Atividade S2 A2

Nome: Adalberto Caldeira Brant Filho

Repositório GitHub: https://github.com/adalbertobrant/lipai

Código das Videoaulas

Aula 01 - Operadores

Os operadores realizam tarefas sobre variáveis , literais e variáveis e literais em conjunto ou separado, na tabela abaixo temos os operadores apresentados na videoaula 1:

Operador	Tipo	Função
`+`	Aritmético	Realiza a soma algébrica ou a concatenação de strings.
`-`	Aritmético	Realiza a subtração.
`*`	Aritmético	Realiza a multiplicação.
`/`	Aritmético	Realiza a divisão (retorna float).
`//`	Aritmético	Realiza a divisão inteira (descarta a parte decimal).
`%`	Aritmético	Módulo: retorna o resto da divisão.
`**`	Aritmético	Exponenciação (potência).
`=`	Atribuição	Atribui um valor a uma variável.
`==`	Comparação	Verifica se dois valores são iguais.
`!=`	Comparação	Verifica se dois valores são diferentes.
`>`, `<`	Comparação	Maior que / Menor que.
`>=`, `<=`	Comparação	Maior ou igual / Menor ou igual.
`and`	Lógico	Retorna `True` se **ambas** as condições forem verdadeiras.
`or`	Lógico	Retorna `True` se **pelo menos uma** condição for verdadeira.
`not`	Lógico	Inverte o valor booleano (True vira False e vice-versa).
`is`	Especial	Identidade: verifica se as variáveis apontam para o mesmo objeto na memória.
`in`	Especial	Associação: verifica se um valor está presente em uma sequência (str, list, dic,set, etc.).

De maneira geral o operador adição trabalha com literais bem como com variáveis

```
n1 = 10.2
n2 = 3.5
resultado = n1 + n2 + 8.5
```

Códigos demonstrativos

```
 #!/usr/bin/env python3
# -*- coding: utf-8 -*-

""" Aula 01 - Operadores """

n1 = 10.2
n2 = 3.5
resultado = n1 + n2 + 8.5

# Operadores Aritméticos
print('1 + 1', 1 + 1, type(1 + 1))
print('1.5 + 1.7', 1.5 + 1.7, type(1.5 + 1.7))
print('resultado ', resultado, type(resultado))
print(3-1)
print(3*2, type(3*2))
print(3/2, type(3/2))
print(3 % 2, type(3 % 2))
print(10 // 3, type(10 // 3))
print(10 ** 2, type(10 ** 2))

# Operador de atribuição
x = 10.0
print(x, type(x))

# Operadores de comparação ou relacionais
if x > 10:
    print(True)

resultado = x == 10
print(resultado, type(resultado))

print('10 == 10', 10 == 10, type(10 == 10))
print('10 != 10', 10 != 10, type(10 != 10))
print('10 > 10', 10 > 10, type(10 > 10))
print('10 >= 10', 10 >= 10, type(10 >= 10))
print('10 < 10', 10 < 10, type(10 < 10))
print('10 <= 10', 10 <= 10, type(10 <= 10))

condicao = x > 10 and resultado < 3

if condicao:
    pass

# Operadores lógicos
# and
print('True and True ->', True and True, type(True and True))
print('True and False ->', True and False, type(True and False))
print('False and True ->', False and True, type(False and True))
```

```python
print('False and False ->', False and False, type(False and False))

# or
print('True or True', True or True, type(True or True))
print('True or False', True or False, type(True or False))
print('False or True', False or True, type(False or True))
print('False or False', False or False, type(False or False))

# not
condicao = True
print('not condicao', not condicao, type(not condicao))

condicao = False
print('not condicao', not condicao, type(not condicao))

# Operadores especiais

# is
# == compara se dois valores são iguais
# is verifica se as variáveis apontam para o mesmo objeto em memória
a = 10
b = 10.0
c = b

print(a == b, a == c, b == c)
print(a is b, a is c, b is c)

# in
# devolve um booleano
# pode ser usado com str, list, tuple, set, dic (apenas na chave)

frase = 'o rato roeu'
print('o' in frase, type('o' in frase))
print('O' in frase, type('O' in frase))

# in in listas
numeros = [1, 3, 5, 7, 11, 2]
print(0 in numeros)
print(-2 in numeros)
print(11 in numeros)

# in in tuplas
numeros = (0, 1, 2, 3, 4, 5, 6, 7, 10)
print(100 in numeros)
print(2 in numeros)

# in in sets
numeros = {1, 2, 34, 5, 7}
print('' in numeros)
print(2 in numeros)

# in in dicionários , apenas busca na chave e não no valor
pessoa = {
    'nome': 'José da Silva',
    'idade': 22,
```

```
    'email': 'jose@mail.com'
}

print('nome' in pessoa)  # True
print('José da Silva' in pessoa)  # False
print('email' in pessoa)  # True
print('jose@mail.com' in pessoa)  # False
```

Aula 02 - if

```
""" Aula 02 - instrução if """

# python usa identação
# if condição:
#     instrucao()
#     instrucao()
#     instrucao()

# outras linguagens c, java, c++

# if(){
#   instrucao()
#   instrucao()
#  }

codigo_cliente = 32
valor_desconto = 30.0
DESCONTO_ESPECIAL = valor_desconto >= 20.0

if DESCONTO_ESPECIAL:
    print('Desconto Especial')
    print(codigo_cliente)
else:
    print('Sem desconto especial')

# atenção com os cuidados com a identação pois pode ocorrer falhas, segundo a
pep8 em sitaução de blocos deve-se usar 4 espaços

# regra do nome pelo menos 3 caracteres
nome = 'Lo'
print(len(nome))  # devolve 2

nome = 'Lois'
print(len(nome))  # devolve 4

if len(nome) < 5:
    print('Nome deve ter pelo menos 6 caracteres')
else:
    print('Nome válido')

NOME_INVALIDO = len(nome) < 3

if NOME_INVALIDO:
    print('Nome deve ter pelo menos 3 caracteres')
```

```
else:
    print('Nome válido')

NOME_VALIDO = len(nome) >= 3
if NOME_VALIDO:
    print('Nome válido')
else:
    print('Nome deve ter pelo menos 3 caracteres')

# regra do nome pelo menos 3 caracteres
# idade tem que ser maior ou igual a 18
# exibir todos os erros no final apenas
nome = 'Lo'
idade = 17
erros = []

NOME_INVALIDO = len(nome) < 3
if NOME_INVALIDO:
    erros.append('Nome deve ter pelo menos 3 caracteres')

IDADE_INVALIDADE = idade < 18
if IDADE_INVALIDADE:
    erros.append('Idade deve ser maior ou igual a 18')

if len(erros) != 0:
    print(erros)
else:
    print('Dados válidos')

# Dentro do if é avaliado como False as seguintes estruturas:
""" False, None, 0 , 0.0, string vazia '', lista, tuple, set vazio """
# Todo o resto é considerado dentro do if como True

# estrutura if elif else

# Verifica se um número é positivo,negativo ou zero

numero = 3

if numero > 0:
    print('Maior que zero')
elif numero == 0:
    print('Zero')
else:
    print('Menor que zero')


numero = 0

if numero > 0:
    print('Maior que zero')
elif numero == 0:
    print('Zero')
else:
    print('Menor que zero')
```

```
numero = -4

if numero > 0:
    print('Maior que zero')
elif numero == 0:
    print('Zero')
else:
    print('Menor que zero')

# faixas de valores devemos tentar particionar para 3 condições
# calcule a média e verifique se as duas notas são válidas antes do cálculo

n1 = 5
n2 = 10.0

NOTA1_VALIDA = 0 < n1 <= 10.0
NOTA2_VALIDA = 0 < n2 <= 10.0

if NOTA1_VALIDA and NOTA2_VALIDA:
    media = (n1 + n2) / 2.0

    if media >= 6.0:
        print('Aprovado')
    elif media >= 4.0:
        print('Recuperação')
    else:
        print('Reprovado')
else:
    print('Notas inválidas')
```

Aula 03 - for

O loop for funciona para repetir instruções e também fazer a iteração em coleções de elementos

```
""" Aula 03 - loop for """

linguagens = ['C', 'Java', 'Python']
print(linguagens[0])
print(linguagens[1])
print(linguagens[2])

# loop for ajuda na iteração

# Sintaxe
# for valor in colecao:
#    instrucao
#    instrucao
#    instrucao

# nessa construção do for valor in colecao , o operador in , verifica se o
elemento está dentro da coleção , retira e coloca no valor

# imprime cada linguagem em uma linha
for linguagem in linguagens:
```

```
    print(linguagem.upper())

# contar elementos e calcular a média

nota1 = 10.0
nota2 = 5.5
nota3 = 8.3

media = (nota1 + nota2 + nota3) / 3
print(media)

# em casos que temos uma lista o for ajuda
soma = 0
notas = [10, 9, 6, 7, 8, 10]
for nota in notas:
    soma = soma + nota
media = soma / len(notas)

# range em python

valores = range(0, 10)  # vai de zero até 9 pois o 10 não entra

for valor in valores:
    print(valor)

# steps in range

valores = range(0, 11, 2)  # vai de 2 em dois até o 10

# steps com decremento  in range

valores = range(10, 0, -1)  # 10, 9 , 8
print(valores)
for i in valores:
    print(i)


valores = range(10, -1, -2)  # 9, 7, 5, 3, 1
print(valores)
for i in valores:
    print(i)

for i in range(len(linguagens)):
    print(linguagens[i])

# o range é mais utilizado em casos em que não iremos percorrer de maneira linear
a nossa coleção
```

Aula 04 - while

```
""" Aula 04 - Loop while """

# Repetir instruções quando não temos uma instrução ou condição clara antes de
verificar valores inicialmente
```

```
# Quando uma lista não está completamente na memória ainda, ou seja está em
construção dinâmica
# muito usado em leitura de arquivos , em leitura com cursores

# while condição_for_verdadeira:
#    instrução
#    instrução
#    instrução

contador = 0

while contador <= 10:
    print(contador)
    contador += 1  # incremento da condição
print('fim')

# exemplo de leituras em arquivo
# arquivo já está aberto
"""
linha = arquivo.readline()
while linha:
    instrucao_linha
    linha = arquivo.readline()  # vai ler até encontrar uma linha final
"""

# O while funciona pelo mesmo princípio de condição ou seja ele precisa de uma
condição verdadeira para entrar em seu laço, sendo condições False a razão de
saída do laço while
```

Aula 05 - continue e break

A instrução break sai do loop , a instrução continue vai para a próxima iteração no loop

```
""" Aula 05 - break e continue """

# Usa-se o break e continue dentro de um looping

for i in range(0, 10, 1):
    if i == 4:
        print('saída do loop')
        break
    print(i)

# encontrar a posição de um número
# em uma lista de inteiros. Caso não
# encontre posição i é igual a -1

busca = 5
numeros = [1, 33, 51, 6, 5, 7, 8, 9, 5, 1, 7]
posicao = -1
contador = 0

for numero in numeros:
    if numero == busca:
        posicao = contador
```

```
        print(f"Achei {busca} na posicao {posicao}")
    contador += 1

# usando o break para achar a primeira posicao
print("Usando o break para cancelar a execução do laço for")
busca = 5
numeros = [1, 33, 51, 6, 5, 7, 8, 9, 5, 1, 7]
posicao = -1
contador = 0

for numero in numeros:
    if numero == busca:
        posicao = contador
        print(f"Achei {busca} na posicao {posicao}")
        print('Saindo do for')
        break
    contador += 1

# exemplo com o range
print("Usando o range e o break no laço for")

busca = 5
numeros = [1, 33, 51, 6, 5, 7, 8, 9, 5, 1, 7]
posicao = -1

for i in range(len(numeros)):
    print('Procurando ... ...')
    if numeros[i] == busca:
        posicao = i
        break
print(f'{busca} encontrada na posicao {posicao}')


# Funcionamento do continue
# Apenas pula a iteração atual indo para a próxima iteração sem sair do loop

numeros = [1, 33, 513, 3, 3, 3, 3, 3, 3, 6, 5,
           7, 8, 9, 5, 1, 7, 3, 3, 3, 3, 3, 3, 3, 333]
for numero in numeros:
    if numero == 3:
        continue  # para a iteração atual do loop e vai para a próxima
    print(numero)
```

Exercícios de Fixação

Ex01

```
""" Ex 01 - Solicite ao usuário 3 notas e apresente o
resultado da média aritmética """

nota1 = float(input('Entre a nota 1-> '))
nota2 = float(input('Entre a nota 2-> '))
nota3 = float(input('Entre a nota 3-> '))

media = nota1 + nota2 + nota3
media = media / 3.0

print(f"Resultado da média das notas {nota1, nota2, nota3} é {media}")
```

Ex02

```
""" Ex02 - Solicite ao usuário, uma única vez, as notas no formato n1, n2, n3,
nm e apresente:a média aritmética das notas;
○ a situação: aprovado (média > 6.0), recuperação (4.0 ≤ média ≤ 6.0) ou
reprovado (média < 4.0)."""

lista_notas = input("Entre todas as notas separadas por ';' -> ")

lista_notas = lista_notas.split(sep=';')

lista_notas = [float(nota.strip()) for nota in lista_notas if nota.strip()]

media = sum(lista_notas) / len(lista_notas)

print(f'Média das notas -> {media}')

if media > 6.0:
    print('Aprovado')
if media >= 4.0 and media <= 6.0:
    print('Recuperação')
if media < 4.0:
    print('Reprovado')
```

Ex03

```
""" Ex 03 - Crie um programa que solicite um identificador ao usuário e
informe se é válido ou inválido
○ o código identificador contém 7 caracteres;
○ começa com BR;
○ seguido de um número inteiro entre 0001 e 9999;
○ por fim, termina com o caractere X.
○ Exemplos válidos: BR0001X, BR1236X, BR9999X
○ Exemplos inválidos: br0001X, BR126X, BR99999X, BR9999Y """

identificador = input('Entre o identificador -> ')

tamanho_identificador = len(identificador) == 7

inicio_br = (identificador[0] == 'B' and identificador[1] == 'R')
```

```
# verifica se os números são válidos

QTD_INTEIROS = 4
SOMA_INTEIROS = 0

for i in range(2, 6):
    if identificador[i] in '0123456789':
        SOMA_INTEIROS += 1

NUMERO_VALIDO = False

if QTD_INTEIROS == SOMA_INTEIROS:
    meio = identificador[2:6]
    numero = int(meio)
    NUMERO_VALIDO = 1 <= numero <= 9999

x = identificador[-1] == 'X'

if tamanho_identificador and inicio_br and NUMERO_VALIDO and x:
    print(f'{identificador} é válido')
else:
    print(f'{identificador} não é válido')
```

Ex04

```
"""
Ex 04 - Baseado no ex03.py, apresente todas as inconsistências
do identificador informado usando uma lista de erros.
"""

identificador = input('Entre o identificador -> ')

erros = []

if len(identificador) != 7:
    erros.append('O identificador não possui exatamente 7 caracteres.')

if len(identificador) >= 2:
    if not (identificador[0] == 'B' and identificador[1] == 'R'):
        erros.append('O identificador não inicia com a sequência BR.')
else:
    erros.append('O identificador não inicia com a sequência BR.')

if len(identificador) >= 6:
    QTD_INTEIROS = 4
    SOMA_INTEIROS = 0

    for i in range(2, 6):
        if identificador[i] in '0123456789':
            SOMA_INTEIROS += 1

    if QTD_INTEIROS == SOMA_INTEIROS:
        meio = identificador[2:6]
        numero = int(meio)
```

```
        if not (1 <= numero <= 9999):
            erros.append(
                'O identificador não apresenta número inteiro entre 0001 e
9999.')
    else:
        erros.append(
            'O identificador não apresenta número inteiro entre 0001 e 9999.')
else:
    erros.append(
        'O identificador não apresenta número inteiro entre 0001 e 9999.')

if len(identificador) >= 1:
    if identificador[-1] != 'X':
        erros.append('O identificador não finaliza com o caractere X.')
else:
    erros.append('O identificador não finaliza com o caractere X.')

print()
if len(erros) == 0:
    print(f'{identificador} é válido')
else:
    print(f'{identificador} não é válido')
    print('\nErros:')
    for erro in erros:
        print(f'# {erro}')
```