

**Instituto de  
Computação**

UNIVERSIDADE ESTADUAL DE CAMPINAS



**MC102 - Aula 05**

**Comandos de Repetição**

Algoritmos e Programação de Computadores

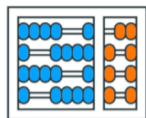
Turmas  
**OVXZ**

**Prof. Lise R. R. Navarrete**

[lrommel@ic.unicamp.br](mailto:lrommel@ic.unicamp.br)

Quinta-feira, 31 de março de 2022

19:00h - 21:00h (CB06)



**Instituto de  
Computação**

UNIVERSIDADE ESTADUAL DE CAMPINAS



UNICAMP

**MC102** – Algoritmos e Programação de Computadores

---

Turmas

**OVXZ**

<https://ic.unicamp.br/~mc102/>

Site da Coordenação de MC102

Aulas teóricas:

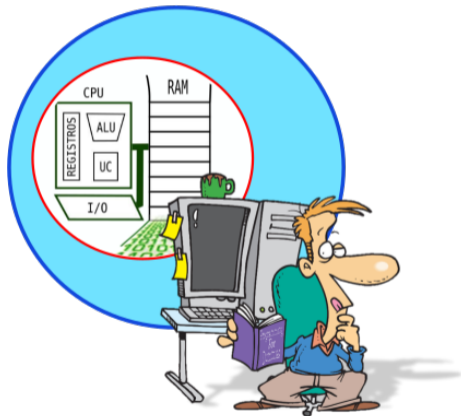
Terça-feira, 21:00h - 23:00h (CB06)

Quinta-feira, 19:00h - 21:00h (CB06)

# Conteúdo

- Comandos de Repetição
- Comando while
- Listas
- Comando for
- Comandos break e continue
- Laços Aninhados
- Exercícios

# Comandos de Repetição



INICIO

AÇÃO 1

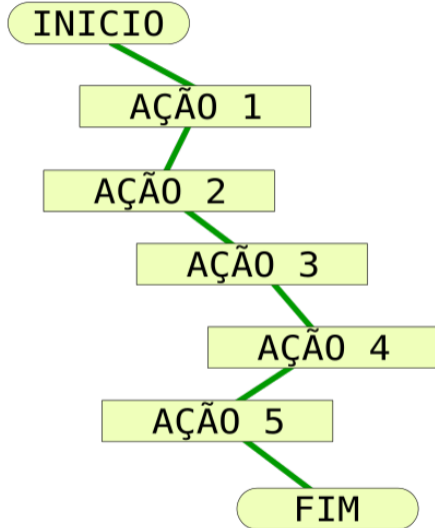
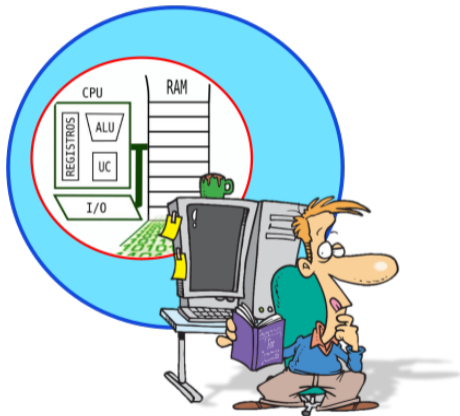
AÇÃO 2

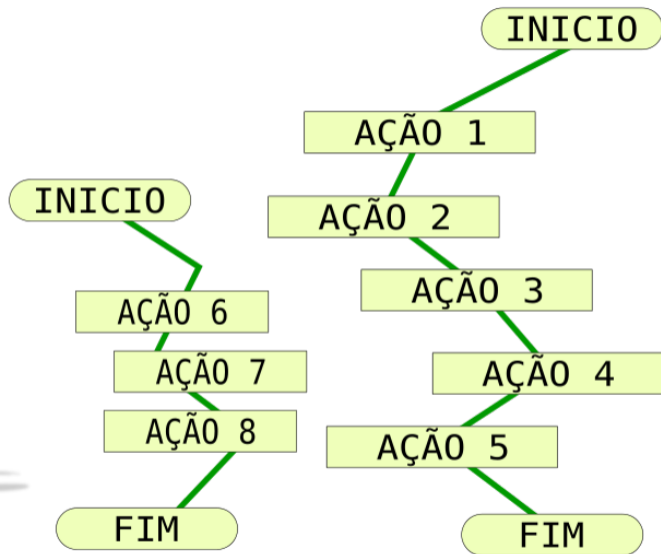
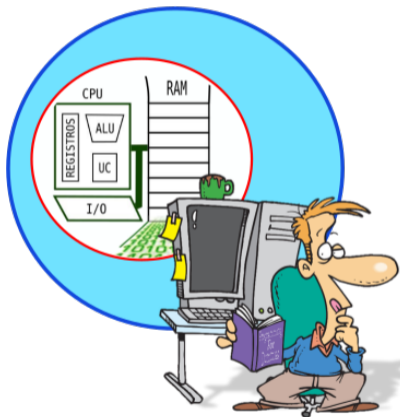
AÇÃO 3

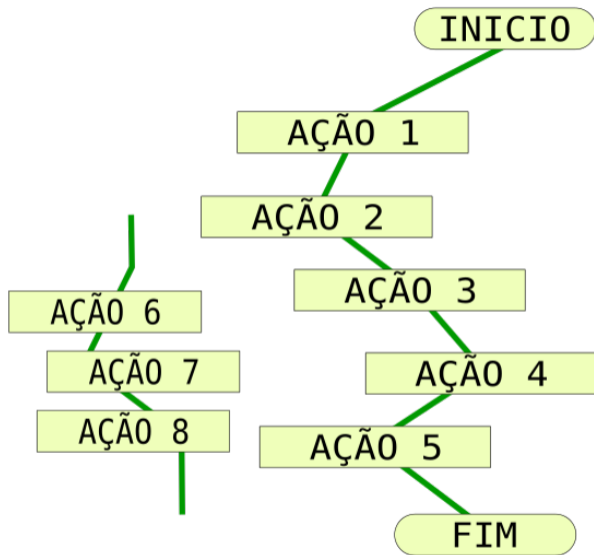
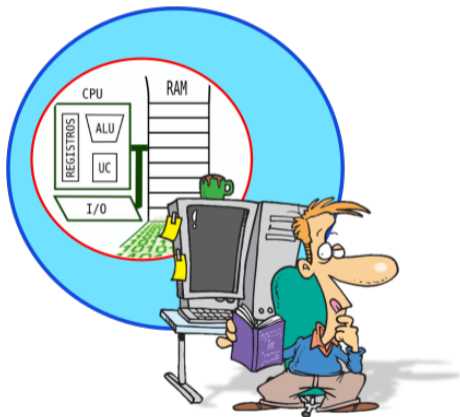
AÇÃO 4

AÇÃO 5

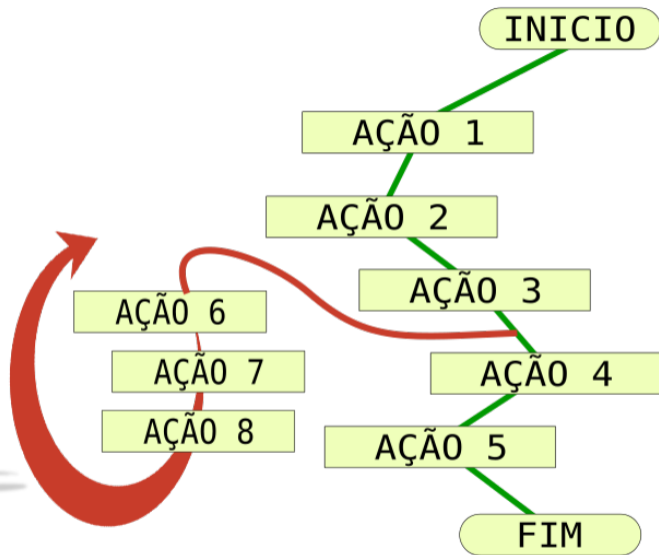
FIM

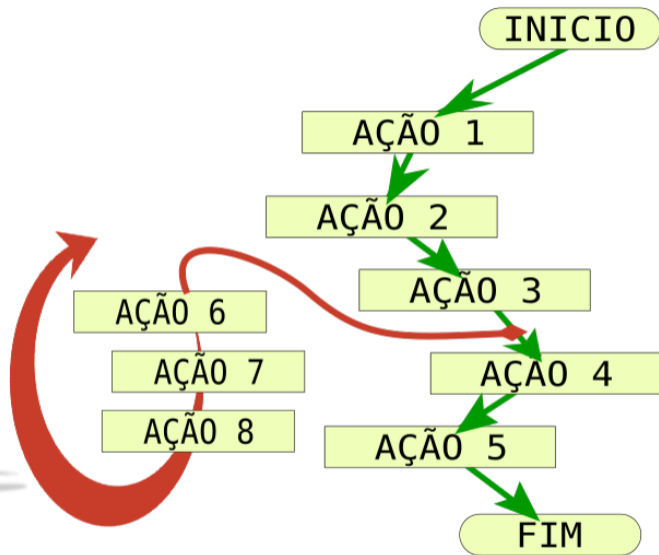
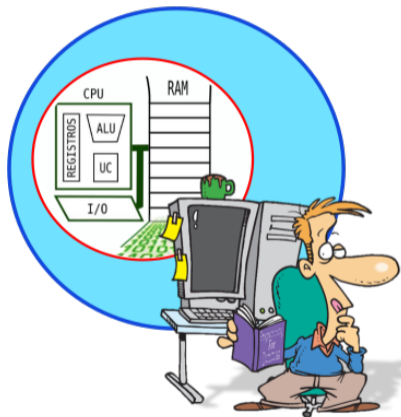


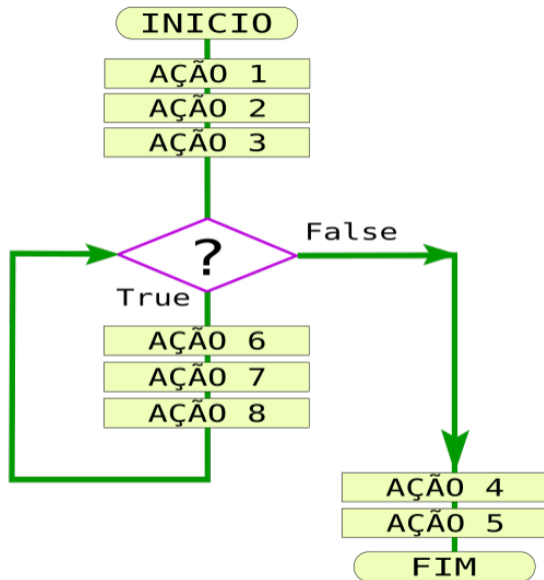
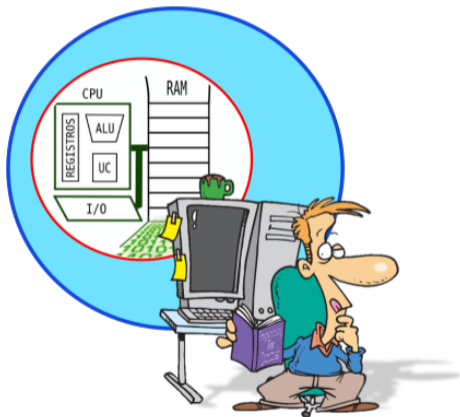












- Imprimindo todos os números inteiros de 1 até 5.

```
1 print(1)
2 print(2)
3 print(3)
4 print(4)
5 print(5)
```

- Imprimindo todos os números inteiros de 1 até 100.

```
1 print(1)
2 print(2)
3 print(3)
4 ...
5 print(100)
```

Queremos imprimir os números de 1 até um  $n$  dado.

<https://ic.unicamp.br/~mc102/aulas/aula04.pdf>

- Imprimindo todos os números inteiros de 1 até n.

```
1 n = int(input("Digite um número: "))
```

- Imprimindo todos os números inteiros de 1 até  $n$ .

```
1 n = int(input("Digite um número: "))
2 if n >= 1:
3     print(1)
4 if n >= 2:
5     print(2)
6 if n >= 3:
7     print(3)
8 if n >= 4:
9     print(4)
10 ...
11 if n >= 100:
12     print(100)
```

- Note que só resolvemos o problema para  $n \leq 100$ .



# Comando while

- O primeiro comando de repetição que aprenderemos é o `while`.

```
1 while <condição>:  
2 # este bloco irá se repetir até a condição ser falsa  
3   <comando1>  
4   <comando2>  
5   ...  
6   <comandoY>
```

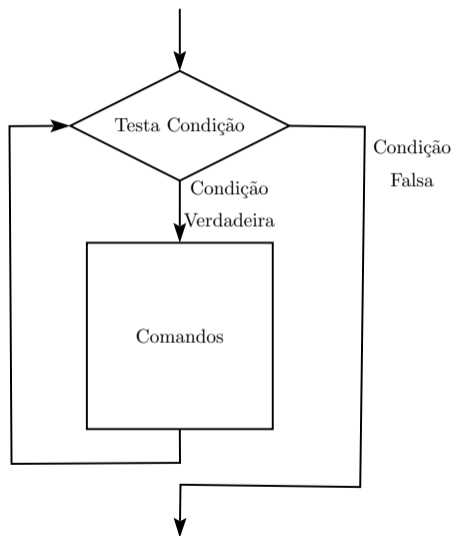
- Funcionamento:

Passo 1: Se a condição for verdadeira, vá para o Passo 2.

Caso contrário, encerre o bloco de repetição (`while`) e prossiga para o próximo comando (conforme o diagrama do próximo slide).

Passo 2: Execute o bloco de comandos.

Passo 3: Volte para o Passo 1.



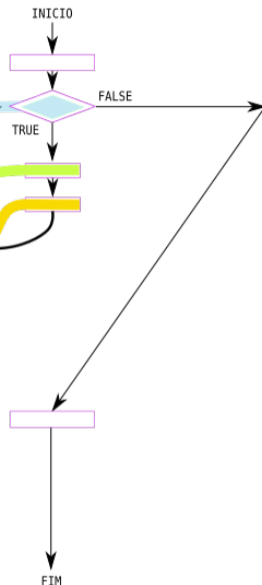
<https://ic.unicamp.br/~mc102/aulas/aula04.pdf>

- Imprimindo todos os números inteiros de 1 até 100.

```
1 i = 1
2 while i <= 100:
3     print(i)
4     i = i + 1
5 print("Fim do programa!")
```

Imprimindo todos os números inteiros de 1 até 100.

```
i = 1  
while i <= 100:  
    print(i)  
    i = i + 1  
print("Fim do programa!")
```



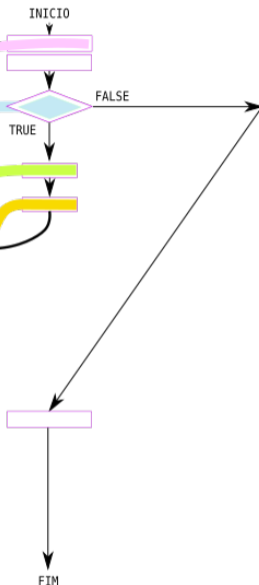
- Imprimindo todos os números inteiros de 1 até n.

```
1 n = int(input("Digite um número inteiro positivo: "))
2 i = 1
3 while i <= n:
4     print(i)
5     i = i + 1
6 print("Fim do programa!")
```

Imprimindo todos os números inteiros de 1 até n.

```

n = int(input("Digite um número inteiro positivo: "))
i = 1
while i <= n:
    print(i)
    i = i + 1
print("Fim do programa!")
  
```



- Contagem regressiva:

```
1 n = int(input("Digite um número inteiro positivo: "))
2
3 while n >= 0:
4     print(n)
5     n = n - 1
6
7 print("BOOM!")
```



Contagem regressiva:

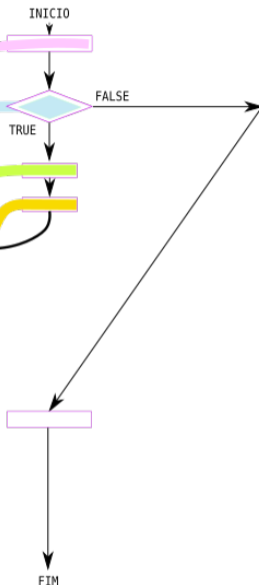
```
n = int(input("Digite um número inteiro positivo: "))
```

```
while n >= 0:
```

```
    print(n)
```

```
    n = n - 1
```

```
print("BOOM!")
```



<https://ic.unicamp.br/~mc102/aulas/aula04.pdf>

- Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

```
1 dividendo = int(input("Entre com o dividendo: "))  
2 divisor = int(input("Entre com o divisor: "))  
3
```

Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

dividendo =

divisor =

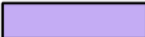
Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

dividendo =

divisor =

Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

dividendo = 

divisor = 

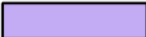


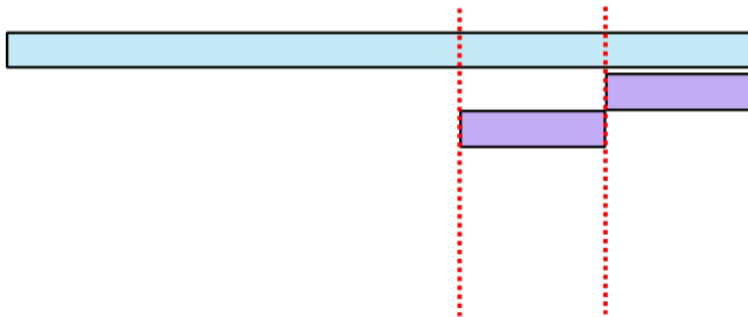




Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

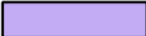
dividendo = 

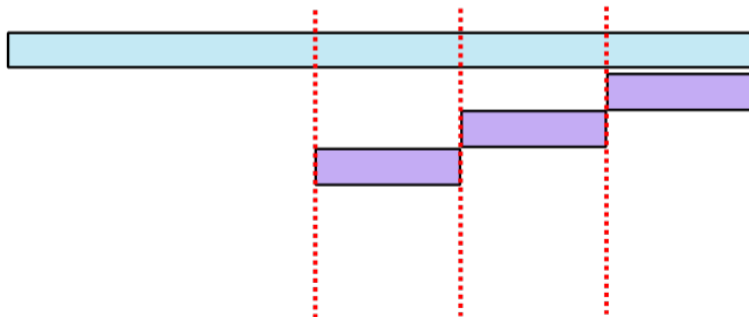
divisor = 



Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

dividendo = 

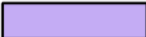
divisor = 

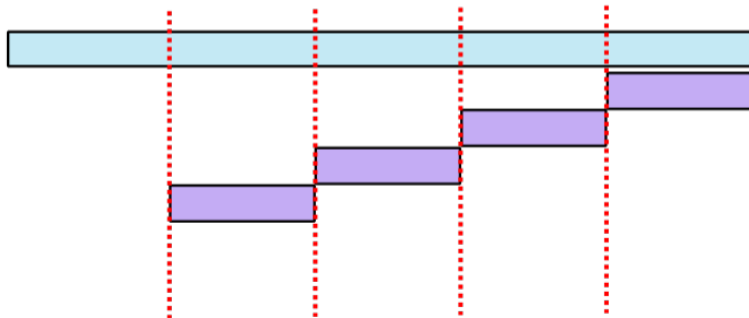




Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

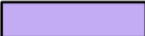
dividendo = 

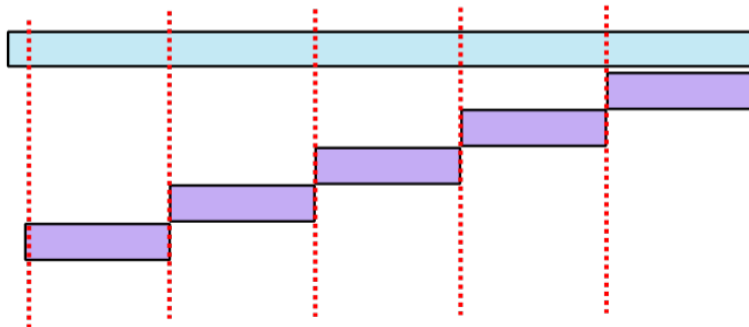
divisor = 



Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

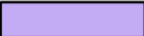
dividendo = 

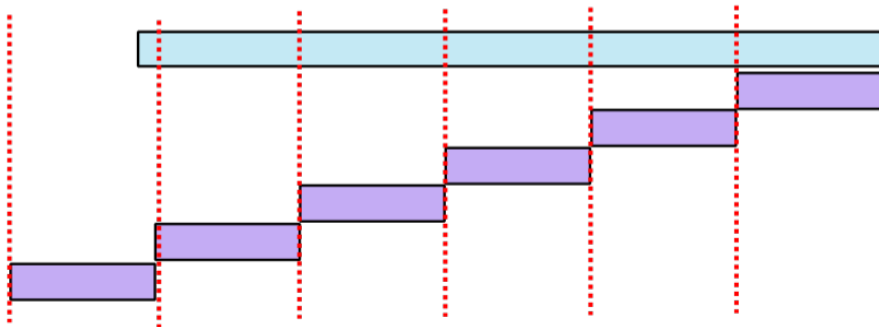
divisor = 



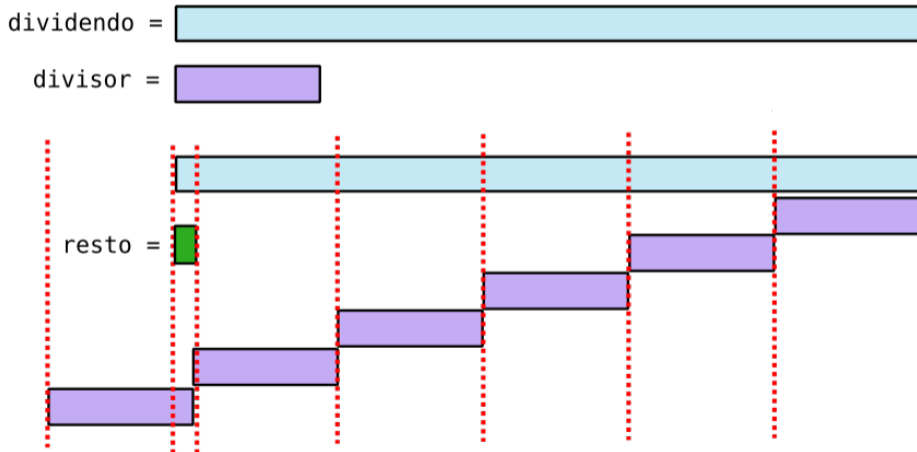
Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

dividendo = 

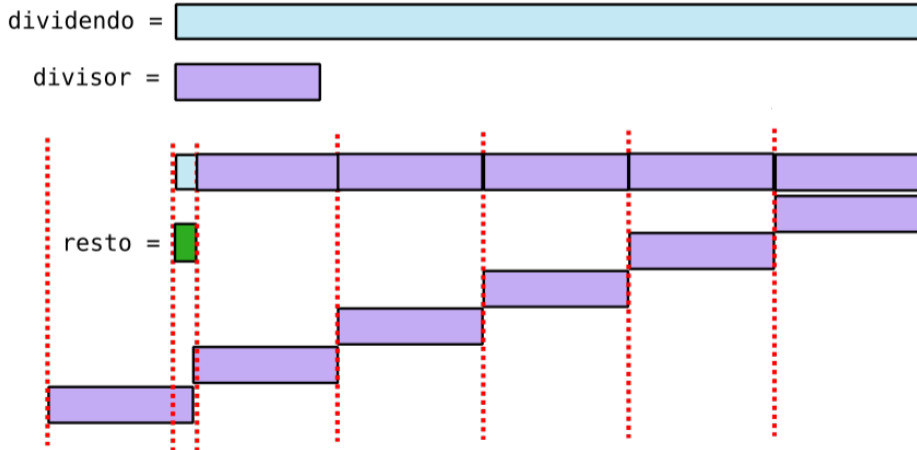
divisor = 



Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.



Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.



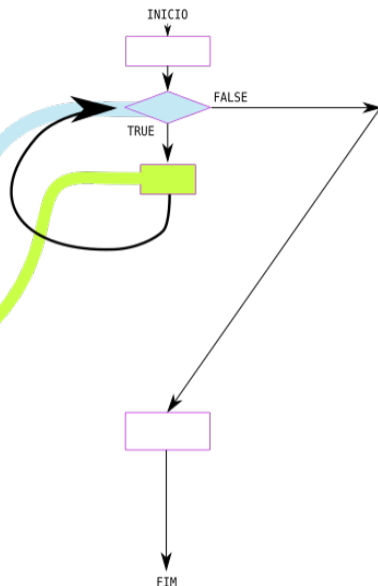
Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

```
dividendo = int(input("Entre com o dividendo: "))
divisor = int(input("Entre com o divisor: "))
```

```
quociente = 0
```

```
while dividendo >= divisor:
    dividendo = dividendo - divisor
    quociente = quociente + 1
```

```
print("Quociente:", quociente)
print("Resto:", dividendo)
```



<https://ic.unicamp.br/~mc102/aulas/aula04.pdf>

- Como calcular  $b^e$  (para  $b \in \mathbb{R}$ ,  $e \in \mathbb{N}$ )?
- Para  $e \geq 1$ , usando  $e - 1$  multiplicações:
  - $b^e = b \times b \times \dots \times b$
- Para  $e \geq 0$ , usando  $e$  multiplicações:
  - $b^e = 1 \times b \times b \times \dots \times b$

- Dado um número real (base) e um número inteiro positivo (expoente), calcule  $base^{expoente}$ , usando apenas somas e multiplicações.

```
1 base = float(input("Base: "))
2 exp = int(input("Expoente: "))
3
4 result = base
5 i = 1
6
7 while i < exp:
8     result = result * base
9     i = i + 1
10
11 print(result)
```



- Dado um número real (base) e um número inteiro não negativo (expoente), calcule  $base^{expoente}$ , usando apenas somas e multiplicações.

```
1 base = float(input("Base: "))
2 exp = int(input("Expoente: "))
3
4 result = 1
5 i = 1
6
7 while i <= exp:
8     result = result * base
9     i = i + 1
10
11 print(result)
```

## O que acontece se a condição for falsa na primeira vez?

```
1 a = 0
2 while a > 0:
3     a = a + 1
4     print(a)
```

- Resposta: o programa nunca entra no bloco de repetição.

## O que acontece se a condição for sempre verdadeira?

```
1 a = 0
2 while a >= 0:
3     a = a + 1
4     print(a)
```

- Resposta: o programa entra no bloco e nunca sai (*loop infinito*).

- Como calcular  $b^e$  (para  $b \in \mathbb{R}$ ,  $e \in \mathbb{Z}$ )?
- Para  $e \geq 0$ , usando  $e$  multiplicações:
  - $b^e = 1 \times b \times b \times \dots \times b$
- Para  $e < 0$ , usando  $|e|$  divisões:
  - $b^e = (((1/b)/b)/\dots)/b$

- Dado um número real (base) e um número inteiro (expoente), calcule  $base^{\text{expoente}}$ , usando apenas somas e multiplicações.

```
1 base = float(input("Base: "))
2 exp = int(input("Expoente: "))
3
4 result = 1
5
6 while exp > 0:
7     result = result * base
8     exp = exp - 1
9
10 while exp < 0:
11     result = result / base
12     exp = exp + 1
13
14 print(result)
```

- Dado um número real (base) e um número inteiro (expoente), calcule  $base^{\text{expoente}}$ , usando apenas somas e multiplicações.

```
1 base = float(input("Base: "))
2 exp = int(input("Expoente: "))
3
4 result = 1
5
6 while exp < 0:
7     result = result / base
8     exp = exp + 1
9
10 while exp > 0:
11     result = result * base
12     exp = exp - 1
13
14 print(result)
```

- Determinando a quantidade de números inteiros positivos fornecidos para um programa (até a leitura de um inteiro não positivo).

```
1 n = 0
2 OK = True
3
4 while OK:
5     x = int(input("Entre com um número inteiro positivo: "))
6     if x > 0:
7         n = n + 1
8     else:
9         OK = False
10
11 print("Quantidade de números positivos fornecidos:", n)
```

- Dado um número inteiro positivo  $n$ , calcular o valor de  $n!$ .

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3 i = 1
4
5 while i <= n:
6     fat = fat * i
7     i = i + 1
8
9 print("Resultado:", fat)
```

- Dado um número inteiro positivo  $n$ , calcular o valor de  $n!$ .

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3 i = 2
4
5 while i <= n:
6     fat = fat * i
7     i = i + 1
8
9 print("Resultado:", fat)
```



- Dado um número inteiro positivo  $n$ , calcular o valor de  $n!$ .

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3
4
5 while n > 0:
6     fat = fat * n
7     n = n - 1
8
9 print("Resultado:", fat)
```

- Dado um número inteiro positivo  $n$ , calcular o valor de  $n!$ .

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3
4
5 while n > 1:
6     fat = fat * n
7     n = n - 1
8
9 print("Resultado:", fat)
```

# Listas

- Uma lista é uma estrutura de Python que armazena múltiplos dados.

```
1 lista = [<dado_1>, <dado_2>, <dado_3>, ..., <dado_n>]
```

- Podemos ter todos os dados do mesmo tipo.

```
1 # Uma lista com dados do tipo int
2 lista_de_int = [40, 3, 61, 7, 3]
3
4 # Uma lista com dados do tipo bool
5 lista_de_bool = [True, False, True]
```

- Podemos ter dados de tipos diferentes misturados.

```
1 lista_mista = ["Gato", 42, True, 5.4, "Cachorro", 73]
```

<https://ic.unicamp.br/~mc102/aulas/aula04.pdf>

- Podemos acessar um elemento em uma lista indicando a sua posição (o primeiro elemento fica na posição 0 da lista).

```
1 lista = ["Azul", 51, "Amarelo", 55, True, 7.2]
2 print(lista[0]) # imprime o primeiro elemento da lista
3 # Azul
4 print(lista[1]) # imprime o segundo elemento da lista
5 # 51
6 print(lista[2]) # imprime o terceiro elemento da lista
7 # Amarelo
8 print(lista[-1]) # imprime o último elemento da lista
9 # 7.2
```

- A função `len()` retorna o número de elementos de uma lista.

```
1 print(len(lista))
2 # 6
```

- Podemos usar um `while` para percorrer todos os elementos de uma lista.

```
1 letras = ["A", "B", "C", "D", "E", "F", "G"]
2 i = 0
3 while i < len(letras):
4     print(letras[i])
5     i = i + 1
```

- Encontrando o máximo de um conjunto de números positivos.

```
1 numeros = [3, 1, 7, 9, 4]
2 maximo = 0
3
4 i = 0
5 while i < len(numeros):
6     if numeros[i] > maximo:
7         maximo = numeros[i]
8     i = i + 1
9
10 print(maximo) # 9
```

- Encontrando o máximo de um conjunto de números quaisquer.

```
1 numeros = [-3, -1, -7, -9, -4]
2 maximo = numeros[0]
3
4 i = 1
5 while i < len(numeros):
6     if numeros[i] > maximo:
7         maximo = numeros[i]
8         i = i + 1
9
10 print(maximo) # -1
```



# Comando for

- Podemos percorrer uma lista de forma mais compacta com o comando **for**.

```
1 for <variavel> in <lista>:  
2 # este bloco irá repetir para todos os valores da lista  
3     <comando1>  
4     <comando2>  
5     ...  
6     <comandoY>
```

- Estes dois códigos são equivalentes.

```
1 letras = ["A", "B", "C", "D", "E", "F", "G"]
2 i = 0
3 while i < len(letras):
4     print(letras[i])
5     i = i + 1
```

```
1 letras = ["A", "B", "C", "D", "E", "F", "G"]
2 for letra in letras:
3     print(letra)
```

- Encontrando o máximo de um conjunto de números positivos.

```
1 numeros = [3, 1, 7, 9, 4]
2 maximo = 0
3
4 for numero in numeros:
5     if numero > maximo:
6         maximo = numero
7
8 print(maximo) # 9
```

- Encontrando o máximo de um conjunto de números.

```
1 numeros = [3, 1, 7, 9, 4]
2 maximo = numeros[0]
3
4 for numero in numeros:
5     if numero > maximo:
6         maximo = numero
7
8 print(maximo) # 9
```

- A função `range` gera uma sequência de inteiros.
- Uma sequência de inteiros pode ser transformada numa lista usando a função `list`.

```
1 print(list(range(2, 6)))  
2 # [2, 3, 4, 5]
```

- A função `range` recebe como argumentos os limites da sequência a ser gerada: o primeiro número é incluído na sequência, mas o último não.

```
1 print(list(range(1, 5)))  
2 # [1, 2, 3, 4]  
3 print(list(range(0, 4)))  
4 # [0, 1, 2, 3]  
5 print(list(range(0, -5)))  
6 # []
```

- Se a sequência começa em zero, o primeiro número pode ser omitido.

```
1 print(list(range(4)))  
2 # [0, 1, 2, 3]
```

- A função `range` pode receber um terceiro argumento (opcional), que define o incremento usado na geração da sequência de inteiros.

```
1 print(list(range(2, 10, 2)))  
2 # [2, 4, 6, 8]  
3 print(list(range(1, 15, 3)))  
4 # [1, 4, 7, 10, 13]  
5 print(list(range(5, 0, -1)))  
6 # [5, 4, 3, 2, 1]
```

- Note que podemos utilizar `range(n)` em conjunto com o `for` para repetir uma operação  $n$  vezes.

```
1 for i in list(range(100)):
2     print("Esta frase será impressa 100 vezes")
```

- No comando `for`, não precisamos explicitamente converter a sequência de inteiros gerada pelo comando `range()` numa lista.

```
1 n = int(input("Digite um número inteiro positivo: "))
2 for i in range(n):
3     print("Esta frase será impressa", n, "vezes")
```



- Imprimindo todos os números inteiros de 1 até 100.

```
1 for i in range(1, 101):  
2     print(i)  
3 print("Fim do programa!")
```

- Imprimindo todos os números inteiros de 1 até n.

```
1 n = int(input("Digite um número inteiro positivo: "))  
2 for i in range(1, n+1):  
3     print(i)  
4 print("Fim do programa!")
```

- Imprimindo as n primeiras potências de 2, sem usar o operador de exponenciação (\*\*).

```
1 n = int(input("Digite um número inteiro positivo: "))
2 potencia = 1
3 for i in range(n):
4     potencia = potencia * 2
5     print(potencia)
```

- Como no exemplo anterior, em vários problemas precisamos combinar dados em uma variável utilizando alguma operação.
- Esse tipo de variável é chamada de acumuladora.
- Exemplo: somando os números inteiros de 1 até n.

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 soma = 0
3 for i in range(1, n+1):
4     soma = soma + i
5 print(soma)
```

- Calculando n!

```
1 n = int(input("Digite um número inteiro positivo: "))
2 fatorial = 1
3 for i in range(1, n+1):
4     fatorial = fatorial * i
5 print(fatorial)
```

- Calculando n!

```
1 n = int(input("Digite um número inteiro positivo: "))
2 fatorial = 1
3 for i in range(2, n+1):
4     fatorial = fatorial * i
5 print(fatorial)
```

- Calculando n!

```
1 n = int(input("Digite um número inteiro positivo: "))
2 fatorial = 1
3 for i in range(n, 1, -1):
4     fatorial = fatorial * i
5 print(fatorial)
```

# Comandos break e continue

<https://ic.unicamp.br/~mc102/aulas/aula04.pdf>

- Vimos que o comando **for** percorre a lista completa.
- Às vezes queremos interromper a execução do comando **for** antes dele percorrer a lista completa.
- O comando **break** faz com que a execução de um laço de repetição seja finalizada, passando a execução para o próximo comando após o laço.



- Exemplo: procurando um valor em uma lista sem elementos repetidos.

```
1 x = int(input("Digite um número: "))
2 for i in [2, 4, 7, 1, 0, 8, 9, 5]:
3     print("Verificando elemento", i)
4     if i == x:
5         print("Elemento", i, "encontrado!")
```

- Após encontrarmos o valor, podemos encerrar a busca.

```
1 x = int(input("Digite um número: "))
2 for i in [2, 4, 7, 1, 0, 8, 9, 5]:
3     print("Verificando elemento", i)
4     if i == x:
5         print("Elemento", i, "encontrado!")
6         break
```

- O que será impresso no seguinte código?

```
1 for i in range(1, 10):  
2     if i == 5:  
3         break  
4     print(i)  
5 print("Fim do programa")
```

- Resposta:

1

2

3

4

Fim do programa

- Determinando a quantidade de números inteiros positivos fornecidos para um programa (até a leitura de um inteiro não positivo).

```
1 n = 0
2
3 while True:
4     x = int(input("Entre com um número inteiro positivo: "))
5     if x <= 0:
6         break
7     n = n + 1
8
9 print("Quantidade de números positivos fornecidos: ", n)
```

- Podemos usar o comando **else** para executar um bloco de comandos apenas caso o comando **for** ou comando **while** tenham sido executados sem interrupção (**break**).
- Encontrando um elemento x em uma lista.

```
1 x = int(input("Digite um número: "))
2 for i in [2, 4, 7, 1, 0, 8, 9, 5]:
3     if i == x:
4         print("Elemento", i, "encontrado!")
5         break
6 else:
7     print("O elemento", x, "não está na lista.")
```

- Quando o **break** é executado, o bloco **else** é ignorado.

- O comando `continue` faz com que a execução atual do bloco de comandos do laço de repetição seja finalizada, passando a execução para a próxima iteração do laço.
- O que será impresso no seguinte código?

```
1 for i in range(1, 6):  
2     if i == 3:  
3         continue  
4     print(i)  
5 print("Fim do programa")
```

- Resposta:

1

2

4

5

Fim do programa

- Estes dois códigos são equivalentes: imprimem apenas os elementos pares da lista.

```
1 for i in [2, 4, 7, 1, 0, 8, 9, 5]:  
2     if i % 2 == 0:  
3         print(i)
```

```
1 for i in [2, 4, 7, 1, 0, 8, 9, 5]:  
2     if i % 2 == 1:  
3         continue  
4     print(i)
```

# Laços Aninhados

<https://ic.unicamp.br/~mc102/aulas/aula04.pdf>

- Em muitas situações é necessário implementar um laço (bloco de repetição) dentro de outro laço.
- Estes blocos de comandos são conhecidos como laços aninhados.



- Imprimindo as tabuadas dos números de 1 a 10.

```
1 for i in range(1, 11):  
2     print("Tabuada do ", i, ":", sep = "")  
3     for j in range(1, 11):  
4         print(i, "x", j, "=", i * j)
```

- Imprimindo um retângulo com  $n$  linhas e  $m$  colunas ( $n \times m$  caracteres #).

```
1 n = int(input("Entre com o número de linhas: "))
2 m = int(input("Entre com o número de colunas: "))
3
4 for i in range(n):
5     for j in range(m):
6         print("#", end = "")
7     print()
```

- Imprimindo um retângulo com  $n$  linhas e  $m$  colunas ( $n \times m$  caracteres #).

```
1 n = int(input("Entre com o número de linhas: "))
2 m = int(input("Entre com o número de colunas: "))
3
4 for i in range(n):
5     print("#" * m)
6
7
```

- Imprimindo um triângulo retângulo isósceles, com catetos de tamanho dado.

```
1 n = int(input("Entre com o tamanho dos catetos: "))
2 c = input("Entre com caractere a ser usado: ")
3
4 for i in range(1, n+1):
5     for j in range(i):
6         print(c, end = " ")
7     print()
```

- Imprimindo um triângulo retângulo isósceles, com catetos de tamanho dado.

```
1 n = int(input("Entre com o tamanho dos catetos: "))
2 c = input("Entre com caractere a ser usado: ")
3
4 for i in range(1, n+1):
5     print(c * i)
6
7
```

- Imprimindo todos os horários de um dia, no formato HH:MM (horas e minutos).

```
1 for h in range(24):  
2     for m in range(60):  
3         print(h, m, sep = ":")  
4
```

- Imprimindo todos os horários de um dia, no formato HH:MM (horas e minutos).

```
1 for h in range(24):  
2     for m in range(60):  
3         print('{:02d}:{:02d}'.format(h, m))  
4
```

- Imprimindo todos os horários de um dia, no formato HH:MM:SS (horas, minutos e segundos).

```
1 for h in range(24):  
2     for m in range(60):  
3         for s in range(60):  
4             print('{:02d}:{:02d}:{:02d}'.format(h, m, s))
```



# Exercícios

- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima os seus divisores.
- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima o número de seus divisores.
- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.
- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima sua fatoração em números primos.
- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima seus fatores primos (cada fator deve ser impresso uma única vez).

<https://ic.unicamp.br/~mc102/aulas/aula04.pdf>

- Escreva um programa que leia um número inteiro ( $n \geq 1$ ) e, para todo inteiro  $j$  de 1 até  $n$ , imprima os valores:

$$\sum_{i=1}^j i.$$

- Escreva um programa que leia um número inteiro ( $n \geq 1$ ) e imprima o valor:

$$\sum_{j=1}^n \sum_{i=1}^j i.$$

- O quadrado de um número inteiro  $n \geq 1$  é dado pela soma dos  $n$  primeiros números ímpares. Escreva um programa que leia um número inteiro ( $n \geq 1$ ) e, para todo inteiro  $j$  de 1 até  $n$ , imprima:

$$\sum_{i=1}^j (2 \times i - 1) = j^2.$$

## Imprimindo Divisores

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima os seus divisores.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2
3 for divisor in range(1, n+1):
4     if n % divisor == 0:
5         print(divisor)
```

## Contando Divisores

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima o número de seus divisores.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisores = 0
3 for divisor in range(1, n+1):
4     if n % divisor == 0:
5         divisores = divisores + 1
6 print(divisores)
```

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisores = 0
3 for divisor in range(1, n+1):
4     if n % divisor == 0:
5         divisores = divisores + 1
6
7 if divisores == 2:
8     print("Primo")
9 else:
10    print("Composto")
```

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 primo = True
3 for divisor in range(2, n):
4     if n % divisor == 0:
5         primo = False
6
7 if primo:
8     print("Primo")
9 else:
10    print("Composto")
```

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 primo = True
3 for divisor in range(2, n):
4     if n % divisor == 0:
5         primo = False
6         break
7 if primo:
8     print("Primo")
9 else:
10    print("Composto")
```



## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 primo = True
3 for divisor in range(2, n//2 + 1):
4     if n % divisor == 0:
5         primo = False
6         break
7 if primo:
8     print("Primo")
9 else:
10    print("Composto")
```

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 primo = True
3 for divisor in range(2, int(n/2)+1):
4     if n % divisor == 0:
5         primo = False
6         break
7 if primo:
8     print("Primo")
9 else:
10    print("Composto")
```

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 primo = True
3 for divisor in range(2, int(n**0.5)+1):
4     if n % divisor == 0:
5         primo = False
6         break
7 if primo:
8     print("Primo")
9 else:
10    print("Composto")
```

## Fatoração em Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima sua fatoração em números primos.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisor = 2
3 while n != 1:
4     if n % divisor == 0:
5         print(divisor)
6         n = n / divisor
7     else:
8         divisor = divisor + 1
```

## Fatoração em Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima sua fatoração em números primos.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisor = 2
3 while n != 1:
4     while n % divisor == 0:
5         print(divisor)
6         n = n / divisor
7     divisor = divisor + 1
8
```

# Perguntas ....

# Referências

- Zanoni Dias, MC102, Algoritmos e Programação de Computadores, IC/UNICAMP, 2021. <https://ic.unicamp.br/~mc102/>
  - Aula Introdutória [ [slides](#) ] [ [vídeo](#) ]
  - Primeira Aula de Laboratório [ [slides](#) ] [ [vídeo](#) ]
  - Python Básico: Tipos, Variáveis, Operadores, Entrada e Saída [ [slides](#) ] [ [vídeo](#) ]
  - Comandos Condicionais [ [slides](#) ] [ [vídeo](#) ]
  - Comandos de Repetição [ [slides](#) ] [ [vídeo](#) ]
  - Listas e Tuplas [ [slides](#) ] [ [vídeo](#) ]
  - Strings [ [slides](#) ] [ [vídeo](#) ]
  - Dicionários [ [slides](#) ] [ [vídeo](#) ]
  - Funções [ [slides](#) ] [ [vídeo](#) ]
  - Objetos Multidimensionais [ [slides](#) ] [ [vídeo](#) ]
  - Algoritmos de Ordenação [ [slides](#) ] [ [vídeo](#) ]
  - Algoritmos de Busca [ [slides](#) ] [ [vídeo](#) ]
  - Recursão [ [slides](#) ] [ [vídeo](#) ]
  - Algoritmos de Ordenação Recursivos [ [slides](#) ] [ [vídeo](#) ]
  - Arquivos [ [slides](#) ] [ [vídeo](#) ]
  - Expressões Regulares [ [slides](#) ] [ [vídeo](#) ]
  - Execução de Testes no Google Cloud Shell [ [slides](#) ] [ [vídeo](#) ]
  - Numpy [ [slides](#) ] [ [vídeo](#) ]
  - Pandas [ [slides](#) ] [ [vídeo](#) ]
- Panda - Cursos de Computação em Python (IME -USP) <https://panda.ime.usp.br/>
  - Como Pensar Como um Cientista da Computação <https://panda.ime.usp.br/pensepy/static/pensepy/>
  - Aulas de Introdução à Computação em Python <https://panda.ime.usp.br/aulasPython/static/aulasPython/>
- Fabio Kon, Introdução à Ciência da Computação com Python <http://bit.ly/FabioKon/>
- Socratica, Python Programming Tutorials <http://bit.ly/SocraticaPython/>
- Google - online editor for cloud-native applications (Python programming) <https://shell.cloud.google.com/>
- w3schools - Python Tutorial <https://www.w3schools.com/python/>
- Outros, citados nos Slides.