotou suas tentativas. A palavra era "banana".
'''

import random

# Lista de palavras para o jogo
palavras = ["elefante", "banana", "computador", "python", "programacao", "jogo", "forca"]

# Escolhe uma palavra aleatória da lista
palavra = random.choice(palavras)

# Converte a palavra em minúsculas para facilitar a comparação
palavra = palavra.lower()
```

```
# Número máximo de tentativas
max_tentativas = 6

# Inicializa a lista de letras adivinhadas
letras_adivinhadas = ["_"] * len(palavra)

# Inicializa a lista de letras erradas
letras_erradas = []

# Inicializa o contador de tentativas
tentativas = 0

# Loop principal do jogo
while True:
    # Exibe a palavra oculta com as letras adivinhadas
    print("Palavra: " + " ".join(letras_adivinhadas))

    # Exibe as letras erradas
    if letras_erradas:
        print("Letras erradas: " + " ".join(letras_erradas))

    # Solicita ao jogador que adivinhe uma letra
    letra = input("Adivinhe uma letra: ").lower()

    # Verifica se a letra é válida (uma letra minúscula)
    if not letra.isalpha() or len(letra) != 1:
        print("Por favor, insira uma letra válida.")
        continue

    # Verifica se a letra já foi adivinhada
    if letra in letras_adivinhadas or letra in letras_erradas:
        print("Você já adivinhou essa letra.")
        continue

    # Verifica se a letra está na palavra
    if letra in palavra:
```

```
        # Atualiza a lista de letras adivinhadas
        for i in range(len(palavra)):
            if palavra[i] == letra:
                letras_adivinhadas[i] = letra
    else:
        # A letra não está na palavra
        letras_erradas.append(letra)
        tentativas += 1

    # Verifica se o jogador ganhou
    if "_" not in letras_adivinhadas:
        print("Resultado: Parabéns, você adivinhou a palavra!")
        break

    # Verifica se o jogador esgotou suas tentativas
    if tentativas == max_tentativas:
        print(f"Resultado: Você esgotou suas tentativas. A palavra era '{palavra}'.")
        break
```

2.12. Exercício 12: Simulador de Banco

A tarefa consiste em criar um programa que simula as operações de um banco, incluindo depósito, saque e verificação de saldo. O programa deve permitir ao usuário realizar essas operações e atualizar o saldo da conta conforme as transações são feitas.

Exemplo do exercício:

```
'''
Exercício 12: Simulador de Banco
Descrição: Crie um programa que simule as operações de um banco, como depósito, saque e verificação de saldo.
Teste de Sistema:

Cenário 1: Depósito e Saque Bem Sucedidos
Entrada:
Saldo inicial: R$ 500
Depósito: R$ 200
```

```
Saque: R$ 100
Saída Esperada: Resultado: Saldo atual: R$ 600

Cenário 2: Tentativa de Saque com Saldo Insuficiente
Entrada:
Saldo inicial: R$ 100
Saque: R$ 200
Saída Esperada: Resultado: Saldo insuficiente para realizar o saque.
'''

print("Cenário 1: Depósito e Saque Bem Sucedidos")

class Banco:
    def __init__(self, saldo_inicial):
        self.saldo = saldo_inicial

    def deposito(self, valor):
        if valor > 0:
            self.saldo += valor
            return True
        else:
            return False

    def saque(self, valor):
        if valor > 0 and valor <= self.saldo:
            self.saldo -= valor
            return True
        else:
            return False

    def verificar_saldo(self):
        return self.saldo

# Solicita o saldo inicial do usuário
saldo_inicial = float(input("Saldo inicial: R$ "))
banco = Banco(saldo_inicial)
```

```
# Solicita o valor do depósito do usuário
valor_deposito = float(input("Depósito: R$ "))
if banco.deposito(valor_deposito):
    print("Depósito bem-sucedido.")
else:
    print("Depósito mal-sucedido.")

# Solicita o valor do saque do usuário
valor_saque = float(input("Saque: R$ "))
if banco.saque(valor_saque):
    print("Saque bem-sucedido.")
else:
    print("Saque mal-sucedido.")

# Verifica o saldo atual
saldo_atual = banco.verificar_saldo()
print(f"Saldo atual: R$ {saldo_atual}")

print("\nCenário 2: Tentativa de Saque com Saldo Insuficiente")

class Banco:
    def __init__(self, saldo_inicial):
        self.saldo = saldo_inicial

    def deposito(self, valor):
        if valor > 0:
            self.saldo += valor
            return True
        else:
            return False

    def saque(self, valor):
        if valor > 0 and valor <= self.saldo:
            self.saldo -= valor
            return True
        else:
            return False
```

```
    def verificar_saldo(self):
        return self.saldo

# Solicita o saldo inicial do usuário
saldo_inicial = float(input("Saldo inicial: R$ "))
banco = Banco(saldo_inicial)

# Solicita o valor do saque do usuário
valor_saque = float(input("Saque: R$ "))
if banco.saque(valor_saque):
    print("Saque bem-sucedido.")
else:
    print("Resultado: Saldo insuficiente para realizar o saque.")

# Verifica o saldo atual
saldo_atual = banco.verificar_saldo()
print(f"Saldo atual: R$ {saldo_atual}")
```

3. CONCLUSÃO

Este relatório destaca os principais achados e resultados das atividades realizadas, sendo possível analisar que os dados coletados e as análises realizadas oferecem “insights” valiosos que podem ser usados para o uso da linguagem Python na busca de resultados e tomada de decisões. Diante dos resultados apresentados, podemos concluir que é recomendado o uso da linguagem, servindo este relatório como referência e guia para uso futuro.