Universidade POSITIVO

AULA 4 – CONSTANTES, VARIÁVEIS E TIPOS DE DADOS EM PYTHON

Objetivos da Aula

- Compreender a diferença entre constantes e variáveis e seu papel em algoritmos.
- Estudar os quatro tipos básicos de dados do Python: **int**, **float**, **str** e **bool**.
- Explorar conversões entre tipos de dados e identificar erros comuns.
- Aplicar boas práticas de nomeação e manipulação de dados.
- Resolver exercícios progressivos diretamente no quadro e testar em prática no VS Code.

---

1. Introdução Teórica

1.1 Variáveis e Constantes

- **Variáveis**: espaços de memória com nomes que podem ser alterados durante a execução.
- **Constantes**: valores fixos usados em cálculos e representados por convenção em letras maiúsculas.

Exemplo:

```python
#Declarando e alterando valores de variáveis
TAXA_IDADE_MINIMA = 18  # constante representando uma idade mínima padrão
nome = "Lucas"
idade_atual = 20

# Exibe se a pessoa tem idade acima do mínimo definido pela constante
if idade_atual >= TAXA_IDADE_MINIMA:
    print(f"{nome} tem {idade_atual} anos e já é maior de idade.")
else:
    print(f"{nome} tem {idade_atual} anos e ainda não atingiu a idade mínima.")


```

1.2 Tipos de Dados em Python

- **Inteiro (int):** números sem casas decimais (34, -10)
- **Float (float):** números com casas decimais (3.14, -2.5)
- **String (str):** texto entre aspas simples ou duplas ("Olá")
- **Booleano (bool):** valores lógicos (True ou False)

Exemplo:

1.3 Conversão Entre Tipos

- A conversão entre tipos é essencial quando precisamos somar, multiplicar ou concatenar valores de tipos diferentes.
- **Quando usar?** Principalmente quando o dado vem de `input()`, que sempre retorna uma string.
- **Funções usadas:** `int()` para inteiros, `float()` para decimais e `str()` para converter números em texto.
- **Atenção:** Se tentar converter algo inválido (ex.: `int('abc')`), o programa gera erro.

Exemplo explicado:

```
print(str(123) + '4')    # Resultado: 1234 (concatenação de strings)
print(int('14') + 28)    # Resultado: 42 (soma numérica)
print(float('1.625') + 0.1)  # Resultado: 1.725 (soma com decimal)
```

Este exemplo demonstra a diferença entre concatenação (strings) e soma (números) e como a conversão altera o comportamento.

1.4 Convenções de Nomeação

- **Use snake_case para variáveis (`nota_final`, `nome_aluno`)**: Este padrão facilita a leitura, separando palavras com underscore. É a convenção mais aceita em Python para nomes de variáveis.
- **Use MAIÚSCULAS para constantes (`TAXA_DESCONTO`)**: Diferencia variáveis comuns de valores que não devem ser alterados.
- **Evite nomes genéricos (`a`, `x`)**: Nomes curtos e sem significado dificultam a leitura e manutenção do código. Prefira nomes que representem o dado armazenado, como `idade_usuario` ou `valor_total`.
- **Evite palavras reservadas (`print`, `class`)**: Palavras que já têm função no Python não devem ser usadas como nome de variáveis. Usar palavras reservadas gera erro de sintaxe e confusão.

1.5 Operadores de Atribuição e Atualização de Variáveis

- Estes operadores permitem atualizar variáveis sem precisar reescrever o nome e valor completo. São úteis para contadores, somatórios e cálculos progressivos.
- Operadores disponíveis: `+=`, `-=`, `*=`, `/=`, `//=`, `%=`.
- **Como funcionam?**
  - `contador += 2` é o mesmo que `contador = contador + 2`.
  - Simplifica a leitura e torna o código mais enxuto.

Exemplo:

```
contador = 5
contador += 2  # soma 2 ao valor atual (resultado: 7)
contador *= 3  # multiplica por 3 (resultado: 21)
print(contador)
```

Este exemplo mostra como atualizar variáveis de forma progressiva em cálculos contínuos.

1.6 Função type() e isinstance()

- **`type()`**: Retorna o tipo exato do dado armazenado (int, float, str, etc.). Útil para depuração e estudo.
- **`isinstance()`**: Verifica se o valor pertence a um tipo específico ou a uma tupla de tipos.
- **Quando usar?**
  - Para confirmar se um dado é do tipo esperado antes de realizar operações.
  - Em funções que devem se comportar diferente conforme o tipo do argumento.

Exemplo explicado:

```
valor = 10.5
print(type(valor))              # Mostra: <class 'float'>
print(isinstance(valor, float))  # Mostra: True
print(isinstance(valor, int))    # Mostra: False
```

Este exemplo reforça a importância de saber o tipo do dado antes de manipulá-lo.

2. Metodologia (Explicação + Quadro + Prática)

1. Explicar no quadro conceitos e exemplos.
2. Mostrar código passo a passo e executar no VS Code.
3. Pedir para alunos testarem mudanças (valores, tipos e conversões).
4. Revisar juntos erros comuns e como corrigi-los.

3. Exemplos Práticos Progressivos

Exemplo 1 – Soma com Conversão

```
a = int(input("Digite um número inteiro: "))
b = int(input("Digite outro número inteiro: "))
print(f"A soma é: {a + b}")
```

Justificativa: Este exemplo reforça a necessidade de conversão para `int`, pois o `input()` retorna string. Sem conversão, haveria concatenação de texto em vez de soma.

Exemplo 2 – Uso de Float e Erro de Conversão

```
# Tentando converter string inválida para float
try:
    print(float("BANANA"))
except ValueError:
    print("Erro: não é possível converter para float.")
```

Justificativa: Demonstra como tratar erros comuns ao converter valores. Ajuda o aluno a entender a importância da validação de dados.

Exemplo 3 – Atualização de Variável e Constante

```python
PI = 3.14
raio = 4
area = PI * (raio ** 2)
print(f"Área do círculo: {area}")
```

Justificativa: Integra constante (PI) com cálculo de área, exemplificando uso prático em fórmulas matemáticas e mostrando boas práticas de nomeação.

4. Exercícios de Fixação

1. Explique o resultado do código:

```
print(str(123) + '4')
```

Resposta: O operador `+` entre strings faz concatenação. `str(123)` gera "123" e somado a `'4'` resulta em "1234".

2. Corrija o código para que o usuário possa somar dois números decimais (float) corretamente. **Resposta esperada:**

```python
a = float(input("Digite o primeiro número decimal: "))
b = float(input("Digite o segundo número decimal: "))
print(f"A soma é: {a + b}")
```

3. Escreva um programa que pergunte ao usuário um número em string e converta para inteiro e float, exibindo ambos. **Explicação:** O `input()` retorna string; é preciso usar `int()` e `float()` para converter e mostrar a diferença.
4. Teste o comando:

```
print(int(2.718))
```

Qual o resultado e por quê?

Resposta: O resultado será `2`, porque a função `int()` descarta a parte decimal, convertendo para número inteiro.

5. Exercício Final (Dificuldade Média)

Tarefa: Crie um programa que:

- Peça o nome do produto e seu preço unitário.
- Peça a quantidade e calcule o valor total.
- Use uma constante para definir o desconto de 10%.
- Exiba o valor bruto, o desconto e o valor final com clareza

```python
# Definindo constante de desconto
DESCONTO = 0.10

# Entrada de dados
produto = input("Digite o nome do produto: ")
preco_unitario = float(input("Digite o preço unitário do produto: "))
quantidade = int(input("Digite a quantidade comprada: "))

# Cálculo dos valores
valor_bruto = preco_unitario * quantidade
valor_desconto = valor_bruto * DESCONTO
valor_final = valor_bruto - valor_desconto

# Exibição formatada dos resultados
print(f"\nProduto: {produto}")
print(f"Quantidade: {quantidade}")
print(f"Preço unitário: R$ {preco_unitario:.2f}")
print(f"Valor total: R$ {valor_bruto:.2f}")
print(f"Desconto: R$ {valor_desconto:.2f}")
print(f"Valor final: R$ {valor_final:.2f}")
```

Saída esperada:

```
Produto: Camiseta
Quantidade: 3
Preço unitário: R$ 50.00
Valor total: R$ 150.00
Desconto: R$ 15.00
Valor final: R$ 135.00
```

Próxima Aula: Entrada de Dados e Conversão Avançada de Tipos em Python.