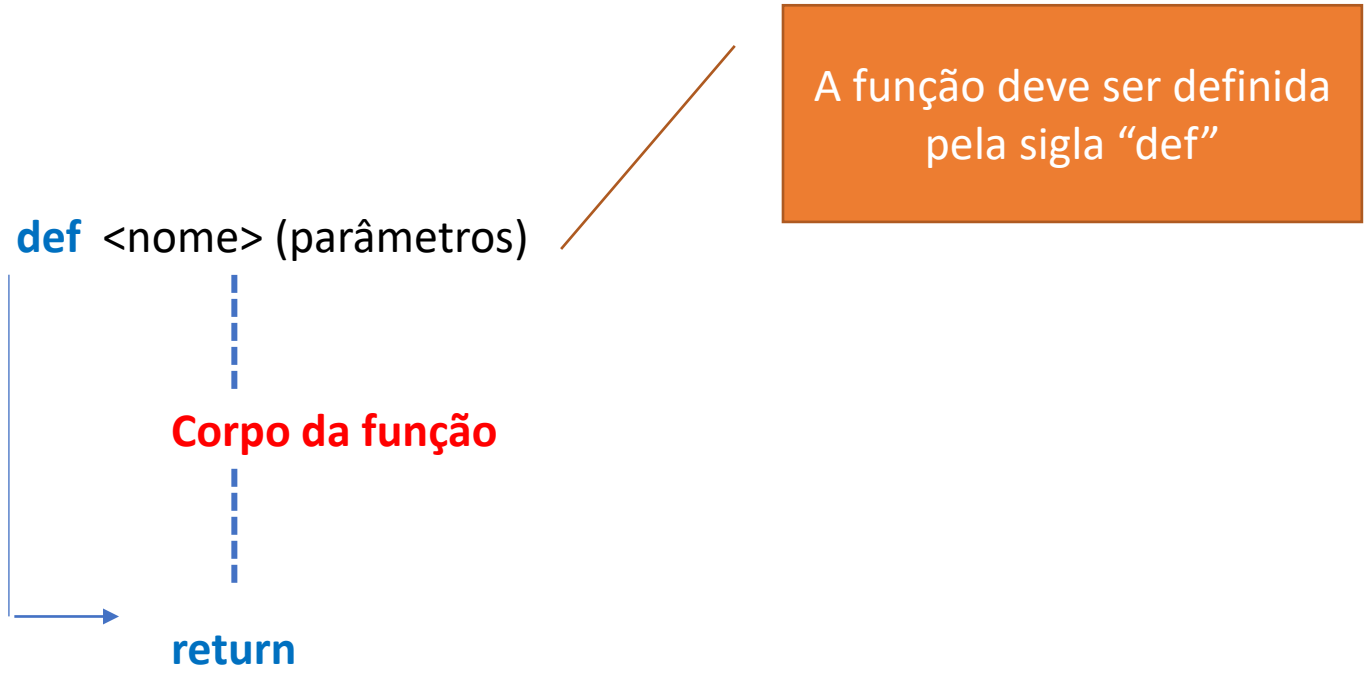
Programando em Python

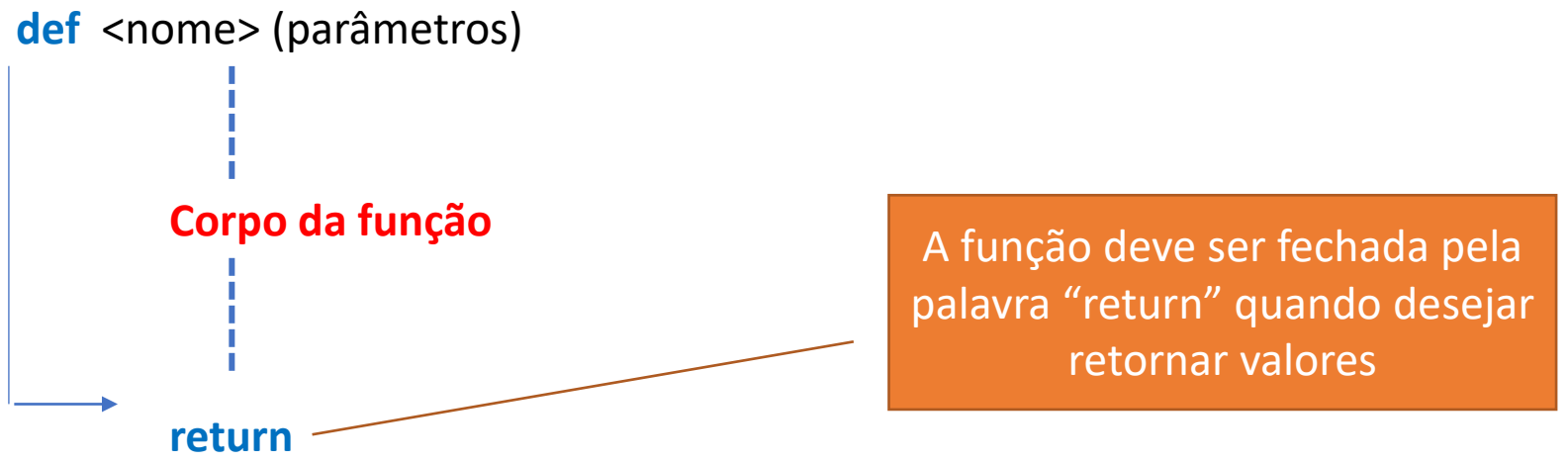
Aula 4

Prof. Dr. Marco Antonio Leonel Caetano

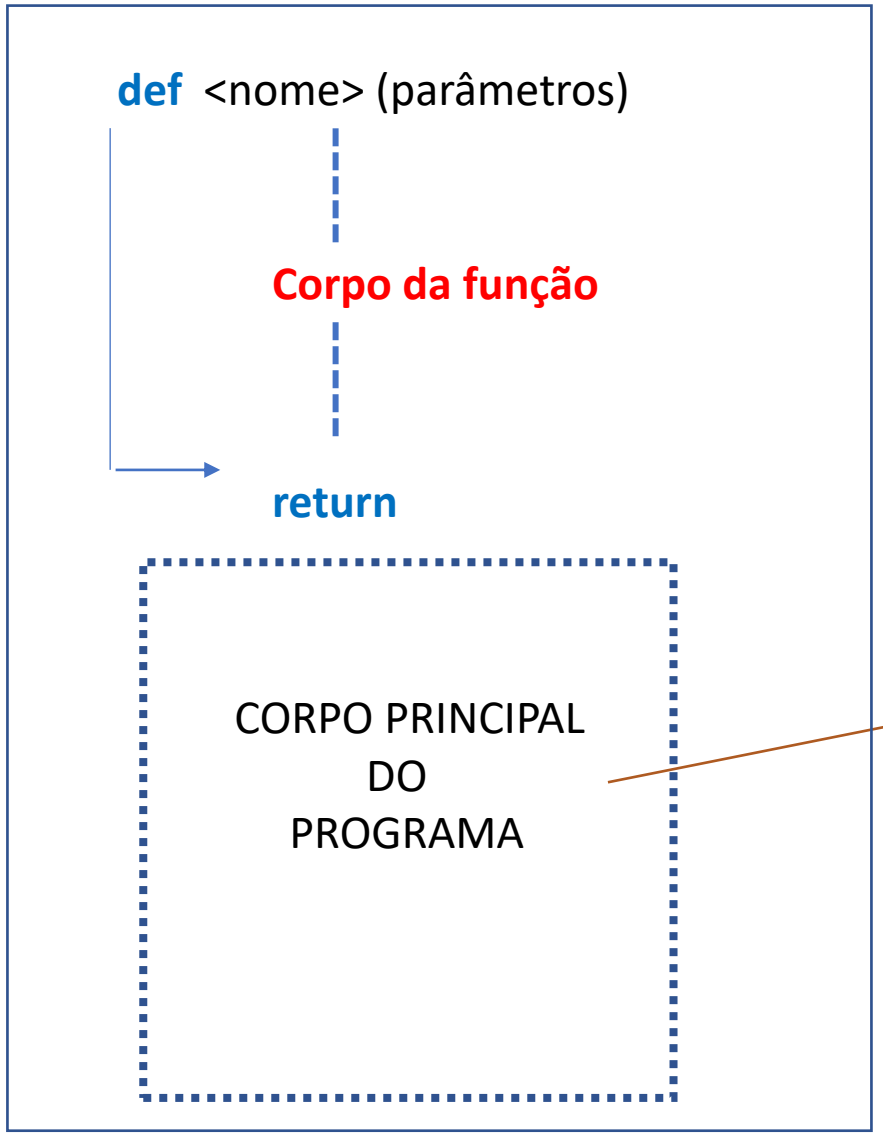
Criando *funções*



Criando *funções*



Criando *funções*



O programa deve vir sempre após a definição da função

Criando *funções*

Exemplo: Função “cidade” define a categoria da cidade pelo número de seus habitantes

```
1# Criando functions
2
3def cidade(tam):
4    if (tam <= 30000):
5        return 'pequena'
6    elif (tam> 30000) and (tam <= 150000):
7        return 'média'
8    elif (tam> 150000) and (tam <= 600000):
9        return 'grande'
10   else:
11       return 'metrópole'
```

Definindo a função

Criando *funções*

Exemplo: Função “cidade” define a categoria da cidade pelo número de seus habitantes

```
1# Criando functions
2
3def cidade(tam):
4    if (tam <= 30000):
5        return 'pequena'
6    elif (tam> 30000) and (tam <= 150000):
7        return 'média'
8    elif (tam> 150000) and (tam <= 600000):
9        return 'grande'
10   else:
11       return 'metrópole'
12
13tamanho=float(input("entre com a população = "))
14a=cidade(tamanho)
15print("++++++ categoria da cidade +++++\n")
16print(a)
```

Programa Principal

Criando *funções*

Exemplo: Função “cidade” define a categoria da cidade pelo número de seus habitantes

Resultado

```
entre com a população = 170000
++++++ categoria da cidade +++++

grande
```

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

```
1 # Criando functions
2 import math
3
4 def delta(a,b,c):
5     delta = b**2 -4*a*c
```

Cálculo de delta

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

```
1 # Criando functions
2 import math
3
4 def delta(a,b,c):
5     delta = b**2 -4*a*c
6     if (delta >0):
7         x1=(-b+math.sqrt(delta))/(2*a)
8         x2=(-b-math.sqrt(delta))/(2*a)
9         return x1,x2
```

Cálculo de duas soluções para delta positivo

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

```
1 # Criando functions
2 import math
3
4 def delta(a,b,c):
5     delta = b**2 -4*a*c
6     if (delta >0):
7         x1=(-b+math.sqrt(delta))/(2*a)
8         x2=(-b-math.sqrt(delta))/(2*a)
9         return x1,x2
```

Cálculo de duas soluções para delta positivo

Retorna as duas soluções para o Programa principal

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

```
1 # Criando functions
2 import math
3
4 def delta(a,b,c):
5     delta = b**2 -4*a*c
6     if (delta >0):
7         x1=(-b+math.sqrt(delta))/(2*a)
8         x2=(-b-math.sqrt(delta))/(2*a)
9         return x1,x2
10    elif (delta == 0):
11        x1=-b/(2*a)
12        return x1
```

Retorna solução única para o Programa principal

Cálculo de única solução para delta nulo

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

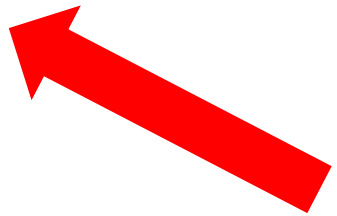
```
1# Criando functions
2import math
3
4def delta(a,b,c):
5    delta = b**2 -4*a*c
6    if (delta >0):
7        x1=(-b+math.sqrt(delta))/(2*a)
8        x2=(-b-math.sqrt(delta))/(2*a)
9        return x1,x2
10   elif (delta == 0):
11       x1=-b/(2*a)
12       return x1
13   else:
14       return 'não existem raízes reais'
```

Não existe solução real para delta negativo

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

```
1 # Criando functions
2 import math
3
4 def delta(a,b,c):
5     delta = b**2 -4*a*c
6     if (delta >0):
7         x1=(-b+math.sqrt(delta))/(2*a)
8         x2=(-b-math.sqrt(delta))/(2*a)
9         return x1,x2
10    elif (delta == 0):
11        x1=-b/(2*a)
12        return x1
13    else:
14        return 'não existem raízes reais'
```



Não existe solução real para delta negativo

Retorna um texto para o programa principal

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

```
1# Criando functions
2import math
3
4def delta(a,b,c):
5    delta = b**2 -4*a*c
6    if (delta >0):
7        x1=(-b+math.sqrt(delta))/(2*a)
8        x2=(-b-math.sqrt(delta))/(2*a)
9        return x1,x2
10   elif (delta == 0):
11       x1=-b/(2*a)
12       return x1
13   else:
14       return 'não existem raízes reais'
15
16a=float(input("a = "))
17b=float(input("b = "))
18c=float(input("c = "))
19resp = delta(a,b,c)
20print("++++++ solução da eq. 2o. grau +++++\n")
21print(resp)
```

Programa Principal

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

Resultado para a=1, b=-5, c=6

```
a = 1

b = -5

c = 6
++++++ solução da eq. 2o. grau +++++

(3.0, 2.0)
```

Criando *funções*

Exemplo: Entrar com a, b e c e dizer a solução da equação de segundo grau $ax^2 + bx + c$

Resultado para a=1, b=-5, c=6

```
a = 1

b = -5

c = 6
++++++ solução da eq. 2o. grau +++++

(3.0, 2.0)
```

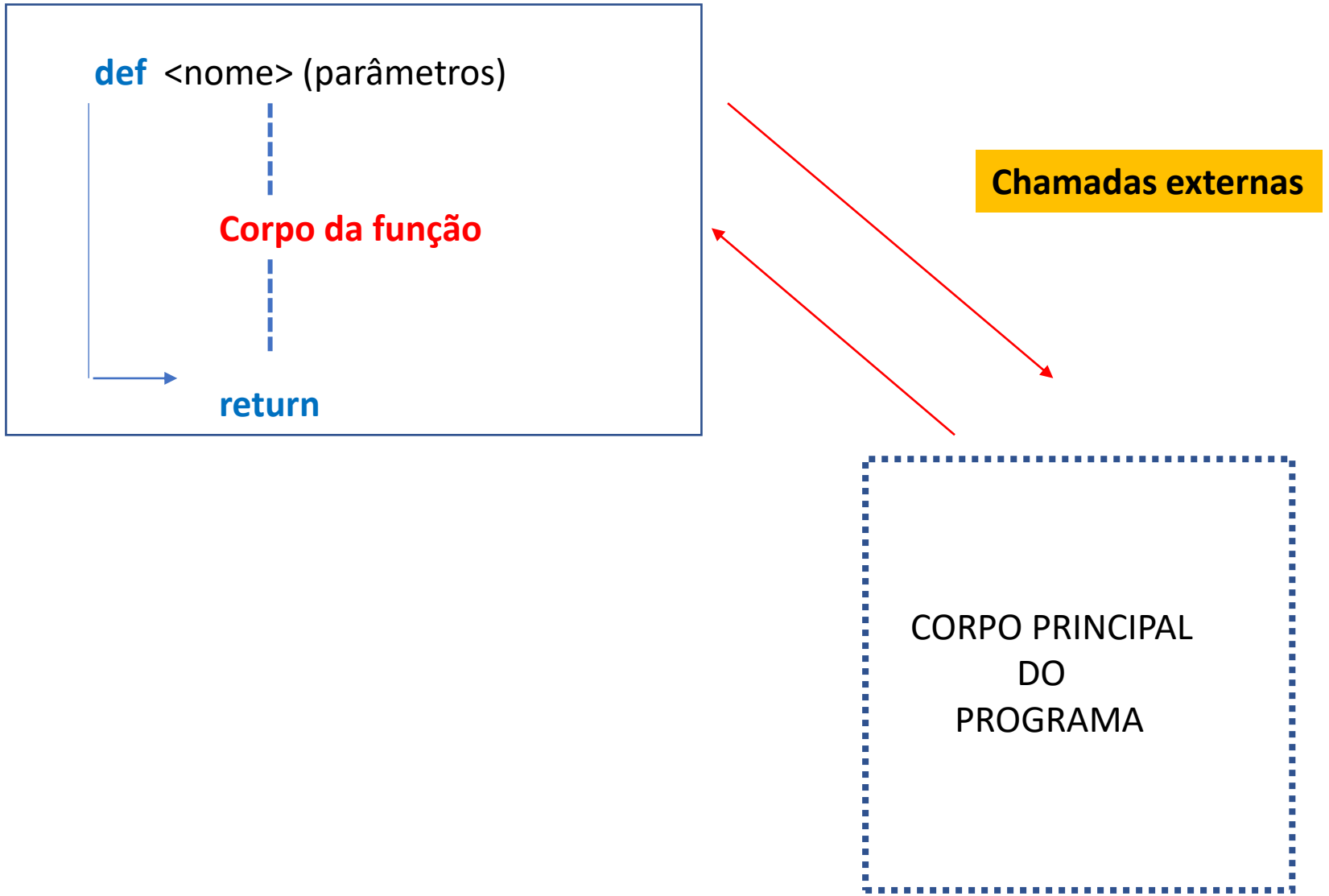
Resultado para a=10, b=1, c=6

```
a = 10

b = 1

c = 6
++++++ solução da eq. 2o. grau +++++

não existem raízes reais
```

Passando *listas inteiras* para *functions*

```
1  #Exemplo listas
2
3  def Imprime(nomes):
4      for i in nomes:
5              print(i)
6
7
8  #++++++++ Aqui começa o program principal +++++++++
9  nomes=['João','Maria','José','Ana']
10
11 Imprime(nomes)
```

Funções como *módulos externos*



Funções como *módulos externos*

Criando a function "minha_funcao" para ser chamada no programa principal

```
1  # Criando functions
2  import math
3
4  def risco(a):
5      if (a <= -0.5):
6          return 'risco alto'
7      elif (a > -0.5) and (a <= 0.5):
8          return 'risco neutro'
9      else:
10         return 'risco baixo'
11
12
```

Salvar esse programa como

"*minha_funcao.py*"

Funções como *módulos externos*

Criando a function "minha_funcao" para ser chamada no programa principal

Programa Principal para chamar "minha_função"

```
1 import minha_funcao
2
3 r=float(input("entre com o risco ="))
4
5 x=minha_funcao.risco(r)
6
7 print("++++++++ cálculo do risco ++++++++")
8 print(x)
```

Funções como *módulos externos*

Criando a function "minha_funcao" para ser chamada no programa principal

Programa Principal para chamar "minha_funcao"

```
1 import minha_funcao
2
3 r=float(input("entre com o risco ="))
4
5 x=minha_funcao.risco(r)
6
7 print("++++++++ cálculo do risco ++++++++")
8 print(x)
```

O arquivo "minha_funcao" não usa *.py na importação

Funções como *módulos externos*

Criando a function "minha_função" para ser chamada no programa principal

Programa Principal para chamar "minha_função"

```
1 import minha_funcao
2
3 r=float(input("entre com o risco ="))
4
5 x=minha_funcao.risco(r)
6
7 print("++++++++ cálculo do risco ++++++++")
8 print(x)
```

A antiga função "risco" tornou-se uma CLASSE e exige (.)

Funções como *módulos externos*

Criando a function "minha_funcao" para ser chamada no programa principal

Resultado para -0.7

```
entre com o risco =-0.7
++++++++ cálculo do risco ++++++++
risco alto
```

Funções como *módulos externos*

Criando a function "minha_funcao" para ser chamada no programa principal

Resultado para -0.7

```
entre com o risco =-0.7
++++++++ cálculo do risco ++++++++
risco alto
```

Resultado para 0.3

```
entre com o risco =0.3
++++++++ cálculo do risco ++++++++
risco neutro
```