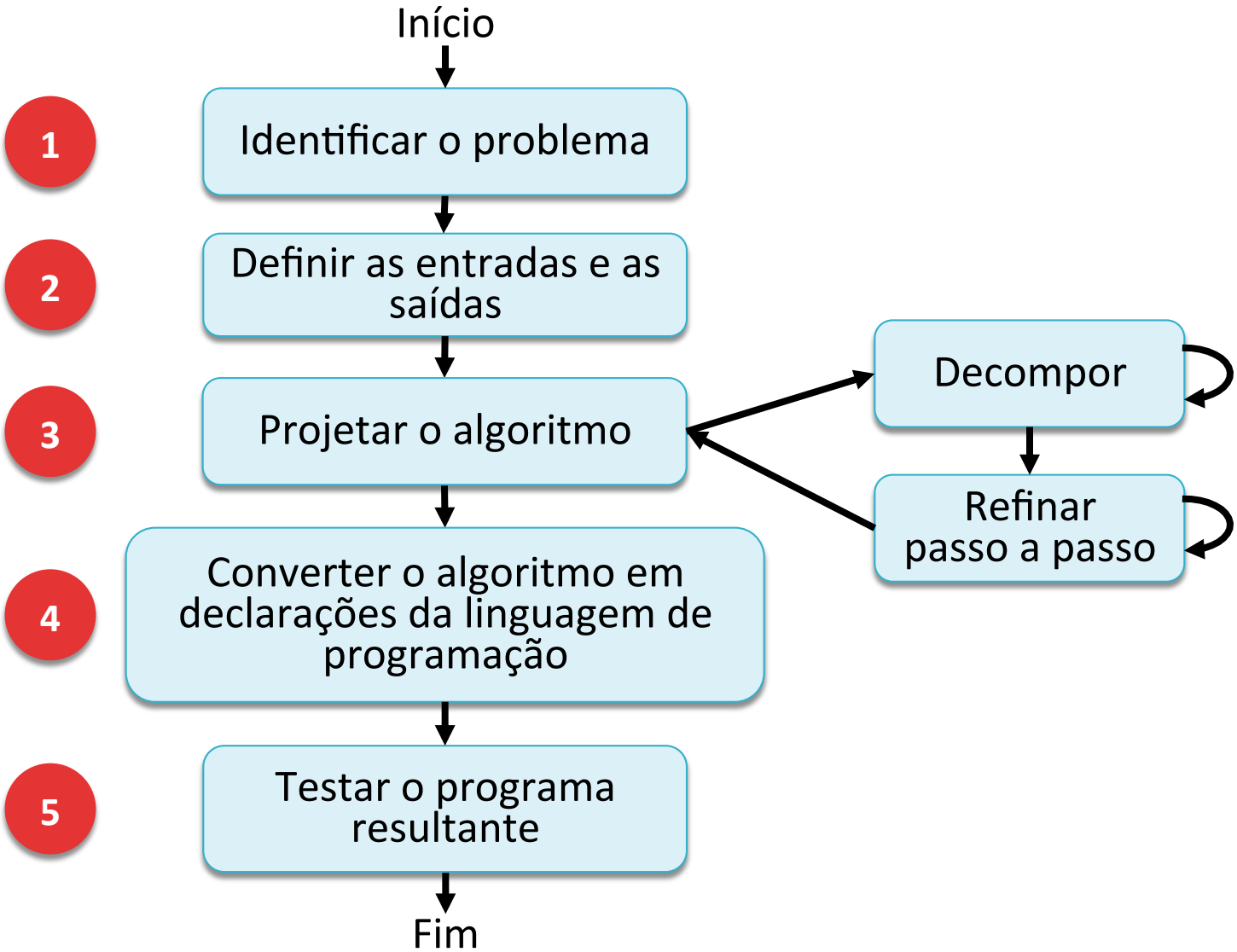
IEC037
Introdução à Programação de Computadores
Aula 09 – Estruturas Condicionais em Python

Turma: 03
Professor: Leandro Galvão
E-mail: galvao@icomp.ufam.edu.br
Página: ipc-t03.weebly.com

Resolução de Problemas Algorítmicos

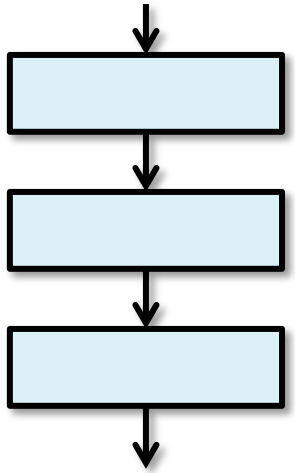


Conteúdo

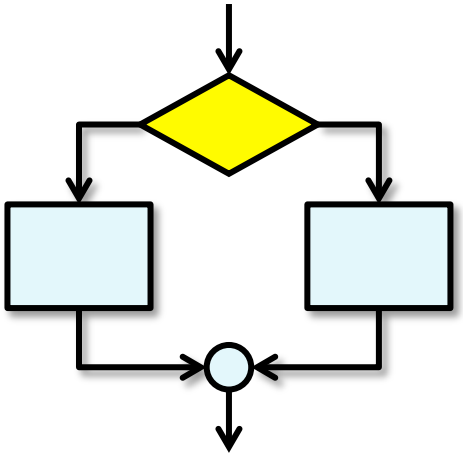
- Estruturas Condicionais Simples
- Estruturas Condicionais Compostas
- Como montar uma condição?
- Estruturas Condicionais Encadeadas

Estruturas de Programação

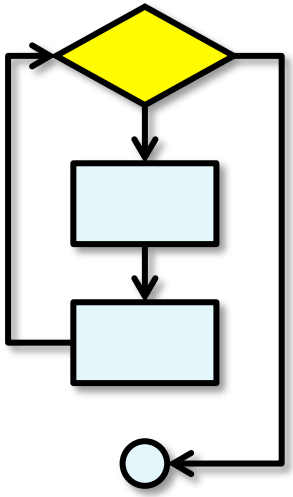
- Qualquer programa de computador pode ser escrito combinando-se os três tipos básicos de estruturas de programação:



Sequencial



Condicional



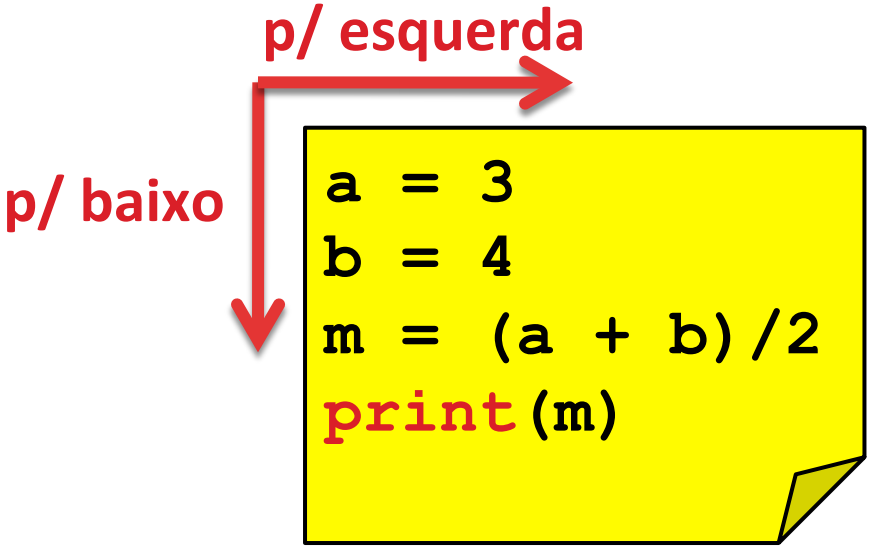
Repetição

Estruturas de Programação

- Teorema provado em 1966 por Corrado Böhm (1923-) e Giuseppe Jacopini (1936-2001) no artigo: "*Flow Diagrams, Turing Machines And Languages With Only Two Formation Rules*".

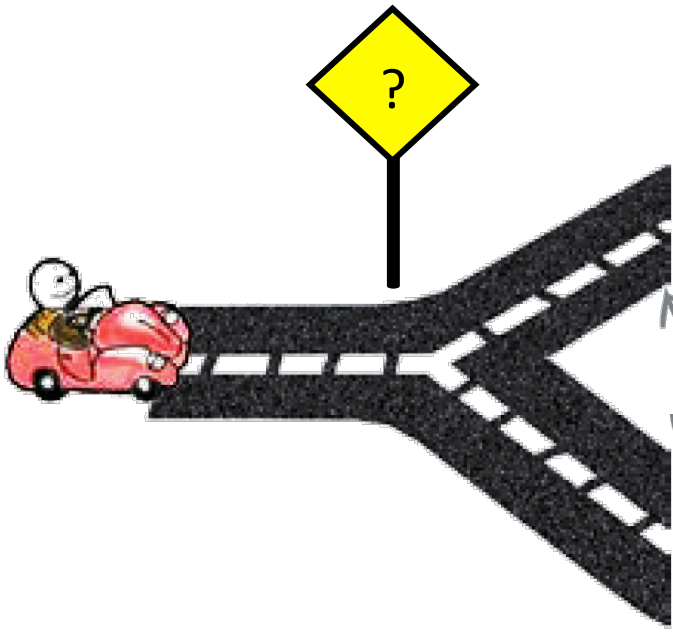
Estrutura Sequencial

- É a estrutura de programação mais simples.
- O fluxo de comandos do algoritmo segue a mesma sequência linear da nossa escrita:
  - De cima para baixo
  - Da esquerda para direita



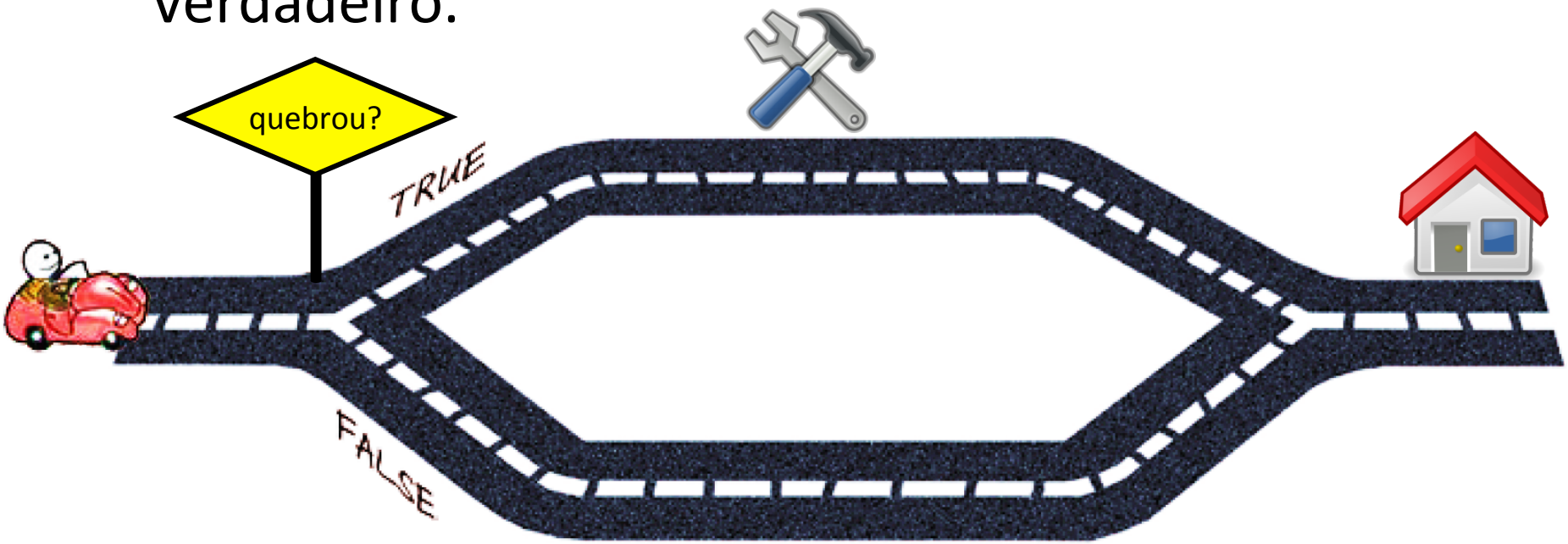
Estrutura Condicional

- Permite alterar o fluxo de execução, de forma a selecionar qual parte do algoritmo deve ser executada.
- Essa decisão é tomada a partir de uma condição, que pode resultar apenas em:
  - Verdade, ou
  - Falsidade



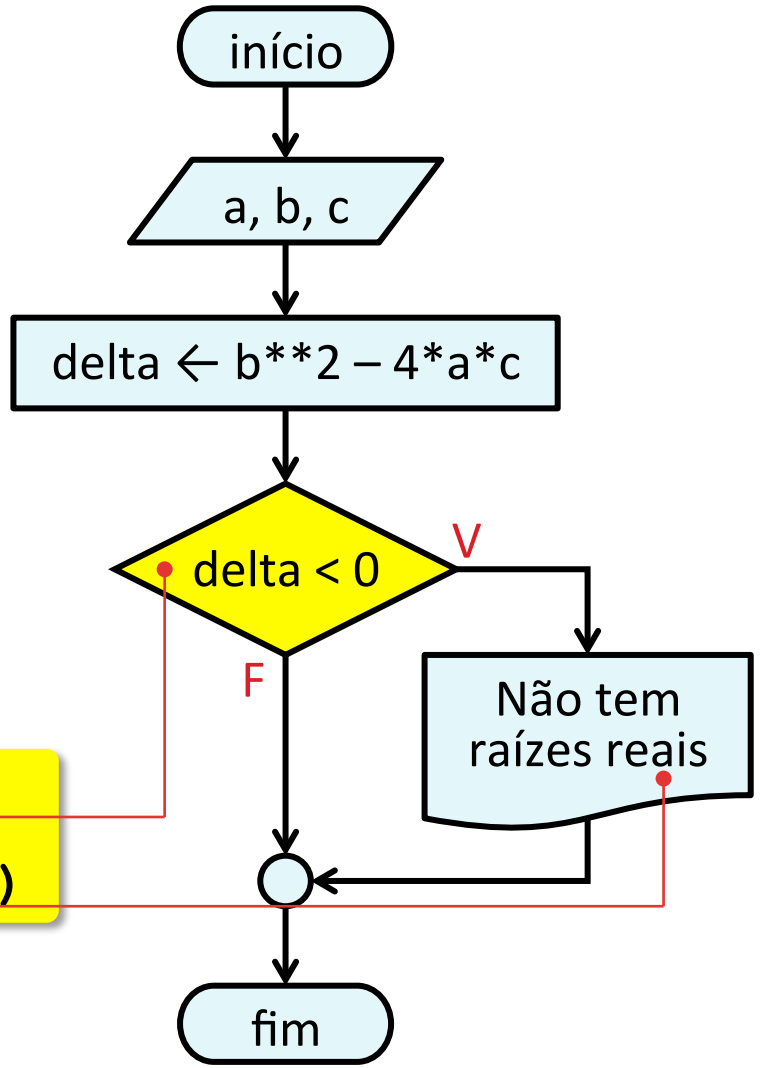
Condição verdadeira, condição falsa

- Verdadeiro ou falso são valores lógicos. São atributos da expressão condicional.
- O funcionamento correto do seu script não está condicionado a resultados lógicos com valor verdadeiro.



Estrutura Condicional Simples

- Quando a condição é verdadeira, o "bloco verdade" é executado.
- Quando a condição é falsa, o "bloco verdade" **não** é executado.




```
if (delta < 0):
    print('Nao tem raizes reais')
```

Estrutura Condicional Simples
:: Em Python

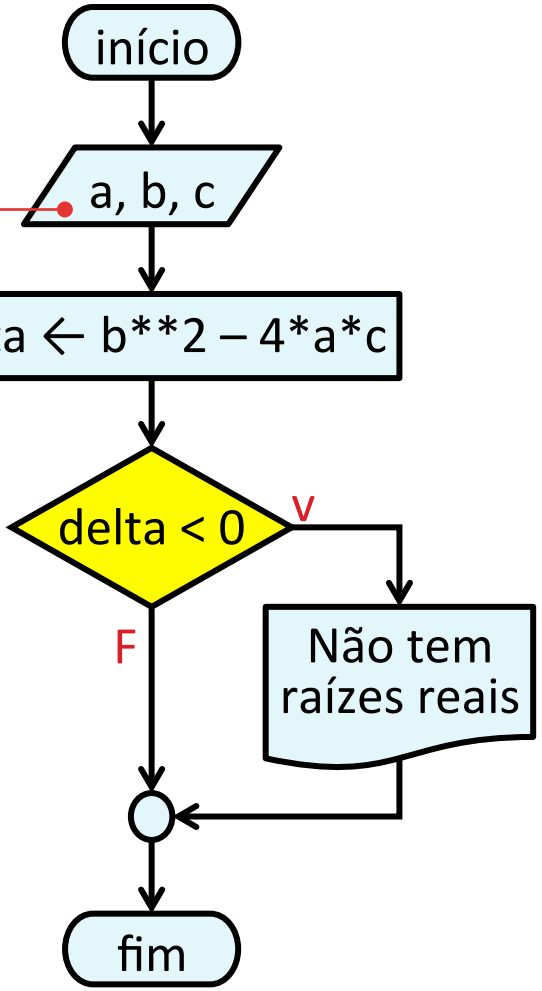
```
a = float(input("Digite a: "))
b = float(input("Digite b: "))
c = float(input("Digite c: "))
```

```
delta = b**2 - 4 * a * c
```

```
if (delta < 0):
    print("Nao tem raizes reais")
```

Condições sempre terminam com sinal de **dois pontos**

Comandos internos às condições **devem ser recuados** (tecla TAB)



Exemplo A

```
# Script que calcula o valor do ingresso
# a depender de se houver meia entrada

op = input("Meia entrada? (S/N) ")
ingresso = 30

if (op == "S"):
    ingresso = ingresso/2

print("Valor do ingresso: ", ingresso)
```

Recuo do comandos dependente
da condição (tecla TAB)

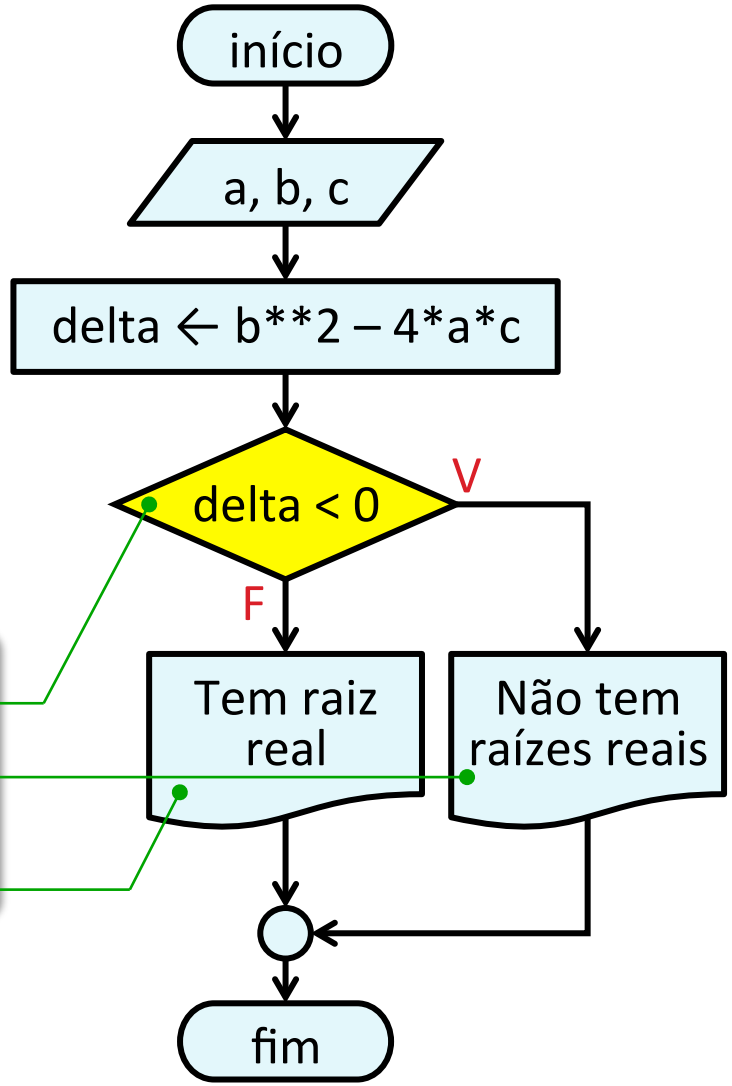
Conteúdo

- Estruturas Condicionais Simples
- **Estruturas Condicionais Compostas**
- Como montar uma condição?
- Estruturas Condicionais Encadeadas

Estruturas Condicionais Compostas

- Quando a condição é verdadeira, o "bloco verdade" é executado.
- Quando a condição é falsa, o "bloco falsidade" é executado.

```
if (delta < 0):
    print("Nao tem raiz real")
else:
    print("Tem raiz real")
```



Estruturas Condicionais Compostas
:: Em Python

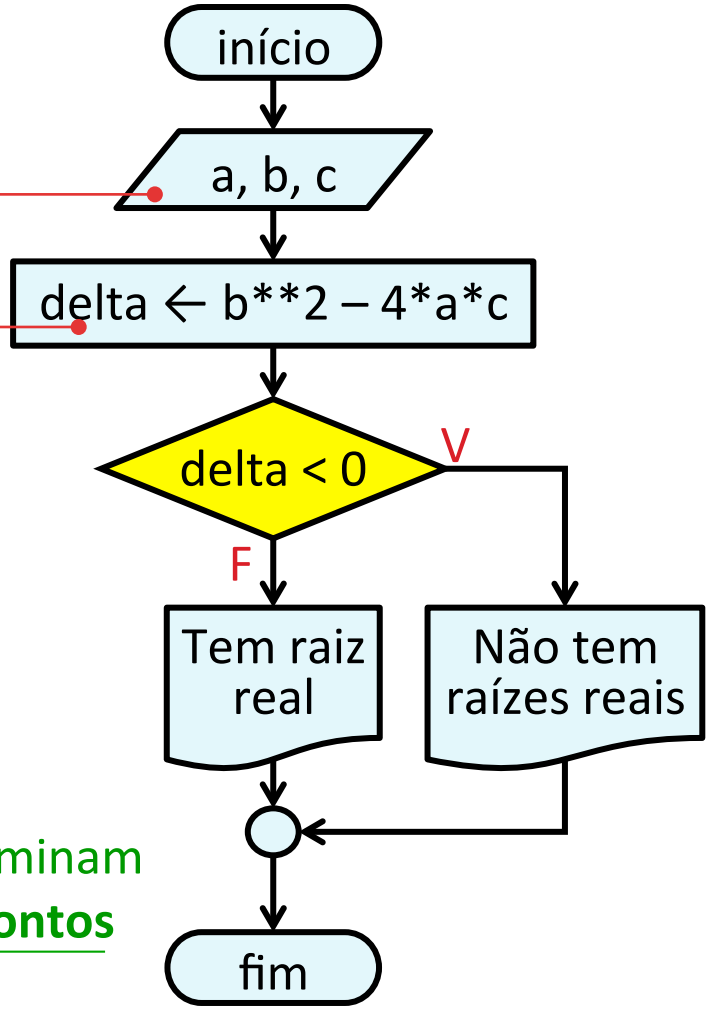
```
a = float(input("Digite a: "))
b = float(input("Digite b: "))
c = float(input("Digite c: "))
```

```
delta = b**2 - 4 * a * c
```

```
if (delta < 0):
    print("Nao tem raiz real")
else:
    print("Tem raiz real")
```

if e **else** sempre terminam com sinal de **dois pontos**

Comandos internos ao **if** e ao **else** devem ser **recuados**



Exemplo B

```
# Script que verifica se o aluno passou
ou nao com base na media

m = float(input("Digite sua media: "))

if (m >= 5.0):
    print("Passou")
else:
    print("Reprovou")
```

if e **else** sempre terminam com sinal de **dois pontos**

Comandos internos ao **if** e ao **else** devem ser **recuados**

Indentação

- O comando **`else`** deve estar alinhado com o comandos **`if`** correspondente.
- Todos os comandos de um mesmo bloco deverão ter o mesmo recuo.

Indentação
:: Cuidados

Indentação Válida

```
if (condição):
    comando
    comando
else:
    comando
    comando
```

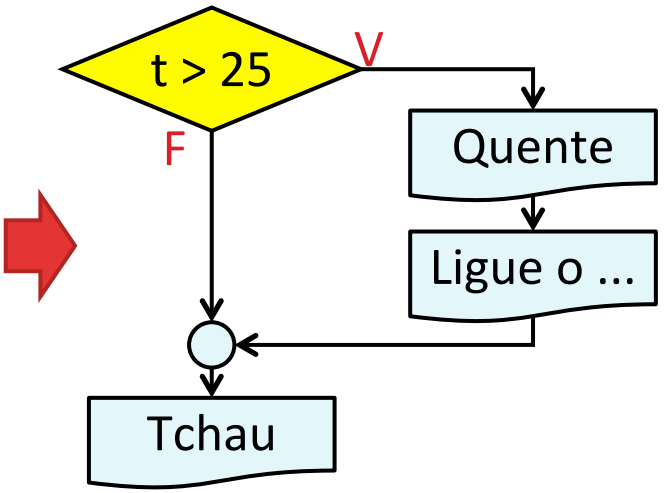
Indentação Inválida

```
if (condição):
    comando
   comando
  else:
     comando
     comando
```

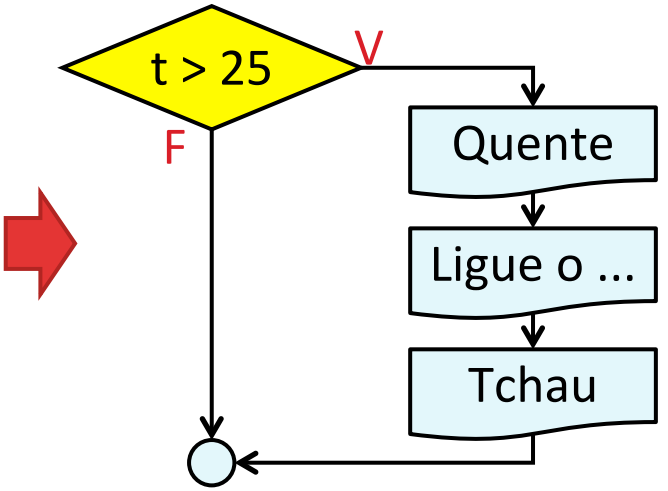
```
if (condição):
    comando
    comando
else:
    comando
      comando
```

Indentação
:: Diferenças

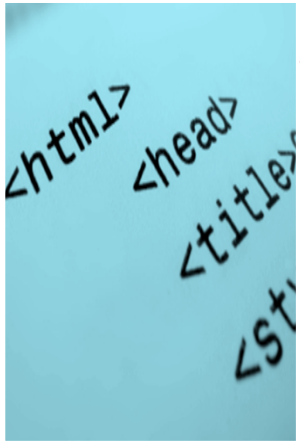
```
if (temp > 25):
    print("Quente")
    print("Ligue o ventilador")
print("Tchau")
```




```
if (temp > 25):
    print("Quente")
    print("Ligue o ventilador")
    print("Tchau")
```

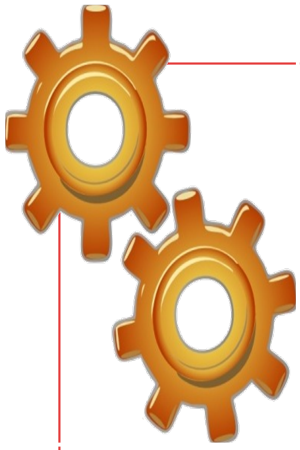


Não confunda



Indentação (~~identação~~)

- Inserção de espaços em um código de linguagem de programação



Endentação

- Encaixe dos dentes de uma peça denteada com os de outra

Problema 1

- Um radar de trânsito verifica a velocidade dos veículos.
- Caso ultrapassem 60 km/h, emite-se um registro de multa.
- O valor da multa é de R $ 200,00 mais R$ 3,00 para cada 1 km/h acima do limite.
- Escreva um programa para determinar o valor da multa.

Problema 1

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Velocidade	km/h	≥ 0
Saídas	Multa	R$	≥ 0

Problema 1

4 – Codificar em Python

```
# Entrada de dados e definicao de constantes
vel = float(input("Informe a velocidade: "))
lim = 60          # Limite de velocidade

# Calculo do valor da multa
if (vel > lim):
    multa = 200 + 3 * (vel - lim)
else:
    multa = 0

# Exibicao de resultados
print(multa)
```

Problema 1

5 – Testar o script resultante

Problema 2

- Projete um algoritmo para uma máquina caça-níquel que gere 3 números aleatórios entre 1 e 10.
- Se os três números forem iguais, o jogador ganha. Caso contrário, ele perde.

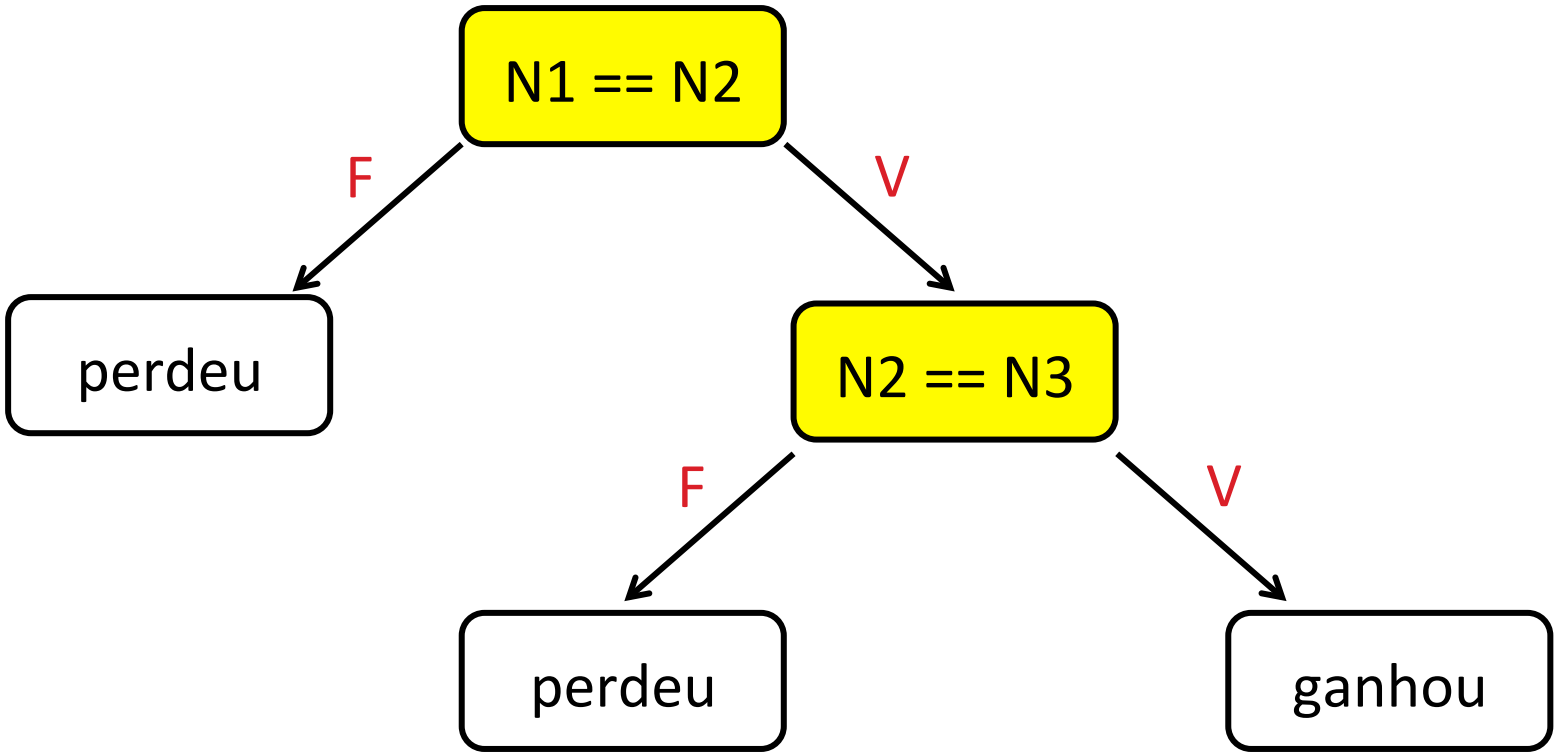
Problema 2

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	N1	---	[1,10]
	N2	---	[1,10]
	N3	---	[1,10]
Saídas	Sucesso no jogo	---	{Perdeu, Ganhou}

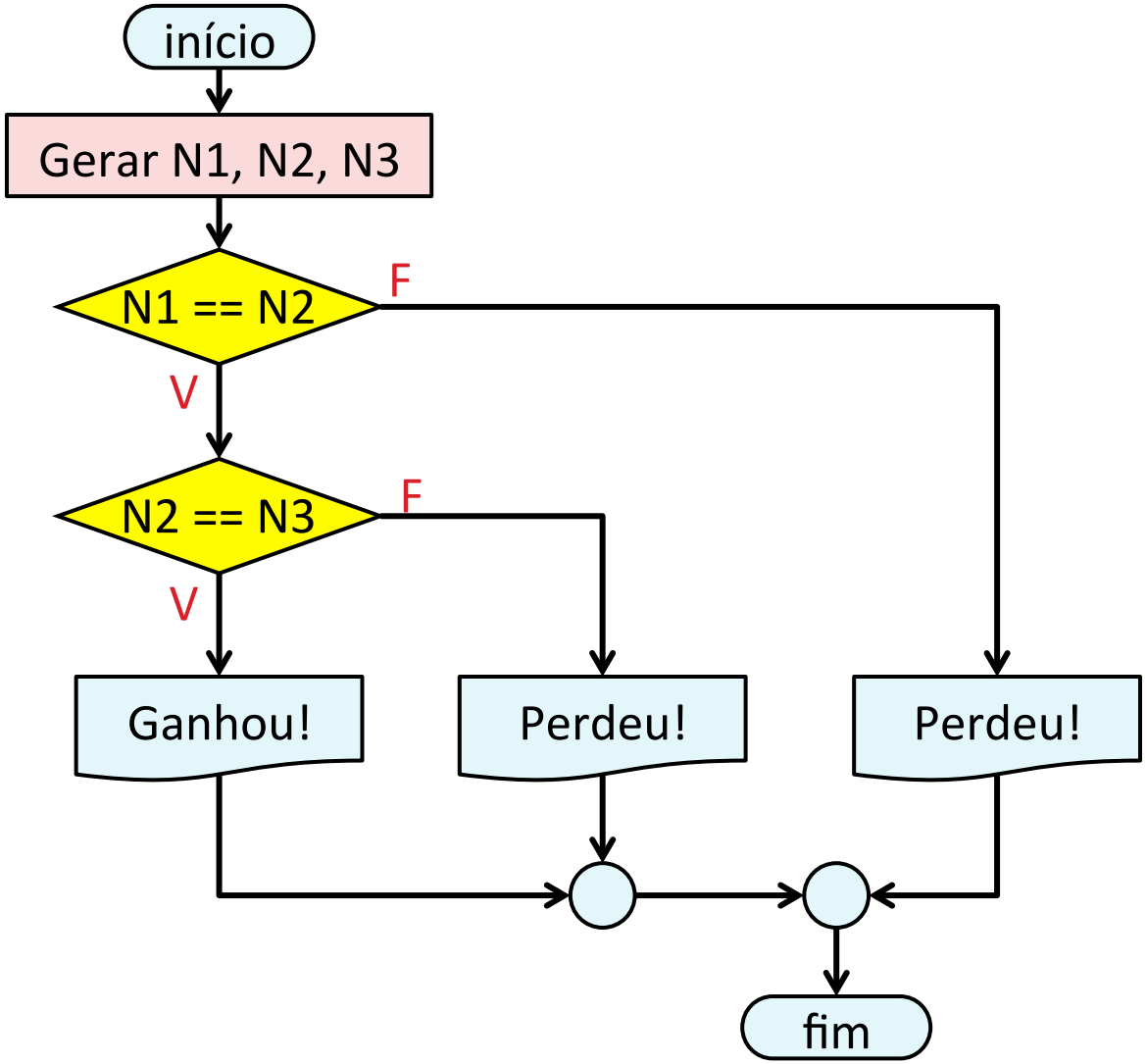
Problema 2

3 – Projetar algoritmo



Problema 2

3 – Projetar algoritmo



Como gerar números aleatórios?

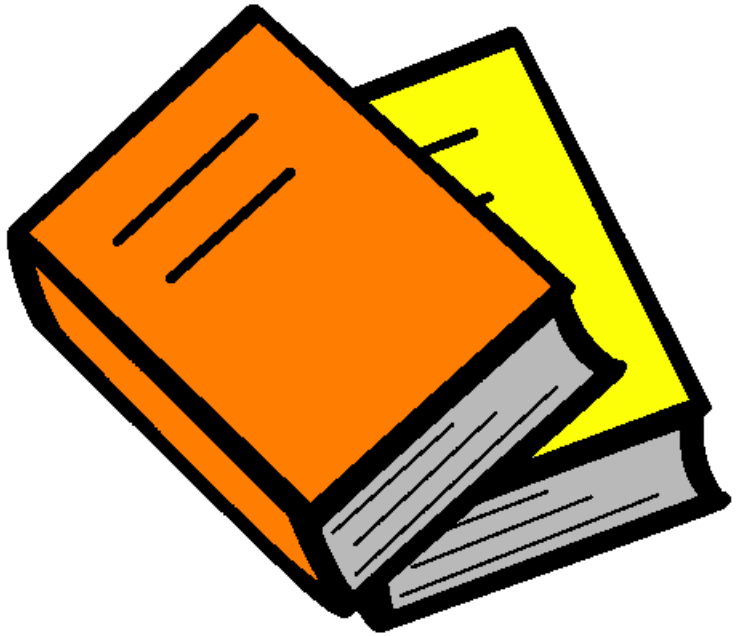
Problema 2

4 – Codificar em Python

```
# Biblioteca de nos. aleatorios
import random

# Gera nos. aleatorios entre 1 e 10
n1 = random.randint(1,10)
n2 = random.randint(1,10)
n3 = random.randint(1,10)

if (n1 == n2):
    if (n2 == n3):
        print("Ganhou")
    else:
        print("Perdeu")
else:
    print("Perdeu")
```

Módulos em Python (Bibliotecas)

- Bibliotecas organizam funções bastante utilizadas em arquivos diferentes.
- Assim, elas podem ser chamadas quando necessário, sem ter de reescrever tudo.
- Em Python, as bibliotecas também são conhecidas como módulos.



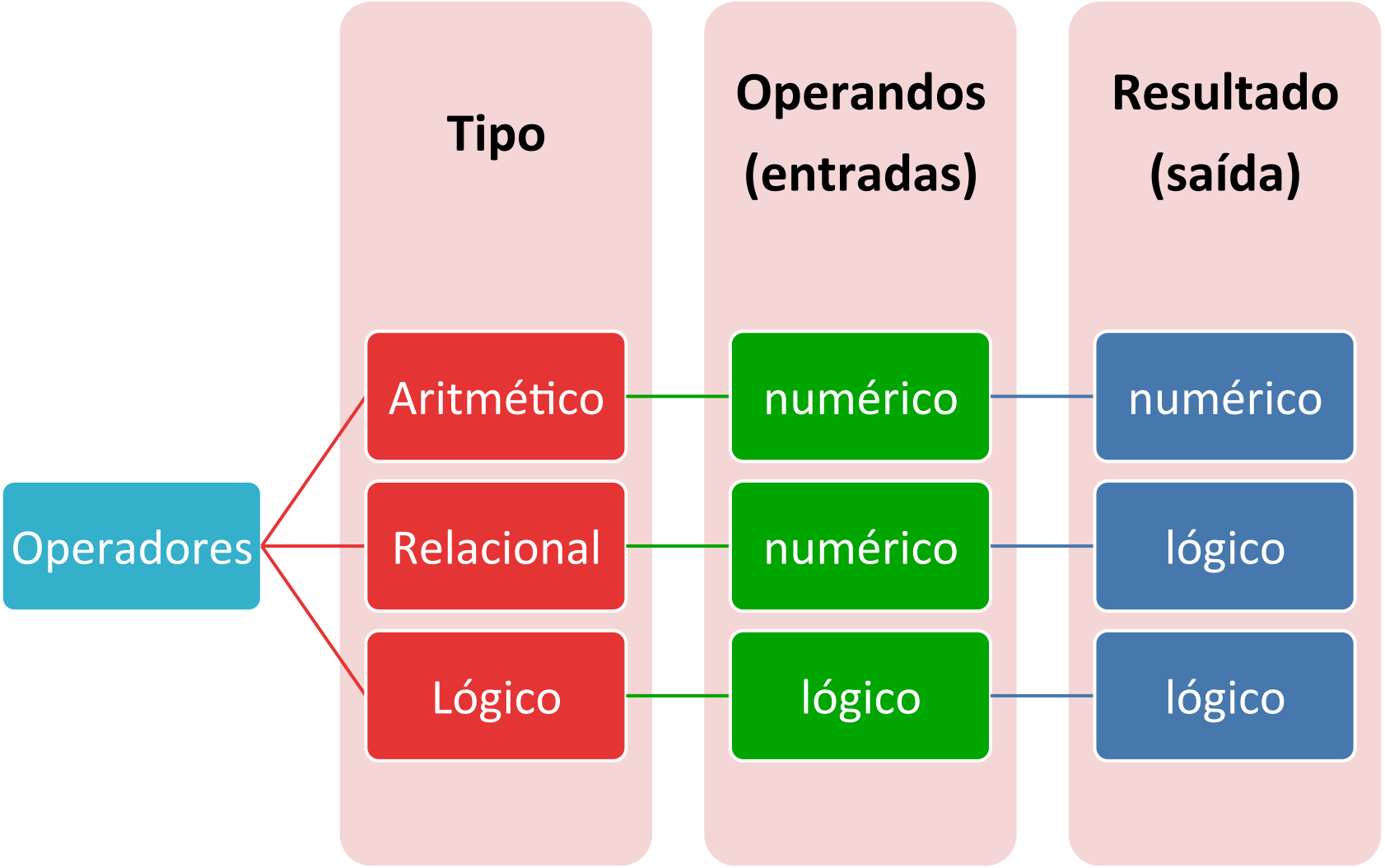
Uso:

<módulo>.<função>

Conteúdo

- Estruturas Condicionais Simples
- Estruturas Condicionais Compostas
- **Como montar uma condição?**
- Estruturas Condicionais Encadeadas

Tipos de operadores



Precedência entre operadores

Operador	Significado	Precedência
`()`	Grupos entre parênteses	+
`**`	Potenciação	
`-`	Negação	
`* / % //`	Multiplicação, divisão real, resto, divisão inteira	
`+ -`	Subtração, adição	
`> >= < <= == !=`	Comparações	
`not`	NÃO lógico	
`and`	E lógico	
`or`	OU lógico	
`=`	Atribuição	–

Exemplos

```
(x <= 20 or x >= 40)
```

```
(salario > 1000 and
idade > 18)
```

x	resultado
10	V
20	V
30	F
40	V
50	V

salario	idade	resultado
900	18	F
1000	19	F
1100	17	F
1200	22	V

Exemplos

```
(m - 4 > m / 2)
```

```
(num % 2 != 0 and
contador < 50)
```

m	resultado
2	F
8	F
30	V

num	contador	resultado
1231	51	F
1232	50	F
1233	49	F
1234	48	V

Atenção

:: Compare variáveis do mesmo tipo

```
x = 4

if ("4" == x):
    print("igual")
else:
    print("diferente")
```

- No exemplo acima, a variável **x** é do tipo inteiro, mas a expressão **"4"** representa um caractere, e não um número.

Armadilhas
:: Números float são aproximações

- Há infinitos números reais.
- A memória do computador é um recurso finito.
- Logo, não há como representar todos os números reais em memória.
- Consequentemente, representamos aproximações.

```
u =  11111113
v = -11111111
w = 7.51111111
print((u + v) + w)
    9.51111111

print(u + (v + w))
    9.511111110448837

u + (v + w) == (u + v) + w
    False
```

Estabeleça um nível mínimo de precisão

Alternativa 1

```
u =  11111113
v = -11111111
w = 7.51111111
x = (u + v) + w
y = u + (v + w)

x == y
  False

round(x,6) == round(y,6)
  True
```

Alternativa 2

```
u =  11111113
v = -11111111
w = 7.51111111
x = (u + v) + w
y = u + (v + w)

x == y
  False

abs(x-y) < 0.0000001
  True
```

Funções `round` e `abs`

`round(x, n)`

- Arredonda um número **`x`** em **`n`** casas decimais.

`abs(z)`

- Determina o módulo de um número real **`z`**, ou seja, sua distância até o zero.

Problema 3

- Um gerente quer medir a eficiência de processos em sua empresa.
- Um processo X começou no horário h1 e terminou no mesmo dia, no horário h2, também medido em horas e minutos.
- Quanto tempo durou o processo?

Problema 3
2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Horário 1 (hh1, mm1)	horas, minutos	[0; 23], [0; 59]
	Horário 1 (hh2, mm2)	horas, minutos	[0; 23], [0; 59]
Saídas	Diferença de tempo (Δh, Δm)	horas, minutos	[0; 23], [0; 59]

Problema 3

3 – Projetar algoritmo

Caso 1

mm2 ≥ mm1

Início: 9h 17min
Fim: 15h 43min

Δm = 43 – 17 = 26min
Δh = 15 – 9 = 6h

Caso 2

mm2 < mm1

Início: 9h 43min
Fim: 15h 17min

Δm = 17 – 43 = 34min (-1h)
Δh = 15 – 9 – 1 = 5h

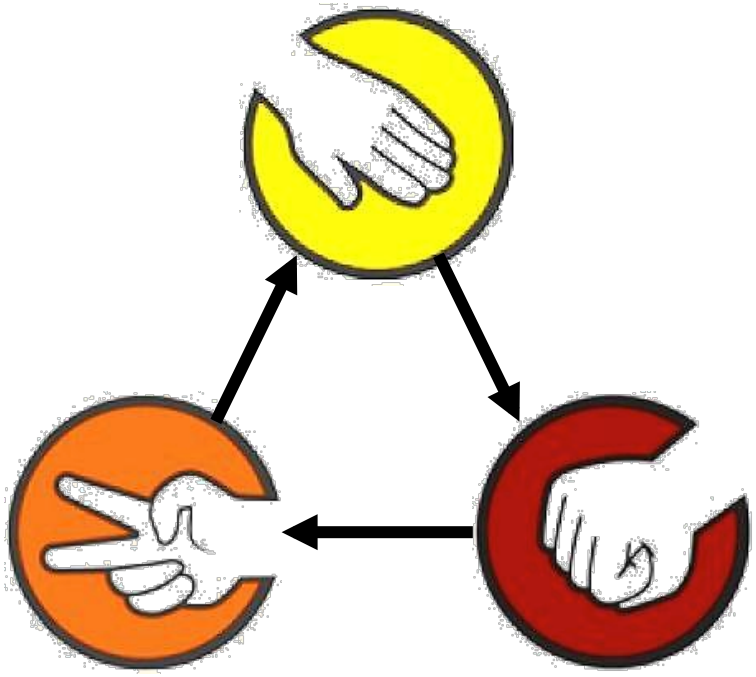
Problema 3

4 – Codificar em Python

```
# Entrada de dados
hh1 = int(input("Hora inicial: "))
mm1 = int(input("Minuto inicial: "))
hh2 = int(input("Hora final: "))
mm2 = int(input("Minuto final: "))

# Diferenca de minutos
dm = (mm2 - mm1) % 60

# Diferenca de horas
if (mm2 >= mm1):
    dh = hh2 - hh1
else:
    dh = hh2 - hh1 - 1

print(dh, dm)
```

Problema 4

- Duas pessoas jogam pedra, papel, tesoura.
- Como determinar quem ganhou?



Problema 4

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Mão do J1	---	{Pedra, Papel, Tesoura}
	Mão do J2	---	{Pedra, Papel, Tesoura}
Saídas	Vencedor	---	{J1, J2}

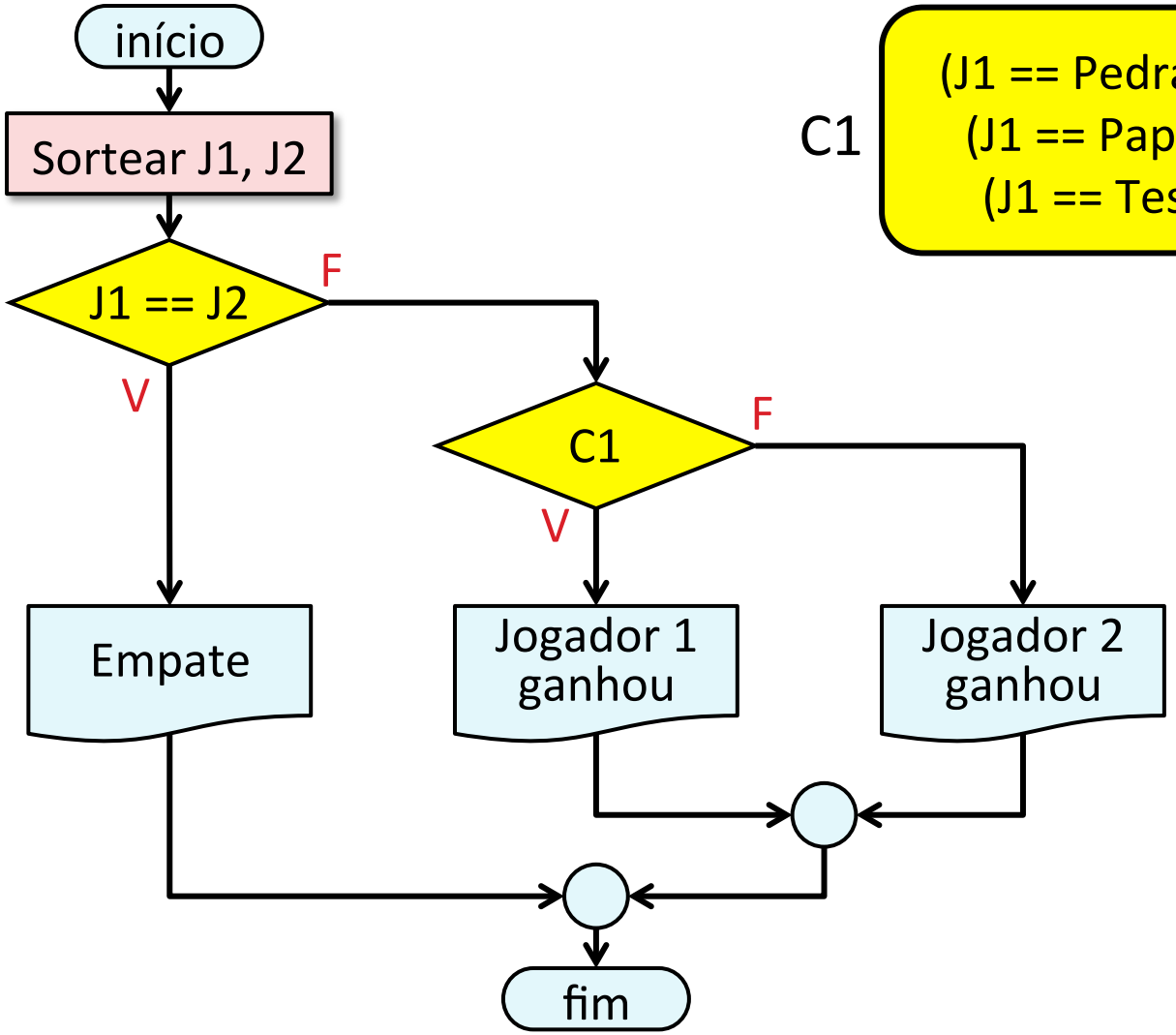
Problema 4

3 – Projetar algoritmo – versão 1

- Se J1 == J2
  - Empate
- J1 ganha quando:
  - (J1 == Pedra **E** J2 == Tesoura) **OU**
  - (J1 == Papel **E** J2 == Pedra) **OU**
  - (J1 == Tesoura **E** J2 == Papel)
- J2 ganha caso contrário

Problema 4

3 – Projetar algoritmo – versão 1



C1

(J1 == Pedra **E** J2 == Tesoura) **OU**
(J1 == Papel **E** J2 == Pedra) **OU**
(J1 == Tesoura **E** J2 == Papel)

?

Como fazer o sorteio?

Problema 4

4 – Codificar em Python – sorteio

```
# Sorteio do jogo "Pedra, Papel, Tesoura"
# Pedra   = 0
# Papel   = 1
# Tesoura = 2
import random

j1 = random.randint(0,2)
j2 = random.randint(0,2)
```

(J1 == Pedra **E** J2 == Tesoura)
OU
(J1 == Papel **E** J2 == Pedra) **OU**
(J1 == Tesoura **E** J2 == Papel)

(J1 == 0 **E** J2 == 2) **OU**
(J1 == 1 **E** J2 == 0) **OU**
(J1 == 2 **E** J2 == 1)

Problema 4

4 – Codificar em Python – versão 1

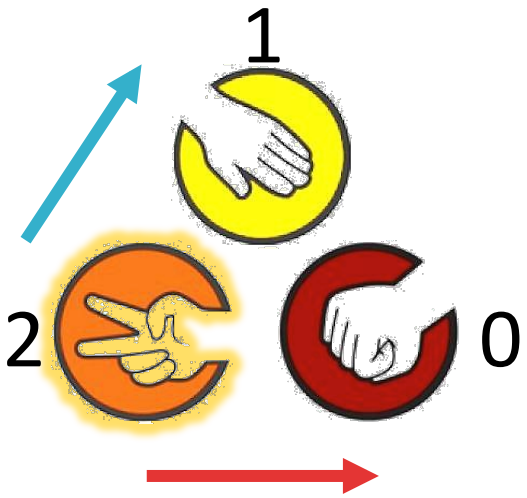
```
if (j1 == j2):
    print("Empate.")
else:
    if (((j1 == 0) and (j2 == 2)) or ((j1 == 1) and (j2
== 0)) or ((j1 == 2) and (j2 == 1))):
        print("Jogador 1 ganhou.")
    else:
        print("Jogador 2 ganhou.")
```

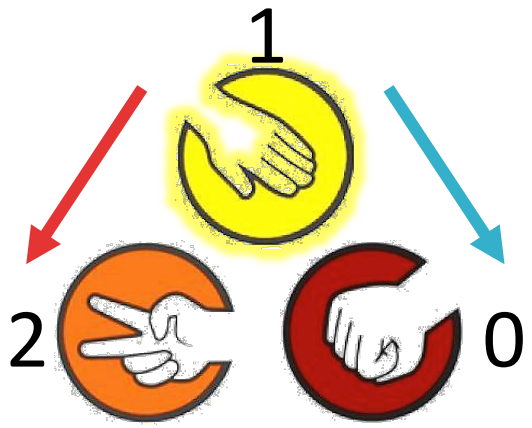
Abstração

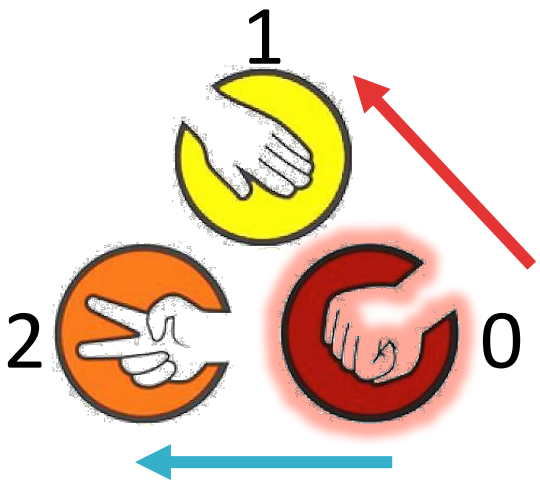
- Associar objetos (pedra, papel, tesoura) a números é uma forma de abstração.
- Adotamos essa abstração para simplificar o código do sorteio.
- Tal simplificação não poderia ser também aplicada ao teste da condição?

Problema 4

:: Repensando o Jogo







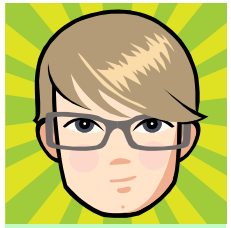
Problema 4

:: Repensando o Jogo

- Por exemplo, é natural pensar que três horas antes de 2h no relógio resulta em 11h.
- Inconscientemente, fazemos as seguintes contas:
  - 2h – 3h = –1h
  - –1h + 12h = 11h

Problema 4

:: Repensando o Jogo



2 1 — 2 – 1 = 1

2 0 — 2 – 0 = 2

1 0 — 1 – 0 = 1

Ganhou quem tirou o número uma unidade maior

1 2 — 1 – 2 = -1

0 2 — 0 – 2 = -2

0 1 — 0 – 1 = -1

Ganhou quem tirou o número uma unidade maior, se pensarmos como em um relógio de 3 posições

Operador %

- O operador % tem uma propriedade interessante:

`a % b`

Se a é positivo

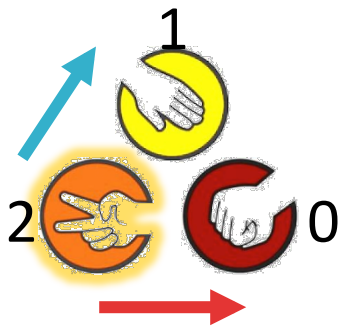
- Resultado: resto da divisão

Se a é negativo

- Resultado: determinado por aritmética circular, como em um relógio.

Problema 4

:: Repensando o Jogo

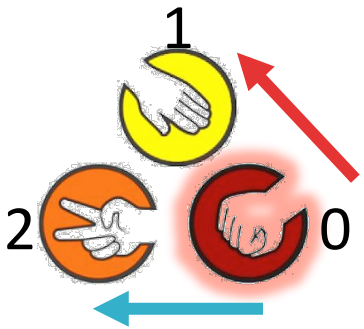



```
(2 - 1) % 3 = 1
```

```
(2 - 0) % 3 = 2
```


```
(1 - 0) % 3 = 1
```

```
(1 - 2) % 3 = 2
```

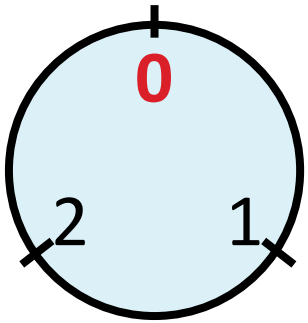



```
(0 - 2) % 3 = 1
```

```
(0 - 1) % 3 = 2
```

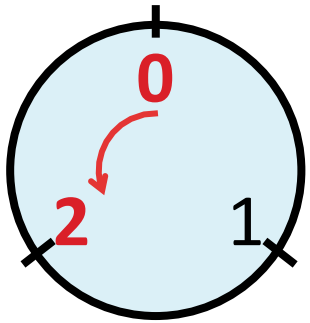
Aritmética circular (ou modular)

Uma casa para trás

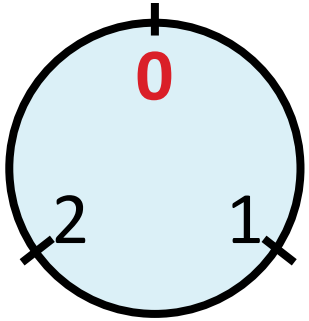


[0 – 1] = -1

(0 – 1) % 3 = 2

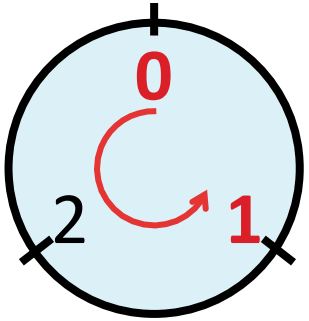


Duas casas para trás

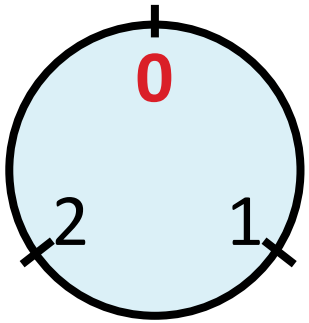


[0 – 2] = -2

(0 – 2) % 3 = 1

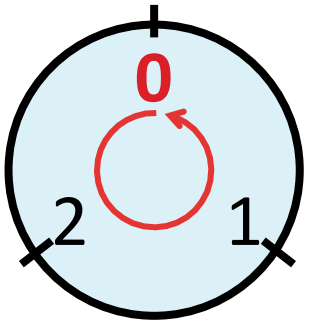


Zero casas para trás



[0 – 0] = 0

(0 – 0) % 3 = 0



Problema 4

4 – Codificar em Python – versão 2

```
# Sorteio do jogo "Pedra, Papel, Tesoura"
# Pedra   = 1
# Papel   = 2
# Tesoura = 3
import random

j1 = random.randint(0,2)
j2 = random.randint(0,2)

if (j1 == j2):
    print("Empate.")
else:
    if ((j1 - j2) % 3 == 1):
        print("Jogador 1 ganhou.")
    else:
        print("Jogador 2 ganhou.")
```

Problema 4

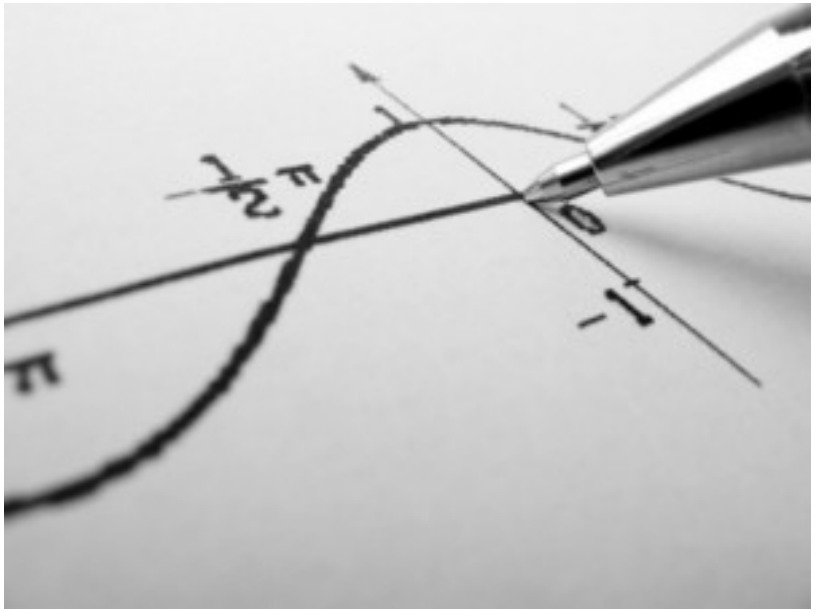
4 – Codificar em Python – versão 2

```
# Sorteio do jogo "Pedra, Papel, Tesoura"
# Pedra   = 1
# Papel   = 2
# Tesoura = 3
import random

j1 = random.randint(0,2)
j2 = random.randint(0,2)

if (j1 == j2):
    print("Empate.")
else:
    if ((j1 - j2) % 3 == 1):
        print("Jogador 1 ganhou.")
    else:
        print("Jogador 2 ganhou.")
```

Teste esta condição no Shell do Python, para todas as possibilidades de J1 e J2

Módulo `math`

- Contém diversas funções que podem ser usadas em cálculos matemáticos.
- Para utilizá-las, não se esqueça de colocar o prefixo `math.` antes do nome da função.



Módulo `math`

:: Funções matemáticas e constantes

`exp(x)`	• Calcula e^x
`log(x)`	• Logaritmo natural de x (base e)
`log10(x)`	• Logaritmo de x na base 10
`sqrt(x)`	• Raiz quadrada de x
`Pi`	• Valor da constante Pi
`e`	• Valor da constante de Euler

Módulo `math`

:: Funções trigonométricas

`sin(x)`	• Calcula o seno de x (em radianos)
`cos(x)`	• Calcula o cosseno de x (em radianos)
`tan(x)`	• Calcula a tangente de x (em radianos)
`asin(x)`	• Calcula o arco-seno de x
`acos(x)`	• Calcula o arco-cosseno de x
`atan(x)`	• Calcula o arco-tangente de x

Módulo `math`

:: Funções trigonométricas

`sin(x)`	• Calcula o seno de x (em radianos)
`cos(x)`	• Calcula o cosseno de x (em radianos)
`tan(x)`	• Calcula a tangente de x (em radianos)
`asin(x)`	• Calcula o arco-seno de x
`acos(x)`	• Calcula o arco-cosseno de x
`atan(x)`	• Calcula o arco-tangente de x

Módulo `math`

:: Funções de arredondamento

`ceil(x)`

- Arredonda x para o inteiro mais próximo em direção a mais infinito

`floor(x)`

- Arredonda x para o inteiro mais próximo em direção a menos infinito

Funções de arredondamento

:: Diferenças

`ceil()` e `floor()`

- Requer módulo **math**.
- Possui apenas **um** argumento de entrada.
- Resulta é um número **inteiro**.

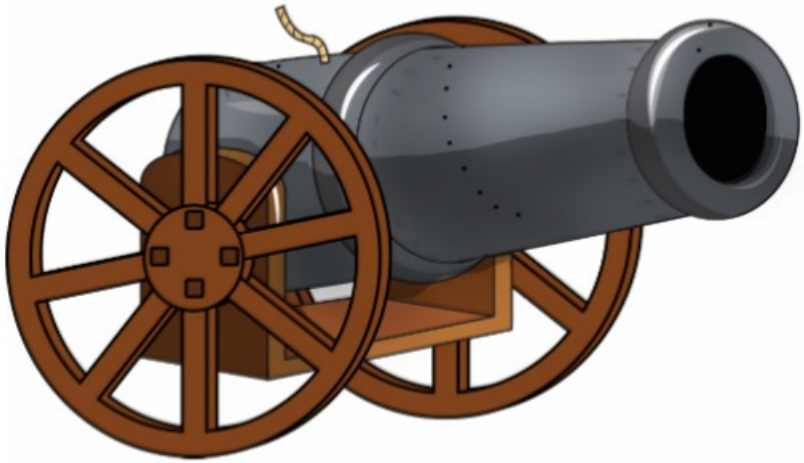
`round()`

- É padrão do Python. Não requer o módulo **math**.
- Possui **dois** argumentos de entrada.
- Resultado é um número **real**.

Problema 5

- Calcular o alcance S de um projétil, dados a velocidade inicial $v\downarrow 0$ e o ângulo θ entre o cano do canhão e o solo. Considere $g=9{,}81 m/s\uparrow 2$.

$$S=v\downarrow 0\uparrow 2 /g \operatorname{sen}(2\theta)$$



Problema 5

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Velocidade inicial	m/s	> 0
	Ângulo com solo	graus	
Saídas	Alcance	m	> 0

Problema 5

4 – Codificar em Python

```
# Entrada de dados e definicao de constantes
v0      = float(input("Velocidade inicial: "))
theta_g = float(input("Angulo: "))
g       = 9.81

# Importar modulo matematico
import math

if (v0 >= 0) and (theta_g > 0) and (theta_g < 90):
    theta_rad = theta_g * math.pi / 180.0
    s = (v0**2 / g) * math.sin(2 * theta_rad)
    print(round(s, 3))
else:
    print("Dados invalidos.")
```

Problema 6

- Uma quantia inicial q é aplicada a uma taxa t de juros. O saldo s desse investimento após m meses é dado por:

$$s = q(1+t)\uparrow m$$

- Para uma taxa t ao mês, quanto tempo (em anos e meses) é necessário para que o saldo dobre em relação ao valor inicial?

Problema 6

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Taxa de juros (t)	---	[0; 1]
Saídas	m (tempo)	anos; meses	[0, +∞[; [0, 11]

Problema 6

3 – Projetar algoritmo

- A saída do problema é a quantidade de meses e anos, mas a saída da equação é o saldo, informação que já conhecemos.
- Portanto, temos de reescrever a equação, isolando o m no lado esquerdo.

$$s = q(1+t)\uparrow m$$

$$2q/q = (1+t)\uparrow m$$

$$\log 2 = \log(1+t)\uparrow m$$

$$\log 2 = m \cdot \log(1+t)$$

$$m = \log 2 / \log(1+t)$$

Problema 6

3 – Projetar algoritmo

- A expressão anterior resultará em um número com parte fracionária.
- Contudo, o valor da saída é inteiro, pois o rendimento acontece a cada mês.
- Logo, o resultado deve ser arredondado para cima (`math.ceil`)

$$m = \log 2 \,/ \log(1+t)$$

Problema 6

4 – Codificar em Python

```
# Entrada de dados
t = float(input("Informe a taxa de aplicacao: "))

if ((t >= 0) and (t <= 1)):
    # Importar modulo matematico
    import math

    m = math.ceil(math.log(2) / math.log(1 + t))

    print(m // 12)    # no. de anos
    print(m % 12)     # no. de meses (0 a 11)
else:
    print("Dados invalidos")
```

Simplificando o uso de módulos Python

- Se ao longo do código você usa diversas vezes funções pertencentes a um módulo Python, a programação pode se tornar cansativa.
- Para usar diretamente o nome da função sem explicitar o nome do módulo como prefixo, use o seguinte comando:

```
from <nome_do_módulo> import *
```

Pode ser o **math**, o **random** ou outro que você necessitar.

Conteúdo

- Estruturas Condicionais Simples
- Estruturas Condicionais Compostas
- Como montar uma condição?
- **Estruturas Condicionais Encadeadas**

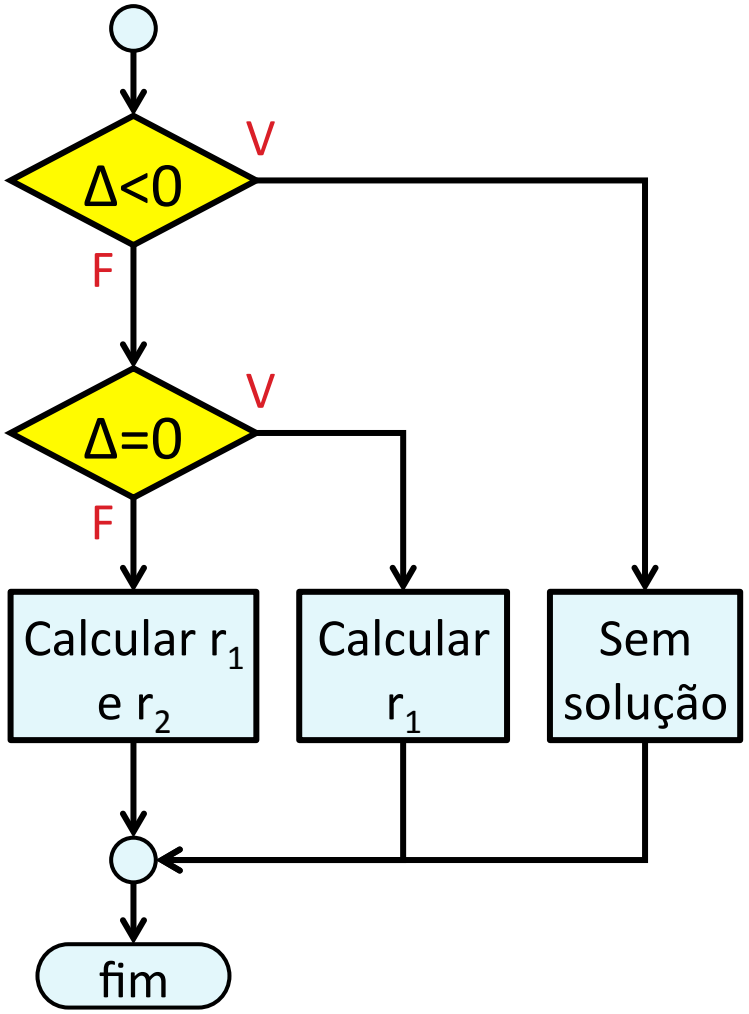
Estruturas Condicionais Encadeadas

- Estruturas condicionais encadeadas (ou aninhadas) são estruturas condicionais dentro de outras estruturas condicionais.
- Quando um problema exige um longo encadeamento de **`if`**s e **`else`**s, a criação de diversos níveis deslocados poderia causar confusão.
- A cláusula **`elif`** substitui um par **`else if`** sem criar um outro nível na estrutura condicional.

Estruturas Condicionais Encadeadas :: Exemplo

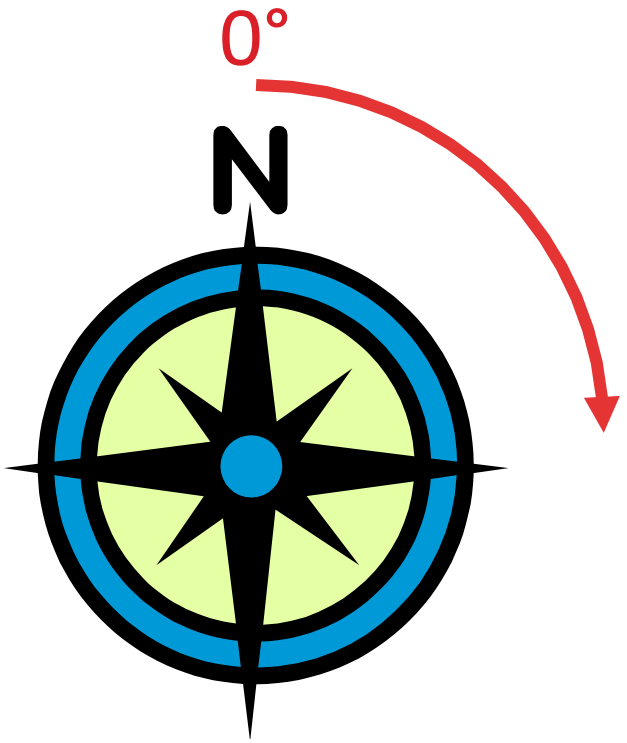
```
if (delta < 0):
    print("Nao tem raiz real")
else:
    if (delta == 0):
        r1 = -b/(2 * a)
    else:
        r1 = (-b+delta**0.5)/(2*a)
        r2 = (-b-delta**0.5)/(2*a)
```

```
if (delta < 0):
    print("Nao tem raiz real")
elif (delta == 0):
    r1 = -b/(2 * a)
else:
    r1 = (-b + delta**0.5)/(2*a)
    r2 = (-b - delta**0.5)/(2*a)
```



Problema 7

- Escrever um script em Python que leia um ângulo entre 0 e 360° e informe o ponto cardeal correspondente.



Problema 7
2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Ângulo	graus	
Saídas	Ponto Cardeal	---	{N, S, L, O}

Problema 7

4 – Codificar em Python

```
# Entrada de dados
ang = int(input("Digite o valor de um angulo: "))

if ((ang == 0) or (ang == 360)):
   print("Norte")
elif (ang == 180):
   print("Sul")
elif (ang == 90):
   print("Leste")
elif (ang == 270):
   print("Oeste")
else:
   print("Desconhecido")
```

Referências bibliográficas

- Menezes, Nilo Ney Coutinho (2010). Introdução à Programação com Python. Editora Novatec.

- HETLAND, Magnus Lie (2008). Beginning Python: From Novice to Professional. Springer eBooks, 2ª edição. Disponível em: http://dx.doi.org/10.1007/978-1-4302-0634-7.
- Gaddis, Tony (2012). Starting out with Python, 2ª edição. Editora Addison-Wesley.
- DIERBACH, Charles. Introduction to Computer Science using Python: a computational problem-solving approach. John Wiley & Sons, 2012.

Dúvidas?