

Python: Comandos de repetição

Galileu Batista de Sousa
Galileu.batista -at +ifrn -edu +br

O fundamento da repetição



Suponha que se deseja somar os números de 1 a 100 ...

```
soma = 1 + 2 + 3 + ... + 100  
print ("soma = ", soma)
```

Ruim.
Específico.

Ainda pior.
Usa mais
código ...

```
soma = 0  
soma = soma + 1  
soma = soma + 2  
soma = soma + 3  
...  
print ("soma = ", soma)
```

O fundamento da repetição



Suponha que se deseja somar os números de 1 a 100 ...

```
soma = 0
x = 1
soma = soma + x
x = 2
soma = soma + x
x = 3
soma = soma + x
...
print ("soma = ", soma)
```

Nada é tão
ruim que não
possa piorar

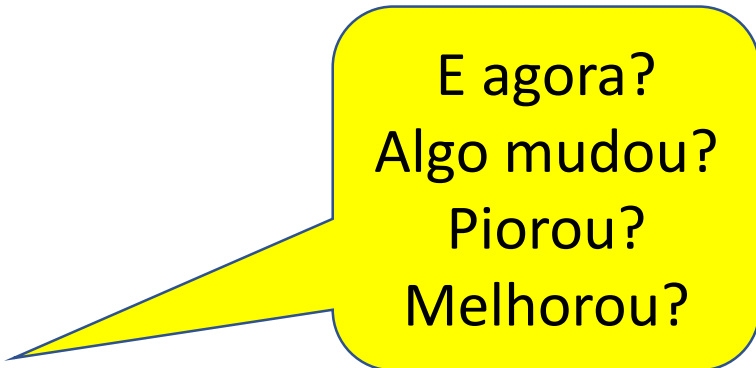
...

O fundamento da repetição



Suponha que se deseja somar os números de 1 a 100 ...

```
soma = 0
x = 1
soma = soma + x
x = x + 1
soma = soma + x
x = x + 1
soma = soma + x
...
print ("soma = ", soma)
```



E agora?
Algo mudou?
Piorou?
Melhorou?

O fundamento da repetição



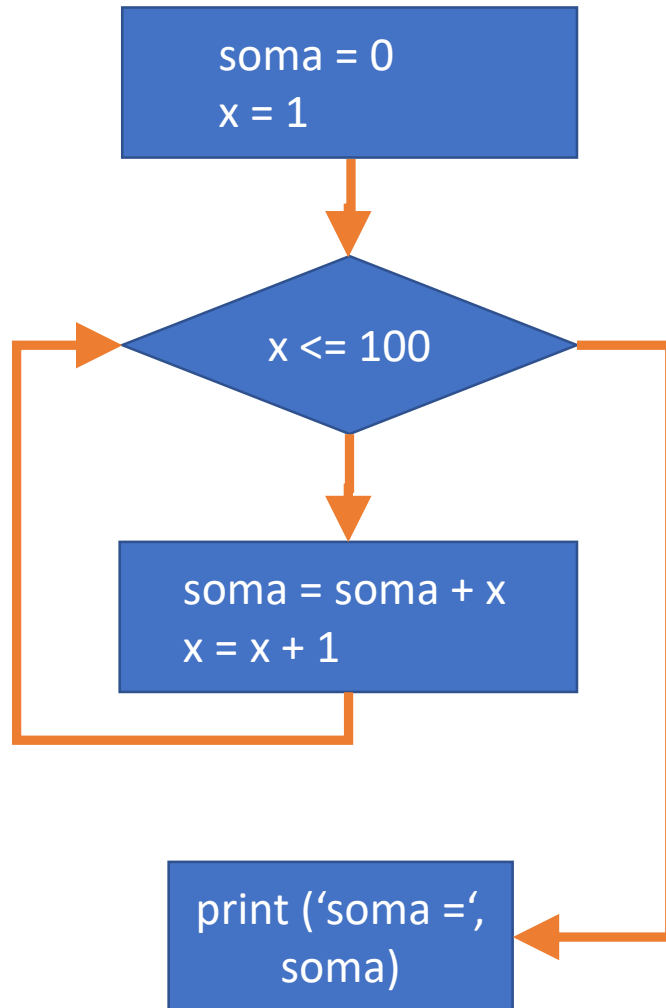
Suponha que se deseja somar os números de 1 a 100 ...

```
soma = 0
x = 1
soma = soma + x
x = x + 1
soma = soma + x
x = x + 1
soma = soma + x
x = x + 1
...
print ("soma = ", soma)
```

Aqui identificamos um padrão que se repete. Podemos usar isso nosso favor.

- Podemos pensar em:
 - Faça a primeira parte
 - repita a destacada
 - Mostre o resultado
- Python oferece essa possibilidade

A ideia de execução repetitiva



```
soma = 0
x = 1
while x <= 100 :
    soma = soma + x
    x = x + 1
print ('soma =', soma)
```

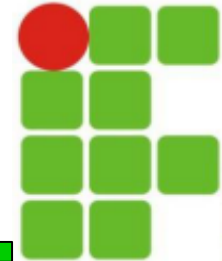
Detalhes sobre while



```
limite = int (input('1 a ?'))  
soma = 0  
x = 1  
while x <= limite:  
    soma = soma + x  
    x = x + 1  
print ('soma =', soma)
```

- Identação
 - Subordinação
- **:** finalizando o **while**
- Lógica
 - A condição pode ser qualquer expressão relacional

Identação



- Estabelece a subordinação de comandos
 - Mais de um comando subordinado, mesma identação
- Alinhamento com comando subordinador
 - Acabou a subordinação
- Use **quatro espaços** com identação (**nunca tab**)
 - Modelo mental para deixar claro quem manda.

Um exemplo mais complexo



- Determinar se um número **71 é primo**
 - Um primo tem exatamente dois divisores distintos.
- A ideia:
 - Observar se 1 é divisor de 71
 - Observar se 2 é divisor de 71
 -
 - Observar se 71 é divisor de 71
 - Ao final: tem 2 divisores?

Cada vez que for observado um divisor, aumente 1 numa variável que guarda o número de divisores

Se a variável que guarda o número de divisores tiver o valor 2, 71 é primo

71 é primo?



```
ndiv = 0
if 71 % 1 == 0:
    ndiv = ndiv + 1
if 71 % 2 == 0:
    ndiv = ndiv + 1
...
if ndiv == 2:
    print ("71 é primo.")
```

Dá pra
melhorar,
né?

71 é primo?



```
ndiv = 0
div = 1
if 71 % div == 0:
    ndiv = ndiv + 1
div = div + 1
if 71 % div == 0:
    ndiv = ndiv + 1
div = div + 1
...
if ndiv == 2:
    print ("71 é primo.")
```

Dá pra
melhorar,
né?

71 é primo?



```
ndiv = 0
div = 1
while div <= 71:
    if 71 % div == 0:
        ndiv = ndiv + 1
        div = div + 1
if ndiv == 2:
    print ("71 é primo.")
```



Bem
melhor ...

x é primo?



Somente ler o valor de x e usá-lo em vez de 71

```
x = int (input("Qual o valor de x: "))
ndiv = 0
div = 1
while div <= x:
    if x % div == 0:
        ndiv = ndiv + 1
        div = div + 1
if ndiv == 2:
    print (x, "é primo.")
```

Pronto. Agora funciona para qualquer número

Mostrar todos os primos até 100



```
x = 1  
while x <= 100:
```

```
    x = x + 1
```

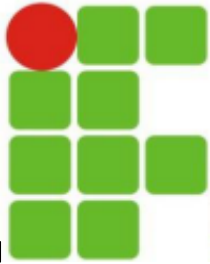
Sequência controlada pelo usuário



- **Calcular a média** dos números digitados pelo usuário
 - Parar quando o usuário digitar número negativo.
- **As diferenças:**
 - O primeiro número é dado pelo usuário
 - O próximo número é dado pelo usuário
 - A condição de parada é número negativo

```
num = int (input('numero?'))  
while num >= 0:  
      
    num = int (input('numero?'))
```

Calculando a média



```
qtde_num = 0  
soma = 0  
num = int (input('numero?'))
```

```
while num >= 0:  
    soma = soma + num  
    qtde_num = qtde_num + 1  
    num = int (input('numero?'))
```

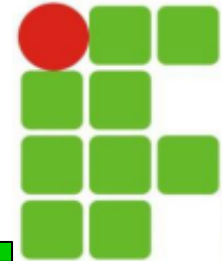
```
if qtde_num > 0:  
    print ('A media é:', soma/qtde_num)  
else:  
    print ('Média não é possível.')
```

Inicializações

Repetições/
iterações

Conclusões

Padrões de loops



- Implicitamente estudamos dois padrões de loops

```
num = int (input('numero?'))  
while num >= 0:  
    [redacted]  
    num = int (input('numero?'))
```

Loop indefinido:
o próximo número não
depende do anterior
(com uma expressão)

```
num = 1  
while num <= 100:  
    [redacted]  
    num = num + 1
```

Loop definido:
o próximo número
depende do anterior

Loops definidos



- Python simplifica por meio do comando **for**
 - Trabalha sobre um conjunto de dados
 - Em cada interação
 - Uma variável assume um valor no conjunto
 - Não necessita fazer inicialização / incrementos

```
num = 1
while num <= 100:
    [redacted]
    num = num + 1
```

```
for num in [redacted]:
    [redacted]
```

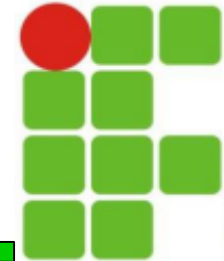
Aqui se informa
o conjunto
{ 1, 2, 3 ..., 100 }

Especificação de conjuntos



- Tecnicamente trata-se de um *iterator*
- O *iterator* mais simples é **range**
 - Gera um conjunto de números inteiros
 - Especifica-se início, fim, incremento
- Exemplos de *range*:
 - **range(100)** - gera números [0, 100 [ou [0, 99]
 - **range(60, 100)** - gera números [60, 100 [ou [60, 99]
 - **range(3, 11, 2)** - gera números (3, 5, 7, 9)

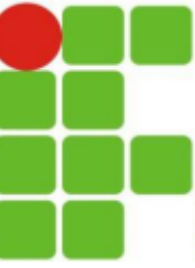
Exemplos de for



```
soma = 0
for x in range (101) :
    soma = soma + x
print ('soma =', soma)
```

```
x = int (input("Qual o valor de x: "))
ndiv = 0
for div in range (1, x+1) :
    if x % div == 0:
        ndiv = ndiv + 1
if ndiv == 2:
    print (x, "é primo.")
```

Qual o maior número ?



12 79 16 15 87 73 26 85 78 66 15 0

Qual? Como você descobriu?

12 79 16 15 87 73 26 85 78 66 15 0

Maior até agora:

87

E o código?



```
maior = -1
num = int (input('numero?'))

while num > 0:
    if num > maior:
        maior = num
    print ("Maior até agora", maior)
    num = int (input('numero?'))

if maior != -1:
    print ('A maior foi:', maior)
else:
    print ('Não houve números naturais.')
```

Inicializações

Repetições/
iterações

Conclusões