

# MC-102 — Aula 07

## Comandos Repetitivos

Prof. Luiz F. Bittencourt

Turmas QR

Instituto de Computação – Unicamp

2019

Conteúdo adaptado de slides fornecidos pelo Prof. Eduardo Xavier.

# Introdução

- Vimos quais são os comandos de repetição em Python.
- Veremos mais alguns exemplos de sua utilização.

# Variável Indicadora

- Um outro uso comum de laços é para a verificação se um determinado objeto, ou conjunto de objetos, satisfaz uma propriedade ou não.
- Um padrão que pode ser útil na resolução deste tipo de problema é o uso de uma **variável indicadora**.
  - ▶ Assumimos que o objeto satisfaz a propriedade (indicadora = True).
  - ▶ Com um laço verificamos se o objeto realmente satisfaz a propriedade. Se em alguma iteração descobrirmos que o objeto não satisfaz a propriedade, então fazemos (indicadora = False).

# Exemplo: Números Primos

## Problema

Determinar se um número  $n$  é primo ou não.

- Um número é primo se seus únicos divisores são 1 e ele mesmo.
- Dado um número  $n$  como detectar se este é ou não primo??
  - 1 Lê um número  $n$ .
  - 2 Testa se nenhum dos números entre 2 e  $(n - 1)$  divide  $n$ .
- Lembre-se que o operador  $\%$  retorna o resto da divisão.
- Portanto  $(a \% b)$  é zero se e somente se  $b$  divide  $a$ .

## Exemplo: Números Primos

```
Ler um número n
div = 2
indicadora = True #assumimos que n é primo
Enquanto div <= (n-1) faça
    Se (n%div) for igual a zero Então
        indicadora = False #descobrimos que n não é primo
    div = div +1
Se indicadora == True então o número é primo
```

# Exemplo: Números Primos

Em Python:

```
n = int(input("Digite um número:"))
div = 2
eprimo=1
while (div<=n-1) and (eprimo) :
    if(n%div == 0):
        eprimo=False
    div = div + 1

if(eprimo):
    print("É primo!!")
else:
    print("Não é primo!!")
```

# Exemplo: Números Primos

Com o uso de **break**:

```
n = int(input("Digite um número:"))
div = 2
eprimo=1
while div<=n-1 :
    if(n%div == 0):
        eprimo=False
        break
    div = div + 1

if(eprimo):
    print("É primo!!")
else:
    print("Não é primo!!")
```

# Exemplo: Números em Ordem

## Problema

Fazer um programa que lê  $n$  números inteiros do teclado e, no final, informa se os números lidos estão ou não em ordem crescente.

- Usaremos uma variável indicadora na resolução deste problema.



## Exemplo: Números em Ordem

- Um laço principal será responsável pela leitura dos números.
- Vamos usar duas variáveis: uma que guarda o número lido na iteração atual e uma que guarda o número lido na iteração anterior.
- Os números estarão ordenados se a condição ( $\text{anterior} \leq \text{atual}$ ) for válida durante a leitura de todos os números.

```
Leia um número e guarde em n
ordenado = 1 #Assumimos que os números estão ordenados
Leia um número e salve em anterior
Repita (n-1) vezes
    Leia um número e salve em atual
    Se atual < anterior
        ordenado = 0
    anterior = atual
```

# Exemplo: Números em Ordem

Em Python:

```
n = int(input("Digite um número:"))
anterior = int(input())
i = 1 #leu um número
ordenado = True

while (i < n) and ordenado :
    atual = int(input())
    i = i + 1 #leu mais um número
    if(atual < anterior):
        ordenado = False
    anterior = atual

if(ordenado):
    print("Sequência está ordenada")
else:
    print("Sequência não está ordenada")
```

# Variável Contadora

- Considere ainda o uso de laços para a verificação se um determinado objeto, ou conjunto de objetos, satisfaz uma propriedade ou não.
- Um outro padrão que pode ser útil é o uso de uma **variável contadora**.
  - ▶ Esperamos que um objeto satisfaça  $x$  vezes uma sub-propriedade. Usamos um laço e uma variável que **conta** o número de vezes que o objeto tem a sub-propriedade satisfeita.
  - ▶ Ao terminar o laço, se contadora for igual à  $x$  então o objeto satisfaz a propriedade.

# Exemplo: Números Primos

- Um número  $n$  é primo se nenhum número de 2 até  $(n - 1)$  dividi-lo.
- Podemos usar uma variável que conta quantos números dividem  $n$ .
- Se o número de divisores for 0, então  $n$  é primo.

```
Leia um número e salve em n
div = 2
divisores = 0 //ninguém divide n ainda
Enquanto div <= (n-1) faça
    Se (n%div) == 0
        divisores = divisores + 1
    div = div + 1

Se divisores == 0 então
    Número é primo
```

# Exemplo: Números Primos

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

div = 2
divisores=0

while div <= n-1:
    if(n%div == 0):
        divisores = divisores + 1
    div = div + 1

if(divisores == 0):
    print("É primo!!")
else:
    print("Não é primo!!")
```

## Exemplo: Números Primos

É claro que é melhor terminar o laço assim que descobrirmos algum divisor de  $n$ .

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

div = 2
divisores=0

while (div <= n-1) and (divisores==0):
    if(n%div == 0):
        divisores = divisores + 1
        div = div + 1

if(divisores == 0):
    print("É primo!!")
else:
    print("Não é primo!!")
```

# Outros Exemplos

- O uso de variáveis **acumuladora**, **indicadora** e **contadora** são úteis em várias situações.
- Mas não existem fórmulas para a criação de soluções para problemas.
- Em outros problemas, o uso destes padrões pode aparecer em conjunto ou mesmo nem aparecer como parte da solução.

# Maior Número

## Problema

Fazer um programa que lê  $n$  números do teclado e informa qual foi o maior número lido.

- O programa deve ter os seguintes passos:
  - 1 Leia um número e salve em  $n$ .
  - 2 Repita  $n$  vezes a leitura de um número determinando o maior.
- Como determinar o maior??



# Maior Número

- A idéia é criar uma variável **maior** que sempre armazena o maior número lido até então.

Leia um número e salve em n

Leia um número e salve em maior

Repita n-1 vezes

    Leia um número e salve em aux

    Se aux > maior então

        maior = aux

# Maior Número

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

maior = int(input())
cont = 1 #leu um número
while cont<n:
    aux = int(input())
    if(aux>maior):
        maior = aux
    cont = cont + 1 #leu mais um número

print("O maior número é: ", maior)
```

# Números de Fibonacci

- A série de Fibonacci é: 1, 1, 2, 3, 5, 8, 13, ...
- Ou seja o  $n$ -ésimo termo é a soma dos dois anteriores

$$F(n) = F(n - 1) + F(n - 2)$$

onde  $F(1) = 1$  e  $F(2) = 1$ .

## Problema

Fazer um programa que imprime os primeiros  $n$  números da série de fibonacci.

# Números de Fibonacci

```
Leia um número e salve em n
contador = 1
f_atual = 1, f_ant = 0
Enquanto contador <= n faça
    Imprima f_atual
    aux = f_atual
    f_atual = f_atual + f_ant
    f_ant = aux
    contador = contador +1
```

# Números de Fibonacci

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

cont = 0 #conta quantidade de num. impressos
f_ant=0
f_atual=1
while cont < n : #enquanto não imprimiu n números
    print( f_atual, ", ", end="")
    cont = cont + 1 #imprimiu mais um número
    f_aux = f_atual
    f_atual = f_atual + f_ant
    f_ant = f_aux
```

# Exercício

- No exemplo dos números primos não precisamos testar todos os números entre  $2, \dots, (n - 1)$ , para verificar se dividem ou não  $n$ . Basta testarmos até  $n/2$ . Por quê? Qual o maior divisor possível de  $n$ ?
- Na verdade basta testarmos os números  $2, \dots, \sqrt{n}$ . Por quê?

# Exercício

- Considere o programa para determinar se uma sequência de  $n$  números digitados pelo usuário está ordenada ou não. Refaça o programa usando uma variável contadora ao invés de indicadora.

# Exercício

- Faça um programa em C que calcule o máximo divisor comum de dois números  $m, n$ . Você deve utilizar a seguinte regra do cálculo do mdc com  $m \geq n$

$$\text{mdc}(m, n) = m \text{ se } n = 0$$

$$\text{mdc}(m, n) = \text{mdc}(n, m \% n) \text{ se } n > 0$$