MAC 115 — Introdução à Ciência da Computação

Aula 6

Nelson Lago

IF noturno – 2023

Previously on MAC 115...

Programando

1. **algoritmo vs implementação**
2. **entrada de dados → processamento → resultado**
   - Mostra o resultado para o usuário
   - **Utiliza o resultado como dado para fazer outra coisa**
3. **Existem *tipos de dados* diferentes em python (*int*, *float*, *string*, *bool*...)**
4. **Expressões são coisas que têm um *valor* (de algum *tipo*)**
   - E podem ser combinadas ou utilizadas como partes de outras expressões

`2 + 3 + 7` (int)

`2 > 3 and 5 > 4` (bool)

Nomes (variáveis)

- **_Nomes_ permitem que pensemos mais sobre o problema a ser resolvido e menos sobre as idiossincrasias do computador**
- **Nomes em geral representam valores que _variam_ (basicamente, alguma informação "real" que está em algum lugar na memória do computador)**
  - Como na matemática!
- **Por isso, chamamos esses nomes de "variáveis"**

Nomes (variáveis)

x ← 5 (atribuição)

Há um número finito de caracteres no teclado, então fazemos atribuição em python com “=”:

```
x = 5
x = x + 1
x = input("Digite seu nome: ")
```

Execução condicional

```
n = int(input("Digite um número natural: "))
if n % 2 == 0:
    print("O número", n, "é par!")
print("Ahazei!")
```

- **`n % 2 == 0` → `True` ou `False`**
  - Embora possamos ler "se condição", na verdade python faz "se o valor da expressão é verdadeiro (`True`)"
  - É **como se** python executasse `if condição == True`
- **Como ele sabe onde "acaba o efeito" do `if`?**
  - Qualquer quantidade de espaços, desde que seja consistente (4 espaços é o mais comum)

Execução condicional – `else`

```
n = int(input("Digite um número natural: "))
if n % 2 == 0:
    print("O número", n, "é par!")
else:
    print("O número", n, "é ímpar!")
print("Ahazei!")
```

- **A *indentação* indica os "lados" do condicional**
  - Sem variável ("n"), o programa sempre executaria o mesmo "lado"
- **O estado da variável só importa no momento do teste**
  - Se ela mudar em seguida, não afeta o condicional
- **Os "lados" são mutuamente excludentes**
  - **Um** e **apenas um** deles é executado

Por que computação?

O computador é extremamente rápido, mas é uma ferramenta com o mesmo nível de "inteligência" que um martelo

Não é mais fácil fazer manualmente?

Repetições e condições

- **"Algo" precisa acontecer para indicar que as repetições chegaram ao fim**

  (ok, às vezes queremos repetir indefinidamente, mas vamos ignorar isso por enquanto)

- **As repetições são controladas por algum tipo de *condição* baseada no estado de uma *variável***

Partes mínimas de um laço

- **Um laço correto precisa**
  - Inicializar a variável de controle antes do início do laço
  - Verificar a condição adequada a cada iteração para que as repetições aconteçam o número correto de vezes
  - Alterar o valor da variável de acordo com a lógica do programa (no mínimo, na última iteração) para garantir que o laço termine

```
usuarioQuerJogar = True
while usuarioQuerJogar:
    # Joga uma partida...
    resposta = input("Você quer jogar novamente? ")
    if resposta != "S":
        usuarioQuerJogar = False
print("Cabô!")
```

Tipos de repetição

- **Dois tipos fundamentais de repetição**
  1. Repetições até atingir um resultado
     - » *Encontrar o próximo primo*
     - » *Reiniciar o jogo até o usuário escolher "sair"*
     - » *...*
  2. Repetições sobre os elementos de um conjunto
     - » *Apresentar todos os pixels de uma foto na tela*
     - » *Trocar todas as letras de um texto para maiúsculas*
     - » *...*

Tipos de repetição

- **Dois tipos fundamentais de repetição**
  1. Repetições até atingir um resultado
     - » *Encontrar o próximo primo*
     - » *Reiniciar o jogo até o usuário escolher "sair"*
     - » *...*
  2. Repetições sobre os elementos de um conjunto
     - » *Apresentar todos os pixels de uma foto na tela*
     - » *Trocar todas as letras de um texto para maiúsculas*
     - » *...*

Mais um exercício!!!!

Exercício

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

Exercício

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: "))
```

Exercício

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: "))
invertido = 0
```

Exercício

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: "))
invertido = 0









print(n, "invertido é", invertido)
```

Exercício

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: "))
invertido = 0
tmp = n
while tmp > 0:




print(n, "invertido é", invertido)
```

Exercício

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: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10



print(n, "invertido é", invertido)
```

Exercício

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: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10

print(n, "invertido é", invertido)
```

Exercício

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: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

Antes do início do laço:

invertido: 0

tmp: 6437

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`invertido = invertido * 10`:

invertido: 0

tmp: 6437

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`invertido = invertido + tmp % 10`:

invertido: 7

tmp: 6437

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`tmp = tmp // 10`:

invertido: 7

tmp: 643

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`while tmp > 0:`

invertido: 7

tmp: 643

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`invertido = invertido * 10`:

invertido: 70

tmp: 643

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`invertido = invertido + tmp % 10`:

invertido: 73

tmp: 643

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`tmp = tmp // 10`:

invertido: 73

tmp: 64

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`while tmp > 0:`

invertido: 73

tmp: 64

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`invertido = invertido * 10`:

invertido: 730

tmp: 64

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`invertido = invertido + tmp % 10`:

invertido: 734

tmp: 64

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`tmp = tmp // 10`:

invertido: 734

tmp: 6

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

```
while tmp > 0:
```

invertido: 734

tmp: 6

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`invertido = invertido * 10`:

invertido: 7340

tmp: 6

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`invertido = invertido + tmp % 10`:

invertido: 7346

tmp: 6

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`tmp = tmp // 10`:

invertido: 7346

tmp: 0

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

Exercício

Vamos experimentar com 6437?

`while tmp > 0:`

invertido: 7346

tmp: 0

```
n = int(input("Digite um número natural: "))
invertido = 0
tmp = n
while tmp > 0:
    invertido = invertido * 10
    invertido = invertido + tmp % 10
    tmp = tmp // 10
print(n, "invertido é", invertido)
```

and now for something completely different

Brincando com *strings*

- **Como imprimir estas frases?**
  - Preciso de um copo d'água

Brincando com *strings*

- **Como imprimir estas frases?**
  - Preciso de um copo d'água

```
print("Preciso de um copo d'água")
```

Brincando com *strings*

- **Como imprimir estas frases?**
  - Preciso de um copo d'água

    ```
    print("Preciso de um copo d'água")
    ```
  - Camões escreveu "Os Lusíadas" no século XVI

Brincando com *strings*

- **Como imprimir estas frases?**
  - Preciso de um copo d'água

    ```
    print("Preciso de um copo d'água")
    ```
  - Camões escreveu "Os Lusíadas" no século XVI

    ```
    print('Camões escreveu "Os Lusíadas" no século XVI')
    ```

Brincando com *strings*

- **Como imprimir estas frases?**

Brincando com *strings*

- **Como imprimir estas frases?**
  - Batatinha quando nasce
    Espalha a rama pelo chão

Brincando com *strings*

- **Como imprimir estas frases?**
  - Batatinha quando nasce
    Espalha a rama pelo chão

```
print("Batatinha quando nasce")
print("Espalha a rama pelo chão")
```

Brincando com *strings*

- **Como imprimir estas frases?**
  - Batatinha quando nasce
    Espalha a rama pelo chão

```
print("Batatinha quando nasce")
print("Espalha a rama pelo chão")
```

VEJA BEM...

Brincando com *strings*

- **Como imprimir estas frases?**
  - Batatinha quando nasce
    Espalha a rama pelo chão

```
print("Batatinha quando nasce")
print("Espalha a rama pelo chão")
```

VEJA BEM...

  - Camões bebeu muitos copos d'água escrevendo "Os Lusíadas"

Brincando com *strings*

- **Como imprimir estas frases?**
  - Batatinha quando nasce
    Espalha a rama pelo chão

```
print("Batatinha quando nasce")
print("Espalha a rama pelo chão")
```

VEJA BEM...

  - Camões bebeu muitos copos d'água escrevendo "Os Lusíadas"

```
print("Camões bebeu muitos copos d'água",
      'escrevendo "Os Lusíadas"')
```

Brincando com *strings*

- **Como imprimir estas frases?**
  - Batatinha quando nasce
    Espalha a rama pelo chão

```
print("Batatinha quando nasce")
print("Espalha a rama pelo chão")
```

VEJA BEM...

  - Camões bebeu muitos copos d'água escrevendo "Os Lusíadas"

```
print("Camões bebeu muitos copos d'água",
      'escrevendo "Os Lusíadas"')
```

VEJA BEM...

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*

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")
```

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\"")
```

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"')
```

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” (\", \' ...)

Brincando com *strings* — escapes

- **Como imprimir esta frase:**

Brincando com *strings* — escapes

- **Como imprimir esta frase:**
  - O caracter '\' resolve "todos" os problemas do mundo

Brincando com *strings* — escapes

- **Como imprimir esta frase:**
  - O caracter '\' resolve "todos" os problemas do mundo

```
print('O caracter \'\\\' resolve "todos" os problemas do mundo')
print("O caracter '\\' resolve \"todos\" os problemas do mundo")
```

Brincando com *strings* — formatação

“Você digitou um total de 12 números: 7 números pares e 5 números ímpares”

```
n = 12
p = 7
i = 5
print("Você digitou um total de", n, "números:", p, "números pares e", i, "números ímpares"
```

Essa linha tem 92 caracteres; o recomendado é não passar de 79 ☹

Brincando com *strings* — formatação

“Você digitou um total de 12 números: 7 números pares e 5 números ímpares”

```
n = 12
p = 7
i = 5

print("Você digitou um total de", n, "números:",
      p, "números pares e", i, "números ímpares")
```

Brincando com *strings* — formatação

“Você digitou um total de 12 números: 7 números pares e 5 números ímpares”

```
n = 12
p = 7
i = 5



print("Você digitou um total de", n, "números:", end=" ")
print(p, "números pares e", i, "números ímpares")
```

Brincando com *strings* — formatação

“Você digitou um total de |12| números: |7| números pares e |5| números ímpares”

```
n = 12
p = 7
i = 5
```

Brincando com *strings* — formatação

“Você digitou um total de |12| números: |7| números pares e |5| números ímpares”

```
n = 12
p = 7
i = 5
print("Você digitou um total de ", n, " números: ", p,
      " números pares e ", i, " números ímpares", sep="|")
```

Brincando com *strings* — formatação

"Você digitou um total de |12| números: |7| números pares e |5| números ímpares"

```
n = 12
p = 7
i = 5
print("Você digitou um total de ", n, " números: ", p,
      " números pares e ", i, " números ímpares", sep="|")
```

Brincando com *strings* — formatação

“Você digitou 12 números (7 pares e 5 ímpares)”

```
n = 12
p = 7
i = 5
```

Brincando com *strings* — formatação

“Você digitou 12 números (7 pares e 5 ímpares)”

```
n = 12
p = 7
i = 5
print("Você digitou {0} números ({1} pares e {2} ímpares)".format(n, p, i))
```

Brincando com *strings* — formatação

“Você digitou 12 números (7 pares e 5 ímpares)”

```
n = 12
p = 7
i = 5
print("Você digitou {0} números ({1} pares e {2} ímpares)".format(n, p, i))
print("Você digitou {} números ({} pares e {} ímpares)".format(n, p, i))
```

Brincando com *strings* — formatação

“Você digitou 12 números (7 pares e 5 ímpares)”

```
n = 12
p = 7
i = 5
print("Você digitou {0} números ({1} pares e {2} ímpares)".format(n, p, i))
print("Você digitou {} números ({} pares e {} ímpares)".format(n, p, i))
print("Você digitou {num} números ({par} pares e {impar} ímpares)"
      .format(num=n, par=p, impar=i))
```

Brincando com *strings* — formatação

```
import math
print("Pi pode ser aproximado para {:.7f} ou {:.4f}".format(math.pi, math.pi))
```

Brincando com *strings* — formatação

```
import math
print("Pi pode ser aproximado para {:.7f} ou {:.4f}".format(math.pi, math.pi))
```

Pi pode ser aproximado para 3.1415927 ou 3.1416

Brincando com *strings* — formatação

```
import math
print("Pi pode ser aproximado para {:.7f} ou {:.4f}".format(math.pi, math.pi))
print("Pi pode ser aproximado para {0:.7f} ou {0:.4f}".format(math.pi))
```

Pi pode ser aproximado para 3.1415927 ou 3.1416

Brincando com *strings* — formatação

```
import math
print("Pi pode ser aproximado para {:.7f} ou {:.4f}".format(math.pi, math.pi))
print("Pi pode ser aproximado para {0:.7f} ou {0:.4f}".format(math.pi))
print("Pi pode ser aproximado para {pi:.7f} ou {pi:.4f}".format(pi=math.pi))
```

Pi pode ser aproximado para 3.1415927 ou 3.1416

Brincando com *strings* — formatação

```
x = 1234.5678
print("x vale {:.7e}".format(x))
```

Brincando com *strings* — formatação

```
x = 1234.5678
print("x vale {:.7e}".format(x))
```

```
x vale 1.2345678e+03
```

Brincando com *strings* — formatação

```
a = 13.22784
b = 1200.20004
c = 13227.84
d = 37.6
e = 0.02
f = 127.0
print("|{:9.3f}| -- |{:9.3f}|".format(a, b))
print("|{:9.3f}| -- |{:9.3f}|".format(c, d))
print("|{:9.3f}| -- |{:9.3f}|".format(e, f))
```

Brincando com *strings* — formatação

```
a = 13.22784
b = 1200.20004
c = 13227.84
d = 37.6
e = 0.02
f = 127.0
print("|{:9.3f}| -- |{:9.3f}|".format(a, b))
print("|{:9.3f}| -- |{:9.3f}|".format(c, d))
print("|{:9.3f}| -- |{:9.3f}|".format(e, f))
```

```
|   13.228| -- | 1200.200|
|13227.840| -- |   37.600|
|    0.020| -- |  127.000|
```

Brincando com *strings* — formatação

```
a = 13.22784
b = 1200.20004
c = 13227.84
d = 37.6
e = 0.02
f = 127.0
print("|{:9.3f}| -- |{:9.3f}|".format(a, b))
print("|{:9.3f}| -- |{:9.3f}|".format(c, d))
print("|{:9.3f}| -- |{:9.3f}|".format(e, f))
```

```
|   13.228| -- | 1200.200|
|13227.840| -- |   37.600|
|    0.020| -- |  127.000|
```

https://docs.python.org/3/library/string.html#formatspec