IEC037
Introdução à Programação de Computadores
Aula 10 – Estruturas de Repetição por Condição em Python

Conteúdo

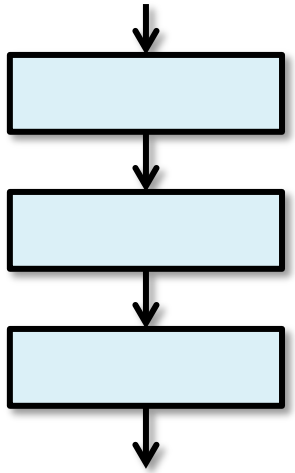
- Estruturas de Repetição por Condição
- Problemas Comuns
- Repetições Aninhadas

Conteúdo

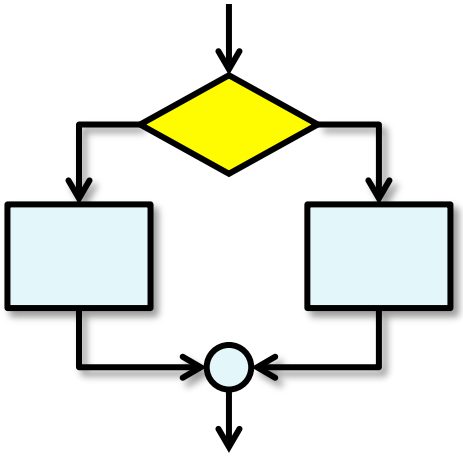
- Estruturas de Repetição por Condição
- Problemas Comuns
- Repetições Aninhadas

Estruturas de Programação

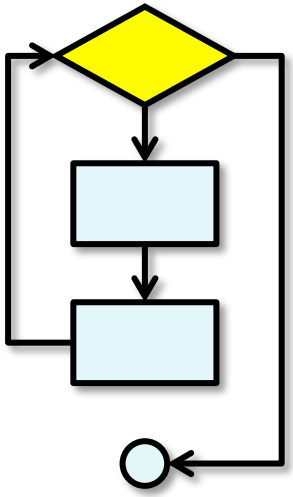
- Qualquer programa de computador pode ser escrito combinando-se os três tipos básicos de estruturas de programação:



Sequencial



Condicional

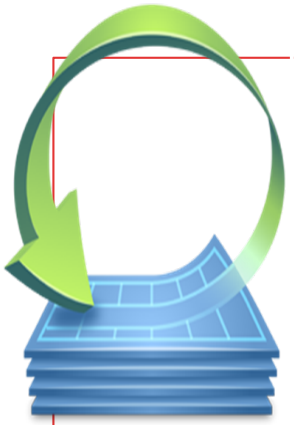


Repetição

Estruturas de Repetição

- Permitem executar mais de uma vez um comando ou um bloco de comandos.
- O trecho do algoritmo em repetição é também chamado de laço (ou *loop*).
- O número de repetições (ou iterações) deve ser conhecido e ser sempre finito.

Não confunda



Iteração

- Repetição de um ato

Interação

- Atualização da influência recíproca de organismos inter-relacionados

Repetição por condição :: Comando `while`

- Permite que um trecho de código seja executado enquanto certa condição for verdadeira.
- Quando a condição for falsa, o bloco de comandos interno deixa de ser repetido e continuamos executando o resto do programa.

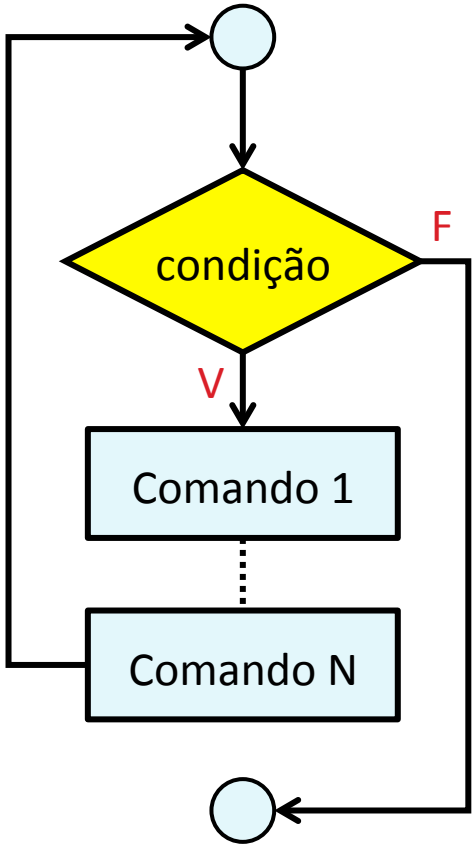
```
while <condição> :
    <comandos_a_repetir>
```

Comandos a serem repetidos devem estar **indentados**

Repetição por condição (`while`)

:: Funcionamento

1. Testar a condição.
2. Se a condição for falsa, então sair do laço.
3. Se a condição for verdadeira, então executar cada um dos comandos do interior do laço.
4. Após executar o último comando do laço `while`, voltar ao passo 1.



Repetição por condição (`while`) :: Exemplo

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

- A condição do comando `while` é construída da mesma forma que a condição do `if`.
- No comando `while`, enquanto a condição for verdadeira, repetimos as linhas do bloco interno.

Comando `while` × `if`

`if`

- Condição determina **se** o trecho de código indentado vai ser executado ou não

`while`

- Condição determina **quantas** vezes o trecho de código indentado vai ser executado

Exercício 1

- Modifique o script anterior para exibir números de 1 a 100.

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

Exercício 1

- Modifique o script anterior para exibir números de 1 a 100.

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

```
x = 1
while (x <= 100):
    print(x)
    x = x + 1
```

Exercício 2

- Modifique o script anterior para exibir números de 50 a 100.

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

Exercício 2

- Modifique o script anterior para exibir números de 50 a 100.

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

```
x = 50
while (x <= 100):
    print(x)
    x = x + 1
```

Exercício 3

- Modifique o script anterior para fazer uma contagem regressiva:

  10, 9, 8, ..., 1, 0

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

Exercício 3

- Modifique o script anterior para fazer uma contagem regressiva:

  10, 9, 8, ..., 1, 0

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

```
x = 10
while (x >= 0):
    print(x)
    x = x - 1
```

Exercício 4

- Modifique o script anterior para exibir números pares de 0 a 100.

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

Exercício 4

- Modifique o script anterior para exibir números pares de 0 a 100.

```
x = 1
while (x <= 5):
    print(x)
    x = x + 1
```

```
x = 0
while (x <= 100):
    print(x)
    x = x + 2
```

Contador de laço

- Nos exercícios anteriores, a variável **x** foi usada para controlar o número de repetições do laço.
- Todo contador de laço deve ajustar os seguintes parâmetros:

1. valor inicial
2. valor final
3. passo (ou incremento)

```
1 x = 1
  while (x <= 5):  2
      print(x)
  3   x = x + 1
```

Contador de laço

:: Parâmetros

valor inicial

- Deve ser ajustado fora do laço.

valor final

- Ajustado na condição do laço. Uma condição mal elaborada pode fazer com que o laço seja executado infinitas vezes (loop infinito).

passo (ou incremento)

- Determina a taxa de crescimento ou decrescimento do contador, desde o valor inicial até o valor final.

Problema 1

- Uma quantia inicial de R $ 20 mil é aplicada a uma taxa de 12% de juros ao ano.
- Qual o valor do saldo após 5 anos de aplicação?

Problema 1

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Taxa de juros	%	12
	Quantia inicial (q)	R$	20 mil
	Tempo	anos	5
Saídas	Saldo	R$	[0, +∞[

Problema 1
3 – Projetar algoritmo

Saldo a cada ano:

Novo saldo = Saldo anterior + Rendi-mento

Rendi-mento = Saldo anterior × Juros/100

Variável Contadora:

`ano`

Valor inicial: 1

Valor final: 5

Incremento: 1

Problema 1

4 – Codificar em Python

```
# Entrada de dados e definicao de constantes
qi    = 20000.00
juros = 12.0
saldo = qi

# Valor inicial
t = 1

# Atualizacao de saldo
while (t <= 5):
    rend = saldo * juros/100
    saldo = saldo + rend
    t = t + 1

# Exibicao de resultados
print("Saldo: R$", round(saldo, 2))
```

Problema 2

- Uma quantia inicial de R $ 10 mil é aplicada a uma taxa de 5% de juros ao ano.
- Quantos anos são necessários para que o saldo dobre em relação ao valor inicial?

Problema 2

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Taxa de juros	%	[0; 100]
	Quantia inicial (q)	R$	10 mil
	Saldo	R$	2 * q
Saídas	Tempo	anos	[0, +∞[

Problema 2
3 – Projetar algoritmo

Saldo a cada ano:

Novo saldo = Saldo anterior + Rendi-mento

Rendi-mento = Saldo anterior × Juros/100

Variável Contadora:

`ano`

Valor inicial: 0

Valor final: ?

Incremento: 1

Problema 2

4 – Codificar em Python

```
# Entrada de dados e definicao de constantes
qi    = 10000.00
juros = 5.0
saldo = qi

# Valor inicial
t = 0

# Atualizacao de saldo
while (saldo < 2 * qi):
    rend = saldo * juros/100
    saldo = saldo + rend
    t = t + 1

# Exibicao de resultados
print("Tempo de investimento:", t, "anos")
```

Contador de laço × Variável acumuladora

Contador de laço

- Auxilia a execução do laço de repetição.
- Valor somado a cada iteração é constante.

Variável acumuladora

- Auxiliam o cômputo de grandezas relacionadas ao problema.
- Valor somado a cada iteração é variável.

Rastreamento

- É uma simulação de execução de código em que você percorre as instruções, uma linha de cada vez, e acompanha os valores das variáveis.
- Quando você rastreia um programa:
  - escreve os nomes das variáveis em uma folha de papel
  - executa mentalmente cada passo do código
  - atualiza as variáveis

Rastreamento
:: Exemplo 1

- No código abaixo, quais são os valores das variáveis `i` e `total` em cada iteração do laço `while`?

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

Rastreamento

:: Exemplo 1

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

i	total
0	?

Rastreamento

:: Exemplo 1

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

i	total
0	?
0	**0**

Rastreamento

:: Exemplo 1

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

i	total
0	?
0	0
1	**0**

Rastreamento

:: Exemplo 1

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

i	total
0	?
0	0
1	0
1	**1**

Rastreamento

:: Exemplo 1

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

i	total
0	?
0	0
1	0
1	1
2	**1**

Rastreamento

:: Exemplo 1

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

i	total
0	?
0	0
1	0
1	1
2	1
2	**3**

Rastreamento

:: Exemplo 1

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

i	total
0	?
0	0
1	0
1	1
2	1
2	3
3	**6**

Rastreamento

:: Exemplo 1

```
i = 0
total = 0
while (total < 10):
    i = i + 1
    total = total + i
    print(i, total)
```

i	total
0	?
0	0
1	0
1	1
2	1
2	3
3	6
4	**10**

Rastreamento
:: Exemplo 2

- Se a condição for mal projetada, o laço poderá ser repetido indefinidamente.
- Chamamos essa situação de *loop* infinito.

```
i = 0
total = 0
while total < 10 :
    i = i + 1
    total = total - 1
    print(i, total)
```

i	total
0	0
1	-1
2	-2
3	-3
4	-4
.....	

Rastreamento
:: Exemplo 3

- Em outras situações, a condição pode não ser satisfeita na primeira tentativa.
- Nesse caso, o laço não será executado.

```
i = 0
total = 0
while total < 0 :
    i = i + 1
    total = total - i
    print(i, total)
```

(Nada mostrado na tela)

Atenção

- Embora permitidas pelo Python, **NÃO** serão aceitas as seguintes práticas:
  - Comando `else` como alternativa ao `while`.
  - Comando `break`.
  - Comando `continue`.

Conteúdo

- Estruturas de Repetição por Condição
- **Problemas Comuns**
- Repetições Aninhadas

Problemas Comuns

- Soma e média
- Contar padrões
- Verificar padrão
- Soma de séries
- Frações contínuas
- Máximo Divisor Comum (MDC)

Problema 3
:: Soma e média

- Uma professora dá aula para uma turma de N alunos e precisa saber a média de notas, que variam de 0 a 10.
- Escreva um script que leia nota por nota e, ao final, exiba a média da turma na tela.

Problema 3

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Número de alunos	---	> 0
	Notas	---	[0, 10]
Saídas	Média	---	[0, 10]

Problema 3

3 – Projetar algoritmo

- Qual a condição para repetição?
  - Enquanto o contador não atingir N (= número de alunos da classe)
- O que deve ser atualizado a cada repetição?
  - Soma das notas inseridas até o momento
  - Contador do laço

Problema 3

4 – Codificar em Python

```
# Pede o numero de alunos
num_alunos = int(input("Digite o no. de alunos:  "))

i = 1          # Variavel contadora
soma = 0       # Variavel acumuladora

while (i <= num_alunos):
    print("Digite a nota do aluno", i, ":")
    nota = float(input())    # Le nota
    soma = soma + nota       # Acumula nota
    i = i + 1                # Atualiza contador

print("Media:", round(soma/num_alunos, 2))
```

Problema 4
:: Soma e média

- Agora, a professora não sabe exatamente a quantidade de notas.
- Escreva um script que leia nota por nota, via teclado, até que seja digitado -1, indicando que a inserção terminou. Ao final, a média da turma deve ser exibida na tela.

Problema 4

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Notas	---	[0, 10]
Saídas	Média	---	[0, 10]

Problema 4

3 – Projetar algoritmo

- Qual a condição para repetição?
  - Enquanto a nota for diferente de -1
- O que deve ser atualizado a cada repetição?
  - Soma das notas inseridas até o momento
  - Quantidade de notas inseridas

Problema 4

4 – Codificar em Python

```
i = 1          # Variavel contadora
soma = 0       # Variavel acumuladora
nota = float(input())    # Le nota

while (nota != -1):
    i = i + 1                    # Atualiza no. de notas
    soma = soma + nota           # Acumula nota
    print("Digite a nota do aluno", i, ":")
    nota = float(input())        # Le nota

print("Media:", round(soma/(i - 1), 2))
```

Como o laço conta o -1 como nota, deve-se descontar uma unidade do contador antes do cálculo da média.

Problema 5

:: Contar padrões

- Um o dado de seis faces é lançado N vezes.
- Contar número de ocorrências de cada face.

Problema 5

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	N	---	[0, +∞[
Saídas	Nº de ocorrências (f1, f2, f3, f4, f5, f6)	---	[0, N]

Problema 5

3 – Projetar algoritmo

1. Ler N (número de lançamentos do dado)
2. Zerar os contadores das faces ci
3. Lançar o dado
4. Se der a face i, atualizar contador ci
5. Repetir passos 2 e 3 até N lançamentos do dado, atualizando o contador
6. Imprimir resultados

Problema 5

4 – Codificar em Python

```
# Valores iniciais
N  = int(input("No. de lancamentos:"))
i  = 0
f1 = 0
f2 = 0
f3 = 0
f4 = 0
f5 = 0
f6 = 0

# Importando biblioteca de nos. aleatorios
from random import *
```

Problema 5

4 – Codificar em Python

```
# Laco de contagem das faces do dado
while (i < N):
    face = randint(1,6)
    if (face == 1):
        f1 = f1 + 1
    elif (face == 2):
        f2 = f2 + 1
    ...
    else:
        f6 = f6 + 1
    i = i + 1

# Impressao de resultados
print(f1, f2, f3, f4, f5, f6)
```

Problema 6
:: Verificar padrão

- Dado um número N, inteiro e positivo, verificar se N é primo, ou seja, se N tem apenas dois divisores: ele mesmo e a unidade.
- Dica: Para verificar se um número N é primo, pode-se limitar o teste de divisibilidade até o valor de $\sqrt{N}$.

Problema 6

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	N	---	[0, +∞[
Saídas	Mensagem	---	{Primo, não-primo}

Problema 6

3 – Projetar algoritmo

1. Ler N (número a ser verificado)
2. Testar se N é divisível por 2, 3, 4, 5, ..., $\sqrt{N}$
3. Caso positivo, terminar laço e imprimir mensagem

- Condições de repetição:
  - Divisor (contador) não atingir $\sqrt{N}$, **E**
  - Número N não ser divisível pelo contador.

Problema 6

4 – Codificar em Python

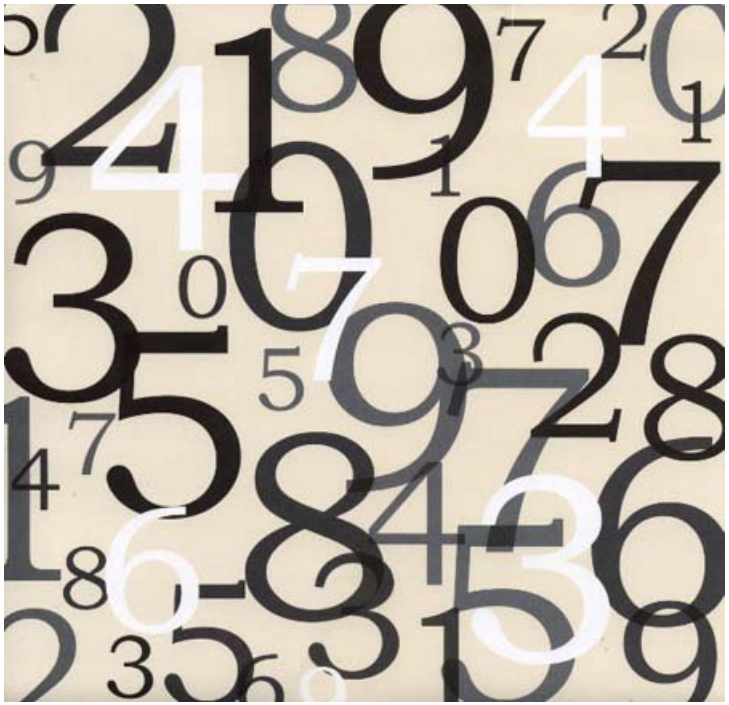
```
# Valores iniciais
N  = int(input("Digite um numero:"))
i  = 2

from math import *

# Repete enquanto nao alcancar raiz de N ou
# nao encontrar divisor de N
while (i <= floor(sqrt(N)) and N % i != 0):
   i = i + 1

# Testou todos os numeros ateh raiz de N?
if (i > floor(sqrt(N))):
   print("Eh primo")
# Se nao testou todos, entao encontrou divisor (resto 0)
else:
   print("Nao eh primo")
```

Problema 7

:: Soma de séries

- Escreva um script Python para encontrar o valor da soma dos 10 primeiros termos da seguinte sequência:

$1+2+4+8+16+\ldots$



Problema 7
3 – Projetar algoritmo

- Determinar, por indução, o termo geral da série:

$1+2+4+8+16+\ldots$

$2^0 \quad 2^1 \quad 2^2 \quad 2^3 \quad 2^4$

$\sum_{i=0}^{9} 2^i$

- Variável contadora (`i`)
  - Valor inicial: 0
  - Valor final: 9
  - Incremento: 1

Problema 7

4 – Codificar em Python

```
# Valores iniciais
soma = 0     # Variavel acumuladora
i    = 0     # Variavel contadora
fim  = 9     # Final da contagem

# Laco de acumulacao
while (i <= fim):
    soma = soma + 2**i
    i = i + 1

# Impressao de resultados
print("Soma dos termos:", soma)
```

Problema 8
:: Soma de séries

- Escreva um script Python para encontrar o valor de S.

$$S = \frac{2}{3} + \frac{2^2}{6} + \frac{2^3}{9} + \cdots + \frac{2^{25}}{75}$$

Problema 8
3 – Projetar algoritmo

- Determinar, por indução, o termo geral da série:

$$S = \underset{3\cdot 1}{\frac{2^1}{3}} + \underset{3\cdot 2}{\frac{2^2}{6}} + \underset{3\cdot 3}{\frac{2^3}{9}} + \cdots + \underset{3\cdot 25}{\frac{2^{25}}{75}} \quad \Rightarrow \quad S = \sum_{i=1}^{25} \frac{2^i}{3 \cdot i}$$

- Variável contadora (**`i`**)
  - Valor inicial: 1
  - Valor final: 25
  - Incremento: 1

Problema 8

4 – Codificar em Python

```
# Valores iniciais
soma = 0     # Variavel acumuladora
i    = 1     # Variavel contadora
fim  = 25    # Final da contagem

# Laco de acumulacao
while (i <= fim):
    soma = soma + 2**i/(3 * i)
    i = i + 1

# Impressao de resultados
print("Soma dos termos:", soma)
```

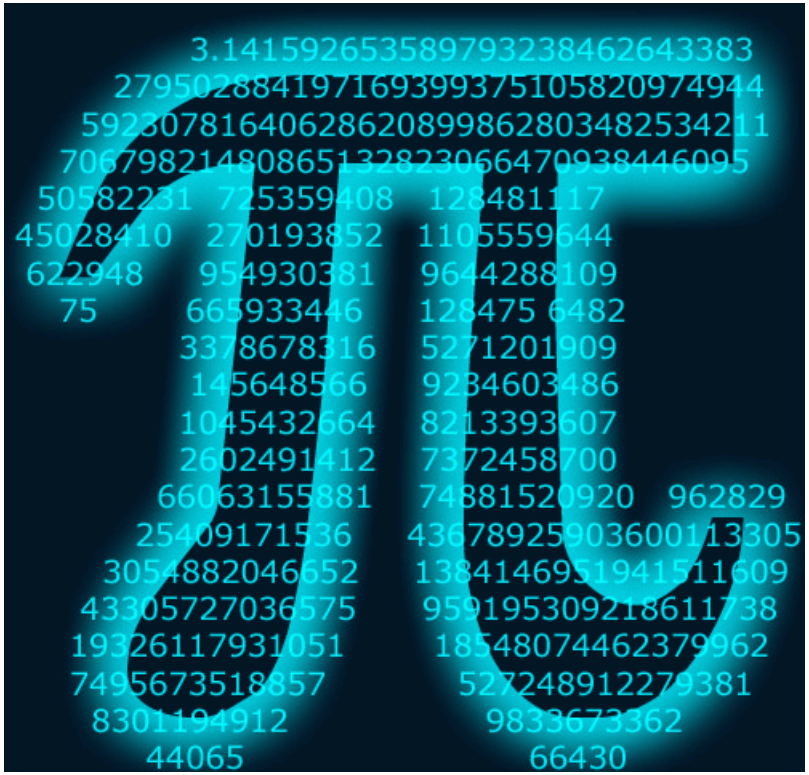
Problema 9

:: Soma de séries

- Calcular o valor de π com os 1000 primeiros termos da seguinte série numérica:

$$S = 1 - \frac{1}{3^3} + \frac{1}{5^3} - \frac{1}{7^3} + \frac{1}{9^3} - \frac{1}{11^3} \cdots$$

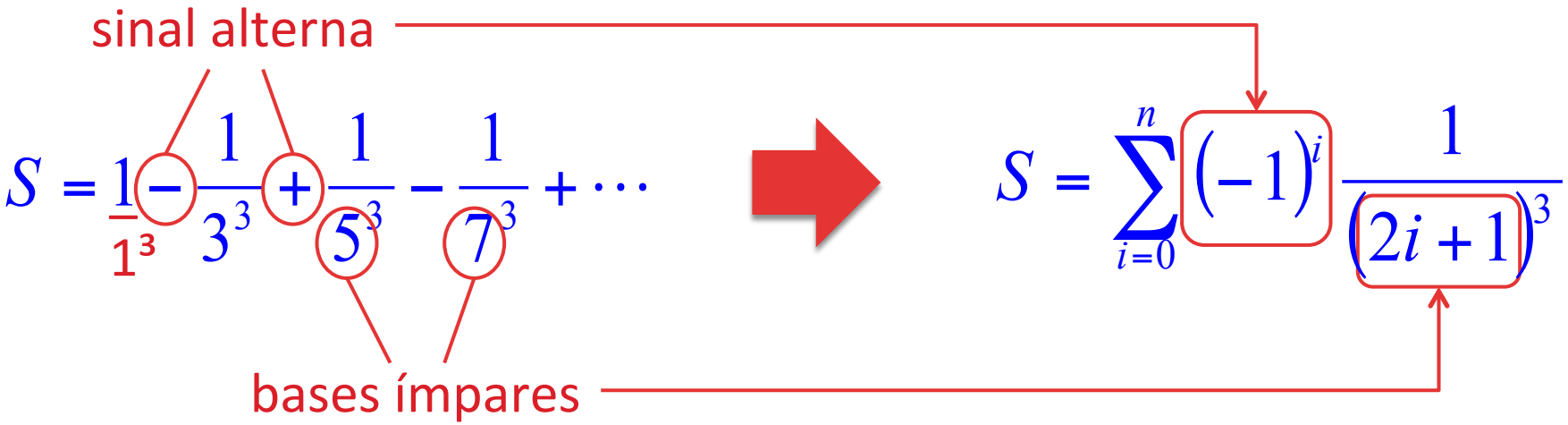
$$\pi = \sqrt[3]{32.S}$$



Problema 9

3 – Projetar algoritmo – versão 1

- Determinar o termo geral da série



- Variável contadora (`i`)
  - Valor inicial: 0
  - Valor final: 999
  - Incremento: 1

Problema 9

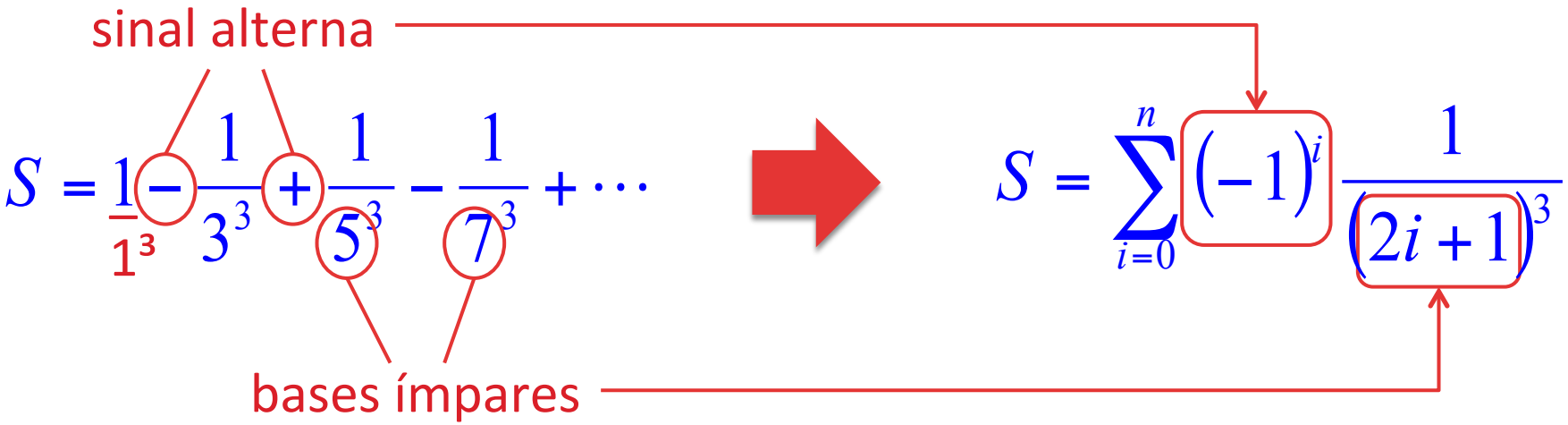
4 – Codificar em Python – versão 1

```
# Valores iniciais
soma = 0     # Variavel acumuladora
i    = 0     # Variavel contadora
fim  = 999   # Final da contagem

# Laco de acumulacao
while (i <= fim):
    soma = soma + (-1)**i / (2 * i + 1)**3
    i = i + 1

meu_pi = (32 * soma)**(1/3)

# Impressao de resultados
print("Valor aprox. de pi:", meu_pi)
```

Problema 9

3 – Projetar algoritmo – versão 2

- Determinar o termo geral da série



- Variável contadora (`j = 2*i`)
  - Valor inicial: 0
  - Valor final: 1998
  - Incremento: 2

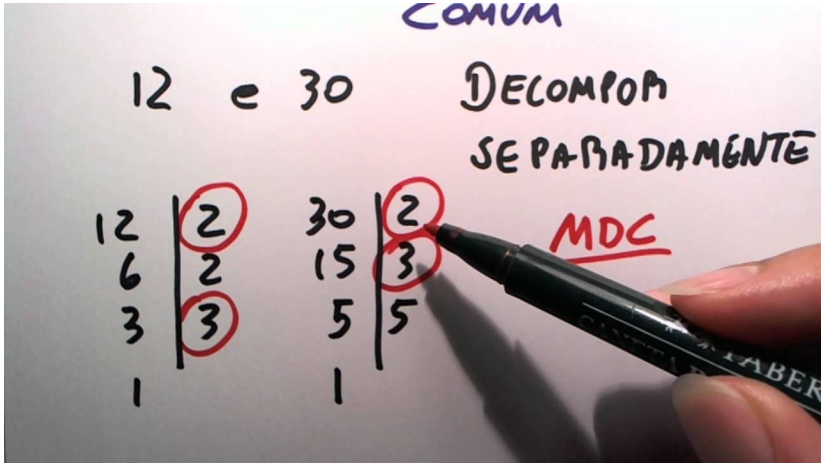
Problema 9

4 – Codificar em Python – versão 2

```
# Valores iniciais
soma = 0      # Variavel acumuladora
j    = 0      # Variavel contadora
fim  = 1998   # Final da contagem

# Laco de acumulacao
while (j <= fim):
    soma = soma + (-1)**j / (j + 1)**3
    j = j + 2

meu_pi = (32 * soma)**(1/3)

# Impressao de resultados
print("Valor aprox. de pi:", meu_pi)
```

Problema 12
:: Máximo Divisor Comum (MDC)

- Dados dois números inteiros positivos, diferentes de 0, calcular o Máximo Divisor Comum (MDC) deles, pelo método de Euclides.



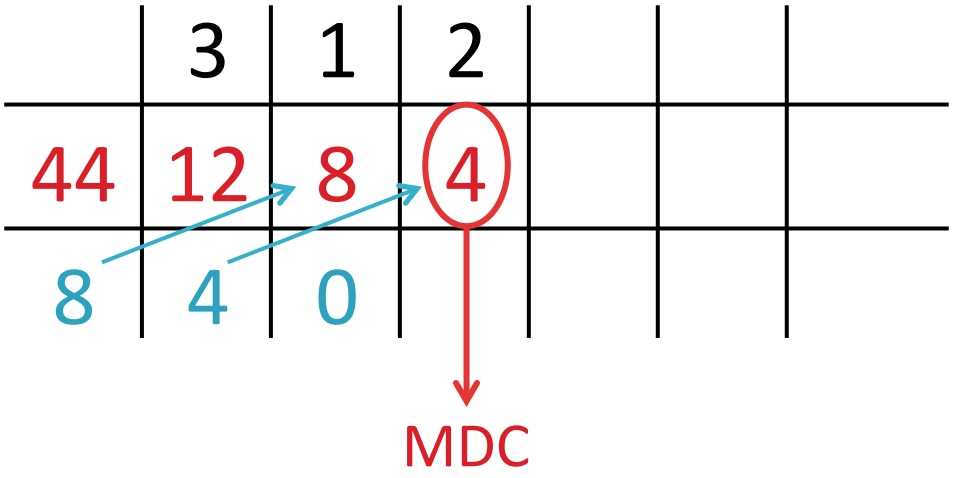
Método de Euclides

1. Calcular o resto da divisão do maior número pelo menor.
2. Se o resto for zero, o MDC é o 2º número.
3. Senão:
    1. substituir o 1º número pelo 2º
    2. substituir o 2º número pelo resto
    3. calcular um novo resto (passo 1)

Método de Euclides

:: Exemplo

- Exemplo: 44 e 12



Problema 12

3 – Projetar algoritmo

1. Ler n1 e n2
2. Calcular resto (n1 % n2)
3. Enquanto resto não for zero, aplicar método de Euclides:
    1. n1 ← n2
    2. n2 ← resto (anterior)
    3. resto ← n1 % n2
4. Imprimir n2 (valor do mdc)

Problema 12

2 – Definir entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	n1	---	[1, +∞[
	n2	---	[1, +∞[, n2 <= n1
Saídas	MDC	---	[1, +∞[

Problema 12

4 – Codificar em Python

```
# Entrada de variaveis
n1 = int(input())
n2 = int(input())

# Valor inicial do resto
resto = n1 % n2

# Metodo de Euclides
while (resto != 0):
    n1 = n2
    n2 = resto
    resto = n1 % n2

# Impressao do MDC
print(n2)
```

Conteúdo

- Estruturas de Repetição por Condição
- Problemas Comuns
- **Repetições Aninhadas**

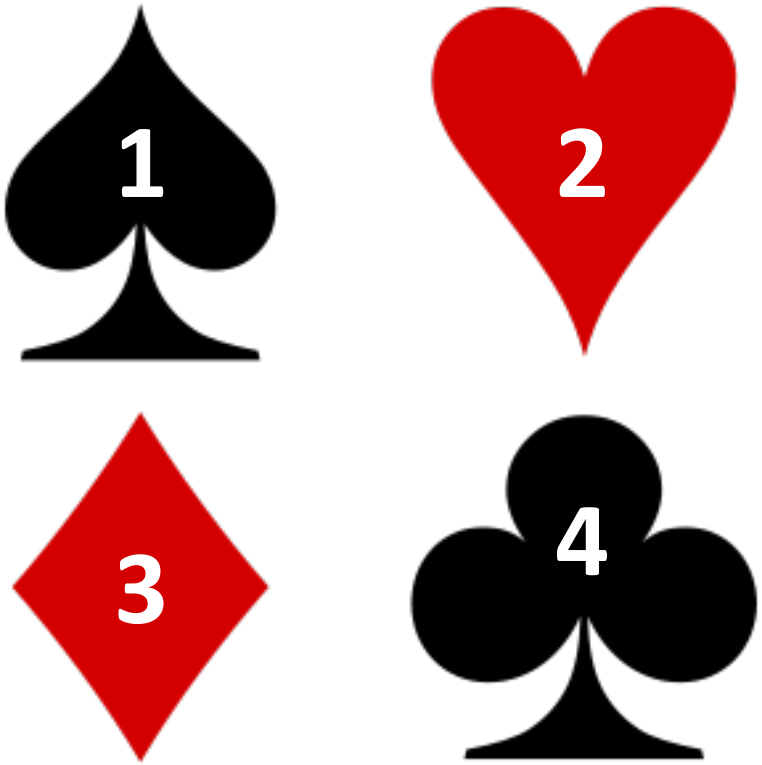
Repetições Aninhadas

- Um comando **`while`** pode ser utilizado dentro de outro comando **`while`**.
- Cuidados devem ser tomados para não aumentar o tempo de execução do algoritmo.
- Variáveis de controle de cada laço **`while`** devem ser diferentes.

Problema 13

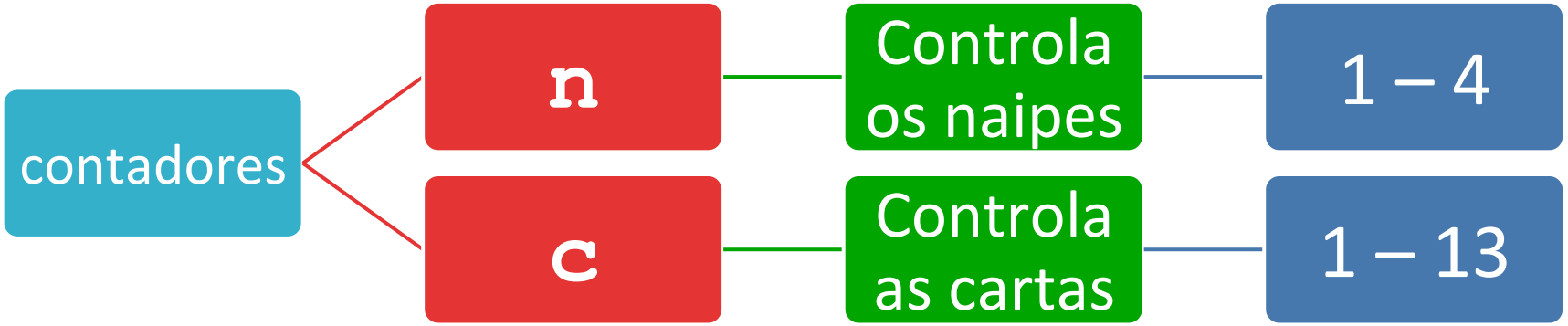
- Crie uma listagem dos valores das 52 cartas do baralho comum.

1 – 1	2 – 1
1 – 2	2 – 2
…	…
1 – 11	4 – 11
1 – 12	4 – 12
1 – 13	4 – 13



Problema 13

3 – Projetar algoritmo



Problema 13

4 – Codificar em Python

```
# Valor inicial do contador do 1o. laco (naipes)
n = 1

# Controla contagem dos naipes
while (n <= 4):
    # Valor inicial do contador do 2o. laco
    c = 1
    # Controla contagem das cartas
    while (c <= 13):
        print(n,"-",c)
        c = c + 1

    n = n + 1
```

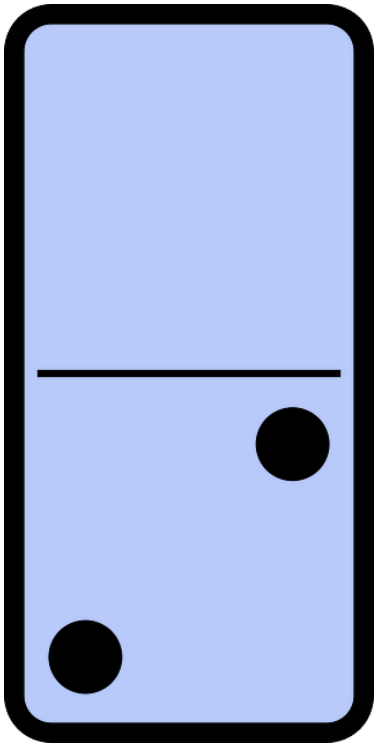
Problema 14

- Crie uma listagem dos valores das 28 peças do jogo de dominó:

0 – 0	0 – 6
0 – 1	1 – 1
0 – 2	1 – 2
0 – 3	…
0 – 4	5 – 6
0 – 5	6 – 6

Problema 14

3 – Projetar algoritmo

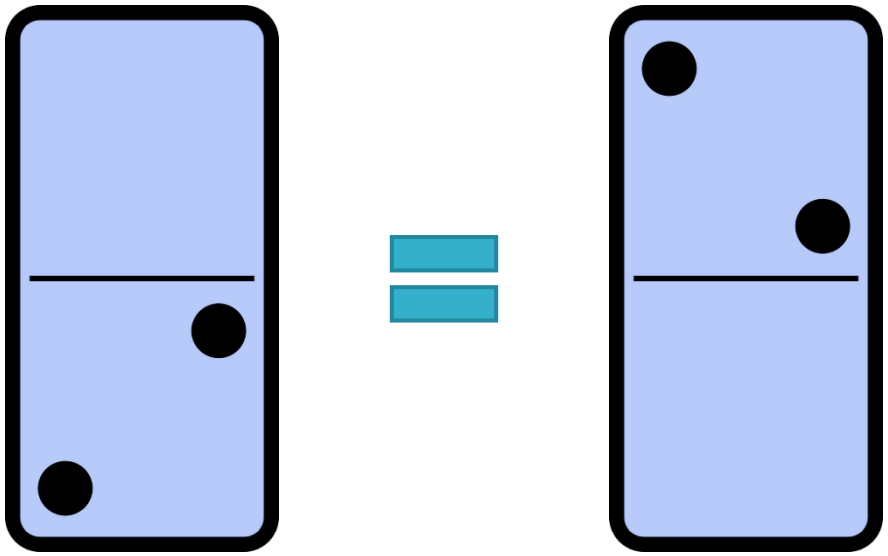


Um laço controla a 1ª metade da peça (`m1`)

Um laço controla a 2ª metade da peça (`m2`)

Problema 14
3 – Projetar algoritmo

- Deve-se ter atenção com a condição de parada do segundo laço, pois a ordem dos números não influencia no tipo de peça:



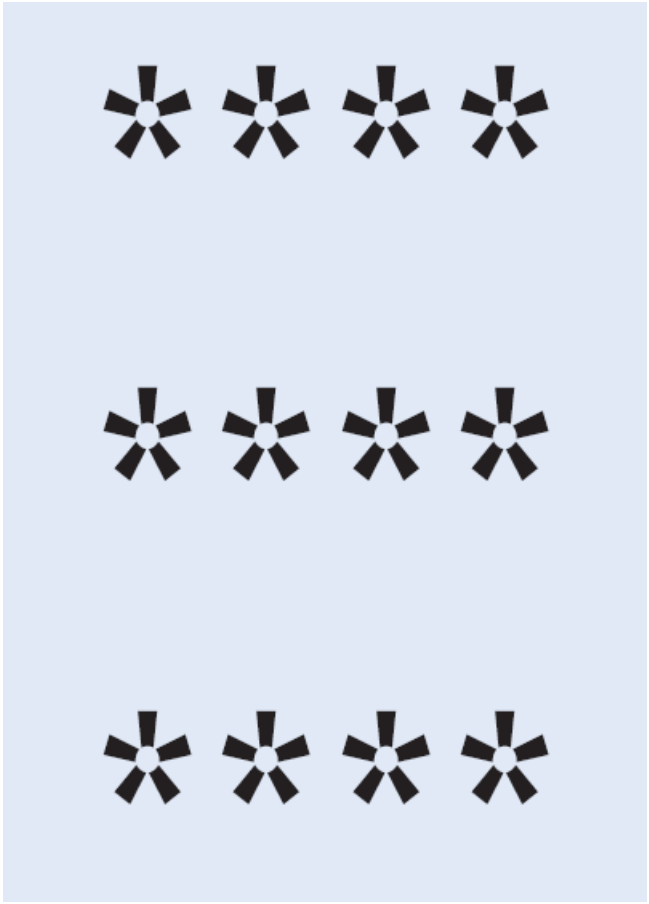
Problema 14

4 – Codificar em Python

```
# Valor inicial do contador do 1o. laco
m1 = 0

# Controla 1a. metade do dominoh
while (m1 <= 6):
    # Valor inicial do contador do 2o. laco
    m2 = 0
    # Controla 2a. metade do dominoh
    while (m2 <= m1):
        print(m1,"-",m2)
        m2 = m2 + 1
    m1 = m1 + 1
```

Contador do 2º laço não é mais constante. Passa a ser o contador do 1º laço.

Problema 15
:: Arte ASCII



- Escreva um script para imprimir vários caracteres asterisco para formar o padrão ao lado.
  - Três linhas de 04 asteriscos.

Problema 15

3 – Projetar algoritmo

1. Iniciar contador de linhas
2. Repetir impressão de asteriscos dispostos em uma mesma linha
3. Repetir passo 2 para cada linha

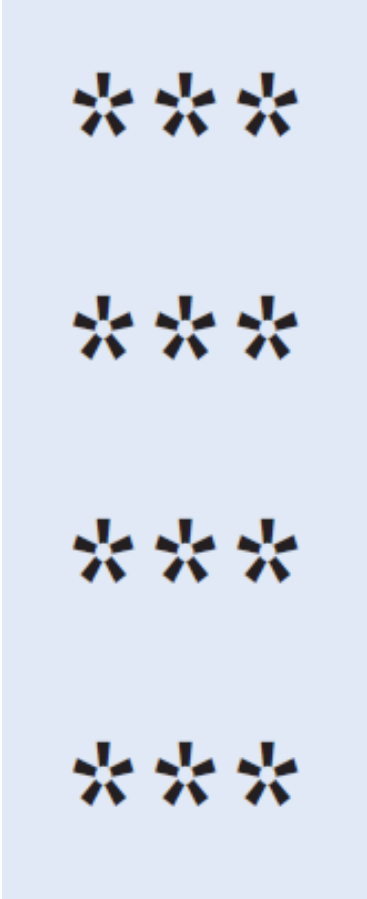
Problema 15

4 – Codificar em Python

```
# Contador do 1o. laco
i = 1

while (i <= 3):
    # Contador do 2o. laco
    j = 1
    while (j <= 4):
        print("*", end="")
        j = j + 1
    print()
    i = i + 1
```

Por padrão, o comando **print()** do Python pula de linha ao terminar. Se você não deseja isso, use o parâmetro **end** para indicar o que vocês deseja que apareça ao fim da impressão.

Problema 16
:: Arte ASCII



- Escreva um script para imprimir vários caracteres asterisco para formar o padrão ao lado:
  - Quatro linhas de 3 asteriscos.

Problema 16

3 – Projetar algoritmo

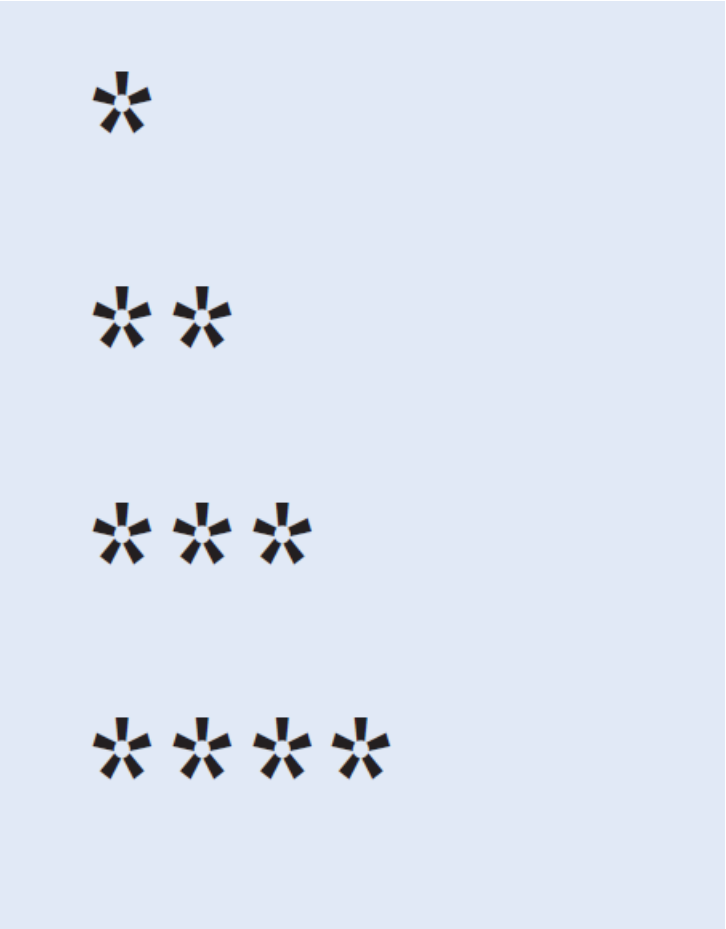
1. Iniciar contador de linhas
2. Repetir impressão de asteriscos dispostos em uma mesma linha
3. Repetir passo 2 para cada linha

Problema 16

4 – Codificar em Python

```
# Contador do 1o. laco
i = 1

while (i <= 4):
    # Contador do 2o. laco
    j = 1
    while (j <= 3):
        print("*", end="")
        j = j + 1
    print()
    i = i + 1
```

Problema 17
:: Arte ASCII



- Escreva um script para imprimir vários caracteres asterisco para formar o padrão ao lado.

Problema 17

3 – Projetar algoritmo

1. Iniciar contador de linhas i
2. Para cada linha, imprimir i asteriscos
3. Repetir passo 1 até atingir o número de iterações

Problema 17

4 – Codificar em Python

```
# Contador do 1o. laco
i = 1
fim = 4

while (i <= fim):
    j = 1
    while (j <= i):
        print("*", end="")
        j = j + 1
    print()
    i = i + 1
```

Contador do 2º laço não é mais constante. Passa a ser o contador do 1º laço.

Arte ASCII

- Como imprimir os seguintes padrões?

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.
- Horstmann, Cay & Necaise, Rance D. (2013). Python for Everyone. John Wiley & Sons.

Dúvidas?