



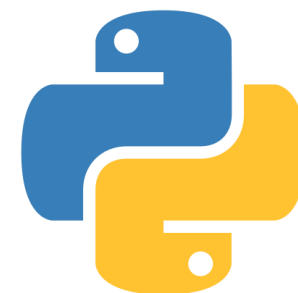
# Programação de Computadores

✓ Métodos

# Conceitos abordados nesta aula

---

A proposta desta aula é apresentar como trabalhar com métodos em Python .



# Métodos

- ✔ O conceito de método está relacionado à divisão de um problema em diversos subproblemas.
- ✔ As soluções dos subproblemas são combinadas numa solução do problema maior.



# Métodos

- ✓ Um programa pode ser simplificado quando dividido em várias sub-rotinas (métodos). Os métodos podem ser classificados em:
  - ✓ **Procedimentos:** quando não há retorno de valor.
  - ✓ **Funções:** quando há retorno de valor.
- ✓ Quando um método é chamado por um programa, ele é executado e ao seu término o controle de processamento retorna automaticamente para a primeira linha de instrução após a linha que efetuou a chamada do método.

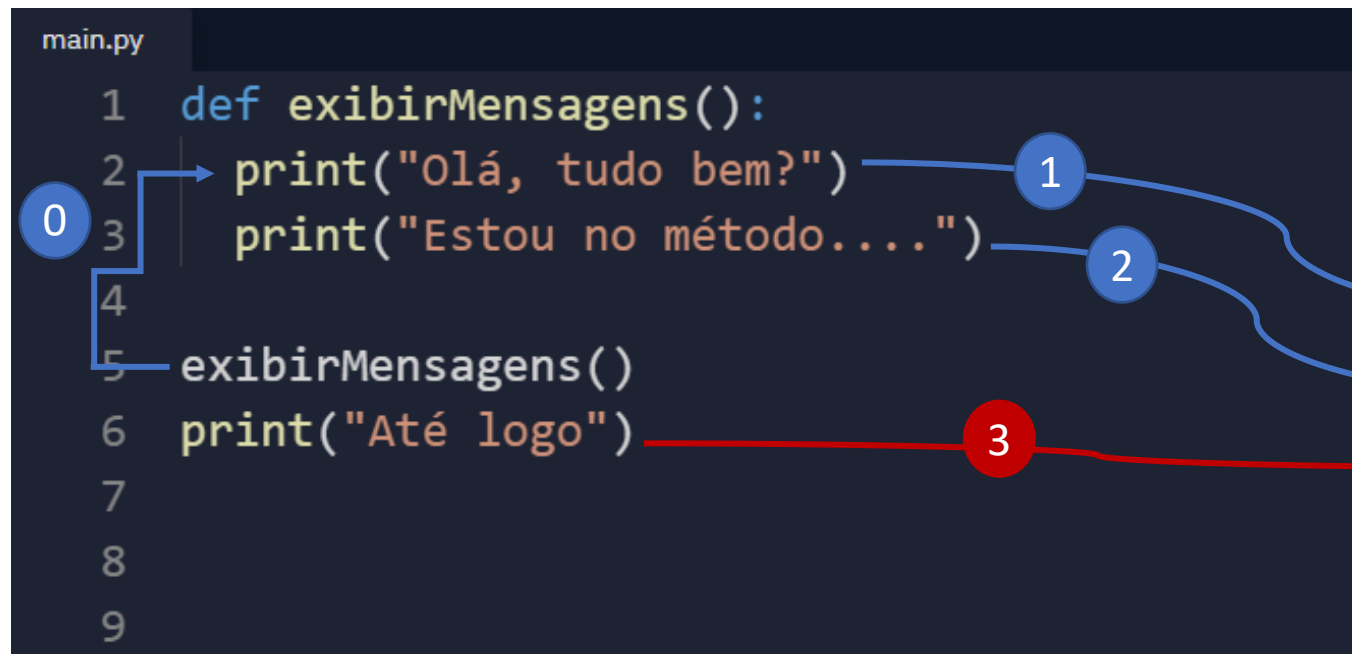


se  
Liga  
Aí

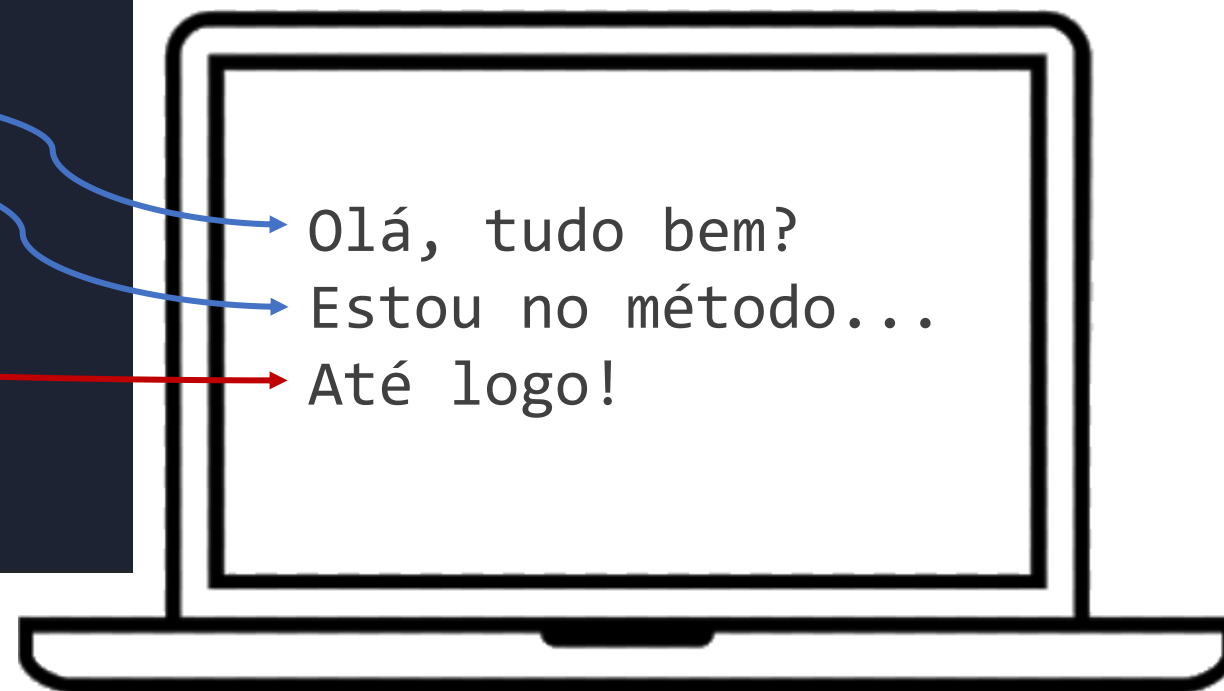
Método é um conceito muito utilizado na Programação Orientada a Objetos, para denominar, em forma geral, funções e procedimentos.

# Exemplo – utilizando um método

```
main.py
1 def exibirMensagens():
2     print("Olá, tudo bem?")
3     print("Estou no método....")
4
5     exibirMensagens()
6     print("Até logo")
7
8
9
```



Na tela:



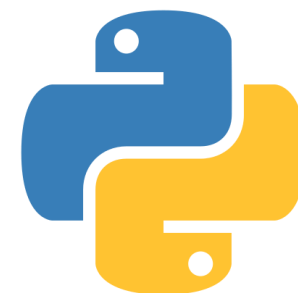
se  
Liga  
Aí

Este método é um procedimento, porque não retorna um valor.

# Métodos em Python - Síntaxe

---

- ✔ Um método em Python é definido pela instrução **def**, seguida pelo nome e parêntesis, que pode (ou não) conter a lista de parâmetros (opcional).
- ✔ Em Python não é necessário especificar os tipos dos dados nos parâmetros.
- ✔ As variáveis declaradas dentro dos métodos tem escopo local, quando o método termina elas são destruídas.



# Métodos em Python - Síntaxe

Segue as mesmas regras  
para nomes de variáveis

Parâmetros são as informações  
necessárias para que o método  
execute o seu papel.

```
1 def nomeMetodo(lista de parâmetros):  
2     lista de comandos
```

Os parâmetros não são  
obrigatórios!!!

Um método pode ter quantos  
comandos forem necessários, mas  
eles precisam ser endentados a  
partir da margem esquerda!!

Caso seja um método com retorno  
(função) deverá terminar com **return**  
seguido do valor retornado.

# Métodos em Python - Síntaxe

- ✔ Python fornece múltiplos retornos de função. Vejamos o exemplo abaixo:

```
def spam(x):  
    return x,x,x,x,x
```

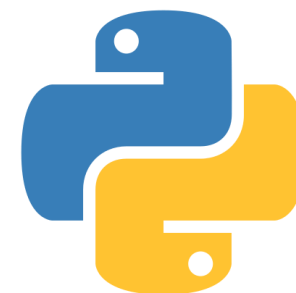
```
spam('spam')
```

```
↳ ('spam', 'spam', 'spam', 'spam', 'spam')
```

```
def spam(x):  
    return x,2*x,3*x,x+1,x**2
```

```
a= spam(4)  
print(a[3])
```

```
↳ 5
```





# Exemplo – um método que soma dois valores

main.py

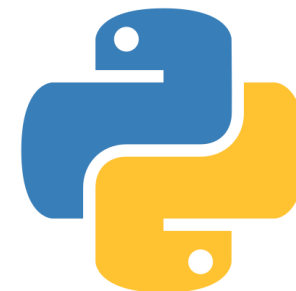
```
1 def somaDoisValores(a, b):  
2     a = a + b  
3     return a  
4
```

As instruções do bloco devem ser endentadas corretamente



se  
Liga  
Aí

Este método é uma função, porque retorna um valor.

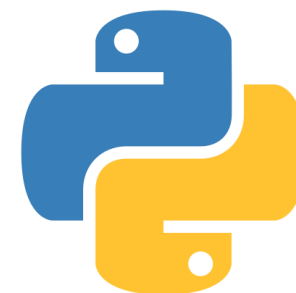


# Exemplo – um método que soma dois valores

O método deve ser criado antes de ser chamado.

main.py

```
1 def somaDoisValores(a, b):  
2     a = a + b  
3     return a  
4  
5 a = int(input("Digite a: "))  
6 b = int(input("Digite b: "))  
7 print(somaDoisValores(a, b))  
8 print(a)
```



# Exemplos de aplicação

1- Vamos elaborar um programa em Python que receba um número e uma mensagem e exiba na tela essas informações.

```
main.py
1 def exibirMensagem(msg, n):
2     print("Mensagem:", msg, "\nNúmero da sorte:", n)
3
4 n = float(input("Digite o número da sorte: "))
5 msg = input("Digite uma mensagem: ")
6
7 exibirMensagem(msg, n)
```



# Exemplos de aplicação

2- Vamos elaborar um programa em Python que receba dois números reais, calcule e exiba na tela a soma desses números.

```
1  # Exemplo: programa Python que utiliza uma função.
2  # A função somaDoisValores deve ser declarada antes
3  # de ser chamada.
4
5  def somaDoisValores(a, b):
6      result = a + b
7      return(result)
8
9  x = float(input("Entre com o 1º valor: "))
10 y = float(input("Entre com o 2º valor: "))
11 print("A soma dos valores é ", somaDoisValores(x, y))
```



# Exemplos de aplicação

---

3- Escreva um método chamado calculaIMC que receba o peso e altura de uma pessoa, calcule e retorne o IMC, de acordo com a fórmula abaixo:

$$\text{imc} = \text{peso} / \text{altura}^2$$

Faça um programa principal que solicite ao usuário seu peso (em kg) e sua altura (em m) e, usando o método definido acima, mostre o IMC.

# Exemplo 3 - IMC

```
# Exemplo de cálculo do IMC em Python
# utilizando duas funções

def calculaIMC(peso, altura):
    res = peso / altura**2
    return(res)

def despedida():
    print("Obrigado por usar este programa!")
    print("Até logo!")

peso = float(input("Digite o peso da pessoa, em Kg: "))
altura = float(input("Digite a estatura da pessoa, em m: "))
print("O IMC é ", calculaIMC(peso, altura), " Kg/m²")
despedida()
```



# Métodos

Comentários sobre a utilização de métodos:

- ✓ Podemos utilizar tantos métodos quanto necessário.
- ✓ Um método poderá ser chamado qualquer quantidade de vezes.
- ✓ Um método poderá chamar a outro método.
- ✓ Um método poderá se auto executar (recursividade).

Vantagens da utilização de métodos:

- ✓ Permitem dividir a lógica de um algoritmo em partes específicas.
- ✓ Facilitam a reutilização de código existente.
- ✓ A programação fica mais clara e organizada.
- ✓ A manutenção de programas fica mais fácil.



# Exemplos de aplicação

---

4- Escreva um método com retorno que receba como parâmetros a base e a altura de um triângulo, calcule e retorne o valor de sua área.

$$\text{area} = \text{base} * \text{altura} / 2$$

Faça um programa em Python que solicite a base e altura de um triângulo ao usuário, e utilizando a função definida acima, calcule e mostre o valor da área.

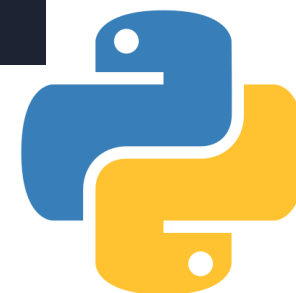
Após o cálculo anterior, defina outros valores no código de base e altura e utilizando o mesmo método acima, mostre o valor da área.



# Exemplos de aplicação

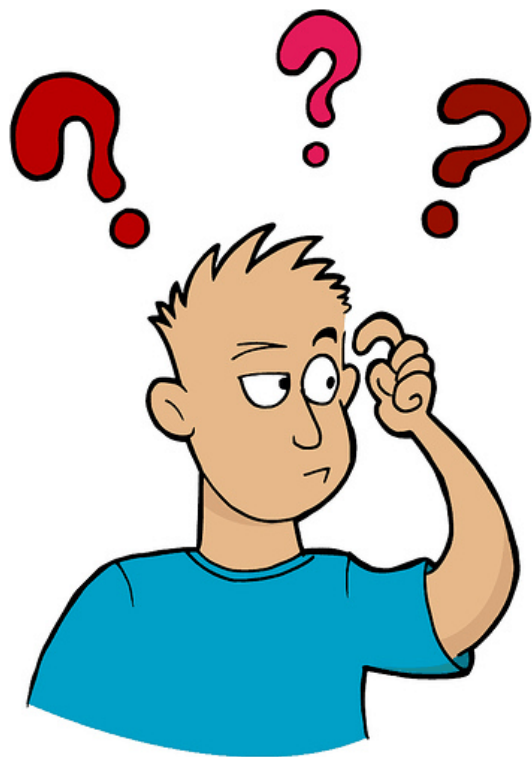
main.py

```
1 def calculaArea(base, altura):
2     area = base * altura/2
3     return area
4
5 base = float(input("Digite a base do triângulo em cm: "))
6 altura = float(input("Digite a altura do triângulo em cm: "))
7
8 print("A área do triângulo é %.2f cm²"%calculaArea(base,altura))
9 print("Um método pode ser utilizado quantas vezes for necessário!!")
10 print(calculaArea(8,5))
11 print(calculaArea(10,10))
12 print(calculaArea(2,5))
```



# Trabalhando com arquivos externos

Se a ideia é dividir o problema em partes menores, porque não separá-lo em arquivos!!!????



# Trabalhando com arquivos externos

Se a ideia é dividir o problema em partes menores, porque não separá-lo em arquivos!!!????



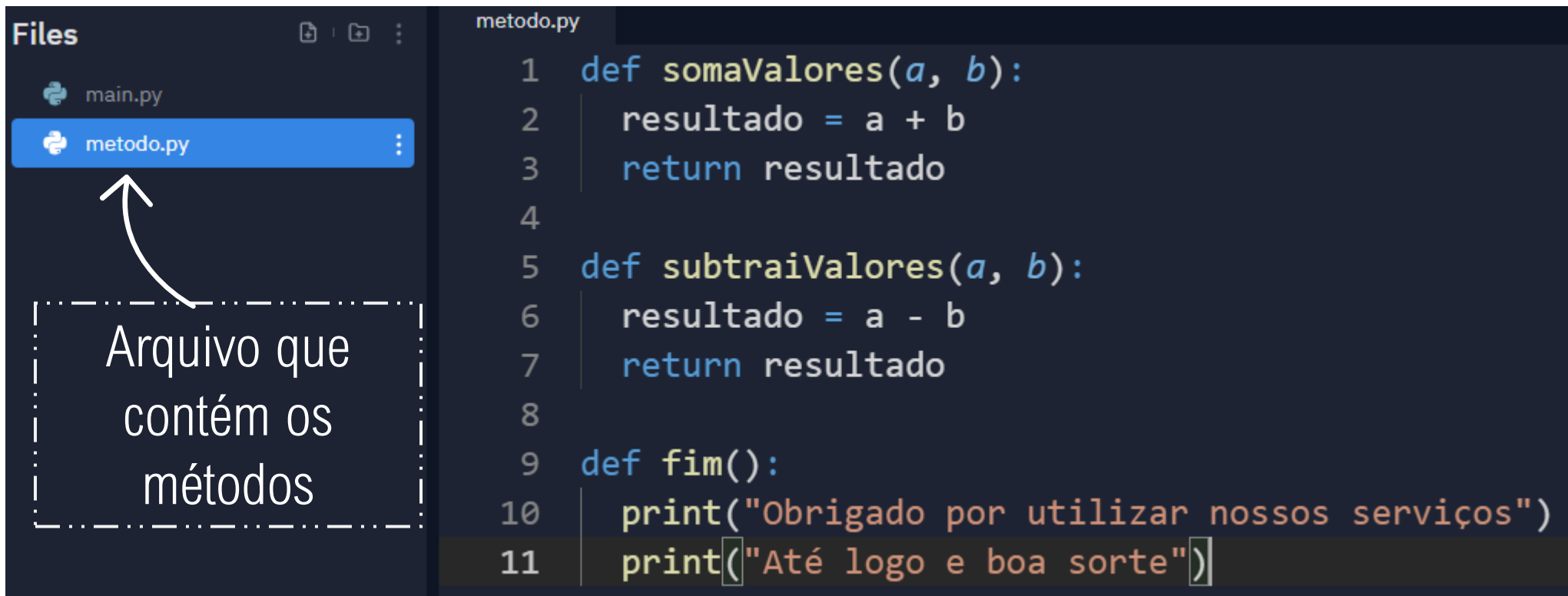
```
Files
├── main.py
└── metodo.py

main.py
1  import metodo
2
3  a = int(input("Digite a: "))
4  b = int(input("Digite b: "))
5
6  op = int(input("1-Soma\n2-Subtração:\nDigite uma opção: "))
7  if op==1:
8      print(metodo.somaValores(a, b))
9  elif op==2:
10     print(metodo.subtraiValores(a, b))
11 else:
12     print("Opção inválida")
13
14 metodo.fim()
```



# Trabalhando com arquivos externos

Se a ideia é dividir o problema em partes menores, porque não separá-lo em arquivos!!!????



The image shows a code editor interface. On the left, a 'Files' sidebar lists 'main.py' and 'metodo.py'. 'metodo.py' is selected and highlighted in blue. A white arrow points from a dashed box containing the text 'Arquivo que contém os métodos' to the 'metodo.py' file. The main editor area shows the content of 'metodo.py' with line numbers 1 through 11. The code defines three functions: 'somaValores', 'subtraiValores', and 'fim'.

```
1 def somaValores(a, b):
2     resultado = a + b
3     return resultado
4
5 def subtraiValores(a, b):
6     resultado = a - b
7     return resultado
8
9 def fim():
10    print("Obrigado por utilizar nossos serviços")
11    print("Até logo e boa sorte")
```

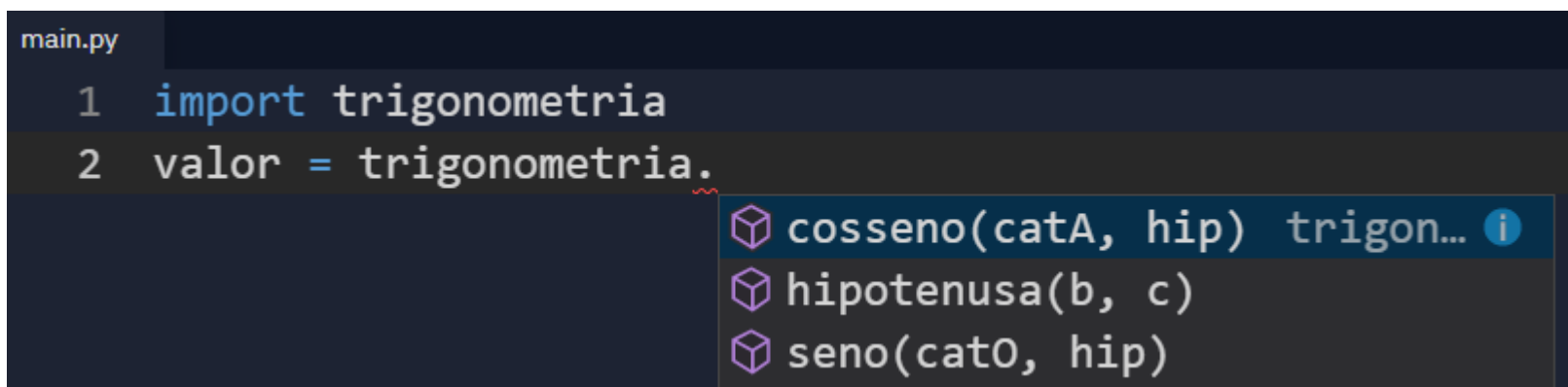
# Trabalhando com arquivos externos

Observações sobre a utilização de arquivos externos:

- ✓ Primeiramente, devemos importar o módulo criado, utilizando **import**.
- ✓ Após a importação, teremos acesso aos métodos utilizando o nome do módulo seguido de ponto (.)
- ✓ Os arquivos devem ter a extensão **.py**

Exemplo:

```
main.py
1 import trigonometria
2 valor = trigonometria.
```



Ao digitar ponto(.) a IDE mostra os métodos disponíveis no arquivo externo!!

# Exemplos de aplicação

5- Vamos criar um programa que calcule o seno, cosseno e a tangente a partir da entrada dos catetos oposto e adjacente de um triângulo retângulo. Para tanto, precisaremos de quatro métodos:

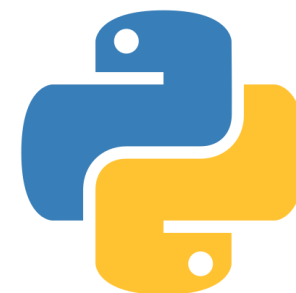
- Cálculo da hipotenusa do triângulo: ①  $\text{hipotenusa} = \sqrt{\text{catO}^2 + \text{catA}^2}$
- Cálculo do seno e cosseno: ②  $\text{seno } \hat{A} = \frac{\text{catO}}{\text{hipotenusa}}$  ③  $\text{cosseno } \hat{A} = \frac{\text{catA}}{\text{hipotenusa}}$
- Cálculo da tangente: ④  $\text{tangente } \hat{A} = \frac{\text{seno } \hat{A}}{\text{cosseno } \hat{A}}$

Faça um programa principal que solicite ao usuário o cateto oposto (catO) e o adjacente (catA) e, usando os métodos definidos acima, mostre o seno, cosseno e a tangente. **OBS: Separe os métodos em um arquivo chamado trigonometria.py e não utilize o módulo math**

# Exemplos de aplicação

main.py

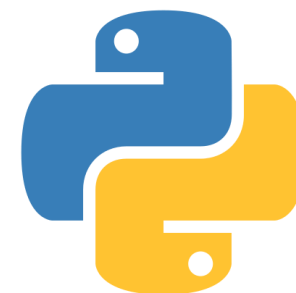
```
1 import trigonometria
2
3 catO = float(input("Digite o cateto oposto: "))
4 catA = float(input("Digite o cateto adjacente: "))
5
6 print("O seno é: %.2f"%trigonometria.seno(catO, catA))
7 print("O cosseno é: %.2f"%trigonometria.cosseno(catO, catA ))
8 print("A tangente é: %.2f"%trigonometria.tangente(catO,catA))
```



# Exemplos de aplicação

trigonometria.py

```
1 def hipotenusa(cat0, catA):
2     return (cat0**2 + catA**2)**0.5
3
4 def seno(cat0, catA):
5     h = hipotenusa(cat0, catA)
6     return cat0/h
7
8 def cosseno(cat0, catA):
9     h = hipotenusa(cat0, catA)
10    return catA/h
11
12 def tangente(cat0, catA):
13    return seno(cat0,catA)/cosseno(cat0,catA)
```





# Expressões lambda em Python

Uma função comum pode ser simplificada com a utilização de uma função lambda. Apesar de muito parecidas, as funções lambda possuem duas diferenças particulares:

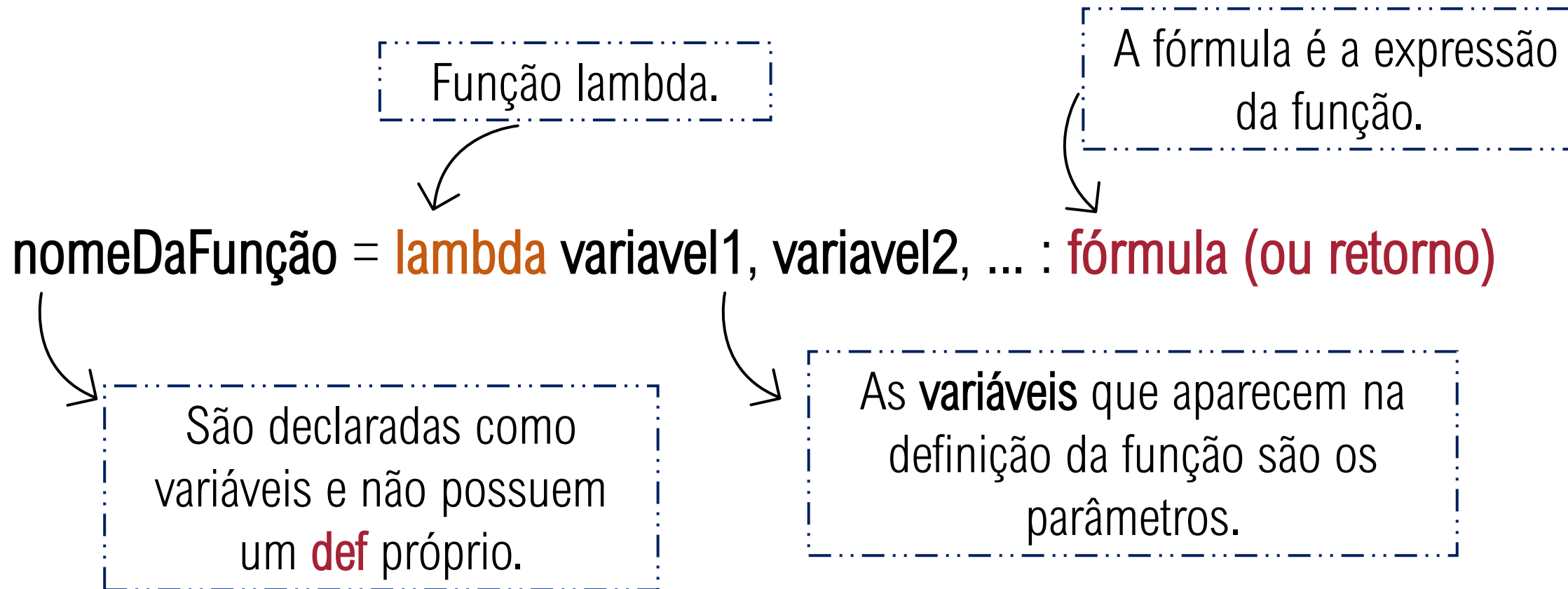
- ✔ não possuem uma definição em código, ou seja, são declaradas como variáveis e não possuem um **def** próprio;
- ✔ são funções de uma linha, que funcionam como se houvesse a instrução **return** antes do comando.

```
main.py
1  #função comum
2  def soma(a,b):
3      return a+b
4
5  #chamada da função
6  print(soma(5,6))
```

```
main.py
1  #função lambda
2  soma = lambda a, b: a+b
3
4  #chamada da função
5  print(soma(5,6))
6
```

# Expressões lambda em Python

A forma geral para criação da Função Lambda é:



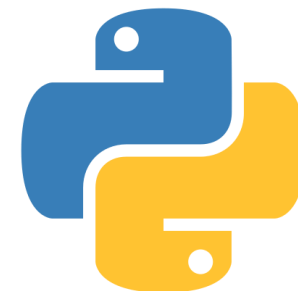
# Expressões lambda em Python

Também podemos ter uma função lambda sem parâmetros, quando for o caso, basta omitirmos a declaração de parâmetros normalmente:

main.py

```
1  #função lambda
2  soma = lambda: 3 + 4
3
4  #chamada da função
5  print(soma())
```

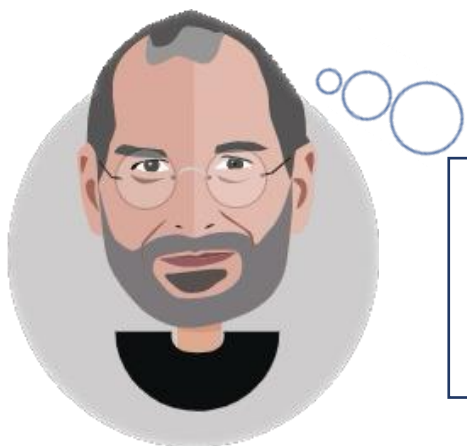
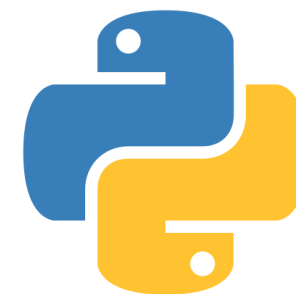
Observe que um dos pontos fortes de expressões lambda é sua síntese, elas são uma forma rápida de definir funções simples.



# Expressões lambda em Python

Mas, como fazemos quando as funções não tem retorno?

```
main.py
1  #função lambda
2  ola = lambda: print("Olá mundo!!!")
3
4  #chamada da função
5  ola()
```



se  
Liga  
Aí

Essa função lambda não possui retorno e nem parâmetros, mas funciona perfeitamente!

# Exemplos de aplicação

6- Vamos criar uma calculadora simples que utilize a função lambda para realizar as operações de soma, subtração, divisão e multiplicação.

Para tanto, criaremos uma estrutura como a apresentada abaixo:

## Console

```
Calculadora Lambda  
[1] Soma  
[2] Subtração  
[3] Multiplicação  
[4] Divisão  
[0] Sair  
Escolha uma operação:  
  
Digite o primeiro número:  
Digite o segundo número:
```

# Exemplos de aplicação

```
main.py
1 soma = lambda a,b: a+b
2 subtracao = lambda a,b: a-b
3 multiplicacao = lambda a,b: a*b
4 divisao = lambda a,b: a/b
5
6 print("Calculadora Lambda")
7 print("[1] Soma\n[2] Subtração\n[3] Multiplicação\n[4] Divisão\n[0] Sair")
8 while True:
9     op = int(input("Digite uma opção: "))
10    if op==0:
11        print("Obrigado por utilizar nossa calculadora!")
12        break
13    elif str(op) not in '1234':
14        print("Opção inválida!!")
15    else:
16        a = float(input("Digite o primeiro número: "))
17        b = float(input("Digite o segundo número: "))
18        if op==1: print(soma(a,b))
19        elif op==2: print(subtracao(a,b))
20        elif op==3: print(multiplicacao(a,b))
21        else: print(divisao(a,b))
```

# Expressões lambda em Python

## Expressões **lambda** em Python

### Função comum:

```
main.py
1  #função comum
2  def soma(a,b):
3      return a+b
4
5  #chamada da função
6  print(soma(5,6))
```

### Função comum:

```
main.py
1  #função lambda
2  soma = lambda a, b: a+b
3
4  #chamada da função
5  print(soma(5,6))
6
```

Saiba mais acessando:

- ✓ <https://gabrielschade.github.io/2018/06/25/basics-python-9-lambda.html>
- ✓ <https://www.codingame.com/playgrounds/52499/programacao-python-intermediario---prof--marco-vaz/funcao-lambda>



# Alguma dúvida????

---





# Exercícios de aplicação

---



# Exercícios de aplicação

---

1- Escreva um método com retorno que receba como parâmetros os lados de um retângulo, calcula e retorna o valor de sua área.

$$\text{area} = \text{lado} * \text{lado}$$

Faça um programa principal que solicite os valores dos lados de um retângulo ao usuário, e utilizando a função definida acima, calcule e mostre o valor de área.

2- Construir um método que receba como parâmetros o valor de uma compra e a quantidade de parcelas e calcula e **retorna** o valor da parcela, sabendo que a loja acrescenta 5% de juros para as compras parceladas.

No algoritmo principal, solicite para o usuário o valor de uma compra e a quantidade de parcelas e utilizando o método descrito acima, mostre o valor da parcela.

# Exercícios de aplicação

---

3- Elabore um programa para calcular a velocidade de três objetos diferentes (com velocidade constante). Conhecemos (são dados digitados pelo usuário), para cada objeto, a distância percorrida e o tempo que necessitou para percorrer essa distância. Utilize um método geral que calcule e retorne a velocidade de um objeto, fornecidos como parâmetros os dados de distância e tempo.

# Créditos

Esta aula foi elaborada com base no material produzido e cedido gentilmente pelos **Professores Alcides, Lédon, Amilton e Cristiane.**







*That's all Folks!*