



# **INFORMÁTICA PARA ENGENHARIA**

## **MODULARIZAÇÃO E ORGANIZAÇÃO DE CÓDIGO**

Prof. Dr. Daniel Caetano

2018 - 2

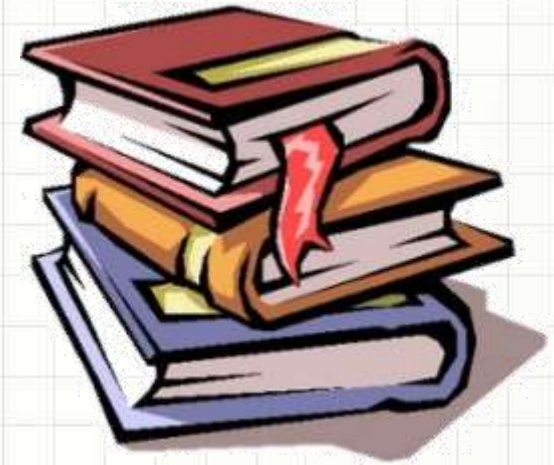
# Objetivos

- Entender a utilidade das funções
- Compreender o escopo das variáveis
- Capacitar o aluno para criar suas próprias funções

- **Atividades Aula 8 – SAVA!**
- **Estudar para Prova!**



# Material de Estudo



---

## Material

## Acesso ao Material

Notas de Aula e  
Apresentação

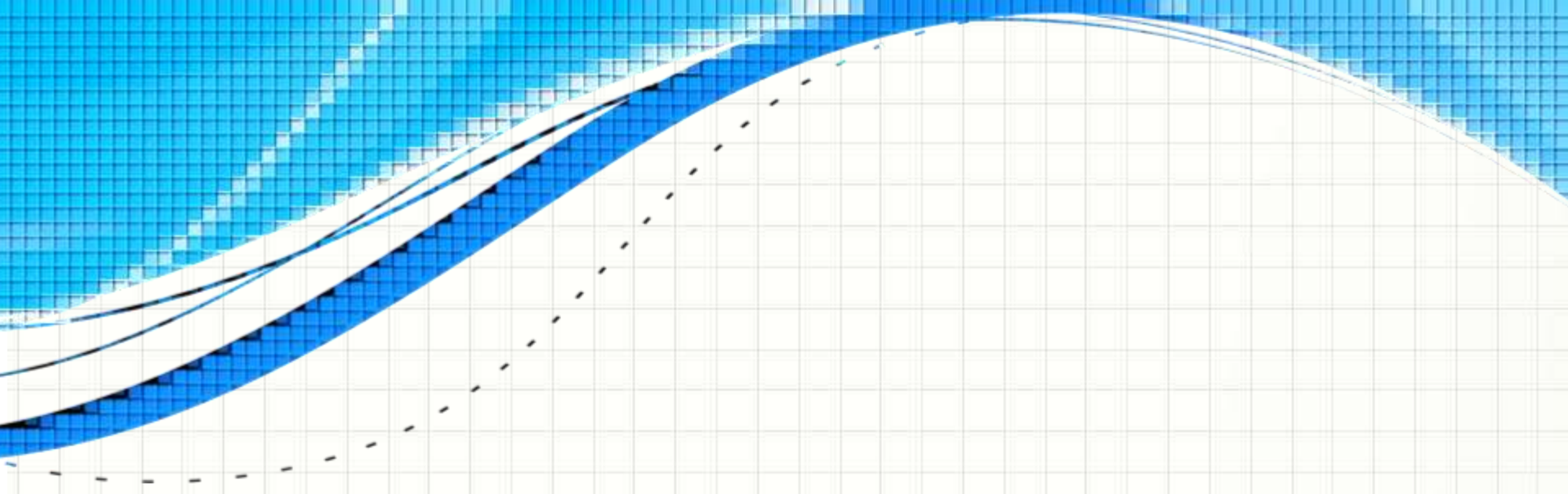
<http://www.caetano.eng.br/>  
(Informática para Engenharia – Aula 8)

Material Didático

Lógica de Programação, págs 173 a 187.

Biblioteca Virtual

“Lógica de Programação – Fundamentos da  
Programação de Computadores”, págs 7 a 47.



# FUNÇÕES SIMPLES

# Funções Simples

- Situação: imprimir 5x o seguinte texto:  
**Sistema de Impressão v.1.0, (c) Daniel Caetano**
- Um jeito de fazer seria usar vários “print”:

aula08ex01.py

```
# Imprime 5x uma mensagem  
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)  
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)  
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)  
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)  
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)
```

**E se eu quiser mudar a versão?**

# Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula08ex01a.py

```
# Imprime 5x uma mensagem
```

```
def mostra_mensagem(): 
```

```
    print ("Sistema de Impressão v.1.0, (c) Daniel Caetano")
```

**Indentação**

**Vamos  
experimental?**

# Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula08ex01a.py

```
# Imprime 5x uma mensagem
```

```
def mostra_mensagem():
```

```
    print ("Sistema de Impressão v.1.0, (c) Daniel Caetano")
```

```
mostra_mensagem()
```

**“chamar” a função:**

Solicitar ao computador que a execute

Observe o uso dos parênteses!

# Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula08ex01a.py

```
# Imprime 5x uma mensagem
```

```
def mostra_mensagem():
```

```
    print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

**E para mudar a versão?**

```
# Repita quantas vezes quiser!
```



# Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula08ex01a.py

## FUNÇÃO

```
# Imprime 5x uma mensagem
```

```
def mostra_mensagem():
```

```
    print ("Sistema de Impressão v.1.0, (c) Daniel Caetano")
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
# F
```

**Declaração da Função:**  
define o nome e o  
código da função

# Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula08ex01a.py

```
# Imprime 5x uma mensagem
```

```
def mostra_mensagem():
```

```
    print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
mostra_mensagem()
```

```
# Rep
```



**As funções devem ser declaradas antes de serem chamadas!**

# Outro Exemplo de Função

- Crie a função para a assinatura do e-mail:

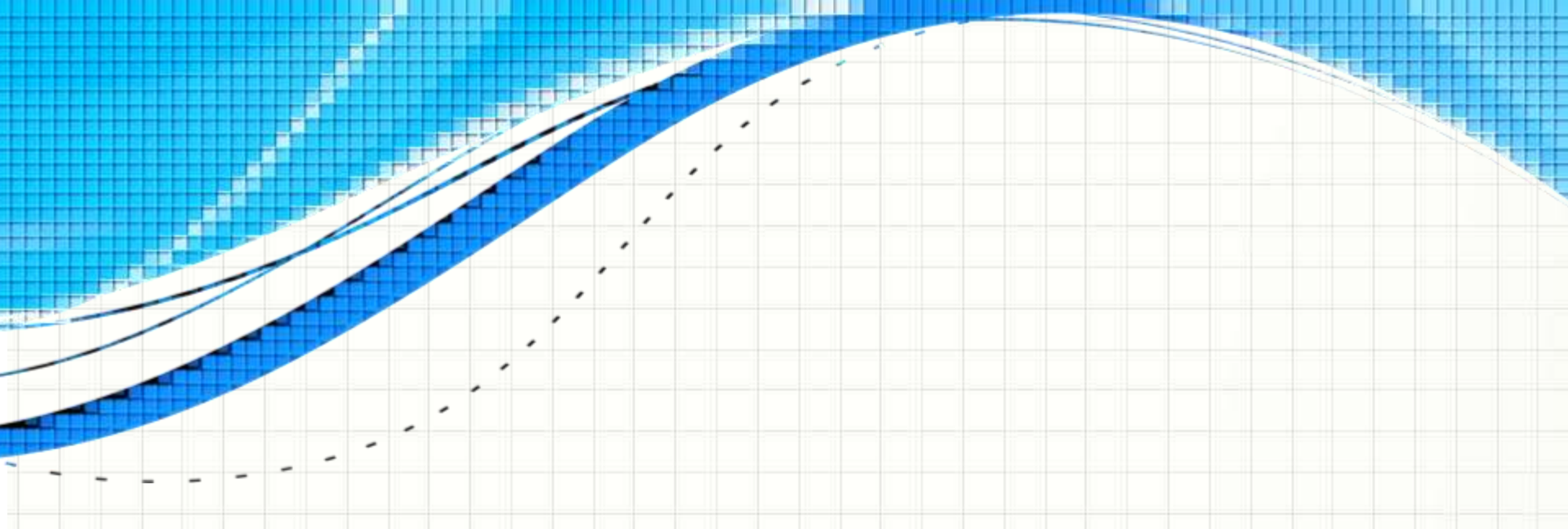
Atenciosamente,  
Prof. Daniel Caetano  
prof@caetano.eng.br

aula08ex02.py

```
# Imprime a assinatura do e-mail  
def assinar():  
    print ("Atenciosamente,")  
    print ("Prof. Daniel Caetano")  
    print ("prof@caetano.eng.br")
```

```
assinar()
```

**Experimentemos!**



# **FUNÇÕES COM PARÂMETROS**

# Funções com Parâmetro

- Até agora, funções funcionam sempre igual!
  - Por exemplo: assinatura de um único professor

Atenciosamente,

Prof. Daniel Caetano

**assinar()**

- Como fazer função de assinatura genérica?

Atenciosamente,

Prof. *[nome]*

**assinar("Daniel Caetano")**

# Funções com Parâmetro

- Crie a função para a assinatura do e-mail:

Atenciosamente,  
Prof. *[nome]*

aula08ex03.py

```
# Imprime a assinatura do e-mail genérica
```

```
def assinar(nome):
```

```
    print ("Atenciosamente,")
```

```
    print ("Prof.", nome)
```

```
assinar()
```

Funcionou?

# Funções com Parâmetro

- Crie a função para a assinatura do e-mail:

Atenciosamente,  
Prof. *[nome]*

aula08ex03.py

```
# Imprime a assinatura do e-mail genérica
```

```
def assinar(nome):  
    print ("Atenciosamente,")  
    print ("Prof.", nome)
```

```
assinar("Daniel Caetano")
```

E agora?

# Funções com Vários Parâmetros

- Crie a função para a assinatura completa:

Atenciosamente,

Prof. *[nome]*

*[e-mail]*

aula08ex04.py

```
# Imprime a assinatura do e-mail genérica completa
```

```
def assinar(nome, email):  
    print ("Atenciosamente,")  
    print ("Prof.", nome)  
    print (email)
```

```
assinar("Daniel Caetano", "prof@caetano.eng.br")
```

**Funcionou?**



# Funções com Parâmetros

- Função para calcular  $IMC = P/A^2$

aula08ex05.py

```
# Calcula e imprime IMC
```

```
def imc(p, a):
```

```
    imc = p/a**2
```

```
    print("O IMC é:", imc)
```

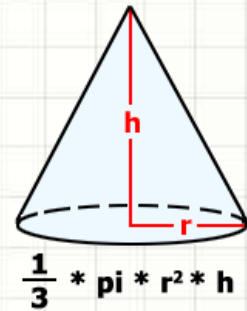
```
# Testa função
```

```
imc(50, 1.60)
```

```
imc(75, 1.70)
```

```
imc(85, 1.75)
```

# Funções com Parâmetros



- Calcular o volume de um cone

aula08ex06.py

```
# Imprime volume do cone
```

```
def volume_cone(r, h):
```

```
    v = (1/3) * 3.1415 * r**2 * h
```

```
    print("O colume do cone é: %.2f" %(v))
```

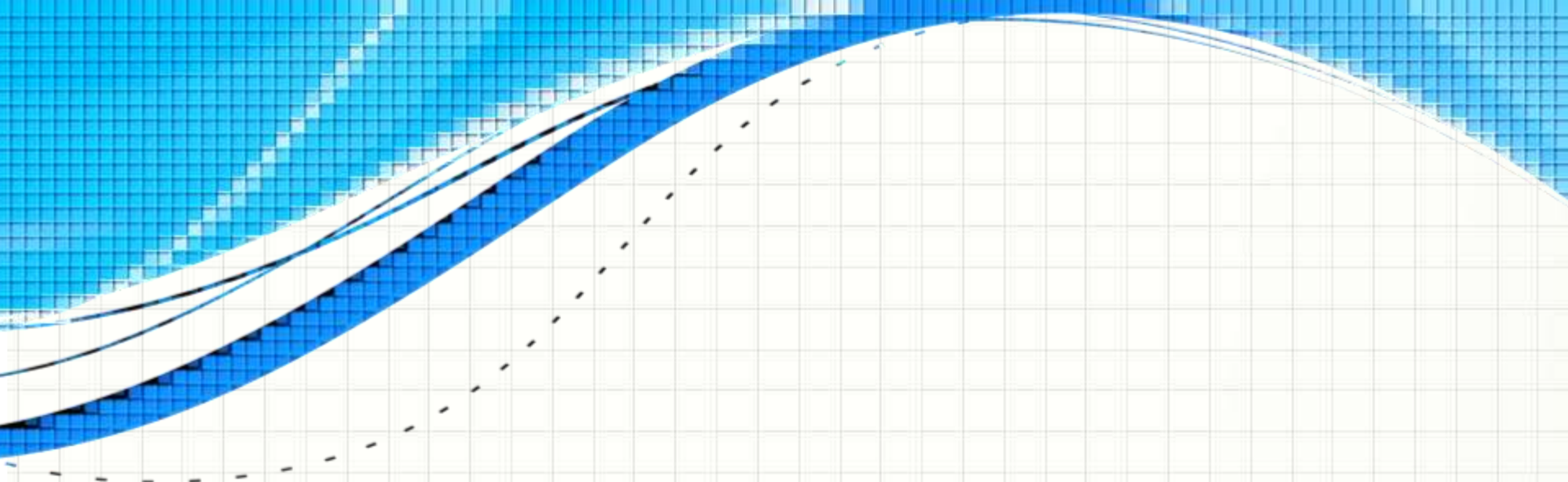
```
# Testa função
```

```
volume_cone(10, 2)
```

```
volume_cone(5, 3)
```

```
volume_cone(2.5, 10.2)
```

Definir uma função é como ensinar uma tarefa nova ao computador!



# **FUNÇÕES COM RETORNO**


# Funções com Retorno

- Vimos: fazer conta e mostrar o resultado
  - Posso usar o resultado depois?
  - Vamos testar?

aula08ex07.py

```
# Calcula área do círculo
def área_circulo(r):
    área = 3.141592 * r**2

# Testa função
área_circulo(2)
print("A área do círculo é:", área)
```



Funcionou?

As variáveis que existem na função são diferentes das que existem no programa principal!


# Funções com Retorno

- Vimos: fazer conta e mostrar o resultado
  - Posso usar o resultado depois?
  - Como resolver?

aula08ex07.py

```
# Calcula área do círculo
def área_circulo(r):
    área = 3.141592 * r**2
    return área

# Testa função
a = área_circulo(2)
print("A área do círculo é:", a)
```



**Funcionou?**

**Return** serve para “devolver” um valor de uma função para o programa principal

# Funções x Procedimentos

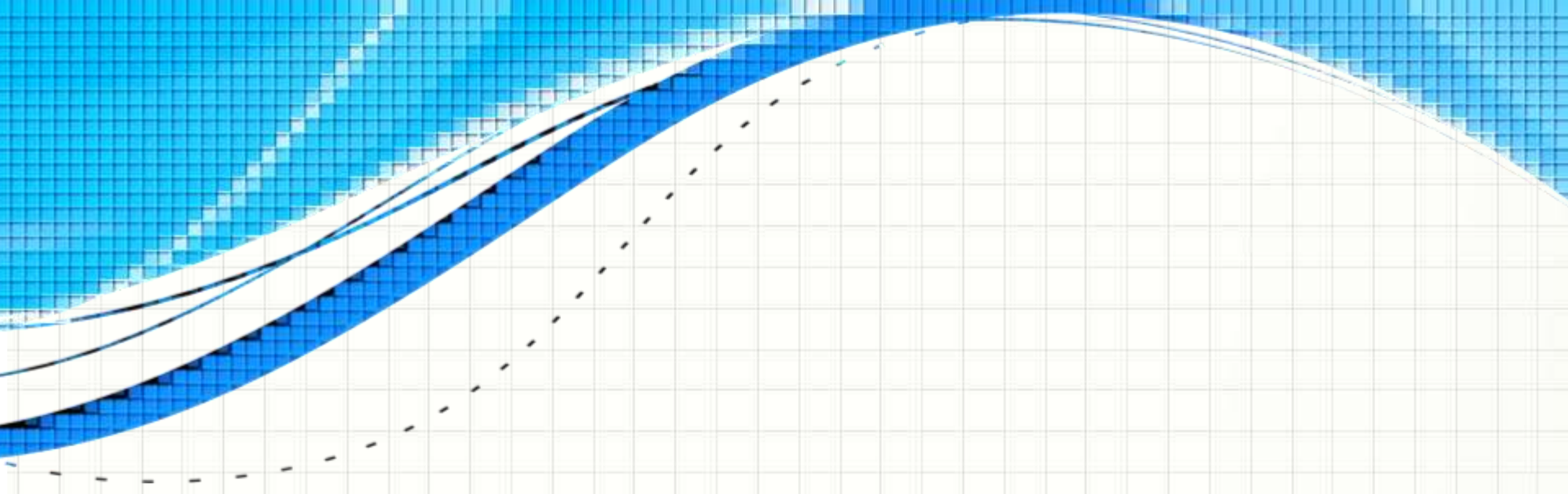
- “Funções” retornam resultados... ou não.
  - Alguns autores dão nomes diferentes

- **Procedimentos** (procedures)
  - Quando **não retorna** um valor

```
print(...)
```

- **Funções** (functions)
  - Quando **retorna** um valor

```
int(...)  
input(...)  
math.sin(n)
```



# ESCOPO DE VARIÁVEIS

# Escopo de Variáveis

- Variáveis diferentes podem ter mesmo nome
  - Seu professor chama “Daniel”
  - Com certeza você conhece mais algum “Daniel”
- Como diferenciar?
  - Contexto (ou Escopo)!
    - Na faculdade, “Daniel” é o professor
    - Em sua casa, “Daniel” pode ser seu irmão
- Em Python, qual o “escopo”?
  - Escopo principal (ou global, fora das funções)
  - Escopo da função (ou local)



# Exemplo de Escopo de Variáveis

- Vamos ver escopos na prática

aula08ex08.py

```
# Função
```

```
def aniversário(idade):  
    idade = idade + 1
```

**Escopo da função**  
**aniversário**

```
# Programa principal
```

```
idade = 10  
print("A idade antes do aniversário:", idade)  
aniversário(idade)  
print("A idade depois do aniversário:", idade)
```

**Escopo**  
**global**

**Funcionou?**

# Exemplo de Escopo de Variáveis

- Vamos ver escopos na prática

aula08ex08.py

```
# Função
```

```
def aniversário(idade):  
    idade = idade + 1  
    return idade
```

```
# Programa principal
```

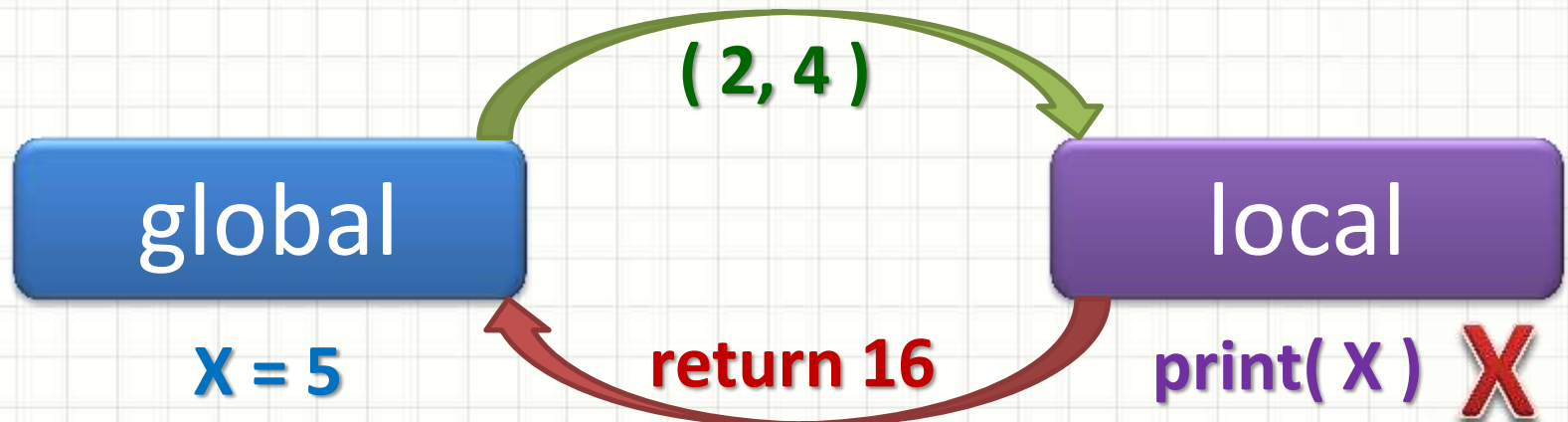
```
idade = 10  
print("A idade antes do aniversário:", idade)  
idade = aniversário(idade)  
print("A idade depois do aniversário:", idade)
```

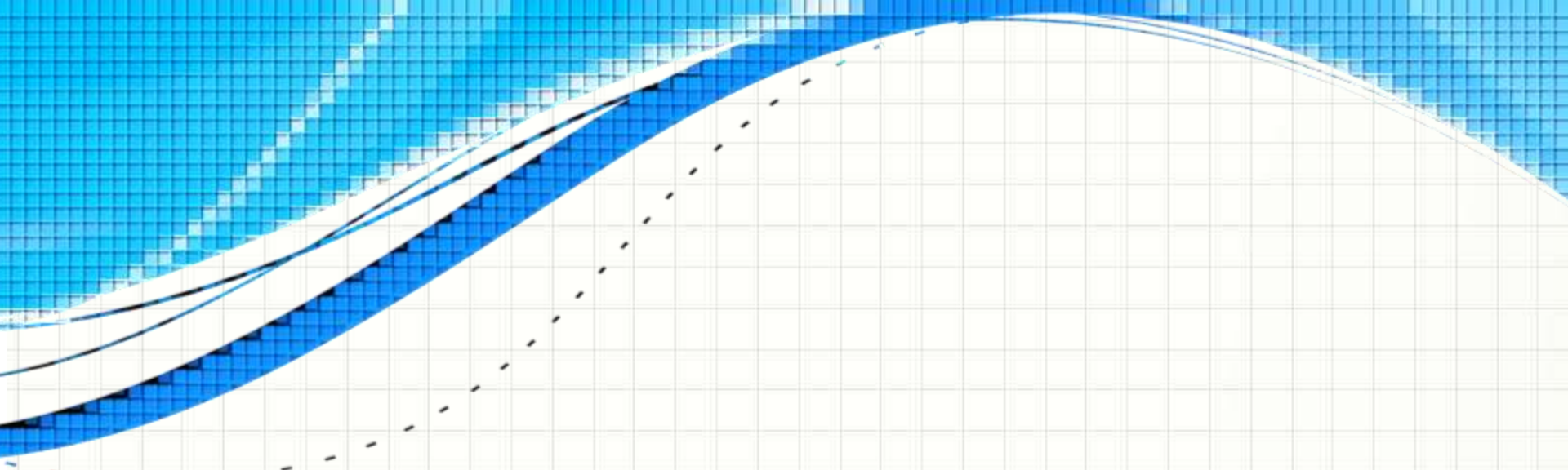
E agora?

Return transfere de um escopo para o outro!

# Resumindo Escopo de Variáveis

- Escopo = Vale Onde? Global x Local
- Variáveis
  - Só valem no escopo em que são criadas
- Passar dados de um escopo para outro?
  - Na chamada: parâmetros
  - Receber respostas: return





**EXEMPLO DE EXECUÇÃO:**

# **FUNÇÕES COM RETORNO DE VALORES**

# Função com Retorno

- Vejamos como funciona o programa abaixo:

aula08ex09.py

```
# Função que lê nome do usuário
```

```
def leitura_de_nome():
```

```
    pnome = input("Por favor, digite seu primeiro nome: ")
```

```
    snome = input("Por favor, digite seu sobrenome: ")
```

```
    nome = pnome + " " + snome
```

```
    return nome
```

```
# Programa de Boas Vindas
```

```
print("Programa Exemplo com Funções")
```

```
print("=====")
```

```
usuario = leitura_de_nome ()
```

```
print("Bom dia,", usuario, "!")
```

# Função com Retorno

- Vejamos como funciona o programa abaixo:

aula08ex09.py

```
# Função que lê nome do usuário
```

```
def leitura_de_nome():
```

```
    pnome = input("Por favor, digite seu primeiro nome: ")
```

```
    snome = input("Por favor, digite seu sobrenome: ")
```

```
    nome = pnome + " " + snome
```

```
    return nome
```

```
# Programa de Boas Vindas
```

```
print("Programa Exemplo com Funções")
```

```
print("=====")
```

```
usuario = leitura_de_nome ()
```

```
print("Bom dia,", usuario, "!")
```

# Função com Retorno

- Vejamos como funciona o programa abaixo:

aula08ex09.py

```
# Função que lê nome do usuário
```

```
def leitura_de_nome():
```

```
    pnome = input("Por favor, digite seu primeiro nome: ")
```

```
    snome = input("Por favor, digite seu sobrenome: ")
```

```
    nome = pnome + " " + snome
```

```
    return nome
```

```
# Programa de Boas Vindas
```

```
print("Programa Exemplo com Funções")
```

```
print("=====")
```

```
usuario = leitura_de_nome ()
```

```
print("Bom dia,", usuario, "!")
```

# Função com Retorno

- Vejamos como funciona o programa abaixo:

aula08ex09.py

```
# Função que lê nome do usuário
```

```
def leitura_de_nome():
```

```
    pnome = input("Por favor, digite seu primeiro nome: ")
    snome = input("Por favor, digite seu sobrenome: ")
    nome = pnome + " " + snome
    return nome
```

```
# Programa de Boas Vindas
```

```
print("Programa Exemplo com Funções")
print("=====")
usuario = leitura_de_nome()
print("Bom dia,", usuario, "!")
```



# Função com Retorno

- Vejamos como funciona o programa abaixo:

aula08ex09.py

```
# Função que lê nome do usuário
```

```
def leitura_de_nome():
```

```
     pnome = input("Por favor, digite seu primeiro nome: ")
```

```
     snome = input("Por favor, digite seu sobrenome: ")
```

```
    nome = pnome + " " + snome
```

```
    return nome
```

```
# Programa de Boas Vindas
```

```
print("Programa Exemplo com Funções")
```

```
print("=====")
```

```
 usuario = leitura_de_nome ()
```

```
print("Bom dia,", usuario, "!")
```

# Função com Retorno

- Vejamos como funciona o programa abaixo:

aula08ex09.py

```
# Função que lê nome do usuário
```

```
def leitura_de_nome():
```

```
    pnome = input("Por favor, digite seu primeiro nome: ")
```

```
     snome = input("Por favor, digite seu sobrenome: ")
```

```
     nome = pnome + " " + snome
```

```
    return nome
```

```
# Programa de Boas Vindas
```

```
print("Programa Exemplo com Funções")
```

```
print("=====")
```

```
 usuario = leitura_de_nome()
```

```
print("Bom dia,", usuario, "!")
```

# Função com Retorno

- Vejamos como funciona o programa abaixo:

aula08ex09.py

```
# Função que lê nome do usuário
```

```
def leitura_de_nome():
```

```
    pnome = input("Por favor, digite seu primeiro nome: ")
```

```
    snome = input("Por favor, digite seu sobrenome: ")
```

```
    nome = pnome + " " + snome
```

```
    return nome
```

```
# Programa de Boas Vindas
```

```
print("Programa Exemplo com Funções")
```

```
print("=====")
```

```
usuario = leitura_de_nome()
```

```
print("Bom dia,", usuario, "!")
```

# Função com Retorno

- Vejamos como funciona o programa abaixo:

aula08ex09.py

Experimente no <http://pythontutor.com/visualize.html>

```
# Função que lê nome do usuário
```

```
def leitura_de_nome():
```

```
    pnome = input("Por favor, digite seu primeiro nome: ")
```

```
    snome = input("Por favor, digite seu sobrenome: ")
```

```
    nome = pnome + " " + snome
```

```
    return nome
```

```
# Programa de Boas Vindas
```

```
print("Programa Exemplo com Funções")
```

```
print("=====")
```

```
usuario = leitura_de_nome ()
```

```
print("Bom dia,", usuario, "!")
```



# **FUNÇÕES NA PRÁTICA**

# Criando Funções

- **Passo 1:** criar um programa que calcule e imprima o perímetro de um círculo de raio 2
  - $P = 2 \cdot \pi \cdot R$
  - $\pi = 3,141592$

# Criando Funções

- **Passo 2:** transformar o cálculo em uma função chamada **perímetro**

# Criando Funções

- **Passo 2:** transformar o cálculo em uma função chamada **perimetro**
  - As variáveis criadas dentro da função só existem dentro desta função
  - Elas são chamadas **variáveis locais**
  - Não é possível acessar uma variável local a não ser de dentro da própria função
  - Os valores das variáveis locais **são destruídos** quando a função finaliza



# Criando Funções

- **Passo 3:** modificar a função **perímetro** para que ela para que ela retorne o resultado, ao invés de imprimi-lo

# Criando Funções

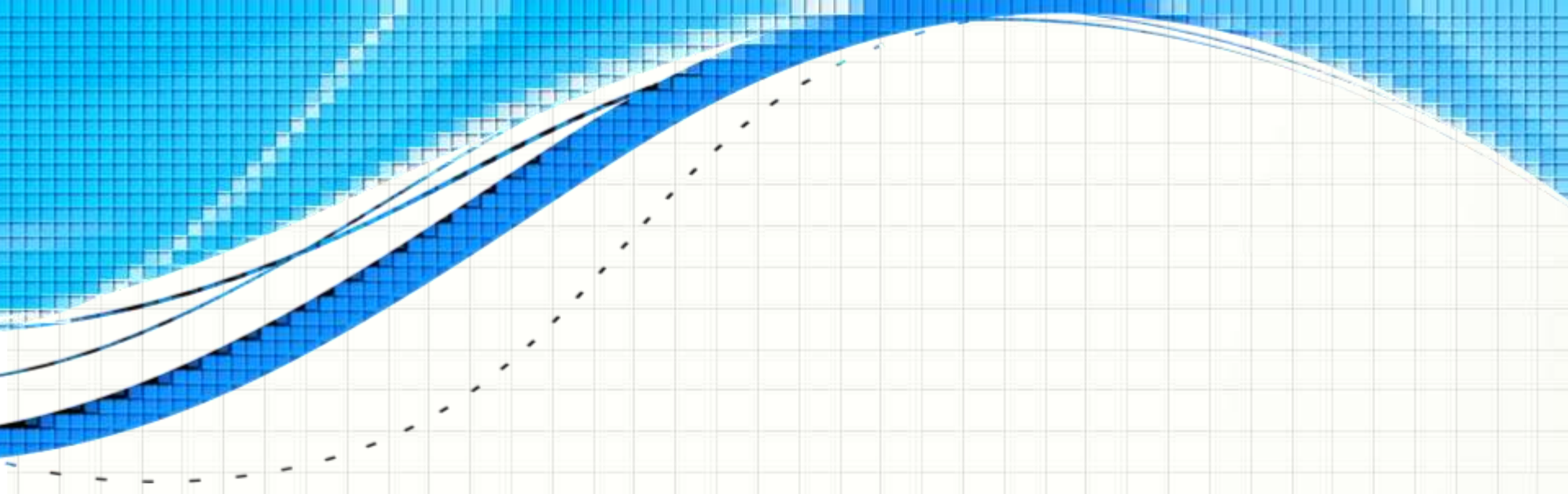
- **Passo 4:** modificar a função **perímetro** para que ela receba o raio do círculo como parâmetro
  - Chame a função com o valor 2 para o raio

# Criando Funções

- **Passo 4:** modificar a função **perímetro** para que ela receba o raio do círculo como parâmetro
  - Os parâmetros funcionam como variáveis locais
  - O valor fornecido como parâmetro (o raio) é **copiado** para essa “variável local”

# Criando Funções

- **Passo 5:** Simule a execução do programa.



# CONCLUSÕES

# Resumo

- O uso de funções simplifica o reaproveitamento de código
- As variáveis possuem um escopo
- As funções podem receber parâmetros e podem retornar resultados

---

**SAVA!**

- **Estudar para a AV1!**



**PERGUNTAS?**