UNIVERSIDADE PAULISTA – UNIP EaD Flex

RELATÓRIO DE AULA PRÁTICA

CURSO: Análise e desenvolvimento de sistemas

DISCIPLINA: Pensamento lógico computacional com python

ALUNO(A): Vitória Freitas

R.A: 2523279 POLO: Vila Dirce

DATA: 27 / 05 / 2025.

ROTEIRO DE AULA PRÁTICA – AULA 01: INTRODUÇÃO À LÓGICA DE PROGRAMAÇÃO E PYTHON

O QUE É LÓGICA DE PROGRAMAÇÃO?

Primeiro vamos entender o que é lógica, lógica é a ciência que estuda as formas e leis do pensamento humano. Já a lógica de programação vai nos ajudar a distinguir raciocínios válidos dos inválidos, ela entrega instruções que conduzem à solução de um problema. Para solucionar esse problema, podemos dividir ele em partes menores e mais manejáveis, pensando em tudo que pode dar errado e tudo que pode dar certo.

COMO EU ESCREVO UM CÓDIGO? E QUE SÃO FLUXOGRAMA?

Podemos começar escrevendo passo a passo, vendo todas as alternativas. Para facilitar a construção de um código podemos usar fluxograma(são representações gráficas que utilizam símbolos padronizados para mapear o fluxo de execução de algoritmos).

O QUE É UM ALGORITMO?

Algoritmo é um pequeno trecho de código para resolver um problema, podemos considerar que algoritmo é uma rotina. E quando se tem um conjunto de rotinas, chamamos de biblioteca. Quando estamos escrevendo um código temos que utilizar os recursos de forma otimizada, mas antes de otimizar o código é importante primeiro resolver o problema que nos foi passado. E também devemos ser compreensíveis com outros programadores que irão ler o código depois.

O QUE É UM PSEUDOCÓDIGO?

É uma linguagem textual, que pode ser considerada um passo anterior a codificação para se entender a lógica de algoritmos, com um pseudocódigo você consegue traduzir para qualquer linguagem de programação.

O QUE É UMA VARIÁVEL E UM ARRAY?

Variável é um lugar reservado na memória do computador e um array é uma variável com várias posições, chamamos de index, os array começam no index 0(menos java que começa com 1).

CÓDIGO DE EXEMPLO MOSTRADO PELO PROFESSOR:

```
lista = [4, 8, 6, 10, 2]
num = 0
soma = 0
while num <= 4:
    soma = soma + lista[num] #adição do elemento da lista
    num += 1 # Incremento correto de num
print("Soma:", soma)
```

COMO MOSTRAR UMA MENSAGEM NA TELA EM PYTHON:

Versão mais recente:

print(f"Olá, {nome}")

Versão mais antiga:

print("Olá, {}".format(nome))

COMO COLOCAR COMENTÁRIO NO CÓDIGO:

Para comentário de linha usamos: #, para comentário de linha usamos: " ".

EXEMPLO DE CÓDIGO DE COMO TRANSFORMAR STRING EM NUMBER:

```
num1 = int(input('valor: '))
num2 = int(input('valor: '))
soma = num1 + num2
print(f'soma: {soma}')
```

VANTAGENS DE SE UTILIZAR PYTHON PARA APRENDER PROGRAMAÇÃO:

Python é uma linguagem de programação considerada fácil de se aprender quando comparada com outras linguagens, pois ela tem uma sintaxe simples e bem intuitiva. Além do fato de ter muito conteúdo disponível para o aprendizado dela na internet.

DESAFIO PASSADO EM SALA DE AULA:

```python
nome = input("Qual é o seu nome: ")
ano = int(input("Qual ano você vem: "))
print(f"Seu nome é {nome}")
print(f"E você veio desse ano: {ano}")
```

ROTEIRO DE AULA PRÁTICA – AULA 02: ESTRUTURAS DE CONTROLE EM PYTHON

IF, ELSE E ELIF:

Fazem parte da estrutura de controle na categoria estruturas de decisão, em python que nos permite gerenciar o fluxo de execução de um programa. Com elas o código é executado de maneira condicional, dependendo das escolhas feitas pelo usuário. Vou mostrar os códigos de exemplo que foi mostrado pelo professor:

```
x = int(input("Digite um número: ")) # Solicita um número ao usuário
if x > 0:
 print("x é positivo")
elif x < 0:
 print("x é negativo")
else:
 print("x é zero")
```

Assim como mostrado no código acima, dependendo do número escolhido pelo usuário, vai mostrar uma mensagem diferente na tela.

FOR, WHILE:

Se refere às estruturas de repetição, em python usamos o for ou while. São ferramentas importantes para a execução iterativa de blocos de código. O for é ideal quando o número de interações é delimitado, ótimo para percorrer por listas ou sequências. No while, a condição lógica é avaliada antes de cada interação, ideal quando o número de repetições é desconhecido. Exemplo de código apresentado pelo professor:

```
# Verificação de permissão temporal
while True:
    try:
        idade = int(input("Qual é a sua idade? "))
        if idade >= 0:
            break  # Sai do while
        else:
            print("Por favor, insira uma idade válida.")
```

```
    except ValueError:
        print("Por favor, insira um número válido.")

if idade < 18:
    print("Acesso negado.")
    exit()  # Encerra o programa se a idade for menor que 18
else:
    print("Acesso permitido.")

# Pergunta quantas viagens no tempo deseja fazer
while True:
    try:
        viagens = int(input("Quantas vezes você deseja viajar no
tempo? "))
        if viagens > 0:
            break
        else:
            print("Por favor, insira um número válido de viagens.")
    except ValueError:
        print("Por favor, insira um número inteiro.")

# Lista para armazenar as viagens
registro_viagens = []

# Solicitação dos detalhes das viagens
for i in range(1, viagens + 1):
    ano = input(f"Digite o ano alvo da viagem {i}: ")

    while True:
        direcao = input("Será para o 'passado' ou 'futuro'?
").strip().lower()
        if direcao in ["passado", "futuro"]:
            break
        else:
            print("Opção inválida! Digite 'passado' ou 'futuro'.")

    registro_viagens.append((ano, direcao))  # Adicionar a viagem à
lista

# Lista de eventos históricos
eventos_historicos = []
```

```python
# Menu de opções para eventos históricos
while True:
    print("\nMenu de Opções:")
    print("1 - Registrar evento histórico")
    print("2 - Mostrar lista de eventos")
    print("0 - Sair")

    opcao = input("Escolha uma opção: ")

    if opcao == "1":
        evento = input("Descreva o evento histórico: ")
        eventos_historicos.append(evento)
        print("Evento registrado com sucesso!")

    elif opcao == "2":
        if eventos_historicos:
            print("\nLista de eventos históricos registrados:")
            for idx, evento in enumerate(eventos_historicos, start=1):
                print(f"{idx}. {evento}")
        else:
            print("Nenhum evento registrado ainda.")

    elif opcao == "0":
        print("Saindo do programa...")
        break

    else:
        print("Opção inválida! Tente novamente.")

# Exibição do resumo das viagens
nome = input("\nInforme seu nome para o resumo: ")
print(f"\nOlá, {nome}! Aqui está o resumo das suas viagens no tempo:")
for i, (ano, direcao) in enumerate(registro_viagens, start=1):
    print(f"Viagem {i}: {direcao.capitalize()} para o ano {ano}.")
```

IMPORTÂNCIA DA LÓGICA BOOLEANA, OPERADORES E PRECEDÊNCIA.

A lógica booleana desenvolvida por George Boole, permite que os nossos computadores processem informações no formato binário(0 e 1), sem ela não teríamos os computadores modernos. Os operadores em python são: or(ou), and (e) e not(não). Or e and retornam um valor lógico true ou false, o not retorna o oposto do valor que foi dado. A precedência é de extrema importância, pois ela determina a ordem em que os operadores são avaliados. O or tem a menor precedência, depois o and e o com a maior é o not.

USO DE BREAK/CONTINUE.

Vou mostrar um exemplo do uso do break e continue em um código simples.

```
numeros = [1, 2, 3]
for numero in numeros:
    print(f"Número: {numero}")
    if numero == 1 or numero == 2:
        continue
    if numero == 3:
        break

print("Acabou")
```

DESAFIO A PASSADO EM SALA DE AULA:

```
viagens = int(input("Quantas viagens você irá fazer: "))
viagens_cadastradas = 0

for i in range(1, viagens + 1):
    passado_futuro = input("Será passado ou futuro: ")
    viagens_cadastradas += 1

print(f"Número de viagens: {viagens_cadastradas}")
```

DESAFIO B PASSADO EM SALA DE AULA:

```
#Enquanto a idade for menor que 0 vai continuar no loop
while True:
    idade = int(input("Sua idade: "))
    if idade >= 0:
        print("Acesso permitido!")
        break
    else:
        print("Idade inválida, tente novamente.")
```

DESAFIO OPCIONAL PASSADO EM SALA DE AULA:

```
#Irá sair do loop quando o usuário apertar 0.
while True:
    print("Opções: ")
    opcoes = int(input("1 - Registrar evento histórico 2 - Mostrar
lista de eventos, 0 - Sair"))
    if opcoes == 0:
        break
    else:
        continue
```

ROTEIRO DE AULA PRÁTICA – AULA 03: TIPOS DE DADOS E VARIÁVEIS EM PYTHON

TIPOS DE DADOS.

int:

```
a = 2
b = 4
print(a + b)
```

float:

```
c = 4.5
d = 5.9
print(c + d)
```

complex:

```
e = 5 + 6j
f = 3 - 4j
print(e + f)
```

str:

```
nome = "Gabriella"
print(f"Olá, {nome}")
```

IMPORTÂNCIA DAS VARIÁVEIS.

Variáveis são de extrema importância para a construção dos códigos, pois vão ser nelas que você vai armazenar os dados de seu programa.

OPERADORES EM PYTHON.

Existem 4 tipos de operadores em python: aritméticos(+, -, /, **, //, %), atribuição(=, -=, +=, *=, //=, /=, %=), comparação(==, <, >, <=, >=, !=) e lógico(and, or e not).

CONVENÇÃO DE NOMES (PEP8) EM PYTHON.

Segundo a pep8 os nomes das funções e variáveis tem que estar em “snake_case”, que são as palavras separadas por sublinhados. Os nomes das classes em “CamelCase”, que seria a primeira letra de cada palavra em letra maiúscula.

CÓDIGOS DESENVOLVIDOS EM SALA DE AULA.

```
numero = int(input("Digite um número?"))
if numero >0:
    print("numero positivo")
elif numero <0:
    print("numero negativo")
else:
    print("numero igual a zero")
```

```
n1  = int (input("Digite o primeiro número"))
n2 = int (input("digite o segundo número"))
n3 = int(input("digite o terceiro número"))
if (n1>n2) and (n1>n3):
        print("n1 é o maior número")
if (n2>n1) and (n2>n3):
        print("n2 é o maior número")
if (n3>n1) and (n3>n2):
        print("n3 é o maior número")
```

```
idade = int(input("Qual sua idade:"))
if idade >=18:
    print("Você pode dirigir..")
else:
    print("você é menor de idade")
```

```
idade = int (input("Digite a idade"))
if idade < 6:
```

```
    print ("Que coisa fofa!.")
elif idade < 18:
    print ("Você pode dirigir")
elif idade > 60:
    print ("Você está na melhor idade")
else:
    print ("Você é o cara")
```

```
idade = int (input("Qual sua idade:"))
if idade >=5 and idade <=7 :
    print ("Infantil A")
elif idade >=8 and idade <= 10 :
    print ("Infantil B")
elif idade >=11 and idade <=13 :
    print ("Juvenil A")
elif idade >=14 and idade <=17 :
    print ("Juvenil B")
elif idade >=17 :
      print ("Senior")
else:
     print ("Não encotrado")
```

```
print ("Olá competidor, Digite a sua idade > 5")
nome = input ("Qual o seu nome?")
idade = int (input("Qual a sua idade:"))
if idade >=5 and idade <=7 :
    print(f"{nome} Infanti A")
elif idade >=8 and idade <= 10 :
    print (f"{nome} Infantil B")
elif idade >=11 and idade <=13 :
    print (f"{nome} juvenil A")
elif idade >=14 and idade <=17 :
    print (f"{nome} Juvenil B")
elif idade >=17 :
      print (f"{nome} Senior")
else:
     print (f"{nome} Não encotrado")
```

```

```

```
saldo = 1000

while True:
    saque = float(input("Digite o valor p/ saque (ou 0 p/ sair)"))

    if saque <= 0:
        print ("Operação Finalizada.")
        break
    elif saque > saldo:
        print ("Saldo Insuficiente.")
    else:
        saldo -= saque
        print (f"Saque realizado: R${saldo:.2f}")
```

```
saldo = 1000
while True:
    try:
        saque = float(input("Digite o valor p/ saque (ou 0 p/ sair)"))
        if saque <= 0:
           print ("Operação Finalizada.")
           break
        elif saque > saldo:
           print ("Saldo Insuficiente.")
        else:
          saldo -= saque
          print (f"Saque realizado: R${saldo:.2f}")
    except ValueError:
       print("Por favor, insira um número válido")
```

DESAFIOS A E B:

```
#pergunta o ano e o ano de nascimento e depois calcula a diferença
entre eles
```

```python
ano = int(input("Ano atual: "))
ano_de_nascimento = int(input("Ano de nascimento: "))
diferenca = ano - ano_de_nascimento

#aqui vamos converter para str para pegar os últimos dois dígitos para saber qual é a decada
#pegamos também os dois primeiros para saber o seculo
digitos = str(ano_de_nascimento)
ultimos_dois = digitos[:-1] + "0"
dois_primeiros = digitos[:2]
seculo = int(dois_primeiros) + 1
decada = int(ultimos_dois)
print(f"Diferença: ",diferenca)
print(f"Década: ",decada)
print(f"Século: ", seculo)
```

```python
#pegamos uma frase e deixamos tudo minúscula e sem os espaços, substituímos as ultimas duas palavras e depois invertemos
frase = input("Frase: ").lower().strip()
substituir_palavra = frase[:-2] + "tr"
frase_inversa = substituir_palavra[::-1]
print(f'Frase invertida: ',frase_inversa)
```

CONCLUSÃO.

Esta aula me ajudou a praticar o uso de dados e variáveis em Python, assim me ajudando a ter um melhor entendimento sobre esses temas passados em sala de aula.

ROTEIRO DE AULA PRÁTICA – AULA 04: FUNÇÕES EM PYTHON

FUNÇÕES, PARÂMETROS E RETORNO:

As funções vão agrupar códigos dentro de bloco para cumprirem alguma operação, nas funções podem ter parâmetros, que servem para passar argumentos para dentro dela. Podem também retornar um valor em sua saída, para isso serve o retorno. São úteis, pois são reutilizáveis, assim deixando o código menos redundante.

DIFERENÇA DE VARIÁVEIS EM ESCOPO GLOBAL E LOCAL:

Uma variável em escopo global pode ser utilizada em todo código, uma que está no escopo local pode apenas ser usada dentro do bloco de código que foi criada, por exemplo variáveis que foram criadas dentro de uma função.

DESAFIO A:

```
from datetime import datetime

def preparar(destino):
    ano_atual = datetime.now().year
    diferenca = abs(destino - ano_atual) # vai retornar um valor
absoluto
    energia_necessaria = diferenca * 10
    return diferenca, energia_necessaria

def executar(energia_necessaria, energia_disponivel=100):
    #se tiver energia retorna verdadeiro, se não retorna falso
    if energia_disponivel >= energia_necessaria:
        return True
    else:
        return False

def verificar_chegada(destino, viagem_bem_sucedida):

    if viagem_bem_sucedida:
        print(f"Viagem bem-sucedida! Você chegou ao ano {destino}.")
    else:
        print("Falha na viagem! Energia insuficiente.")
```

```
def viagem_tempo():
    destino = int(input("Digite o ano de destino para a viagem no
tempo: "))

    diferenca, energia_necessaria = preparar(destino)
    print(f"Preparando viagem para {destino}...")
    print(f"Diferença de anos: {diferenca}, Energia necessária:
{energia_necessaria}")

    viagem_bem_sucedida = executar(energia_necessaria)

    verificar_chegada(destino, viagem_bem_sucedida)

# Executar o organizador de viagem no tempo
viagem_tempo()
```

DESAFIO B:

```
def calcular(distancia_anos, velocidade_segundo):
    #calcular em segundos
    tempo_segundos = distancia_anos / velocidade_segundo
    return tempo_segundos

def converter(tempo_segundos):
    #converter em minutos e horas
    tempo_minutos = tempo_segundos / 60
    tempo_horas = tempo_minutos / 60
    return tempo_minutos, tempo_horas


distancia = float(input("Digite a distância da viagem (em anos): "))
velocidade = float(input("Digite a velocidade da viagem (em
anos/segundo): "))

segundos = calcular(distancia, velocidade)

minutos, horas = converter(segundos)
```

```
print(f"\nTempo de viagem:")
print(f"- Segundos: {segundos:.2f} s")
print(f"- Minutos: {minutos:.2f} min")
print(f"- Horas: {horas:.2f} h")
```

CONCLUSÃO:

Funções são ótimas para evitar duplicação, em python para definirmos funções usamos o def depois coloque o nome abra parênteses e finalize usando “:”. Para chamá-las basta colocar o nome delas e os parênteses.

ROTEIRO DE AULA PRÁTICA – AULA 05: MANIPULAÇÃO DE LISTA E DICIONÁRIOS.

LISTAS, ÍNDICES, MÉTODOS E FATIAMENTO.

Listas são utilizadas para guardar vários itens em apenas em uma variável, índices são a posição de um elemento dentro dessa lista.

Um método é uma função associada a um objeto e em python tudo é um objeto.

O fatiamento serve para acessar partes em uma sequência.

DICIONÁRIOS, CHAVES E VALORES.

Dicionários em python é uma estrutura de dados que armazena dados em pares, as chaves são o nome dos atributos dentro desse dicionário e o valores são cada dado daquela chave.

DESAFIO A:

```
agenda  = {} # guardar os compromissos

quant_compromissos = int(input("Quantos compromissos você irá inserir:
"))

for i in range(1, quant_compromissos + 1):
    ano = input("Ano: ")
    compromisso = input("Compromisso: ")
    if ano in agenda:
        agenda[ano].append(compromisso)
    else:
        agenda[ano] = [compromisso]
    print("Compromisso adicionado com sucesso.")

print(f"Agenda: {agenda}")
```

DESAFIO B:

```
#todas as gírias
```

```
giria = {
        "truta": "meu caro",
        "suave": "saudação respeitosa",
        "pego a visão": "compreende",
        "parça": "meu amigo",
        "zoação": "brincadeiras",
    }
#processo de tradução
print(f"Girias: {giria}")
frase_completa = input("Digite uma frase para traduzir: ").lower()
frase_dividida = frase_completa.split()
frase_traduzida = []

for palavra in frase_dividida:
    if palavra in giria:
        frase_traduzida.append(giria[palavra])
    else:
        frase_traduzida.append(palavra)

print("\nFrase traduzida:")
print(" ".join(frase_traduzida))
```

CONCLUSÃO:

O que foi mais fácil criar os dicionários, mais difícil adicionar itens neles de forma interativa.

ROTEIRO DE AULA PRÁTICA – AULA 06: ENTRADA E SAÍDAS DE DADOS EM PYTHON.

ENTRADAS DE DADOS EM PYTHON.

Para entradas de dados usamos o input e o que sai na tela com o resultado é o output, para mostrar esse resultado usamos o print. Para formatação podemos utilizar str.format().

LEITURA/ESCRITA DE ARQUIVOS EM PYTHON(JSON, W, R E A).

Usamos o módulo json da biblioteca de python para lermos um arquivo json.

Modo W para a escrita, cria um novo arquivo ou substitui um. Modo R para a leitura abre o arquivo. Modo A vai acrescentar sem apagar o arquivo anterior.

DESAFIOS A E B:

```
print(" CADASTRO INTERATIVO ")

# Validação do nome não pode estar vazio
while True:
    nome = input("Digite seu nome: ").strip()
    if nome:
        break
    print("Erro: Nome não pode ser vazio!")

# Validação da idade tem ser número positivo
while True:
    idade = int(input("Digite sua idade: "))
    if(idade > 0):
        break
    else:
        print("Idade inválida. Tente novamente.")

cidade = input("Digite sua cidade: ").strip()

# Exibição
print("DADOS CADASTRADOS")
print(f"Nome: {nome.title()}")
print(f"Idade: {idade} anos")
```

```
print(f"Cidade: {cidade.title()}")
```

```
ARQUIVO_REGISTROS = "registros.txt"
#função para salvar
def salvar(ano, descricao):
    with open(ARQUIVO_REGISTROS, 'a', encoding='utf-8') as arquivo:
        arquivo.write(f"{ano}|{descricao}\n")
#função para ler
def ler():
    try:
        with open(ARQUIVO_REGISTROS, 'r', encoding='utf-8') as
arquivo:
            return arquivo.readlines()
    except FileNotFoundError:
        return []
#função para gerenciar

while True:
    print("GERENCIADOR DE VIAGENS")
    print("1. Registrar nova viagem")
    print("2. Ver todas as viagens")
    print("3. Sair")
    opcao = input("Escolha: ")
    if opcao == "1":
        ano = input("Ano visitado: ")
        descricao = input("Descrição: ")
        salvar(ano, descricao)
        print("Viagem registrada!")
    elif opcao == "2":
        registros = ler()
        if not registros:
            print("Nenhuma viagem registrada ainda.")
        else:
            print("\nREGISTROS")
            for linha in registros:
                ano, descricao = linha.strip().split('|')
                print(f"Ano {ano}: {descricao}")
    elif opcao == "3":
```

```
        print("Saindo...")
        break
    else:
        print("Opção inválida!")
```

CONCLUSÃO

Após a aula ficou simples manipular E/S.

ROTEIRO DE AULA PRÁTICA – AULA 07: DEPURAÇÃO E TESTE DE ALGORITMOS.

DIFERENÇA ENTRE DEPURAÇÃO E TESTE AUTOMATIZADOS.

Os testes são feitos para verificar a funcionalidade e desempenho de um software e a depuração para encontrar e corrigir os erros dentro dele.

PARA DEPURAÇÃO UTILIZEI OS PRINTS.

Para a realização da depuração de alguns códigos feitos em sala de aula foi utilizados os prints.

DESAFIOS A, B FEITO EM SALA DE AULA.

```python
from datetime import datetime

def dias_para_viagem(data_viagem_str):
    hoje = datetime.now()
    data_viagem = datetime.strptime(data_viagem_str, "%d/%m/%Y")
    diferenca = data_viagem - hoje
    #print(f"Hoje", hoje)
    #print(f"Viagem", data_viagem)
    #print(f"Diferença", diferenca)

    return diferenca.days
#exemplo de uso
print(dias_para_viagem("30/12/2025"))
```

```python
import unittest
import re

class Validacoes:
    def validar_senha(self, senha):
        senha_num = re.search(r'\d', senha)
        senha_letraMaius = re.search(r'[A-Z]', senha)
        senha_letraMinus = re.search(r'[a-z]', senha)

        if senha_num and senha_letraMaius and senha_letraMinus:
```

```
            return True
        else:
            return False

    def validar_idade(self, idade):

        if idade >= 18:
            return True
        else:
            return False

class Teste_de_unidade(unittest.TestCase):
    def setUp(self):
        self.teste = Validacoes()
    def teste_senha(self):
        self.assertTrue(self.teste.validar_senha("abc12DE"))
        self.assertFalse(self.teste.validar_senha("abc"))
        self.assertFalse(self.teste.validar_senha("abc12"))
    def teste_idade(self):
        self.assertFalse(self.teste.validar_idade(17))
        self.assertTrue(self.teste.validar_idade(18))
        self.assertTrue(self.teste.validar_idade(19))

#para chamar no terminal e fazer a verificação python -m unittest
teste10.py -v
```

CONCLUSÃO:

Utilizar a depuração ou os testes automatizados são essenciais para construção de um software bem desenvolvido, excelentes para encontrar erros, assim facilitando o processo de conserto deles.

ROTEIRO DE AULA PRÁTICA – AULA 08: INTRODUÇÃO À PROGRAMAÇÃO ORIENTADA A OBJETOS(POO) EM PYTHON.

Princípios da POO: classes, objetos, atributos, métodos:

Classes servem como um molde para criação de instâncias. Em python são criadas pela palavra 'class', métodos são as funções associadas a classe. Os objetos são as instâncias das classes, atributos representam características ou estado de um objeto.

DESAFIOS:

```
class Conta:
    #função para inicializar
    def __init__(self, titular, saldo=0):
        self.titular = titular
        self.saldo = saldo
    #função para depositar
    def depositar(self, valor):
        if valor > 0:
            self.saldo += valor
            print(f"Depósito de R${valor:.2f} realizado. Novo saldo:
R${self.saldo:.2f}")
        else:
            print("Valor inválido para depósito.")
     #função para sacar
    def sacar(self, valor):
        if valor > 0 and valor <= self.saldo:
            self.saldo -= valor
            print(f"Saque de R${valor:.2f} realizado. Novo saldo:
R${self.saldo:.2f}")
        else:
            print("Saldo insuficiente ou valor inválido.")

    def exibir_informacoes(self):
        print(f"Titular: {self.titular}\nSaldo: R${self.saldo:.2f}")

conta= Conta("Fernado Alves", 3000)
conta.depositar(1000)
conta.sacar(500)
```

```
conta.exibir_informacoes()
```

```
class Evento:
    def __init__(self, ano, descricao):
        self.ano = ano
        self.descricao = descricao

    def __str__(self):
        return f"Ano {self.ano}: {self.descricao}"

class Viajante:
    def __init__(self, nome):
        self.nome = nome
        self.eventos_vividos = []

    def registrar(self, evento):
        self.eventos_vividos.append(evento)
        print(f"{self.nome} registrou: {evento}")

    def relatorio_pos_viagem(self):
        print(f"RELATÓRIO DE {self.nome.upper()} ")
        for evento in self.eventos_vividos:
            print(evento)

class MaquinaDoTempo:
    def __init__(self, modelo):
        self.modelo = modelo

    def viajando(self, viajante, ano, descricao_evento):
        print(f"\nViajando para o ano {ano} com a máquina
{self.modelo}...")
        evento = Evento(ano, descricao_evento)
        viajante.registrar(evento)

viajante = Viajante("Vitoria")
maquina = MaquinaDoTempo("MX-YZ14")

maquina.viajando(viajante, 2050, "Tecnologia novas")
maquina.viajando(viajante, 3040, "Mundo Novo")
```

```
viajante.relatorio_pos_viagem()
```

CONCLUSÃO

Com essa aula tivemos um excelente aprendizado sobre a programação orientada a objeto, aperfeiçoando nossos conhecimentos sobre este tema com os exercícios passados.

REFERÊNCIAS:

https://www.datacamp.com/pt/tutorial/python-operators-tutorial acesso: 26/05/2025

https://realpython-com.translate.goog/python-pep8/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=tc acesso: 26/05/2025

https://learn.microsoft.com/pt-br/cpp/cpp/functions-cpp?view=msvc-170

https://www-w3schools-com.translate.goog/python/python_lists.asp?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=tc acesso: 26/05/2025

https://hub.asimov.academy/tutorial/funcao-vs-metodo-em-python-entendendo-as-diferencas/ acesso: 26/05/2025

https://hub.asimov.academy/tutorial/fatiamento-em-python-como-funciona-o-famoso-slicing/ acesso: 26/05/2025

https://hub.asimov.academy/blog/dicionario-python/acesso: 26/05/2025

https://www-programiz-com.translate.goog/python-programming/input-output-import?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=tc acesso: 26/05/2025

https://pythonacademy.com.br/blog/input-e-print-entrada-e-saida-de-dados-no-python acesso: 26/05/2025

https://hub.asimov.academy/tutorial/lendo-e-escrevendo-arquivos-json-em-python/ acesso: 26/05/2025

file:///C:/Users/vitor/OneDrive/Documentos/faculdade/pensamento%20computacional%20com%20python/slides/Slide%20de%20Aula%20-%20Unidade%20III.pdf acesso: 26/05/2025

https://www-browserstack-com.translate.goog/guide/difference-between-testing-and-debugging?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=wa acesso: 26/05/2025

https://medium.com/@guilhermerdcarvalho/paradigma-orientado-ao-objeto-poo-em-python-a107d35bee3f acesso: 26/05/2025