

MAC 110 – Introdução à Ciência da Computação

Aula 7

Nelson Lago

BMAC – 2024



Previously on MAC110...

Brincando com *strings* — escapes

- O caracter “\” indica o início de uma *sequência de escape*: uma sequência de caracteres que é usada para representar algum outro em uma *string*

```
print("Batatinha quando nasce\nEspalha a rama pelo chão")  
print("Camões bebeu muitos copos d'água escrevendo \"Os Lusíadas\"")  
print('Camões bebeu muitos copos d\'água escrevendo "Os Lusíadas"')
```

- ▶ Espaços em branco “especiais” (\n, \t...)
- ▶ Caracteres “problemáticos” (\", \'...)

Truques com `print()` — `end` e `sep`

- `print()` não termina mudando para a próxima linha, mas sim com o que é definido por `end`
 - ▶ (se você não definir `end`, python utiliza `\n`)
- `print()` não separa os itens com um espaço, mas sim com o que é definido por `sep`
 - ▶ (se você não definir `sep`, python utiliza um espaço)

Truques com `print()` — `end` e `sep`

```
print("super", "cali", "fragilistic",  
      "expiali", "docious", sep="")  
print("Batatinha quando nasce", end="\n.\n.\n.\n")  
print("Espalha a rama pelo chão")
```

supercalifragilisticexpialidocious

Batatinha quando nasce

.
.
.

Espalha a rama pelo chão

Truques com `print()` — `end` e `sep`

```
print("Maria", 45678, "tem noção", sep="\t")  
print("João", 123, "não tem noção", sep="\t")  
print("Ana", 9, "tem noção (mas não muita)", sep="\t")
```

Maria	45678	tem noção
João	123	não tem noção
Ana	9	tem noção (mas não muita)

Truques com `print()` — `end` e `sep`

Dado um número natural $n > 0$, imprima-o com os dígitos invertidos (“de trás para a frente”). Por exemplo, $6437 \rightarrow 7346$

```
n = int(input("Digite um número natural: "))
while n > 0:
    print(n % 10, end="")
    n = n // 10
print()
```

Brincando com *strings* — formatação

Em geral: {**que:como**}

que: nome, número (índice) ou vazio (fica subentendido o índice)

como: A.BC

A: largura mínima; o que falta é preenchido por espaços (à esquerda para números, à direita para strings). Para números, pode ser precedido por zero → preenchimento com zeros (à esquerda)

B: casas depois da vírgula (só para floats)

C: força formato: f (float) ou e (notação científica)

como pode ser omitido (aí não precisa de “:”) → {0}, {b\lah}

que pode ser omitido → {:05}, {:7.2f}

ambos podem ser omitidos (aí não precisa de “:”) → {}

Truques com *strings* — formatação

```
import math
print("Você digitou {0} números ({1} pares e {2} ímpares)".format(12, 7, 5))
print("Você digitou {} números ({} pares e {} ímpares)".format(12, 7, 5))
print("Pi pode ser aproximado para {pi:.7f} ou {pi:.4f}".format(pi=math.pi))
print("x vale {:.7e}".format(1234.5678))
print("|{:9.3f}| -- |{:9.3f}|".format(13.22784, 1200.20004))
print("|{:9.3f}| -- |{:9.3f}|".format(13227.84, 37.6))
print("|{:9.3f}| -- |{:9.3f}|".format(0.0, 127))
```

strings vs `print()`

- escapes são recursos das *strings*
- `sep` e `end` são recursos de `print()`
- comandos de formatação (`.format()`) são recursos das *strings*

Álgebra booleana

Propriedades Comutativas

$$A \text{ and } B = B \text{ and } A$$

$$A \text{ or } B = B \text{ or } A$$

Propriedades Distributivas

$$A \text{ and } (B \text{ or } C) = (A \text{ and } B) \text{ or } (A \text{ and } C)$$

$$A \text{ or } (B \text{ and } C) = (A \text{ or } B) \text{ and } (A \text{ or } C)$$

Propriedades Associativas

$$(A \text{ or } B) \text{ or } C = A \text{ or } (B \text{ or } C)$$

$$(A \text{ and } B) \text{ and } C = A \text{ and } (B \text{ and } C)$$

Propriedades Idempotentes

$$A \text{ and } A = A$$

$$A \text{ or } A = A$$

Dupla Negação

$$\text{not not } A = A$$

Elementos Absorventes

$$A \text{ or } \text{True} = \text{True}$$

$$A \text{ and } \text{False} = \text{False}$$

Elementos Neutros

$$A \text{ or } \text{False} = A$$

$$A \text{ and } \text{True} = A$$

Leis de De Morgan

$$\text{not } (A \text{ or } B) = (\text{not } A) \text{ and } (\text{not } B)$$

$$\text{not } (A \text{ and } B) = (\text{not } A) \text{ or } (\text{not } B)$$

- pizza
- sushi
- Sou guloso
 - ▶ pizza **ou** hambúrguer
- moqueca
- hambúrguer
- Sou alérgico a peixes
 - ▶ **nem** sushi **nem** moqueca

Equivalentes! Qual usar?

O que facilita o entendimento

Álgebra booleana

- **Leis de De Morgan:**

- ▶ $\text{not } (A \text{ or } B) = (\text{not } A) \text{ and } (\text{not } B)$
- ▶ $\text{not } (A \text{ and } B) = (\text{not } A) \text{ or } (\text{not } B)$

- **nem** sushi **nem** moqueca — **(not sushi) and (not moqueca)**

- **não** quero se for **(sushi ou moqueca)** — **not (sushi or moqueca)**

- **Topa lição de casa, jantar e depois cinema?**

- ▶ Não tenho grana para jantar e cinema!
 - » **not** (*jantar and cinema*)
- ▶ Tem que abrir mão de (pelo menos) um deles
 - » **(not jantar) or (not cinema)**
 - » *lição and ((not jantar) or (not cinema))*

Exercícios

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

Exercício — formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000

Exercício — formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.000000

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.000000
- 200000000000.0

Exercício — formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.000000
- 200000000000.0
- 2.0000000e+10

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.0000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

- 2e-10

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

- 2e-10
- 2.0e-10

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

- 2e-10
- 2.0e-10
- 02e-10

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.0000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

- 2e-10
- 2.0e-10
- 02e-10
- 0.0000000

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.0000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

- 2e-10
- 2.0e-10
- 02e-10
- 0.0000000
- 0.0000000000

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.0000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

- 2e-10
- 2.0e-10
- 02e-10
- 0.0000000
- 0.0000000000
- 0.00000000002

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.0000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

- 2e-10
- 2.0e-10
- 02e-10
- 0.0000000
- 0.0000000000
- 0.00000000002
- 0.000000000020

Exercício – formatando números

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{10}$

- 200000000000
- 0200000000000
- 200000000000
- 200000000000.0000000
- 200000000000.0
- 2.0000000e+10
- 2.0e+10
- 2e+10

No shell do IDLE, experimente diferentes maneiras de imprimir o resultado de $2 \cdot 10^{-10}$

- 2e-10
- 2.0e-10
- 02e-10
- 0.0000000
- 0.0000000000
- 0.00000000002
- 0.000000000020
- 2.000000000000000000007286e-10

Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 9 e o preço na coluna 17 da tela.



Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 9 e o preço na coluna 17 da tela.

```
nome = input("Qual o nome do item? ")
```

Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 9 e o preço na coluna 17 da tela.

```
nome = input("Qual o nome do item? ")
cor = input("Qual a cor do item? ")
preço = input("Qual o preço do item? ")
```

Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 9 e o preço na coluna 17 da tela.

```
nome = input("Qual o nome do item? ")
cor = input("Qual a cor do item? ")
preço = input("Qual o preço do item? ")
print(nome, cor, preço, sep="\t")
```

Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 10 e o preço na coluna 16 da tela.



Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 10 e o preço na coluna 16 da tela.

```
nome = input("Qual o nome do item? ")  
cor = input("Qual a cor do item? ")  
preço = input("Qual o preço do item? ")
```

Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 10 e o preço na coluna 16 da tela.

```
nome = input("Qual o nome do item? ")
cor = input("Qual a cor do item? ")
preço = input("Qual o preço do item? ")
print("{ }{ }{ }".format(nome, cor, preço))
```

Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 10 e o preço na coluna 16 da tela.

```
nome = input("Qual o nome do item? ")
cor = input("Qual a cor do item? ")
preço = input("Qual o preço do item? ")
print("{:9}{ }{ }".format(nome, cor, preço))
```


Exercício – formatando tabelas

Escreva um programa que obtém do usuário os dados “nome”, “cor” e “preço” e imprime esses dados em uma linha de maneira que a cor apareça na coluna 10 e o preço na coluna 16 da tela.

```
nome = input("Qual o nome do item? ")
cor = input("Qual a cor do item? ")
preço = input("Qual o preço do item? ")
print("{:9}{:6}{ }".format(nome, cor, preço))
```

Exercício – contagem regressiva

Faça um programa que imprime uma contagem regressiva



Exercício – contagem regressiva

Faça um programa que imprime uma contagem regressiva

```
print("pffft!")
```

Exercício – contagem regressiva

Faça um programa que imprime uma contagem regressiva

```
while i >= 0:
```

```
    print("pffft!")
```

Exercício – contagem regressiva

Faça um programa que imprime uma contagem regressiva

```
i = 10
while i >= 0:

    print("pffft!")
```

Exercício – contagem regressiva

Faça um programa que imprime uma contagem regressiva

```
i = 10
while i >= 0:

    i -= 1

print("pffft!")
```

Exercício – contagem regressiva

Faça um programa que imprime uma contagem regressiva

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

print("pffft!")
```

Exercício – contagem regressiva

Faça um programa que imprime uma contagem regressiva

```
import time
i = 10
while i >= 0:
    print(i)
    i -= 1
    time.sleep(1)

print("pffft!")
```


Exercício – contagem regressiva

Modifique o programa anterior para que os números sejam seguidos por elipses (...)

```
import time
i = 10
while i >= 0:
    print(i)
    i -= 1
    time.sleep(1)

print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números sejam seguidos por elipses (...)

```
import time
i = 10
while i >= 0:
    print(i,          )
    i -= 1
    time.sleep(1)

print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números sejam seguidos por elipses (...)

```
import time
i = 10
while i >= 0:
    print(i, end="...\n")
    i -= 1
    time.sleep(1)

print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números sejam seguidos por elipses (...)

```
import time
i = 10
while i > 0:
    print(i, end="...\n")
    i -= 1
    time.sleep(1)
print("zero!")
print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números sejam seguidos por elipses (...)

```
import time
i = 10
while i > 0:
    print("{:02}".format(i), end="...\n")
    i -= 1
    time.sleep(1)
print("zero!")
print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números apareçam em uma só linha

```
import time
i = 10
while i >= 0:
    print(i)
    i -= 1
    time.sleep(1)

print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números apareçam em uma só linha

```
import time
i = 10
while i >= 0:
    print(i,          )
    i -= 1
    time.sleep(1)

print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números apareçam em uma só linha

```
import time
i = 10
while i >= 0:
    print(i, end=" ")
    i -= 1
    time.sleep(1)

print("pffft!")
```


Exercício – contagem regressiva

Modifique o programa anterior para que os números apareçam em uma só linha

```
import time
i = 10
while i >= 0:
    print(i, end=" ", flush=True)
    i -= 1
    time.sleep(1)

print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números apareçam em uma só linha

```
import time
i = 10
while i >= 0:
    print(i, end=" ", flush=True)
    i -= 1
    time.sleep(1)

print("pffft!")
```

10 9 8 7 6 5 4 3 2 1 0 pffft!

Exercício – contagem regressiva

Modifique o programa anterior para que os números apareçam em uma só linha

```
import time
i = 10
while i >= 0:
    print(i, end=" ", flush=True)
    i -= 1
    time.sleep(1)
print()
print("pffft!")
```

Exercício – contagem regressiva

Modifique o programa anterior para que os números apareçam em uma só linha

```
import time
i = 10
while i > 0:
    print(i, end=" ", flush=True)
    i -= 1
    time.sleep(1)
print("zero!")
print("pffft!")
```

Exercícios

Dado um número n , diga seu equivalente em dúzias e unidades (“27 corresponde a 2 dúzias e 3 unidades”).



Exercícios

Dado um número n , diga seu equivalente em dúzias e unidades (“27 corresponde a 2 dúzias e 3 unidades”).

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

Exercícios

Dado um número n , diga seu equivalente em dúzias e unidades (“27 corresponde a 2 dúzias e 3 unidades”).

```
n = int(input("Digite um número natural: "))  
dúzias = n // 12
```

Exercícios

Dado um número n , diga seu equivalente em dúzias e unidades (“27 corresponde a 2 dúzias e 3 unidades”).

```
n = int(input("Digite um número natural: "))  
dúzias = n // 12  
unidades = n % 12
```


Exercícios

Dado um número n , diga seu equivalente em dúzias e unidades (“27 corresponde a 2 dúzias e 3 unidades”).

```
n = int(input("Digite um número natural: "))
dúzias = n // 12
unidades = n % 12
print("{} corresponde a {} dúzias e {} unidades"
      .format(n, dúzias, unidades))
```

Exercícios

Dado um número n , diga seu equivalente em grosas, dúzias e unidades (“665 corresponde a 4 grosas, 7 dúzias e 5 unidades”).



Exercícios

Dado um número n , diga seu equivalente em grosas, dúzias e unidades (“665 corresponde a 4 grosas, 7 dúzias e 5 unidades”).

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

print("{} corresponde a {} grosas, {} dúzias e {} unidades"
      .format(n, grosas, dúzias, unidades))
```

Exercícios

Dado um número n , diga seu equivalente em grosas, dúzias e unidades (“665 corresponde a 4 grosas, 7 dúzias e 5 unidades”).

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

unidades = n % 12
print("{} corresponde a {} grosas, {} dúzias e {} unidades"
      .format(n, grosas, dúzias, unidades))
```

Exercícios

Dado um número n , diga seu equivalente em grosas, dúzias e unidades (“665 corresponde a 4 grosas, 7 dúzias e 5 unidades”).

```
n = int(input("Digite um número natural: "))
grosas = n // 144

unidades = n % 12
print("{} corresponde a {} grosas, {} dúzias e {} unidades"
      .format(n, grosas, dúzias, unidades))
```

Exercícios

Dado um número n , diga seu equivalente em grosas, dúzias e unidades (“665 corresponde a 4 grosas, 7 dúzias e 5 unidades”).

```
n = int(input("Digite um número natural: "))
grosas = n // 144
soburô = n % 144
dúzias = soburô // 12
unidades = n % 12
print("{} corresponde a {} grosas, {} dúzias e {} unidades"
      .format(n, grosas, dúzias, unidades))
```

Exercícios

Dado um número n , diga seu equivalente em grosas, dúzias e unidades (“665 corresponde a 4 grosas, 7 dúzias e 5 unidades”).

```
n = int(input("Digite um número natural: "))
grosas = n // 144
soburô = n % 144
dúzias = soburô // 12
unidades = soburô % 12
print("{} corresponde a {} grosas, {} dúzias e {} unidades"
      .format(n, grosas, dúzias, unidades))
```

Exercícios

Dado um número n , diga seu equivalente em grosas, dúzias e unidades (“665 corresponde a 4 grosas, 7 dúzias e 5 unidades”).

```
n = int(input("Digite um número natural: "))
m = n
divisor = 12**2
print(n, "corresponde a", end=" ")
print(m // divisor, "grosas,", end=" ")
m %= divisor
divisor //= 12
print(m // divisor, "dúzias e", end=" ")
m %= divisor
divisor //= 12 # == 1
print(m // divisor, "unidades")
```


Exercícios

Dado um número n , diga seu equivalente em grosas, dúzias e unidades (“665 corresponde a 4 grosas, 7 dúzias e 5 unidades”).

```
n = int(input("Digite um número natural: "))
m = n
divisor = 12**2
print(n, "corresponde a", end=" ")
print(m // divisor, "grosas,", end=" ")
m %= divisor
divisor //= 12
print(m // divisor, "dúzias e", end=" ")
m %= divisor
divisor //= 12 # == 1
print(m // divisor, "unidades")
```

Isso tem cara de laço!

Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).



Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))
```

Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))  
dias = n // (60 * 60 * 24)
```

Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))
dias = n // (60 * 60 * 24)
soburô = n % (60 * 60 * 24)
```

Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))
dias = n // (60 * 60 * 24)
soburô = n % (60 * 60 * 24)
horas = soburô // (60 * 60)
```

Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))
dias = n // (60 * 60 * 24)
soburô = n % (60 * 60 * 24)
horas = soburô // (60 * 60)
soburô %= (60 * 60)
```

Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))
dias = n // (60 * 60 * 24)
soburô = n % (60 * 60 * 24)
horas = soburô // (60 * 60)
soburô %= (60 * 60)
minutos = soburô // 60
```


Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))
dias = n // (60 * 60 * 24)
soburô = n % (60 * 60 * 24)
horas = soburô // (60 * 60)
soburô %= (60 * 60)
minutos = soburô // 60
soburô %= 60
```

Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))
dias = n // (60 * 60 * 24)
soburô = n % (60 * 60 * 24)
horas = soburô // (60 * 60)
soburô %= (60 * 60)
minutos = soburô // 60
soburô %= 60
print("{} dias, {} horas, {} minutos e {} segundos"
      .format(dias, horas, minutos, soburô))
```

Exercícios

Dado um número de segundos, como 150328, informe o tempo correspondente em dias, horas, minutos e segundos (neste exemplo, “1 dias, 17 horas, 45 minutos e 28 segundos”).

```
n = int(input("Digite o número de segundos: "))
dias = n // (60 * 60 * 24)
soburô = n % (60 * 60 * 24)
horas = soburô // (60 * 60)
soburô %= (60 * 60)
minutos = soburô // 60
soburô %= 60
print("{} dias, {} horas, {} minutos e {} segundos"
      .format(dias, horas, minutos, soburo))
```

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

```
nota = int(input("Digite a nota: "))
```

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

```
nota = int(input("Digite a nota: "))
```

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

```
nota = float(input("Digite a nota: "))
```

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

```
nota = float(input("Digite a nota: "))  
rec = 0
```


Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

```
nota = float(input("Digite a nota: "))  
rec = 0  
while nota >= 0:
```

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

```
nota = float(input("Digite a nota: "))
rec = 0
while nota >= 0:

    nota = float(input("Digite a nota: "))
```

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

```
nota = float(input("Digite a nota: "))
rec = 0
while nota >= 0:

    nota = float(input("Digite a nota: "))
    print("{} alunos ficaram de recuperação".format(rec))
```

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

```
nota = float(input("Digite a nota: "))
rec = 0
while nota >= 0:
    if nota >= 3 and nota < 5:
        rec += 1
    nota = float(input("Digite a nota: "))
print("{} alunos ficaram de recuperação".format(rec))
```

Exercícios

Dada uma sequência de notas $0 \leq \text{nota} \leq 10$ terminada por um número negativo, diga quantos alunos ficaram de recuperação (um aluno está de recuperação se sua nota foi ao menos 3, mas menor que 5)

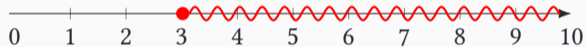
```
nota = float(input("Digite a nota: "))
rec = 0
while nota >= 0:
    if nota >= 3 and nota < 5:
        rec += 1
    nota = float(input("Digite a nota: "))
print("{} alunos ficaram de recuperação".format(rec))
```

Às vezes vale a pena entender o operador “and” como “intersecção de conjuntos”

Exercícios

Às vezes vale a pena entender o operador “and” como “intersecção de conjuntos”

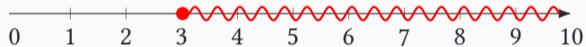
$$3 \leq \text{nota}$$



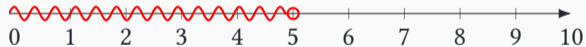
Exercícios

Às vezes vale a pena entender o operador “and” como “intersecção de conjuntos”

$$3 \leq \text{nota}$$



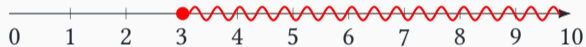
$$\text{nota} < 5$$



Exercícios

Às vezes vale a pena entender o operador “and” como “intersecção de conjuntos”

$$3 \leq \text{nota}$$



$$\text{nota} < 5$$



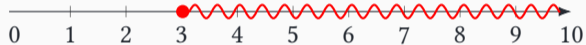
$$3 \leq \text{nota} \text{ and } \text{nota} < 5$$



Exercícios

Às vezes vale a pena entender o operador “and” como “intersecção de conjuntos”
(e o operador “or” como “união de conjuntos”)

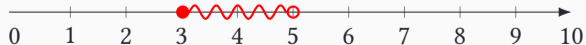
$$3 \leq \text{nota}$$



$$\text{nota} < 5$$



$$3 \leq \text{nota} \text{ and } \text{nota} < 5$$



Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

Qual algoritmo vamos usar?

Exercícios

- Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido

Exercícios

- Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido
- Encontra os n primeiros múltiplos de i e os n primeiros múltiplos de j ; junta todos, coloca em ordem e pega apenas os n primeiros resultados

Exercícios

- Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido
- Encontra os n primeiros múltiplos de i e os n primeiros múltiplos de j ; junta todos, coloca em ordem e pega apenas os n primeiros resultados
- Vai testando os múltiplos de i e j “alternadamente” (na verdade, sempre o menor múltiplo ainda não testado) até encontrar n números

Exercícios

- **Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido**
 - ▶ Simples, porém ineficiente
- **Encontra os n primeiros múltiplos de i e os n primeiros múltiplos de j ; junta todos, coloca em ordem e pega apenas os n primeiros resultados**

- **Vai testando os múltiplos de i e j “alternadamente” (na verdade, sempre o menor múltiplo ainda não testado) até encontrar n números**

Exercícios

- **Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido**
 - ▶ Simples, porém ineficiente
- **Encontra os n primeiros múltiplos de i e os n primeiros múltiplos de j ; junta todos, coloca em ordem e pega apenas os n primeiros resultados**
 - ▶ Absurdamente complicado e ineficiente (e envolve coisas que ainda não vimos)
- **Vai testando os múltiplos de i e j “alternadamente” (na verdade, sempre o menor múltiplo ainda não testado) até encontrar n números**

Exercícios

- **Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido**
 - ▶ Simples, porém ineficiente
- **Encontra os n primeiros múltiplos de i e os n primeiros múltiplos de j ; junta todos, coloca em ordem e pega apenas os n primeiros resultados**
 - ▶ Absurdamente complicado e ineficiente (e envolve coisas que ainda não vimos)
- **Vai testando os múltiplos de i e j “alternadamente” (na verdade, sempre o menor múltiplo ainda não testado) até encontrar n números**
 - ▶ Medianamente complexo, mas bastante eficiente

Exercícios

- **Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido**
 - ▶ Simples, porém ineficiente
- **Encontra os n primeiros múltiplos de i e os n primeiros múltiplos de j ; junta todos, coloca em ordem e pega apenas os n primeiros resultados**
 - ▶ Absurdamente complicado e ineficiente (e envolve coisas que ainda não vimos)
- **Vai testando os múltiplos de i e j “alternadamente” (na verdade, sempre o menor múltiplo ainda não testado) até encontrar n números**
 - ▶ Medianamente complexo, mas bastante eficiente

Exercícios

- **Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido**
 - ▶ Simples, porém ineficiente
- **Encontra os n primeiros múltiplos de i e os n primeiros múltiplos de j ; junta todos, coloca em ordem e pega apenas os n primeiros resultados**
 - ▶ Absurdamente complicado e ineficiente (e envolve coisas que ainda não vimos)
- **Vai testando os múltiplos de i e j “alternadamente” (na verdade, sempre o menor múltiplo ainda não testado) até encontrar n números**
 - ▶ Medianamente complexo, mas bastante eficiente — estamos começando!

Exercícios

- **Vai testando todos os naturais até encontrar n números que satisfaçam o que foi pedido**
 - ▶ Simples, porém ineficiente
- **Encontra os n primeiros múltiplos de i e os n primeiros múltiplos de j ; junta todos, coloca em ordem e pega apenas os n primeiros resultados**
 - ▶ Absurdamente complicado e ineficiente (e envolve coisas que ainda não vimos)
- **Vai testando os múltiplos de i e j “alternadamente” (na verdade, sempre o menor múltiplo ainda não testado) até encontrar n números**
 - ▶ Medianamente complexo, mas bastante eficiente — estamos começando!

“Premature optimization is the root of all evil” (Knuth?)

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
n = int(input("Digite n: "))  
i = int(input("Digite i: "))  
j = int(input("Digite j: "))  
...
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0  
x = 1
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j != 0:
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:

    elif x % i == 0:
        print(x, end=" ")
        encontrados += 1
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:

    elif x % i == 0:
        print(x, end=" ")
        encontrados += 1
    elif x % j == 0:
        print(x, end=" ")
        encontrados += 1
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:

    elif x % i == 0:
        print(x, end=" ")
        encontrados += 1
    elif x % j == 0:
        print(x, end=" ")
        encontrados += 1
    else:
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:

    elif x % i == 0:
        print(x, end=" ")
        encontrados += 1
    elif x % j == 0:
        print(x, end=" ")
        encontrados += 1
    else:

x += 1
```


Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:

    elif x % i == 0 or x % j == 0 :
        print(x, end=" ")
        encontrados += 1
    else:

x += 1
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:
        pass
    elif x % i == 0 or x % j == 0 :
        print(x, end=" ")
        encontrados += 1
    else:
        pass
    x += 1
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:
        pass
    elif x % i == 0 or x % j == 0 :
        print(x, end=" ")
        encontrados += 1
    else:
        pass
    x += 1
print()
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:
        pass
    elif x % i == 0 or x % j == 0 :
        print(x, end=" ")
        encontrados += 1

    x += 1
print()
```

Condições mutuamente excludentes

```
if delta < 0:
    print("não há raízes reais")
elif delta == 0:
    ...
    print("A raiz dupla é", raiz)
else:
    ...
    print("As raízes são {} e {}".format(raiz1, raiz2))
```

- A indentação deixa mais claro que, na verdade, são casos do mesmo “nível” e mutuamente excludentes (**um e apenas um** dos casos é executado)

Condições mutuamente excludentes

```
if delta < 0:
    print("não há raízes reais")
elif delta == 0:
    ...
    print("A raiz dupla é", raiz)
else:
    ...
    print("As raízes são {} e {}".format(raiz1, raiz2))
```

- A indentação deixa mais claro que, na verdade, são casos do mesmo “nível” e mutuamente excludentes (**um e apenas um** dos casos é executado)

Mas a ordem pode fazer diferença!

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:
        pass
    elif x % i == 0 or x % j == 0 :
        print(x, end=" ")
        encontrados += 1

    x += 1
print()
```

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:
        pass
    elif (x % i == 0 or x % j == 0) and not (x % i == 0 and x % j == 0):
        print(x, end=" ")
        encontrados += 1

    x += 1
print()
```


Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:
        pass
    elif (x % i == 0 or x % j == 0) and not (x % i == 0 and x % j == 0):
        print(x, end=" ")
        encontrados += 1

    x += 1
print()
```

(implícito)

Exercícios

Dados os naturais positivos n , i e j , imprimir em ordem crescente os n primeiros naturais que sejam múltiplos de i ou de j , mas não de ambos.

```
encontrados = 0
x = 1
while encontrados < n:
    if x % i == 0 and x % j == 0:
        pass
    elif (x % i == 0 or x % j == 0) and not (x % i == 0 and x % j == 0):
        print(x, end=" ")
        encontrados += 1

    x += 1
print()
```

(implícito)

A ordem faz diferença!

Lição de casa:
o que acontece se i e j são iguais?