

# Primeiros Programas em Python



- Aula 02 -  
Pensamento Computacional

Prof. Me. Lucas R. C. Pessutto

# Na aula de hoje...

**01**

## Resolvendo Problemas

Processo  
Na prática!

**02**

## Elementos Léxicos

Variáveis / Constantes  
Identificadores  
Comentários

**03**

## Comando de Atribuição

**04**

## Entrada/Saída

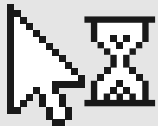
Comando input  
Saída Formatada de dados

**05**

## Operadores Matemáticos

Operadores Aritméticos Nativos  
Biblioteca matemática do Python

...





## Análise do Problema

### ANALISE

- Entender claramente o problema
- Conhecer o que compõe uma solução



## Descrição dos Dados e Algoritmos

### PROJETO

- Listar quais tipos de dados são necessários
- Determinar como os dados são estruturados
- Encontrar os algoritmos apropriados



## Implementação do Programa

### IMPLEMENTAÇÃO

- Representar os dados e algoritmos na linguagem de programação



## Depuração e Testes

### TESTES

- Testar o programa em um conjunto selecionado de instâncias do problema
- Corrigir e entender as causas dos erros encontrados no programa

# Problema 01: Soma



Enunciado de um problema:

Ler dois valores informados pelo usuário através do teclado, calcular e informar a soma destes valores.



Análise do Problema

Entender e Identificar o objetivo, as entradas e saídas do problema

**Objetivo:** realizar a soma de dois valores informados

**Entradas:** dois valores numéricos

**Saída:** um valor numérico (soma)

# Problema 01: Soma



Enunciado de um problema:

Ler dois valores informados pelo usuário através do teclado, calcular e informar a soma destes valores.



Listar os dados necessários. Elaborar um algoritmo

Descrição dos Dados  
e Algoritmos

Escreva um algoritmo para resolver esse problema!

# Teste de mesa

**ALGORITMO** calcular\_soma ; ✓

**VAR** ✓

num1 , num2 , soma : real ; ✓

**INICIO** ✓

**ESCREVER** ( ' Informe dois números: ' ) ;

**LER** ( num1 , num2 ) ;

soma  $\leftarrow$  num1 + num2 ;

**ESCREVER** ( ' O resultado da soma é ', soma ) ;

**FIM.**

Memória

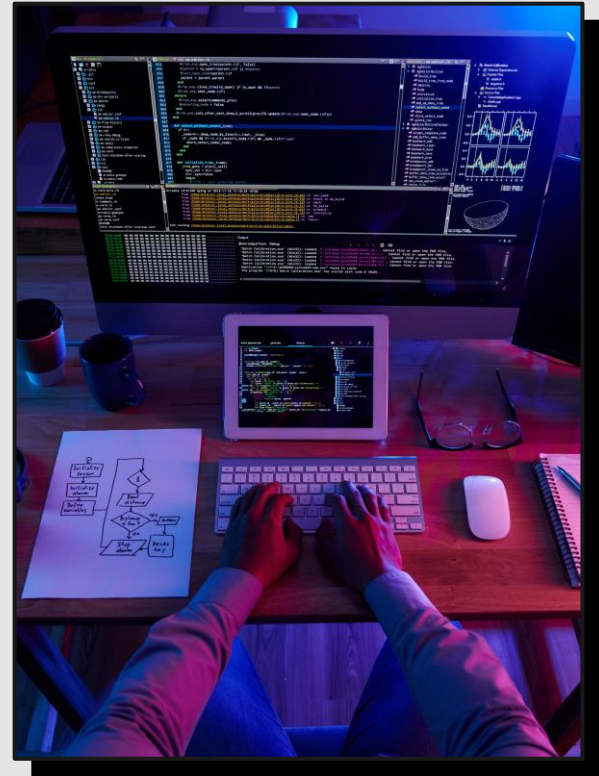
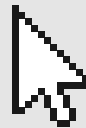
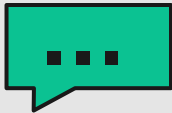
num1	num2	soma	

Monitor



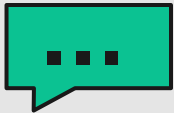
# Linguagem de Programação

Fornece conjunto de **convenções e regras** que possibilitam instruir o computador na execução de tarefas



# Ambiente de Programação

- \* Permite edição do programa na linguagem específica (linguagem de programação)
- \* Verifica sintaxe (compilação): se correta, traduz para linguagem de máquina
- \* Executa programa em linguagem de máquina (execução)
- \* Obs: Qualquer editor de textos e compilador podem ser usados para codificar programas.





# Programação Sequencial em Python

- \* **Entrada de Dados:**

Função input: lê valores digitados no teclado

- \* **Saída de Dados:**

Função print: impressão de dados

- \* **Atribuição:**

Operador atribuição =

# Variáveis

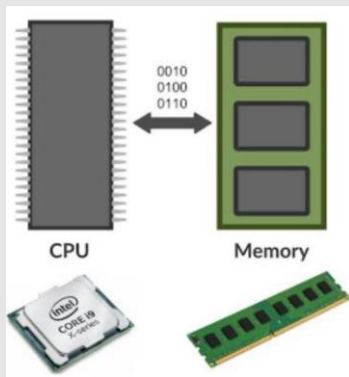
conteúdo

identificador

tipo

Posições de memória capazes de **armazenar** e representar um valor

Quando em execução possuem nomes associados, chamados de identificadores



Endereço	Identificador	Valor	Tipo
0x0000	num1	5	int
0x0001	num2	2	int
0x0002	subtr	3	int
0x0003	-	-	-
...	...	...	...

Representação simplificada da memória (sem levar em conta tipos e tamanhos)

# Variáveis em Python

São declaradas assim que recebem um valor, por exemplo:

`x = 5` indica que a variável recém criada `x` possui o valor `5`

`x` é o identificador, `=` é o símbolo de atribuição (versão de `←`) e `5` é uma constante

Variáveis possuem tipos de dados específicos:

inteiro (int): `42`

ponto flutuante/reais (float): `3.14`

textos (string): `"João da Silva"`

booleano (bool): `False`

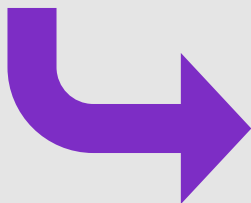
complexo (complex): `1.5 + 5.3j`

Suportam operações como adição, subtração, multiplicação, divisão e podem se relacionar logicamente

Case-sensitive: identificadores `nome` e `Nome` representam variáveis diferentes

# Variáveis

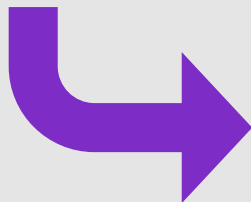
Para saber o tipo de uma variável utiliza-se a função `type(variável)`



```
x = 2
tipo = type(x)
print(tipo)

saída: <class 'int'>
```

Variáveis podem mudar seu tipo a qualquer momento



```
x = 4
tipo = type(x)
print(tipo)
x = 1.42
tipo = type(x)
print(tipo)

saída: <class 'int'>
saída: <class 'float'>
```

# Boas Práticas para Identificadores

Geral: iniciar com letra minúscula + letras minúsculas ou números

Não usar caracteres especiais (&, \*, \$, -, +, etc.) ou acentuação

Identificadores com múltiplas palavras devem ser separadas por sublinhado (\_)

Utilizar identificadores significativos que descrevam o que representam

## Identificadores desejáveis

x, y, endereco,  
valor\_do\_pedido, jogador1,  
jogador2



## Identificadores indesejáveis/incorretos

Endereco, 12pessoas,  
valorDoPedido, var&avel



# Constantes

- \* Valores que **não se alteram** durante a execução de um programa

Ex:  $\text{PI} = 3.1415$

- \* Utilizadas para facilitar a manutenção do código
- \* Constantes devem ser identificadas por **letras maiúsculas** e **\_ (sublinhado)** para nomes compostos

Ex: `NUM_EMPREGADOS`, `IDADE_MINIMA`

# Comentários

Estruturas **desconsideradas** pelo compilador

Buscam facilitar a leitura do código pelo programador

Importantes uma vez que programas complexos costumam ser muito mais vezes lidos do que escritos

Comentários em linha (#)

# Exemplo de comentário em linha

Comentários em bloco (""" """)

"""

Exemplo de comentário em bloco

"""

# Comando de Atribuição

Armazena um valor em uma variável pelo operador =

Sintaxe: `variável = expressão`

Ex:

```
idade = 21
```

```
altura = 1.82
```

```
resposta = "Sim"
```

```
outra_idade = idade
```

Endereço	Identificador	Valor	Tipo
0x0000	idade	21	int
0x0001	altura	1.82	float
0x0002	resposta	"Sim"	string
0x0003	outra_idade	21	int
...	...	...	...

Representação simplificada da memória (sem levar em conta tipos e tamanhos)



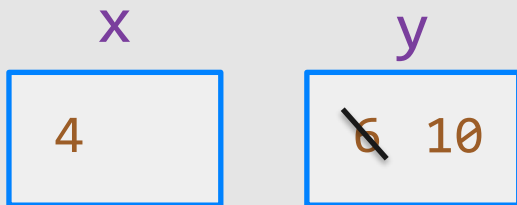
# Comando de Atribuição

Outro Exemplo:

$x = 4$       *# A variável x receberá o valor 4*

$y = x + 2$

$y = y + 4$       *# conteúdo inicial da variável utilizado na expressão*



# Entrada e Saída

- \* Operações que permitem a comunicação do usuário com o programa
- \* Saída de dados pelo comando print
- \* Sintaxe:  
`print("texto", variaveis)`
- \* texto e variáveis são impressas na ordem especificada entre vírgulas

# Saída de Dados

```
idade = 21  
print("O valor de idade é", idade)
```

O valor de idade é 21

O Python insere um espaço antes  
de imprimir o valor da variável!

```
idade = 21  
print(idade, "O valor de idade é")
```

21 O valor de idade é

```
idade = 21  
print("O valor de idade é", "idade")
```

O valor de idade é idade

# Comando de Entrada

A operação de **Entrada** de dados (via teclado) é feita através do comando **input**

Sintaxe:

```
<variável> = input("texto")
```

Funcionamento:

- \* Imprime "texto" e aguarda até que o usuário digite algo e pressione Enter
- \* O valor digitado pelo usuário é retornado pelo **input** sempre como texto (**str**)
- \* O valor digitado é armazenado em <variável>

*Para tratar desse valor digitado como número (por exemplo, **int** ou **float**) é preciso fazer a conversão com a função apropriada*

# Comando de Entrada

```
msg = input("Digite uma mensagem: ")  
print("Você digitou:", msg)
```

```
Digite uma mensagem: Alo mamãe!  
Você digitou: Alo mamãe!
```

```
idade = int(input("Digite sua idade: "))  
print("Você tem", idade, "anos.")  
print("Ano que vem você terá", idade + 1, "anos")
```

```
Digite sua idade: 21  
Você tem 21 anos.  
Ano que vem você terá 22 anos
```

Exemplo de entrada de dados  
do tipo inteiro (int)

# Saída Formatada

Python oferece uma série de formas de formatar texto com strings. Suponha que temos três variáveis para imprimir: nome (str), matricula (int), nota (float).

```
nome = "Lucas"  
matricula = 305016  
nota = 9.7539
```

```
print("Aluno:", nome, "(", matricula, ") - Nota:", nota)
```

```
Aluno: Lucas ( 305016 ) - Nota: 9.7539
```

# Saída Formatada

## Forma 1: printf style

baseado no operador `%`, permite uma série de sequências de caracteres de formatação de acordo com os tipos das variáveis.

The diagram illustrates the mapping of variables to format specifiers in the following Python code snippet:

```
print("Aluno: %s (%d) - Nota: %.1f" % (nome, matricula, nota))
```

Arrows indicate the following mappings:

- `nome` (blue box) maps to `%s` (blue box) via a blue arrow labeled `str(%s)`.
- `matricula` (red box) maps to `(%d)` (red box) via a red arrow labeled `int(%d)`.
- `nota` (purple box) maps to `%.1f` (purple box) via a purple arrow labeled `float(%f)`.

```
Aluno: Lucas (305016) - Nota: 9.8
```

Aqui o `.1f` indica que será impressa uma casa decimal no float

# Saída Formatada

## Forma 2: str.format()

permite especificar campos para substituição delimitados por chaves {}.

The diagram illustrates the mapping of arguments to format specifiers in the following code snippet:

```
print("Aluno: {} ({{}}) - Nota: {:.1f}".format(nome, matricula, nota))
```

Arrows indicate the following mappings:

- Posição 0 (blue):** Points from the first pair of curly braces `{}` to the `nome` argument.
- Posição 1 (red):** Points from the second pair of curly braces `{{}}` to the `matricula` argument.
- Posição 2 (purple):** Points from the float format specifier `{:.1f}` to the `nota` argument.

Também é possível especificar a posição (0, 1, 2...) do argumento dentro das chaves

Assim como no printf-style `.1f` pode ser usado para formatar a precisão do float



# Saída Formatada

## Forma 3: f-string

strings prefixadas com um `f` permitem incluir variáveis ou expressões entre chaves `{}` diretamente no conteúdo da string

**Atenção:** O prefixo `f` fica antes das aspas

```
print(f"Aluno: {nome} ({matricula}) - Nota: {nota:.1f}")
```

A formatação da precisão do float funciona da mesma forma que nos outros métodos

# Formatadores

```
valor = 8.29563
```

```
print(f"{{valor:6.2f}}")
```

São impressos dois  
espaços antes do 8

8.30

Número mínimo de caracteres a serem impressos. Se o resultado a é menor do que esse número, o resultado é completado com espaços em branco.

Número de dígitos a serem impressos após o ponto decimal, com arredondamento

Importante: o valor impresso na tela não altera o valor da variável armazenada na variável!

# Formatadores

```
valor = 99.7567892
print(f"Valor com 9f      = [{valor:9f}]")
print(f"Valor com 9.0f    = [{valor:9.0f}]")
print(f"Valor com 9.1f    = [{valor:9.1f}]")
print(f"Valor com 9.2f    = [{valor:9.2f}]")
print(f"Valor com 9.3f    = [{valor:9.3f}]")
print(f"Valor com 1.2f    = [{valor:1.2f}]")
```

```
Valor com 9f      = [99.756789]
Valor com 9.0f    = [      100]
Valor com 9.1f    = [     99.8]
Valor com 9.2f    = [     99.76]
Valor com 9.3f    = [     99.757]
Valor com 1.2f    = [99.76]
```

# Formatadores e strings

```
nome = "Maria"
```

```
print(f"[{nome}]")  
print(f"[{nome:10}]")  
print(f"[{nome:>20}]")  
print(f"[{nome:^50}]")
```

Símbolo	Alinhamento
:>	Direita
:^	Centro
:<	Esquerda

```
[Maria]  
[Maria      ]  
[                Maria]  
[                Maria                ]
```

# Formatadores e inteiros

```
num = 1234
```

```
print(f"[{num:2}]")  
print(f"[{num:6}]")  
print(f"[{num:08}]")
```

```
[1234]  
[ 1234]  
[00001234]
```

Preenche as casas  
vazias com zeros!

# Problema 01: Soma



Enunciado de um problema:

Ler dois valores informados pelo usuário através do teclado, calcular e informar a soma destes valores.



Implementação do  
Programa

Traduzir o algoritmo para Python

```
numero1 = int(input("Informe o primeiro número: "))
numero2 = int(input("Informe o segundo número: "))

soma = numero1 + numero2

print(f"{numero1} + {numero2} = {soma}")
```

# Problema 01: Soma



Enunciado de um problema:

Ler dois valores informados pelo usuário através do teclado, calcular e informar a soma destes valores.



Depuração e

Testes

Testar o programa e corrigir erros

Testar o programa:

- \* Números pequenos
- \* Números grandes
- \* Números negativos
- \* etc.

## Problema 02: Conversão



Enunciado de um problema:

Dado o preço de um produto em reais, converter este valor para o equivalente em dólares. O programa deverá ler do teclado o preço do produto e a taxa de conversão para o dólar.

Escreva o algoritmo, traduza-o para Python e teste seu programa



# Expressões e Operadores Aritméticos

Expressões aritméticas seguem o formato:

expressão1 operador expressão2

Principais  
operadores  
aritméticos



Operador	Formato	Função
+	e1 + e2	soma
-	e1 - e2	subtração
*	e1 * e2	multiplicação
/	e1 / e2	divisão
//	e1 // e2	divisão inteira
**	e1 ** e2	potenciação
%	e1 % e2	resto da divisão
-	- e1	negação

# Precedência de Operadores Aritméticos

**Ordem** pela qual a linguagem executa as expressões e seus operadores. Parênteses podem alterar a precedência padrão das expressões

Ordem de precedência	Operadores	Função
1º	(   )	Grupos entre parênteses
2º	**	Potenciação
3º		Negação
4º	*   /   //   %	Multiplicação, divisão, resto
5º	+   -	Adição, subtração

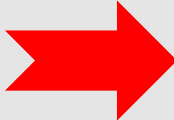
Lista completa de operadores e precedência em Python:


[docs.python.org/3/reference/expressions.html#operator-precedence](https://docs.python.org/3/reference/expressions.html#operator-precedence)

# Precedência de Operadores Aritméticos

Como representar a seguinte expressão em Python?

$$x = \frac{a + b}{2}$$

Opção 1)  $x = a + b / 2$   INCORRETO!

Opção 2)  $x = (a + b) / 2$   CORRETO!

# Operadores Matemáticos

```
'''
```

Programa que ilustra o funcionamento dos operadores aritméticos

```
'''
```

```
val_1 = 13
```

```
val_2 = 5
```

```
divisao = val_1 / val_2
```

```
div_inteira = val_1 // val_2
```

```
resto = val_1 % val_2
```

```
print(f"Divisão Real : {divisao:.1f}")
```

```
print(f"Divisão Inteira: {div_inteira}")
```

```
print(f"Resto Divisão : {resto}")
```

```
Divisão Real      : 2.6
```

```
Divisão Inteira: 2
```

```
Resto Divisão    : 3
```

# Resto da Divisão Inteira

O resto da divisão inteira é útil em várias situações!

Vamos supor o cálculo do número de botes salva-vidas necessários num transatlântico

Vamos supor 950 passageiros e tripulantes. Quantos botes de 23 lugares são necessários? E botes com 25 lugares?

Se o resto da divisão não der zero, significa que precisamos de mais um bote que irá ser parcialmente utilizado

# Resto da Divisão Inteira

```
'''  
Cálculo de Botes  
'''
```

```
passageiros = 950
```

```
# Com botes de 23 Lugares
```

```
lugares_no_bote = 23
```

```
print(f"São necessários {passageiros // lugares_no_bote} botes de {lugares_no_bote}  
lugares.")
```

```
print(f"Sobram {passageiros % lugares_no_bote} passageiros")
```

```
# Com botes de 25 Lugares
```

```
lugares_no_bote = 25
```

```
print(f"São necessários {passageiros // lugares_no_bote} botes de {lugares_no_bote}  
lugares.")
```

```
print(f"Sobram {passageiros % lugares_no_bote} passageiros")
```

São necessários 41 botes de 23 lugares.  
Sobram 7 passageiros  
São necessários 38 botes de 25 lugares.  
Sobram 0 passageiros

## Problema 03: Troco



Enunciado de um problema:

Faça um programa que:

- leia um valor inteiro.
- calcule o menor número de notas necessárias para completar este valor. Lembre que as notas correntes no real são 100, 50, 20, 10, 5, 2 e 1 (incluindo aqui também a moeda).

OBS: não vamos analisar ainda o caso da parte fracionária. Podemos implementar o mesmo algoritmo para ele tratando-o como um inteiro.

# Problema 03: Troco

Exemplo de Execução:



```
Informe um valor inteiro: 1299
```

```
Valor lido: R$1299,00
```

```
notas de 100: 12
```

```
notas de 50: 1
```

```
notas de 20: 2
```

```
notas de 10: 0
```

```
notas de 5: 1
```

```
notas de 2: 2
```

```
notas de 1: 0
```



# Funções da Biblioteca math

Função	Exemplo	Descrição
ceil	<code>ceil(x)</code>	Arredonda o número real para cima: <code>ceil(3.2) = 4</code>
floor	<code>floor(x)</code>	Arredonda o número real para baixo: <code>floor(3.8) = 3</code>
sin	<code>sin(x)</code>	Seno de x (em radianos)
cos	<code>cos(x)</code>	Cosseno de x (x em radianos)
tan	<code>tan(x)</code>	Tangente de x (em radianos)
exp	<code>exp(x)</code>	e elevado na potência x
log	<code>log(x)</code>	Logaritmo natural de x (base e)
log10	<code>log10(x)</code>	Logaritmo de x na base 10
pow	<code>pow(x, y)</code>	Calcula x elevado à potência y
sqrt	<code>sqrt(x)</code>	Raiz quadrada de x
fabs	<code>fabs(x)</code>	Valor absoluto de x – Retorna um float: <code>fabs(5.1) = 5.1</code>
fmod	<code>fmod(x, y)</code>	Resto da divisão de dois floats: <code>fmod(9.8, 4.0) = 1.8</code>

# Funções da Biblioteca math

```
import math # Lembre-se de sempre importar a biblioteca
```

```
# Função ceil
```

```
print(f"ceil(9.2) = {math.ceil(9.2)}")  
print(f"ceil(-9.8) = {math.ceil(-9.8)}")
```

```
# Função floor
```

```
print(f"floor(9.2) = {math.floor(9.2)}")  
print(f"floor(-9.8) = {math.floor(-9.8)}")
```

```
# Funções Trigonométricas
```

```
print(f"sin(0) = {math.sin(0)}")  
print(f"cos(0) = {math.cos(0)}")  
print(f"tan(0) = {math.tan(0)}")
```

# Funções da Biblioteca math

...

*# Exponencial e Logaritmo*

```
print(f"exp(1) = {math.exp(1):.7}")
print(f"exp(2) = {math.exp(2):.7}")
print(f"log(2.718282) = {math.log(2.718282):.2f}")
print(f"log(7.389056) = {math.log(7.389056):.2f}")
print(f"log10(10) = {math.log10(10)}")
print(f"log10(100) = {math.log10(100)}")
```

*# Raiz quadrada e Potência*

```
print(f"2^7 = {math.pow(2, 7)} ou {2 ** 7}")
print(f"9^0.5 = {math.pow(9, 0.5)} ou {9 ** 0.5}")
print(f"raiz(9) = {math.sqrt(9)}")
```

# Funções da Biblioteca math

...

*# Valor absoluto inteiro (não está na biblioteca matemática!)*

```
print(f"abs(5) = {abs(5)}")
```

```
print(f"abs(-5) = {abs(-5)}")
```

*# Valor absoluto para float*

```
print(f"fabs(5.1) = {math.fabs(5.1)}")
```

```
print(f"fabs(-5.1) = {math.fabs(-5.1)}")
```

*# mod para float*

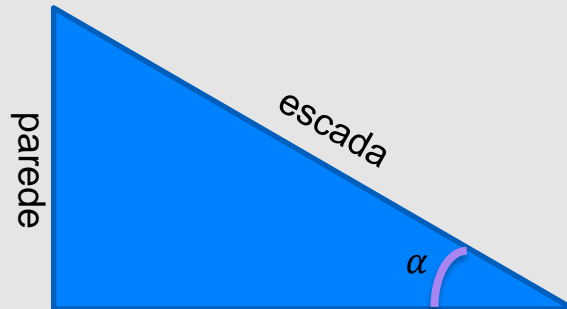
```
print(f"fmod(9.8, 4) = {math.fmod(9.8, 4):.2f}")
```

# Problema 04: Escada



Enunciado de um problema:

Faça um programa que receba a medida do ângulo (em graus) formado por uma escada apoiada no chão e encostada na parede e a altura da parede onde está a ponta da escada. Calcule e mostre a medida dessa escada.



# Problema 04: Escada

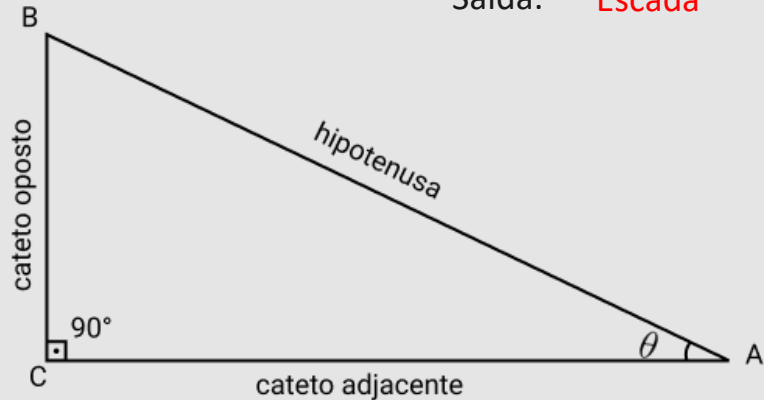


## Análise do Problema

Relações trigonométricas no triângulo retângulo!

Entrada: Ângulo Parede

Saída: Escada



$$\textit{seno}(\theta) = \frac{\textit{cateto oposto}}{\textit{hipotenusa}} = \frac{BC}{AB}$$

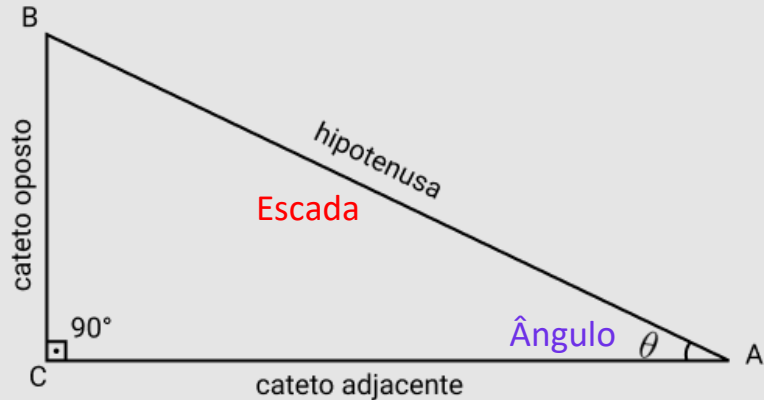
$$\textit{cosseno}(\theta) = \frac{\textit{cateto adjacente}}{\textit{hipotenusa}} = \frac{AC}{AB}$$

$$\textit{tangente}(\theta) = \frac{\textit{cateto oposto}}{\textit{cateto adjacente}} = \frac{BC}{AC}$$

# Problema 04: Escada



## Análise do Problema



Relações trigonométricas no triângulo retângulo!

$$\textit{seno}(\theta) = \frac{\textit{cateto oposto}}{\textit{hipotenusa}}$$

$$\textit{seno}(\textit{ângulo}) = \frac{\textit{parede}}{\textit{escada}}$$

$$\textit{seno}(\textit{ângulo}) \times \textit{escada} = \textit{parede}$$

$$\textit{escada} = \frac{\textit{parede}}{\textit{seno}(\textit{ângulo})}$$

## Problema 04: Escada

```
import math

angulo = float(input("Informe o ângulo da escada com o chão (em graus): "))
parede = float(input("Informe a altura da parede (em metros): "))

# Converte o ângulo para radianos
radianos = angulo * math.pi / 180

escada = parede / math.sin(radianos)

print(f"A altura da escada é de {escada:.2f} metros")
```