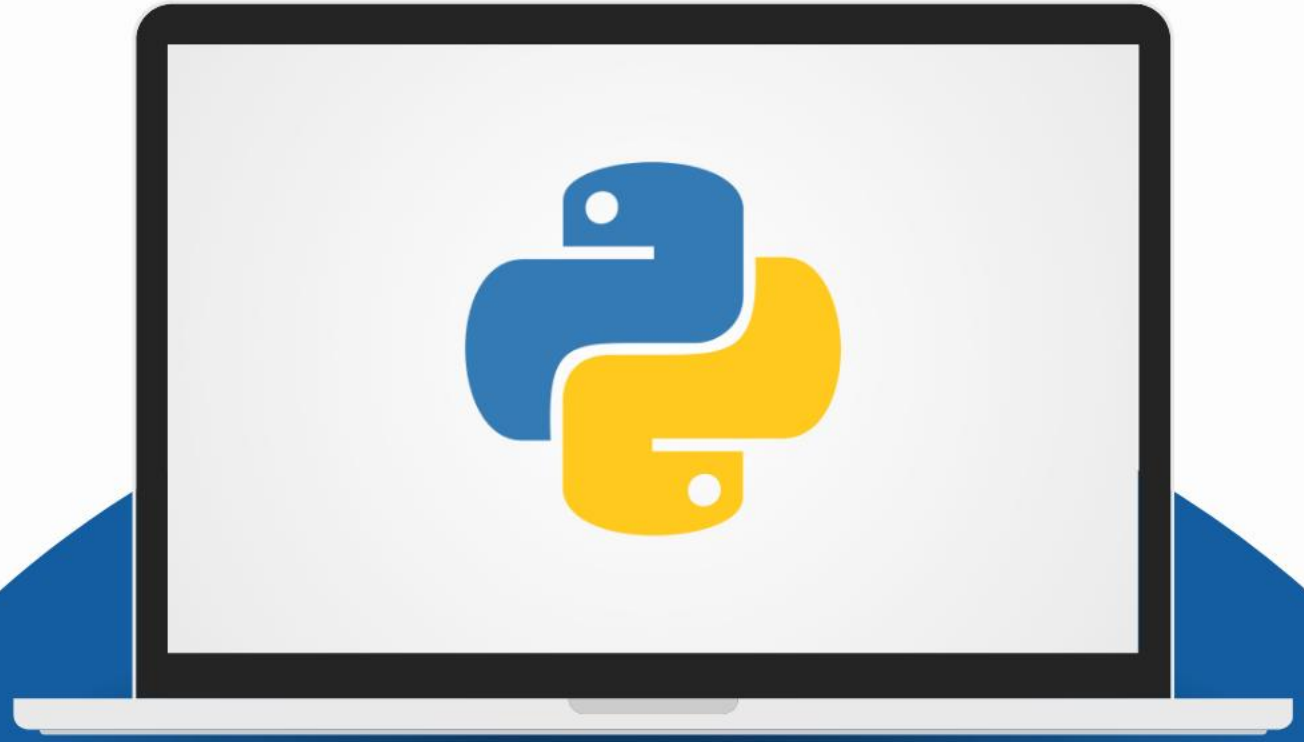
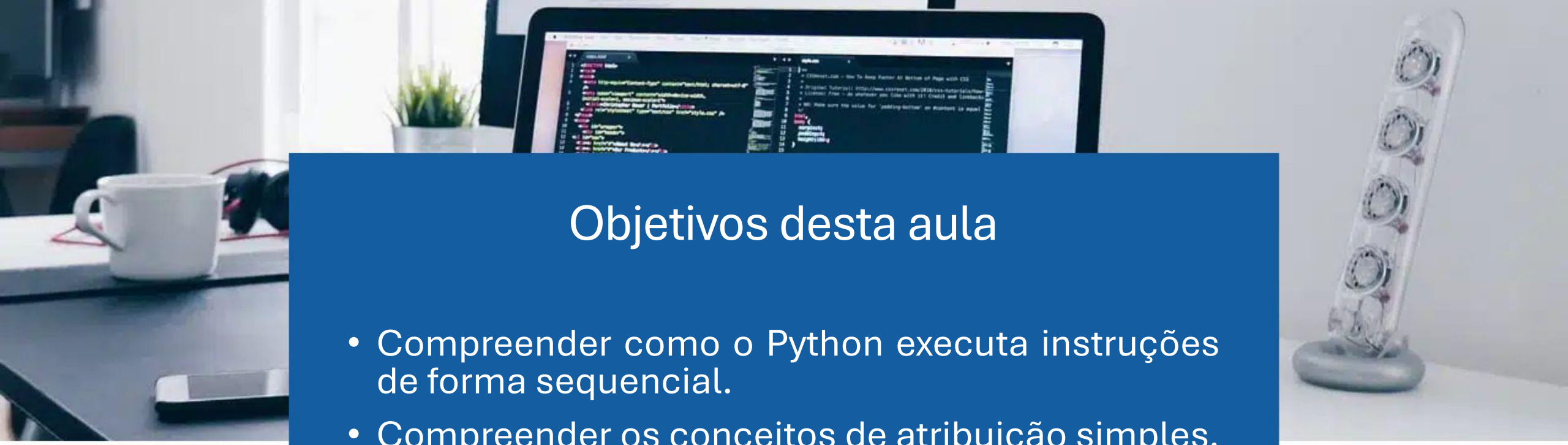
Aula #4:

Estruturas sequenciais

Prof. Dr. Luiz Álvaro de Oliveira Júnior





Objetivos desta aula

- Compreender como o Python executa instruções de forma sequencial.
- Compreender os conceitos de atribuição simples, composta e múltipla.
- Utilizar funções básicas de entrada (input) e saída (print).
- Aplicar esses conceitos em pequenos algoritmos práticos.

Sumário

Estruturas sequenciais

Estruturas **sequenciais** são o tipo mais simples de construção lógica em um programa e representam uma **sequência direta** de instruções que são **executadas** linearmente, isto é, **na ordem em que aparecem no código**, de cima para baixo, sem desvios, repetições ou decisões.

Ideal para operações simples como cálculo, exibição de mensagens, coleta de dados, entre outras aplicações.

Estruturas sequenciais

Exemplo: O código abaixo lê a base e a altura de um retângulo, calcula a área e, em seguida, mostra na tela o valor com duas casas decimais.

```
b = float(input("Digite a base: ")
h = float(input("Digite a altura: ")
A = b * h
print(f"A área do retângulo é de {A:.2f} m2")
```

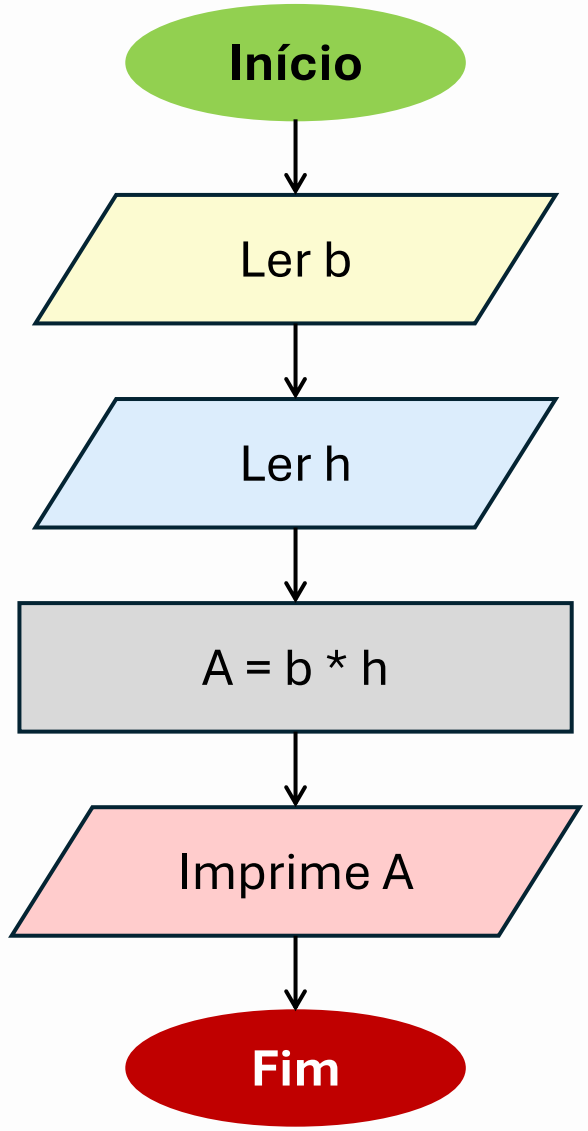
Fluxo do código

```
b = float(input("Digite a base : ")
```

```
h = float(input("Digite a altura : ")
```

```
A = b * h
```

```
print(f"A área do retângulo é de {A:.2f} m2")
```



Atribuição composta

Até o momento, temos trabalhado com variáveis cujos valores eram atribuídos de maneira simples. No entanto, há outras duas formas relevantes ao contexto da nossa disciplina – as atribuições composta e múltipla.

A **atribuição composta** é muito útil na atualização de valores de variáveis já existentes. Nesse caso o operador de atribuição simples (=) é modificado inserindo antes dele o operador correspondente à operação que se deseja realizar. É útil em operações recursivas.

Atribuição composta

Operadores de atribuição composta:

Operador	Exemplo	Equivale a
+=	x += 5	x = x + 5
-=	x -= 2	x = x - 2
*=	x *= 3	x = x * 3
/=	x /= 4	x = x / 4
//=	x //= 2	x = x // 2
%=	x %= 3	x = x % 3
**=	x **= 5	x = x ** 5

Atribuição múltipla

Na **atribuição múltipla** fazemos a atribuição de mais de uma variável em uma mesma linha, separando-as por vírgula. As funções de entrada de dados que vão após o sinal de atribuição também devem ser separadas por vírgulas.

Vejamos essa abordagem aplicada ao código anterior:

```
b,h = float(input("Digite a base : "),float(input("Digite a altura: ")
A = b * h
print(f"A área do retângulo é de {A:.2f} m2")
```

Exercícios

Exercício #1

Escreva um programa que leia o número de matrícula de um aluno da PUC Goiás e gere o endereço de e-mail institucional desse aluno.

```
matricula = input("Digite os números da sua matrícula: ")
print(f"E-mail institucional: {matricula}@pucgo.edu.br")
```

Exercício #2

Escreva um programa que leia o dia e o número correspondente ao mês de nascimento de uma pessoa e some os dois números. **Use atribuição múltipla**.

```
dia, mes = int(input("Digite o dia: ")), int(input("Digite o mês: "))
soma = dia + mês
print("A soma é: ",soma)
```

Exercício #3

Escreva um programa que leia duas variáveis inteiras, troque os valores de uma pela outra e imprima o resultado na tela. **Use atribuição simples**.

```
x = int(input("Digite o valor de x: "))
y = int(input("Digite o valor de y: "))
aux = x
x = y
y = aux
print(x, y)
```

Exercício #4

Modifique o programa anterior para que ele leia duas variáveis inteiras e troque os valores de uma pela outra sem usar variável auxiliar. **Use atribuição múltipla**.

```
x, y = int(input("Digite x: ")), int(input("Digite y: "))
x, y = y, x
print(x, y)
```