

# PYTHON: Função

**Jean Eduardo Glazar**

*Programação I*

*Curso de Sistemas de Informação  
Ifes Campus Colatina*

# Função

- Funções são blocos de código identificado por um nome.
- Um problema maior pode ser dividido em diversos subproblemas.
- Reutilização de código.
- Podem receber parâmetros pré-determinados.
- Podem retornar valores ou objetos.

# Função

# PROGRAMA  
# PRINCIPAL

Comando 1

Funcao\_A

Comando 2

Comando 3

Funcao\_B

Comando 4

Funcao\_C

Funcao\_A

# Funcao\_A

Inicio\_A

Bloco de comandos A

Fim\_A

# Funcao\_B

Inicio\_B

Bloco de comandos B

Funcao\_C

Fim\_B

# Funcao\_C

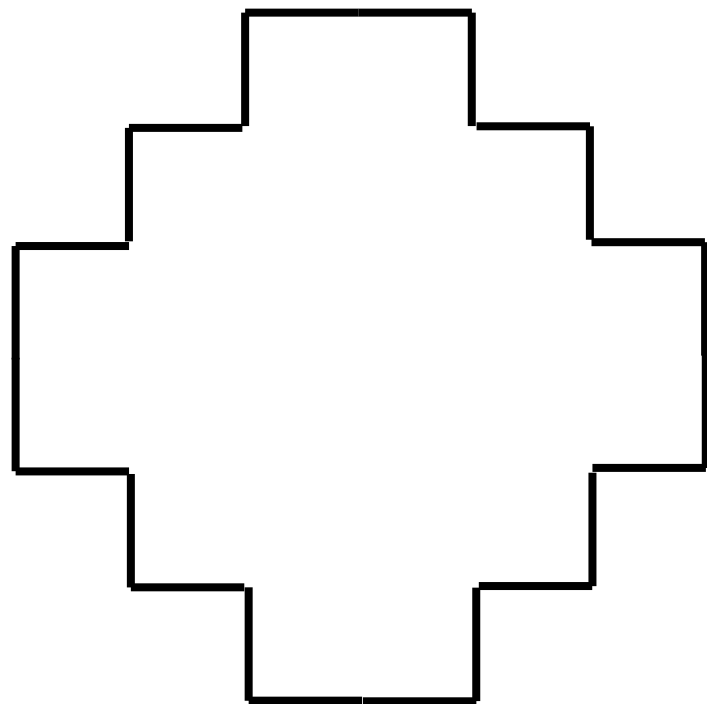
Inicio\_C

Bloco de comandos C

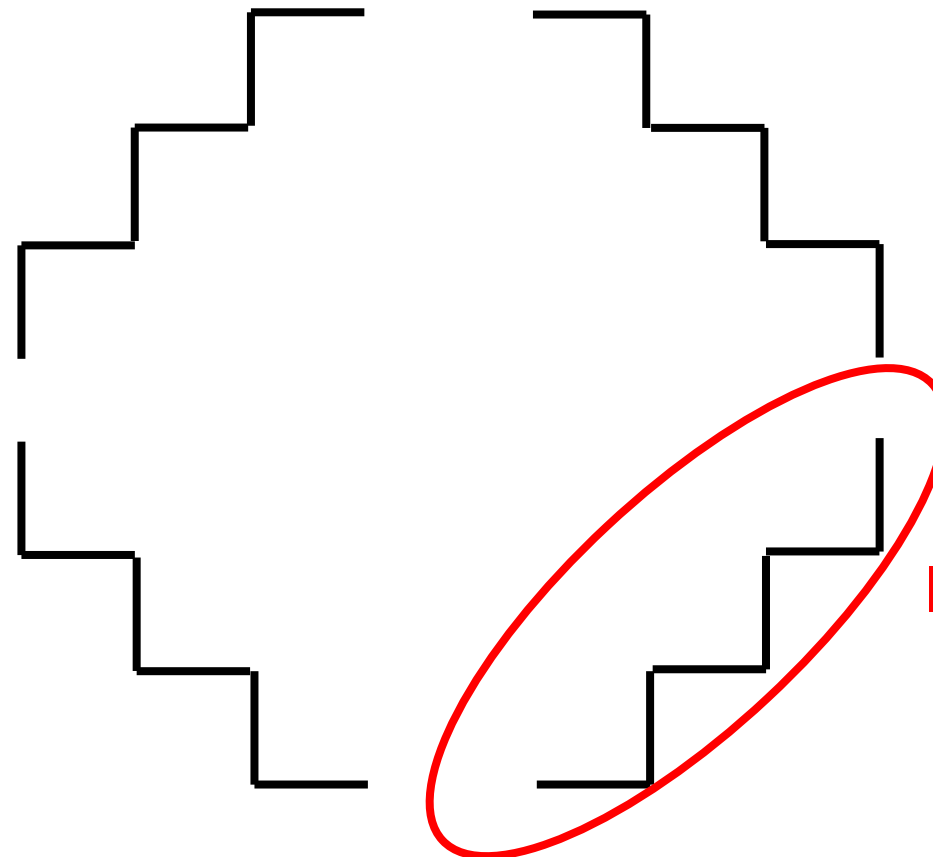
Fim\_C



# Divisão em problemas menores



# Divisão em problemas menores



# PROGRAMA  
# PRINCIPAL

4 x

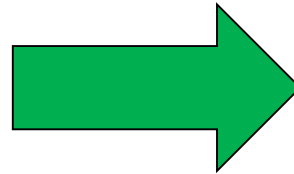
Função Lado

Função Lado

# Programa Principal - Simplificado

# PROGRAMA  
# PRINCIPAL

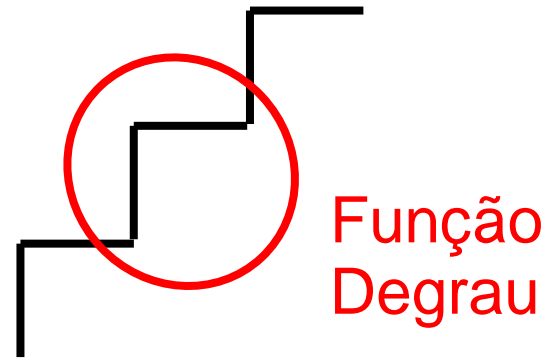
Funcao Lado  
Virar 90º horário  
Funcao Lado  
Virar 90º horário  
Funcao Lado  
Virar 90º horário  
Funcao Lado



# PROGRAMA  
# PRINCIPAL

```
cont = 1
while cont <= 4 :
    Funcao Lado
    Virar 90º horário
    cont = cont + 1
```

# Função Lado



# Função Lado

3 x

Função Degrau



# Programa Completo

# PROGRAMA  
# PRINCIPAL

```
cont = 1
while cont <= 4 :
    Funcao Lado
    Virar 90º horário
    cont = cont + 1
```

# Função Lado

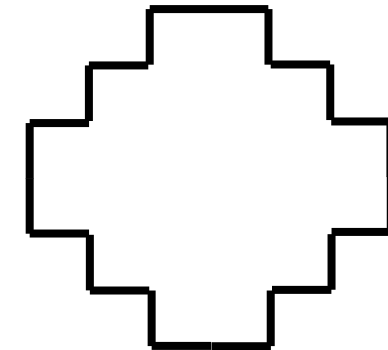
```
cont = 1
while cont <= 3 :
    Funcao Degrau
    cont = cont + 1
```

# Função Degrau

Desenhar:



# Resultado





# Sintaxe

**def nomeFuncao ():**

**comando 1**

**comando 2**

**...**

**comando N**



# Indentação

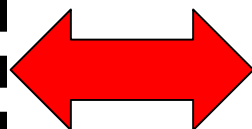
**def nomeFuncao ():**

**comando 1**

**comando 2**

**...**

**comando N**



**Outro comando**

# Dois pontos



NÃO ESQUEÇA!



# Exemplo

```
def cabecalho() :  
    print("-----")  
    print("\t AULA DE FUNÇÃO")  
    print("Autor: Jean Eduardo Glazar")  
    print("IFES Campus Colatina")  
    print("PROGRAMAÇÃO I")  
    print("-----")
```

# Chamada da Função

- A função não executa sozinha.
- É preciso “chamar” a função.
- A chamada da função é realizada usando seu nome.

`cabecalho ()`

# Exemplo

```
def cabecalho():  
    print("-----")  
    print("\t AULA DE FUNÇÃO")  
    print("Autor: Jean Eduardo Glazar")  
    print("IFES Campus Colatina")  
    print("PROGRAMAÇÃO I")  
    print("-----")
```

```
# Inicio do programa começa aqui  
cabecalho()
```

# Função com retorno

- Uma função pode retornar valores.
- O valor de retorno pode ser:
  - **Número (inteiro ou real)**
  - **String**
  - **Booleano (True ou False)**
  - **Lista**
  - **Objeto**
  - **Etc.**



# Função com retorno

- Ao encontrar o comando **return**, a função termina imediatamente e o controle do programa volta ao ponto onde a função foi chamada.
- Se uma função chega a seu fim sem nenhum valor de retorno ter sido especificado, o valor de retorno é **None**

# Sintaxe com retorn

**def** nomeFuncao ():

comando 1

comando 2

...

comando N

**return** valor

# Exemplo: ler 3 notas

```
n1 = float( input("Digite uma nota: "))
while n1 < 0 or n1 > 10 :
    print("Nota inválida!")
    n1 = float( input("Digite uma nota: "))

n2 = float( input("Digite uma nota: "))
while n2 < 0 or n2 > 10 :
    print("Nota inválida!")
    n2 = float( input("Digite uma nota: "))

n3 = float( input("Digite uma nota: "))
while n3 < 0 or n3 > 10 :
    print("Nota inválida!")
    n3 = float( input("Digite uma nota: "))
```



# Exemplo com função

```
def lerNota():  
    n = float(input("Digite uma nota: "))  
    while n < 0 or n > 10 :  
        print("Nota inválida!")  
        n = float(input("Digite uma nota: "))  
    return n
```

# Função com retorno

- Quando uma função retorna um valor, é preciso guardar esse valor retornado.
- Cria-se uma variável para guardar esse valor.
- Essa variável recebe a chamada da função.

`n1 = lerNota()`

variável

Chamada da  
função

# Função com retorno

# PROGRAMA PRINCIPAL

. . .

n1 = lerNota()

. . .

Retorno

n1 = n

def lerNota():

n = float(input("Nota: "))

while n < 0 or n > 10 :

print("Nota inválida!")

n = float(input("Nota: "))

return n



# Ler 3 notas: com função

```
def lerNota():  
    n = float(input("Digite uma nota: "))  
    while n < 0 or n > 10 :  
        print("Nota inválida!")  
        n = float(input("Digite uma nota: "))  
    return n
```

## Início do programa começa aqui

```
n1 = lerNota()
```

```
n2 = lerNota()
```

```
n3 = lerNota()
```

```
...
```




# Outro exemplo: fatorial

```
def fatorial():  
    n = 10  
    fat = 1  
    while n > 1:  
        fat = fat * n  
        n = n - 1  
    return fat  
  
## Inicio do programa começa aqui  
res = fatorial()  
print("Fatorial = %d" %res)
```

# Outro exemplo: fatorial

```
def fatorial():  
    n = 10  
    fat = 1  
    while n > 1:  
        fat = fat * n  
        n = n - 1  
    return fat
```



PROBLEMA: Se eu quiser o fatorial de outro número ??!!

Da forma que está não posso aproveitar a função

```
## Inicio do programa começa aqui  
res = fatorial()  
print("Fatorial = %d" % res)
```

# Função com parâmetros

- Os parâmetros são informações que passamos para dentro da função.
- Podemos ter vários parâmetros.
- Identificados por um nome, separados por vírgula.

# Função com parâmetros

# PROGRAMA PRINCIPAL

```
. . .  
n1 = minhaFuncao()  
. . .
```

PARÂMETROS  
(entrada)

```
def minhaFuncao():  
    . . .  
    return n
```

RETORNO  
(saída)

# Sintaxe: função com parâmetros

```
def nomeFuncao (par1, par2, ... ):
    comando 1
    comando 2
    ...
    comando N
return valor
```

# Exemplo: fatorial

```
def fatorial(n):
```

```
    fat = 1
```

```
    while n > 1:
```

```
        fat = fat * n
```

```
        n = n - 1
```

```
    return fat
```

```
# Início do programa
```

```
res = fatorial(10)
```

```
print("Fatorial = %d" %res)
```

$n = 10$

O valor de ***n*** vem de fora da função

O valor 10 é enviado para a função:



# Exemplo: fatorial

- Podemos agora chamar a função fatorial várias vezes com valores diferentes.
- Por exemplo: mostrar o fatorial de 1 até 15.



# Exemplo: fatorial

```
# Inicio do programa
```

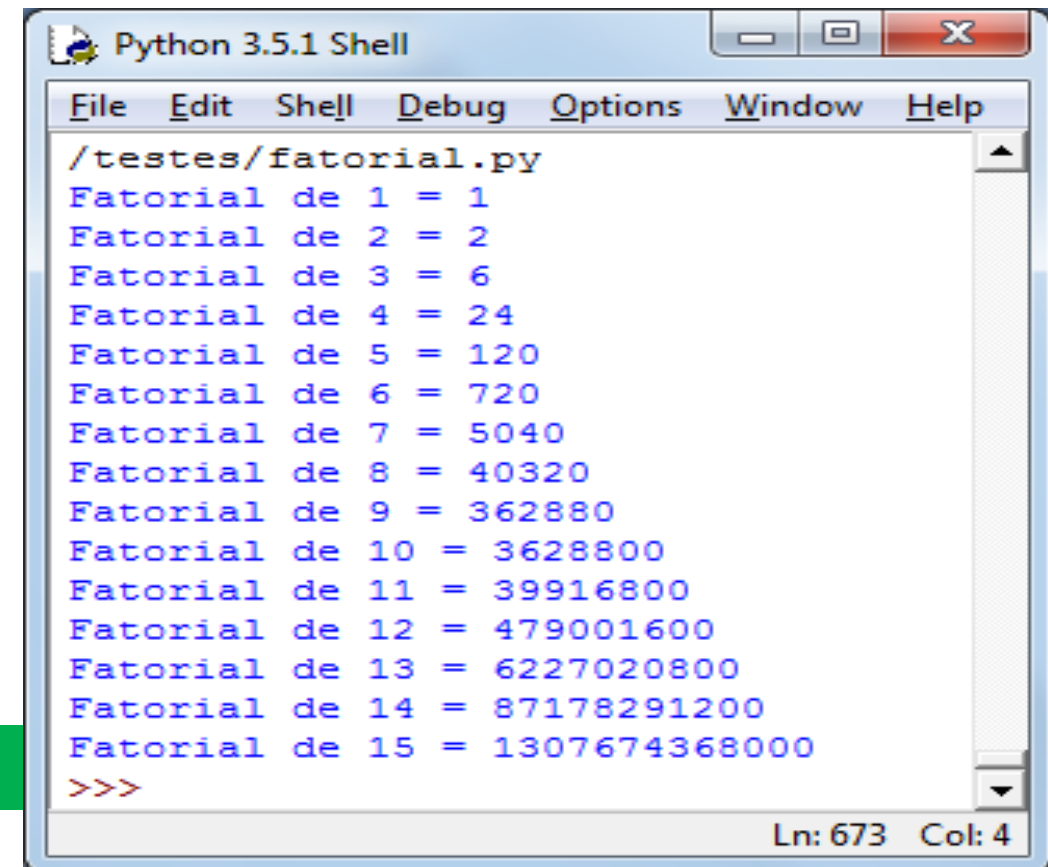
```
cont = 1
```

```
while cont <= 15:
```

```
    res = fatorial(cont)
```

```
    print("Fatorial de %d = %d" % (cont, res))
```

```
    cont = cont + 1
```



Python 3.5.1 Shell

```
File Edit Shell Debug Options Window Help
/testes/fatorial.py
Fatorial de 1 = 1
Fatorial de 2 = 2
Fatorial de 3 = 6
Fatorial de 4 = 24
Fatorial de 5 = 120
Fatorial de 6 = 720
Fatorial de 7 = 5040
Fatorial de 8 = 40320
Fatorial de 9 = 362880
Fatorial de 10 = 3628800
Fatorial de 11 = 39916800
Fatorial de 12 = 479001600
Fatorial de 13 = 6227020800
Fatorial de 14 = 87178291200
Fatorial de 15 = 1307674368000
>>>
```

Ln: 673 Col: 4

# Dicas

Dado o enunciado de um problema, siga os seguintes passos:

1. Identifique os verbos no enunciado. Provavelmente, cada verbo será uma função.
2. Comece pelo programa principal, admitindo que as funções estejam prontas. Nesse momento você deve identificar as informações dos parâmetros e o retorno de cada função.
3. Implemente uma função de cada vez (se possível) e teste

# Exercício 1

Faça um programa com funções para ler o nome do aluno, ler e validar suas três notas. Calcule a média final do aluno e imprima: “APROVADO” se a média for maior ou igual a 7; “REPROVADO” se a média for menor que 6; e “PROVA FINAL” se a média estiver entre 6 e 7. Imprima também as notas na ordem decrescente.

# Exercício 1

**Faça** um programa com funções para **ler** o nome do aluno, **ler** e **validar** suas três notas. **Calcule** a média final do aluno e **imprima**: “APROVADO” se a média for maior ou igual a 7; “REPROVADO” se a média for menor que 6; e “PROVA FINAL” se a média estiver entre 6 e 7. **Imprima** também as notas na ordem decrescente.

- **lerNome**
- **lerNota** (ler e validar uma nota)
- **calcularMedia**
- **imprimirResultado**
- **imprimirDecrescente**

# Exercício 1: programa principal

```
# Inicio do programa
```

```
nome = lerNome()
```

```
n1 = lerNota()
```

```
n2 = lerNota()
```

```
n3 = lerNota()
```

```
media = calcularMedia(n1, n2, n3)
```

```
imprimirResultado(media)
```

```
imprimirDecrescente(n1, n2, n3)
```

# Exercício 2

Desenvolver um programa para verificar a nota do aluno em uma prova com 10 questões de múltipla escolha. Primeiro leia e valide (de “A” até “E”) o gabarito de cada questão e armazene em 10 variáveis (por exemplo, gab1, gab2, etc.). Em seguida, o programa deve perguntar ao aluno a resposta de cada questão e comparar com o gabarito da prova e assim calcular o total de acertos e a nota (atribuir 1 ponto por resposta certa). Após cada aluno utilizar o sistema deve ser feita uma pergunta se outro aluno vai utilizar o sistema. Após todos os alunos terem respondido informar: a maior e a menor nota; o total de alunos que utilizaram o sistema; a média das notas da turma.

# Exercício 2

Desenvolver um programa para verificar a nota do aluno em uma prova com 10 questões de múltipla escolha. Primeiro **leia e valide (de “A” até “E”)** o gabarito de cada questão e armazene em 10 variáveis (por exemplo, gab1, gab2, etc.). Em seguida, o programa deve **perguntar ao aluno a resposta de cada questão** e **comparar** com o gabarito da prova e assim **calcular o total de acertos** e a nota (atribuir 1 ponto por resposta certa). Após cada aluno utilizar o sistema deve **ser feita uma pergunta** se outro aluno vai utilizar o sistema. Após todos os alunos terem respondido informar: a **maior** e a **menor** nota; o **total** de alunos que utilizaram o sistema; a **média** das notas da turma.



# Exercício 2

- **lerResposta :**  
Chamar 10 vezes para o gabarito e depois mais 10 vezes para as respostas do aluno
- **comparar:**  
Compara a resposta do aluno com a do gabarito
- **calcularTotal :**  
Somar a pontuação (talvez não precise ser função)
- **desejaContinuar :**
- **calcularMedia:**
- **Variáveis: maior, menor, somaNotas e total de alunos.**

# Variáveis locais e globais

```
def alterar():  
    x = 10  
    print("x dentro da função = %d" %x)
```

```
# Inicio do programa
```

```
x = 20  
print("Antes de alterar: %d" %x)  
alterar()  
print("Depois de alterar: %d" %x)
```

São variáveis  
diferentes

Saída:

Antes de alterar: 20  
x dentro da função = 10  
Depois de alterar: 20



# Parâmetros opcionais

- Se não for passado o parâmetro será igual ao *default* definido na função;
- Os parâmetros opcionais devem ficar após aqueles que não são opcionais.

# Exemplo

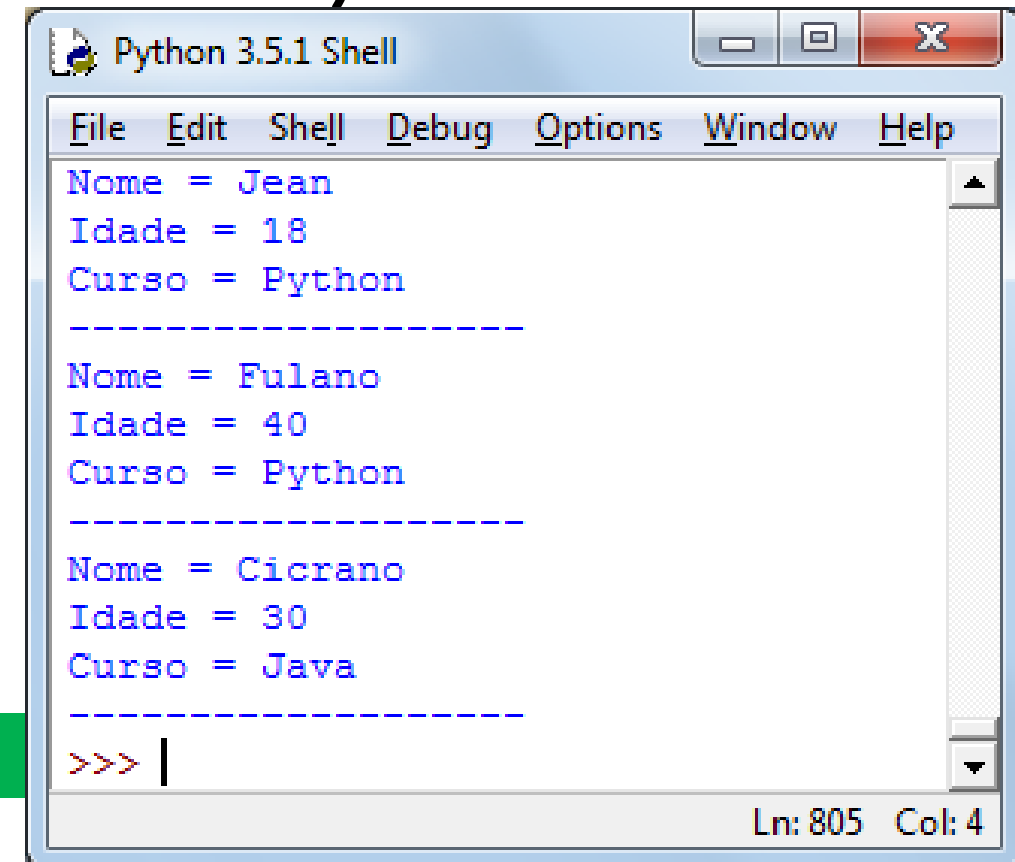
```
def info(nome, idade=18, curso="Python") :  
    print("Nome = %s" %nome)  
    print("Idade = %d" %idade)  
    print("Curso = %s" %curso)  
    print("-----")
```

# Inicio do programa

```
info("Jean")
```

```
info("Fulano", 40)
```

```
info("Cicrano", 30, "Java")
```



The screenshot shows a 'Python 3.5.1 Shell' window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The output of the code is displayed in the main text area, showing the results of three function calls. The first call for 'Jean' shows age 18 and course 'Python'. The second call for 'Fulano' shows age 40 and course 'Python'. The third call for 'Cicrano' shows age 30 and course 'Java'. The window status bar at the bottom right indicates 'Ln: 805 Col: 4'.

```
Python 3.5.1 Shell  
File Edit Shell Debug Options Window Help  
Nome = Jean  
Idade = 18  
Curso = Python  
-----  
Nome = Fulano  
Idade = 40  
Curso = Python  
-----  
Nome = Cicrano  
Idade = 30  
Curso = Java  
-----  
>>> |  
Ln: 805 Col: 4
```

# Exemplo: erro

```
# Inicio do programa  
info()
```

**ERRO:**

```
TypeError: info() missing 1 required  
positional argument: 'nome'
```

# Exercícios

“Na procura de conhecimentos, o primeiro passo é o silêncio, o segundo ouvir, o terceiro relembrar, o quarto praticar e o quinto ensinar aos outros.”

Pensamento Judaico