**Exercícios Detalhados: Variáveis ​​em Python**

**Exercícios Práticos**

1. **Escopo de Variáveis**​​Crie um programa que demonstre a diferença entre variáveis ​​locais e globais. Defina uma variável global, crie uma função que tente modificar essa variável global (sem usar a palavra-chave global), e outra função que a modifique corretamente usando global. Imprima o valor da variável após convocar ambas as funções.

variavel\_global = 10

def modificar\_sem\_global():

variavel\_global = 20

print(f"Na função (sem global), a variável é: {variavel\_global}")

def modificar\_com\_global():

global variavel\_global

variavel\_global = 30

print(f"Na função (com global), a variável é: {variavel\_global}")

modificar\_sem\_global()

print(f"Valor da variável global após a primeira função: {variavel\_global}")

modificar\_com\_global()

print(f"Valor da variável global após a segunda função: {variavel\_global}")



1. **Mutabilidade vs. Imutabilidade** Crie uma lista e uma string. Tente modificar um elemento da lista e um caractere da string. Use a função id()para mostrar como o ID da lista permanece o mesmo após a alteração, enquanto o ID da string muda quando você tenta "modificá-la".

lista = [1, 2, 3]

texto = "Olá"

print(f"ID da lista antes: {id(lista)}")

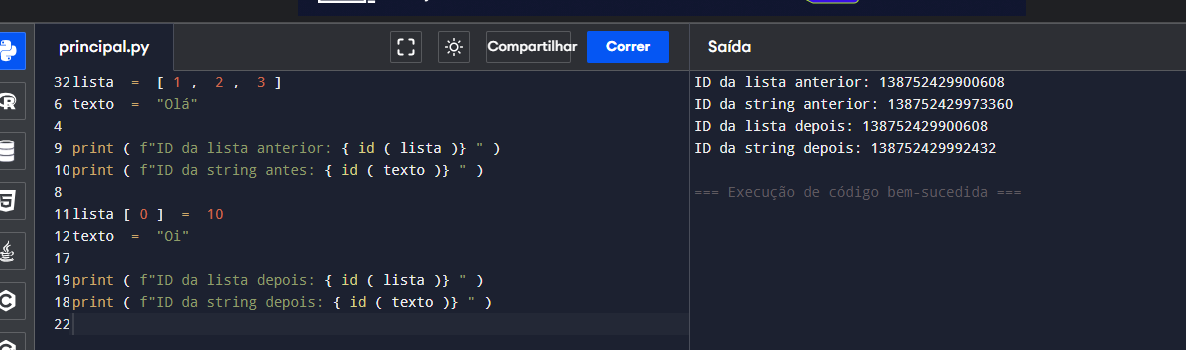
print(f"ID da string antes: {id(texto)}")

lista[0] = 10

texto = "Oi"

print(f"ID da lista depois: {id(lista)}")

print(f"ID da string depois: {id(texto)}")



1. **Tipagem Dinâmica** Escreva um programa que demonstre a tipagem dinâmica do Python. Crie uma variável e atribua a ela valores de diferentes tipos (int, float, string, lista) em sequência. Utilize a função type()para imprimir o tipo de variável após cada atribuição.

variavel = 10

print(f"Tipo após atribuição de inteiro: {type(variavel)}")

variavel = 3.14

print(f"Tipo após atribuição de float: {type(variavel)}")

variavel = "Olá"

print(f"Tipo após atribuição de string: {type(variavel)}")

variavel = [1, 2, 3]

print(f"Tipo após atribuição de lista: {type(variavel)}")



1. **Gerenciamento de Memória** Crie duas variáveis ​​que referenciam a mesma lista. Modifique a lista através de uma das variáveis ​​e mostre que a mudança é refletida na outra. Em seguida, use o módulo copypara criar uma cópia superficial e uma cópia profunda da lista, e demonstre a diferença entre elas ao modificar elementos aninhados.

import copy

lista\_original = [1, 2, [3, 4]]

lista\_ref = lista\_original

lista\_ref[0] = 10

print(lista\_original)

lista\_copia\_superficial = copy.copy(lista\_original)

lista\_copia\_superficial[2][0] = 99

print(lista\_original)

lista\_copia\_profunda = copy.deepcopy(lista\_original)

lista\_copia\_profunda[2][1] = 88

print(lista\_original)



1. **Conversão de Tipos e Operações** Crie um programa que solicite ao usuário a inserção de dois números como strings. Converta-os para float, realize algumas operações matemáticas (adição, subtração, multiplicação, divisão), e então converta os resultados de volta para strings para exibição. Lide com possíveis erros de conversão.

try:

num1 = float(input("Digite o primeiro número: "))

num2 = float(input("Digite o segundo número: "))

print("Soma:", num1 + num2)

print("Subtração:", num1 - num2)

print("Multiplicação:", num1 \* num2)

if num2 != 0:

print("Divisão:", num1 / num2)

else:

print("Erro: divisão por zero")

except ValueError:

print("Erro: Por favor, insira números válidos.")

**Exercícios Teóricos**

1. **Explique o conceito de tipagem dinâmica em Python. Como isso difere de linguagens com tipagem estática? Quais são as vantagens e vantagens da tipagem dinâmica?**

A tipagem dinâmica oferece flexibilidade e agilidade, mas pode ser mais propensa a erros e desafios

Resposta atenciosa:

* + Definição de tipagem dinâmica
  + Comparação com tipagem estática

Tipo de variação

* + Vantagens: flexibilidade, menos código detalhado
  + Desvantagens: possíveis erros de tipo em tempo de execução, possível impacto no desempenho

1. **Descreva a diferença entre objetos mutáveis ​​e imutáveis ​​em Python. Dê exemplos de cada um e explique como isso afeta o comportamento das variáveis ​​que referenciam esses objetos.**

Resposta atenciosa:

* + Definição de mutabilidade e imutabilidade
  + Exemplos: listas (mutáveis) vs.
  + Explicação de como a mutabilidade afeta operações e atribuições
  + Discussão sobre como isso está relacionado com o conceito de identidade de objetos (função id())

1. **Explique o conceito de escopo de variáveis ​​em Python. Como o Python lida com variações locais e globais? O que é a regra LEGB e como ela se aplica à resolução de nomes de variáveis?**

Resposta atenciosa:

* + Definição de escopo local e global
  + Explicação da palavra-chaveglobal
  + Descrição da regra LEGB (Local, Enclosing, Global, Built-in)
  + Exemplos de como Python resolve nomes de variáveis ​​em diferentes contextos