



Engenharia de Software

Laboratório de Programação com interfaces com o usuário

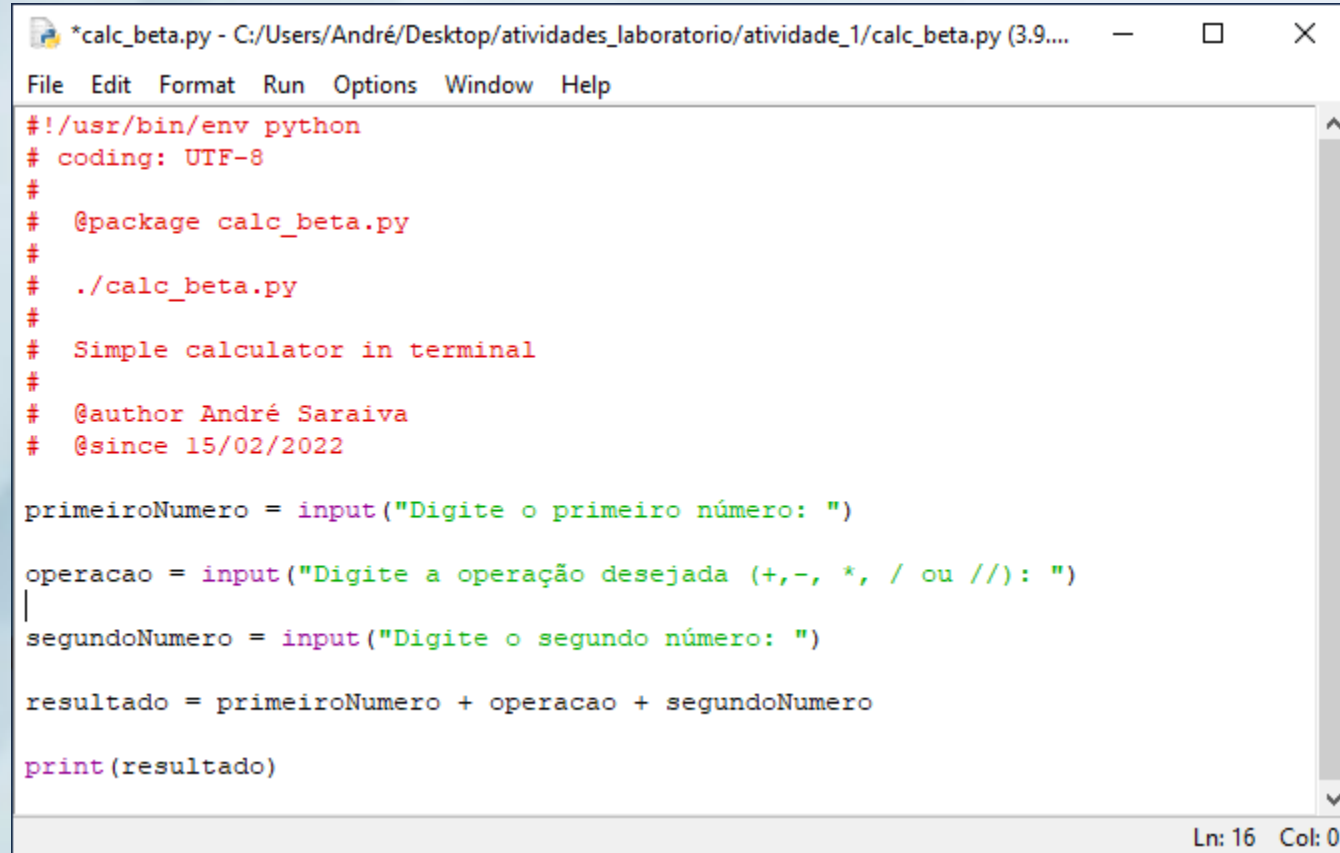
Prof. M.Sc. André Saraiva

2022

Atividade – Aula Passada

- Criar uma calculadora no terminal para as operações básicas (soma, subtração, multiplicação e divisão). Seu software deve:
 - Permitir a entrada de um número;
 - Permitir a entrada de uma das operações citadas;
 - Soma(+), subtração(-), multiplicação(*) e divisão(/ ou //).
 - Permitir a entrada do segundo número;
 - Apresentar na saída padrão (vídeo) o resultado.
- Neste momento não iremos fazer nenhum tratamento ou condicionais

Atividade – Aula Passada



The image shows a screenshot of a Python script editor window. The title bar indicates the file is `*calc_beta.py` located at `C:/Users/André/Desktop/atividades_laboratorio/atividade_1/calc_beta.py` using Python 3.9. The menu bar includes `File`, `Edit`, `Format`, `Run`, `Options`, `Window`, and `Help`. The script content is as follows:

```
#!/usr/bin/env python
# coding: UTF-8
#
# @package calc_beta.py
#
# ./calc_beta.py
#
# Simple calculator in terminal
#
# @author André Saraiva
# @since 15/02/2022

primeiroNumero = input("Digite o primeiro número: ")

operacao = input("Digite a operação desejada (+, -, *, / ou //): ")
|
segundoNumero = input("Digite o segundo número: ")

resultado = primeiroNumero + operacao + segundoNumero

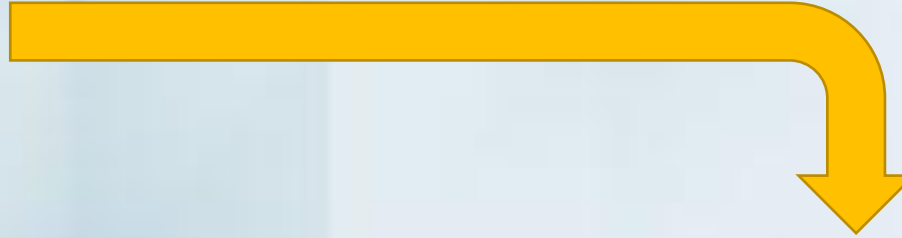
print(resultado)
```

The status bar at the bottom right shows the cursor position: `Ln: 16 Col: 0`.

Atividade – Aula Passada

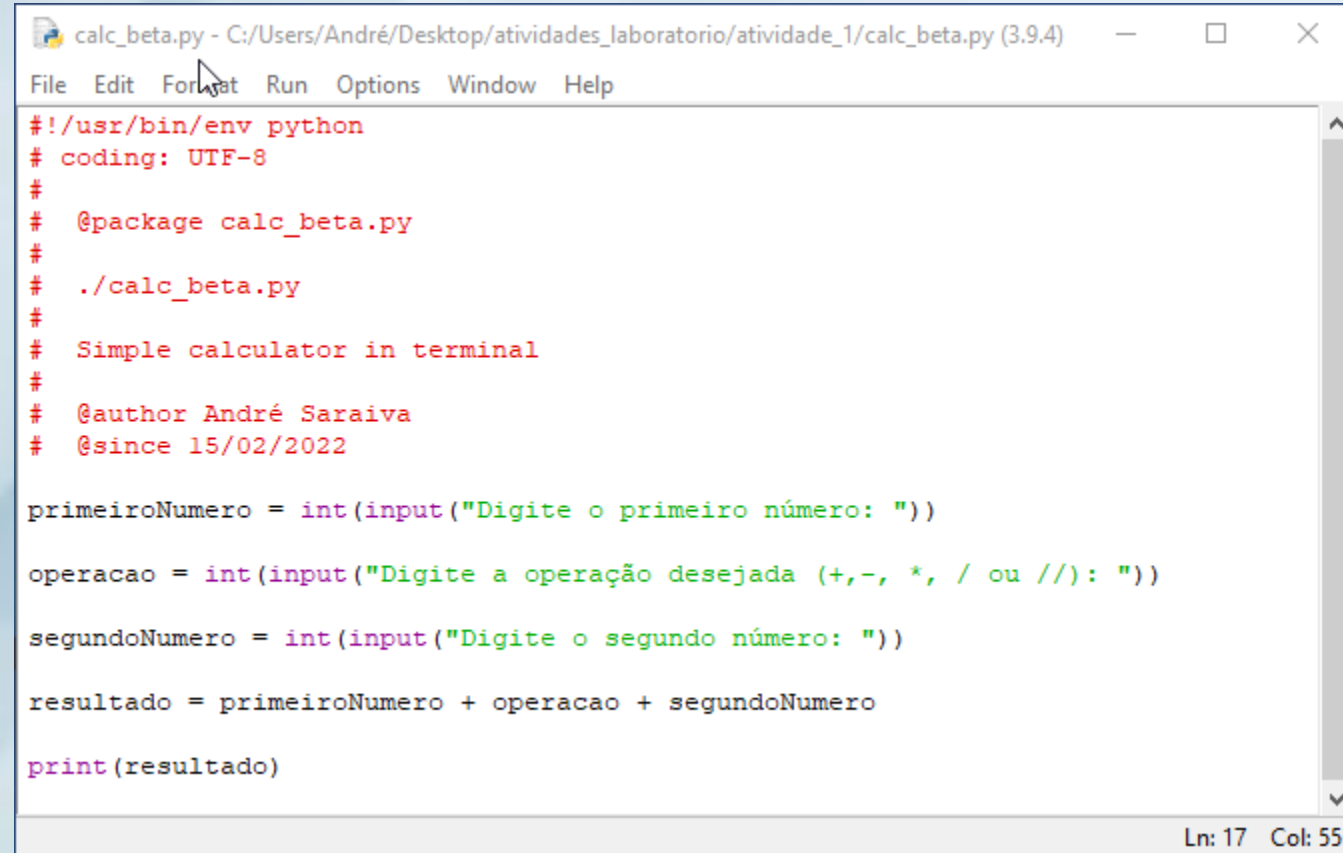
```
*calc_beta.py - C:/Users/André/Desktop/atividades_laboratorio/atividade_1/calc_beta.py (3.9...
File Edit Format Run Options Window Help
#!/usr/bin/env python
# coding: UTF-8
#
# @package calc_beta.py
#
# ./calc_beta.py
#
# Simple calculator in terminal
#
# @author André Saraiva
# @since 15/02/2022

primeiroNumero = input("Digite o primeiro número: ")
operacao = input("Digite a operação desejada (+, -, *, / ou //): ")
segundoNumero = input("Digite o segundo número: ")
resultado = primeiroNumero + operacao + segundoNumero
print(resultado)
```



```
IDLE Shell 3.9.4
File Edit Shell Debug Options Window Help
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/André/Desktop/atividades_laboratorio/atividade_1/calc_beta.py
Digite o primeiro número: 3
Digite a operação desejada (+, -, *, / ou //): *
Digite o segundo número: 3
3*3
>>>
```

Atividade – Aula Passada



```
calc_beta.py - C:/Users/André/Desktop/atividades_laboratorio/atividade_1/calc_beta.py (3.9.4)
File Edit Format Run Options Window Help
#!/usr/bin/env python
# coding: UTF-8
#
# @package calc_beta.py
#
# ./calc_beta.py
#
# Simple calculator in terminal
#
# @author André Saraiva
# @since 15/02/2022

primeiroNumero = int(input("Digite o primeiro número: "))

operacao = int(input("Digite a operação desejada (+, -, *, / ou //): "))

segundoNumero = int(input("Digite o segundo número: "))

resultado = primeiroNumero + operacao + segundoNumero

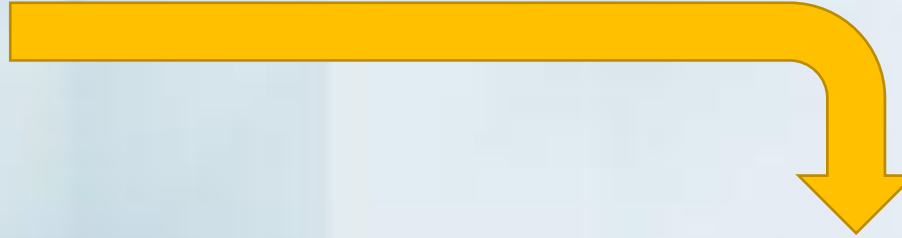
print(resultado)
```

Ln: 17 Col: 55

Atividade – Aula Passada

```
calc_beta.py - C:/Users/André/Desktop/atividades_laboratorio/atividade_1/calc_beta.py (3.9.4)
File Edit Format Run Options Window Help
#!/usr/bin/env python
# coding: UTF-8
#
# @package calc_beta.py
#
# ./calc_beta.py
#
# Simple calculator in terminal
#
# @author André Saraiva
# @since 15/02/2022

primeiroNumero = int(input("Digite o primeiro número: "))
operacao = int(input("Digite a operação desejada (+,-, *, / ou //): "))
segundoNumero = int(input("Digite o segundo número: "))
resultado = primeiroNumero + operacao + segundoNumero
print(resultado)
```



```
IDLE Shell 3.9.4
File Edit Shell Debug Options Window Help
>>>
= RESTART: C:/Users/André/Desktop/atividades_laboratorio/atividade_1/calc_beta.py
Digite o primeiro número: 3
Digite a operação desejada (+,-, *, / ou //): *
Traceback (most recent call last):
  File "C:/Users/André/Desktop/atividades_laboratorio/atividade_1/calc_beta.py",
    line 15, in <module>
        operacao = int(input("Digite a operação desejada (+,-, *, / ou //): "))
ValueError: invalid literal for int() with base 10: '*'
>>>
>>>
>>> |
```

Revisão

- A função **eval()** analisa o argumento da expressão e o avalia como uma expressão. Se a expressão for uma instrução Python legal, ela será executada.

- ***eval(expression, globals, locals)***

Ex.: `eval ('3*3') = 9`

https://www.w3schools.com/python/ref_func_eval.asp

Hoje

- **Hoje veremos:**
 - **Estruturas de Controle:**
 - **If / Else;**
 - **For;**
 - **While.**

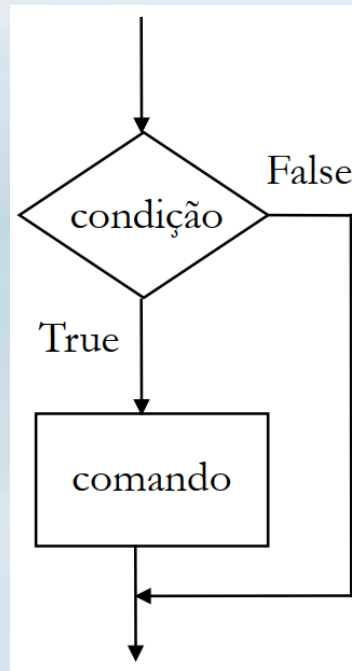
Revisão - If / Else

- **Comentários:**

- Comentário simples, de uma linha: #
- Comentário de duas ou mais linhas: `'''e'''` ou `""" e """`
- **Nossa Padronização:** não deve ter acento e deve possuir apenas caracteres alfa-numéricos. Usar notação de camelo (`minhaNota`, `meuSaldo`, `saqueDiario`,...)

Revisão – If / Else

- **if**
 - É uma estrutura de condição e utiliza-se quando se deseja executar UM comando (ou conjunto de comandos) apenas se uma condição (expressão booleana) for satisfeita;



Revisão – If / Else

- **if**

```
valor = float(input("Entre com um valor: "))
```

```
if (valor > 0):
```

```
    print(valor, " é maior do que zero.")
```

```
valor = float(input("Entre com um valor: "))
```

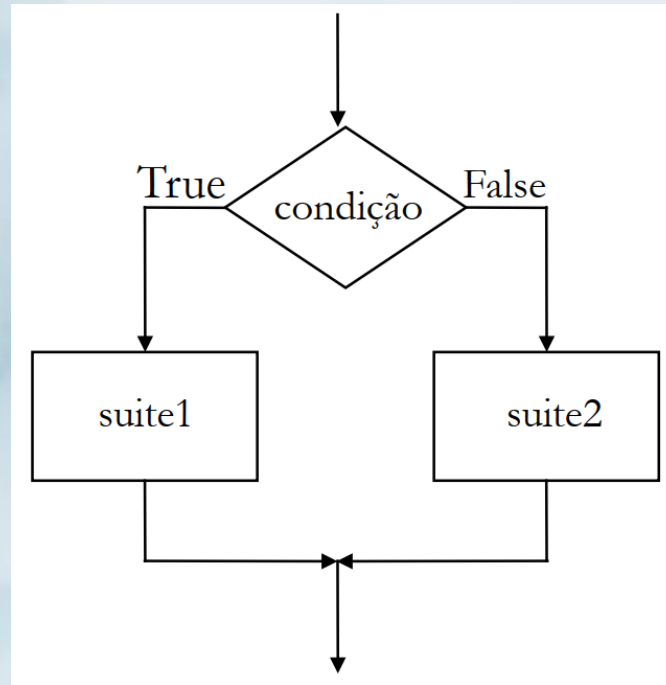
```
if (valor > 0) and (valor < 100):
```

```
    print(valor, " está compreendido entre 1 e 99.")
```

Revisão – If / Else

- **if/else**

- As estruturas de condição são utilizadas quando se deseja executar um entre DOIS comandos (ou conjunto de comandos) dependendo do resultado da condição (expressão booleana);



Revisão – If / Else

- **if/else**

```
valor = float(input("Entre com um valor: "))
```

```
if (valor > 0):
```

```
    print(valor, " é maior do que zero.")
```

```
else:
```

```
    print(valor, "é menor ou igual do que zero.")
```

Revisão – If / Else

- **if/elif/else**

- As estruturas de condição são utilizadas quando se deseja executar um entre VÁRIOS comandos (ou conjunto de comandos) dependendo do resultado da condição (expressão booleana);

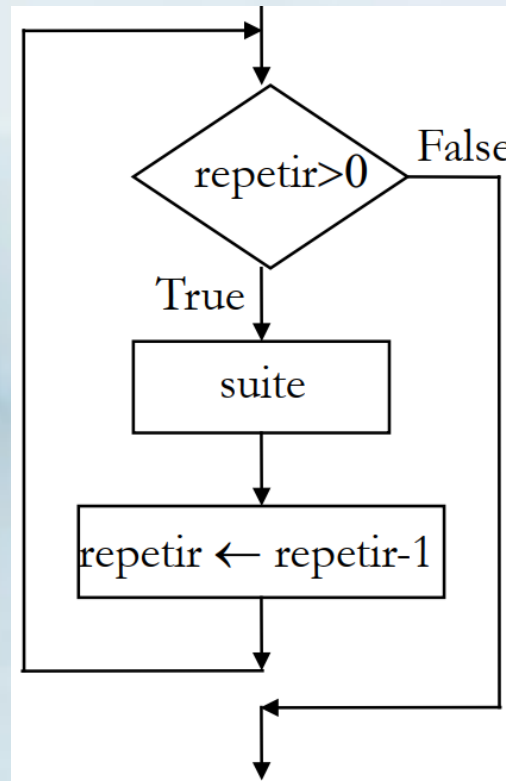
```
if <expressao1>:  
    <suite1>  
elif <expressao2>:  
    <suite2>  
...  
elif <expressaoN>:  
    <suiteN>  
else:  
    <suiteCasoContrário>
```

Atividade

- Criar uma calculadora no terminal. Seu software deve:
 - Permitir a entrada de um número;
 - Permitir a entrada de uma das operações citadas;
 - Soma(+), subtração(-), multiplicação(*) e divisão(/ ou //).
 - Permitir a entrada do segundo número;
 - Apresentar na saída padrão (vídeo) o resultado.
 - Encerrar a execução se na primeira entrada o usuário digitar “q” ou “Q”
- Neste momento é obrigatório o uso de condicionais e tratamento para evitar divisões por 0 (zero)

Revisão – For

- As estruturas de repetição **for** é utilizada quando se deseja executar um comando (ou conjunto de comandos) um número específico de vezes.



Revisão – For

- **for**

```
for x in [1,2,3,4,5,6,7,8,9]:
```

```
    print(x, end=" ")
```

```
print()
```

```
1 2 3 4 5 6 7 8 9 <enter>
```

Revisão – For

- **for**

```
for x in range(9):  
    print(x, end=" ")  
print()
```

range(valor limite) cria uma
lista [0,1,2,3,4,5,6,7,8]

```
for x in range(1,10):  
    print(x, end=" ")  
print()
```

range(valor inicial, valor
limite) cria uma lista
[1,2,3,4,5,6,7,8,9]

Revisão – For

- **for**

```
for x in range(2,18,5):  
    print(x, end=" ")  
print()
```

range(valor inicial, valor limite, avanço) cria uma lista [2,5,8,11,14,17]

2 5 8 11 14 17 <enter>

Revisão – For

- **for**

```
numero = int(input("Digite um número > 0:"))
```

```
fat = 1
```

```
for x in range(1,numero+1):
```

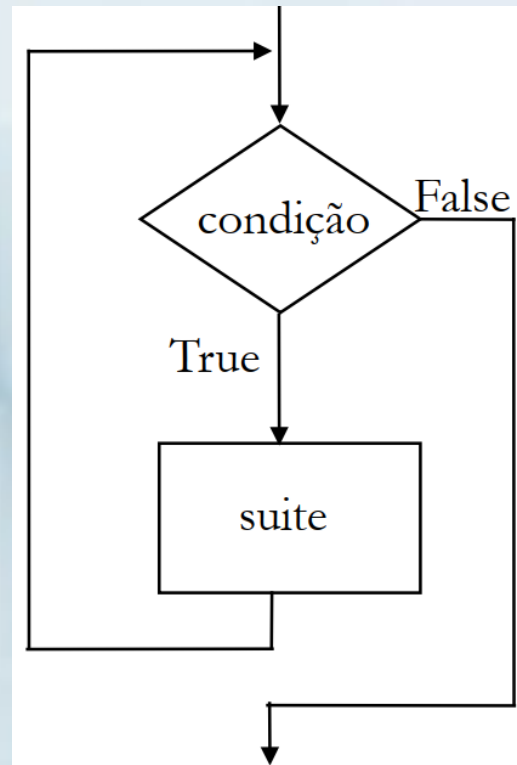
```
    fat = fat * x
```

```
print(fat)
```

O que o print irá imprimir na tela se o valor digitado for 4.

Revisão – While

- As estruturas de repetição **while** é utilizada quando se deseja executar um comando (ou conjunto de comandos) enquanto uma condição for True.



Revisão – While

- **while**

```
indice = 1
```

```
while indice < 10:
```

```
    print(indice, end=" ")
```

```
    indice = indice + 1
```

```
print()
```

```
1 2 3 4 5 6 7 8 9 <enter>
```

Atividade - While

- Criar uma programa para calcular Fatorial. Seu software deve:
 - Permitir a entrada de um número inteiro;
 - Apresentar na saída padrão (vídeo) o resultado.
- Neste momento é obrigatório o uso do While
- Você deve tratar seu programa para não aceitar valores menores negativos
- Fatorial de 0 será sempre 1 (propriedade matemática)

Revisão – Funções

- Trata-se de um grupo de sentenças (suite), comando(s) e/ou estrutura(s) de controle (e funções internas), ao qual é atribuído um nome, que após a sua execução retorna um valor.

Revisão – Funções

- Trata-se de um grupo de sentenças (suite), comando(s) e/ou estrutura(s) de controle (e funções internas), ao qual é atribuído um nome, que após a sua execução retorna um valor.
- A utilização de funções permite que:
 - Diferentes partes do programa possam ser desenvolvidas e testadas separadamente;
 - Partes do código possam ser reutilizadas em diferentes pontos do programa;
 - Programas complexos possam ser montados a partir de unidades menores já desenvolvidas e testadas.

Revisão – Funções

def nomeFunção(*lista de parâmetros*):
 suíte do corpo da função

A *lista de parâmetros* pode ter zero ou mais parâmetros, separados por vírgulas

A *suíte do corpo da função* deve possuir zero ou mais retornos de valores, expressos por **return** *valor apropriado*

Caso nenhum valor seja retornado, corresponde a **return** ou **return None**

Revisão – Funções

```
# Programa Completo
# Subprogramas
def nomeFun1(<listaParametros1>):
    <corpo nomeFun1>
...
def nomeFunN(<listaParametrosN>):
    <corpo nomeFunN>
```

Revisão – Funções

Programa Completo

 **# Subprogramas**

def nomeFun1(<listaParametros1>):

 <corpo nomeFun1>

...

def nomeFunN(<listaParametrosN>):

 <corpo nomeFunN>

Programa Principal

utiliza nomeFun1(...) ou/e nomeFunN(...)

utiliza nomeFunN(...) ou/e nomeFunN(...)

Revisão – Funções

Programa Completo com Variáveis Globais

global var1

...

global varM

Subprogramas

def nomeFun1(<listaParametros1>):

 <corpo nomeFun1>

...

def nomeFunN(<listaParametrosN>):

 <corpo nomeFunN>

Programa Principal

utiliza var1 e/ou varM e/ou nomeFun1(...) e/ou nomeFunN(...)

utiliza var1 e/ou varM e/ou nomeFun1(...) e/ou nomeFunN(...)

Próxima Aula

- **Nas próximas aulas abordaremos:**
 - **Subprogramação:**
 - **Funções (continuação);**
 - **Passagem de Parâmetros;**
 - **Recursividade.**

Fim

Obrigado!!!

Contato:

andresaraiva@id.uff.br