Notas de aula de Programação
Curso de Python
Encontro 3 - Estruturas de repetição

Prof. Louis Augusto
louis.augusto@ifsc.edu.br

Instituto Federal de Santa Catarina
Campus São José

Índice

Sumário

Estruturas de repetição

Estruturas de repetição permitem ao programador definir que uma determinada sequência de instruções deve ser executada de maneira repetitiva até certa condição ser satisfeita.

Há dois tipos de controle de repetição (também chamados de laços, ou loops):

Laços de controle finito Utiliza o comando for.

Laços de controle infinito Utiliza o comando while.

Estruturas de repetição

Estruturas de repetição permitem ao programador definir que uma determinada sequência de instruções deve ser executada de maneira repetitiva até certa condição ser satisfeita.

Há dois tipos de controle de repetição (também chamados de laços, ou loops):

Laços de controle finito. Utiliza o comando for.

```
PARA contador DE valor inicial ATÉ meta [PASSO incremento] FAÇA
    Instruções a serem executadas até o contador atingir a meta
FIM-PARA
```

Laços de controle infinito. Utiliza o comando while.

```
ENQUANTO condição FAÇA
    Instruções a serem executadas enquanto a condição for verdadeira
FIM-ENQUANTO
```

Estruturas de repetição

Estruturas de repetição permitem ao programador definir que uma determinada sequência de instruções deve ser executada de maneira repetitiva até certa condição ser satisfeita.

Há dois tipos de controle de repetição (também chamados de laços, ou loops):

Laços de controle finito Utiliza o comando for.

Lógica do controle

```
PARA contador DE valor inicial ATE meta [PASSO incremento ] FAÇA
    Instruções a serem executadas até o contador atingir a meta
FIM-PARA
```

Laços de controle infinito Utiliza o comando while.

Lógica do controle

```
ENQUANTO condição FAÇA
    Instruções a serem executadas enquanto a condição for verdadeira
FIM-ENQUANTO
```

Estruturas de repetição

Estruturas de repetição permitem ao programador definir que uma determinada sequência de instruções deve ser executada de maneira repetitiva até certa condição ser satisfeita.

Há dois tipos de controle de repetição (também chamados de laços, ou loops):

Laços de controle finito Utiliza o comando for.

Lógica do controle

PARA contador DE valor inicial ATE meta [PASSO incremento] FAÇA
    Instruções a serem executadas até o contador atingir a meta
FIM-PARA

Laços de controle infinito Utiliza o comando while.

Estruturas de repetição

Estruturas de repetição permitem ao programador definir que uma determinada sequência de instruções deve ser executada de maneira repetitiva até certa condição ser satisfeita.

Há dois tipos de controle de repetição (também chamados de laços, ou loops):

Laços de controle finito Utiliza o comando for.

Lógica do controle

PARA contador DE valor inicial ATE meta [PASSO incremento] FAÇA
    Instruções a serem executadas até o contador atingir a meta
FIM-PARA

Laços de controle infinito Utiliza o comando while.

Estruturas de repetição

Estruturas de repetição permitem ao programador definir que uma determinada sequência de instruções deve ser executada de maneira repetitiva até certa condição ser satisfeita.

Há dois tipos de controle de repetição (também chamados de laços, ou loops):

Laços de controle finito Utiliza o comando for.

Lógica do controle

```
PARA contador DE valor inicial ATE meta [PASSO incremento ] FAÇA
    Instruções a serem executadas até o contador atingir a meta
FIM-PARA
```

Laços de controle infinito Utiliza o comando while.

Lógica do controle

```
ENQUANTO condição FAÇA
    Instruções a serem executadas enquanto a condição retornar VERDADE
ELSE
    Instruções a serem executadas quando a condição retornar FALSO
```

Estruturas de repetição

Estruturas de repetição permitem ao programador definir que uma determinada sequência de instruções deve ser executada de maneira repetitiva até certa condição ser satisfeita.

Há dois tipos de controle de repetição (também chamados de laços, ou loops):

Laços de controle finito Utiliza o comando for.

Lógica do controle

PARA contador DE valor inicial ATE meta [PASSO incremento] FAÇA
    Instruções a serem executadas até o contador atingir a meta
FIM-PARA

Laços de controle infinito Utiliza o comando while.

Lógica do controle

ENQUANTO condição FAÇA
    Instruções a serem executadas enquanto a condição retornar VERDADE
ELSE
    Instruções a serem executadas quando a condição retornar FALSO

Sumário

Laço com controle finito

O comando for utiliza em sua forma mais simples um inteiro que serve de contador para a variação em uma quantidade de termos.

Usamos para isto:

```
for i in range(n):
    bloco de código.
```

Neste caso o contador inicia em `0` e termina em `n-1`.

Exemplo: Beecrowd Problema 1059

beecrowd | 1059

Números Pares

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Faça um programa que mostre os números pares entre 1 e 100, inclusive.

Entrada

Neste problema extremamente simples de repetição não há entrada.

Saída

Imprima todos os números pares entre 1 e 100, inclusive se for o caso, um em cada linha.

Exemplo de Entrada	Exemplo de Saída
	2 4 6 ... 100

Laço com controle finito

O comando for utiliza em sua forma mais simples um inteiro que serve de contador para a variação em uma quantidade de termos.

Usamos para isto:

```
for i in range(n):
    bloco de código.
```

Neste caso o contador inicia em `0` e termina em `n-1`.

Exemplo: Beecrowd Problema 1059

beecrowd | 1059

Números Pares

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Faça um programa que mostre os números pares entre 1 e 100, inclusive.

Entrada

Neste problema extremamente simples de repetição não há entrada.

Saída

Imprima todos os números pares entre 1 e 100, inclusive se for o caso, um em cada linha.

Exemplo de Entrada	Exemplo de Saída
	2 4 6 ... 100

Laço com controle finito

O comando for utiliza em sua forma mais simples um inteiro que serve de contador para a variação em uma quantidade de termos.

Usamos para isto:

```
for i in range(n):
    bloco de código.
```

Neste caso o contador inicia em `0` e termina em `n-1`.

Exemplo: Beecrowd Problema 1059

beecrowd | 1059

Números Pares

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Faça um programa que mostre os números pares entre 1 e 100, inclusive.

Entrada

Neste problema extremamente simples de repetição não há entrada.

Saída

Imprima todos os números pares entre 1 e 100, inclusive se for o caso, um em cada linha.

Exemplo de Entrada	Exemplo de Saída
	2 4 6 ... 100

Laço com controle finito

O comando for utiliza em sua forma mais simples um inteiro que serve de contador para a variação em uma quantidade de termos.

Usamos para isto:

```
for i in range(n):
    bloco de código.
```

Neste caso o contador inicia em `0` e termina em `n-1`.

Exemplo: Beecrowd Problema 1059

beecrowd | 1059

Números Pares

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Faça um programa que mostre os números pares entre 1 e 100, inclusive.

Entrada

Neste problema extremamente simples de repetição não há entrada.

Saída

Imprima todos os números pares entre 1 e 100, inclusive se for o caso, um em cada linha.

Exemplo de Entrada	Exemplo de Saída
	2 4 6 ... 100

Laço com controle finito

Todavia para o comando for temos algumas variantes.

Variável sem acesso: Podemos ao invés de definir uma variável explícita, como i em `for i in range(n)`: utilizar uma variável inacessível, representada por _.

Exemplo:

```
for _ in range (10):
    print("Teste_")
```

Isto gerará a impressão 10 vezes da string `Teste_`

Variável com valor inicial e final Se definirmos `for i in range(1,50)` teremos o primeiro valor de i como 1 e o último como 49.

Variável com valor inicial e final e passo Se definirmos `for i in range(2,101,2)` teremos o primeiro valor de i como 1 e o último como 100, variando de 2 em 2.

Em listas, o comando for é mais flexível, e permite caminhar na lista. Ao estudarmos listas, teremos este item retomado.

Laço com controle finito

Todavia para o comando for temos algumas variantes.

Variável sem acesso: Podemos ao invés de definir uma variável explícita, como i em `for i in range(n)`: utilizar uma variável inacessível, representada por _.

Exemplo:

```
for _ in range (10):
    print("Teste_")
```

Isto gerará a impressão 10 vezes da string `Teste_`

Variável com valor inicial e final Se definirmos `for i in range(1,50)` teremos o primeiro valor de i como 1 e o último como 49.

Variável com valor inicial e final e passo Se definirmos `for i in range(2,101,2)` teremos o primeiro valor de i como 1 e o último como 100, variando de 2 em 2.

Em listas, o comando for é mais flexível, e permite caminhar na lista. Ao estudarmos listas, teremos este item retomado.

Laço com controle finito

Todavia para o comando for temos algumas variantes.

Variável sem acesso: Podemos ao invés de definir uma variável explícita, como i em `for i in range(n):` utilizar uma variável inacessível, representada por _.
Exemplo:

```
for _ in range (10):
    print("Teste_")
```

Isto gerará a impressão 10 vezes da string `Teste_`

Variável com valor inicial e final Se definirmos `for i in range(1,50)` teremos o primeiro valor de i como 1 e o último como 49.

Variável com valor inicial e final e passo Se definirmos `for i in range(2,101,2)` teremos o primeiro valor de i como 1 e o último como 100, variando de 2 em 2.

Em listas, o comando for é mais flexível, e permite caminhar na lista. Ao estudarmos listas, teremos este item retomado.

Laço com controle finito

Todavia para o comando for temos algumas variantes.

Variável sem acesso: Podemos ao invés de definir uma variável explícita, como i em `for i in range(n):` utilizar uma variável inacessível, representada por _.
Exemplo:

```
for _ in range (10):
    print("Teste_")
```

Isto gerará a impressão 10 vezes da string `Teste_`

Variável com valor inicial e final Se definirmos `for i in range(1,50)` teremos o primeiro valor de i como 1 e o último como 49.

Variável com valor inicial e final e passo Se definirmos `for i in range(2,101,2)` teremos o primeiro valor de i como 1 e o último como 100, variando de 2 em 2.

Em listas, o comando for é mais flexível, e permite caminhar na lista. Ao estudarmos listas, teremos este item retomado.

Laço com controle finito

Todavia para o comando for temos algumas variantes.

Variável sem acesso: Podemos ao invés de definir uma variável explícita, como i em `for i in range(n):` utilizar uma variável inacessível, representada por _.
Exemplo:

```
for _ in range (10):
    print("Teste_")
```

Isto gerará a impressão 10 vezes da string `Teste_`

Variável com valor inicial e final Se definirmos `for i in range(1,50)` teremos o primeiro valor de i como 1 e o último como 49.

Variável com valor inicial e final e passo Se definirmos `for i in range(2,101,2)` teremos o primeiro valor de i como 1 e o último como 100, variando de 2 em 2.

Em listas, o comando for é mais flexível, e permite caminhar na lista. Ao estudarmos listas, teremos este item retomado.

Laço com controle finito

Todavia para o comando for temos algumas variantes.

Variável sem acesso: Podemos ao invés de definir uma variável explícita, como i em `for i in range(n):` utilizar uma variável inacessível, representada por _.
Exemplo:

```
for _ in range (10):
    print("Teste_")
```

Isto gerará a impressão 10 vezes da string `Teste_`

Variável com valor inicial e final Se definirmos `for i in range(1,50)` teremos o primeiro valor de i como 1 e o último como 49.

Variável com valor inicial e final e passo Se definirmos `for i in range(2,101,2)` teremos o primeiro valor de i como 1 e o último como 100, variando de 2 em 2.

Em listas, o comando for é mais flexível, e permite caminhar na lista. Ao estudarmos listas, teremos este item retomado.

Laço com controle finito

Todavia para o comando for temos algumas variantes.

Variável sem acesso: Podemos ao invés de definir uma variável explícita, como i em `for i in range(n):` utilizar uma variável inacessível, representada por _.
Exemplo:

```
for _ in range (10):
    print("Teste_")
```

Isto gerará a impressão 10 vezes da string `Teste_`

Variável com valor inicial e final Se definirmos `for i in range(1,50)` teremos o primeiro valor de i como 1 e o último como 49.

Variável com valor inicial e final e passo Se definirmos `for i in range(2,101,2)` teremos o primeiro valor de i como 1 e o último como 100, variando de 2 em 2.

Em listas, o comando for é mais flexível, e permite caminhar na lista. Ao estudarmos listas, teremos este item retomado.

Sumário

Laços com controle infinito

Conforme dito antes, o comando `while` serve principalmente para realizar um laço infinito, que é um laço que deve ser interrompido por uma condição.

Exemplo de aplicação: Leia uma certa quantidade de números e imprima a média dos mesmos, com duas casas decimais. O usuário irá informar 0 para finalizar a entrada.

```
#-*- coding: utf-8 -*-
soma = 0
qde = 0
n = float(input())#Leitura inicial do numero
while n!= 0:
        soma+=n
        qde = qde +1
        n = float(input())
else:
        print("Saída do laço com n=0.")
print('{:.2f}'.format(soma/qde))
```

Laços com controle infinito

Conforme dito antes, o comando `while` serve principalmente para realizar um laço infinito, que é um laço que deve ser interrompido por uma condição.

Exemplo de aplicação: Leia uma certa quantidade de números e imprima a média dos mesmos, com duas casas decimais. O usuário irá informar 0 para finalizar a entrada.

```
#-*- coding: utf-8 -*-
soma = 0
qde = 0
n = float(input())#Leitura inicial do numero
while n!= 0:
        soma+=n
        qde = qde +1
        n = float(input())
else:
        print("Saída do laço com n=0.")
print('{:.2f}'.format(soma/qde))
```

Laços com controle infinito

Conforme dito antes, o comando `while` serve principalmente para realizar um laço infinito, que é um laço que deve ser interrompido por uma condição.

Exemplo de aplicação: Leia uma certa quantidade de números e imprima a média dos mesmos, com duas casas decimais. O usuário irá informar 0 para finalizar a entrada.

```
#-*- coding: utf-8 -*-
soma = 0
qde = 0
n = float(input())#Leitura inicial do numero
while n!= 0:
        soma+=n
        qde = qde +1
        n = float(input())
else:
        print("Saída do laço com n=0.")
print('{:.2f}'.format(soma/qde))
```

Alternativas para uso de while

O uso normal do `while` é:

```
while(condição):
    bloco de código
```

Todavia há alternativas. Podemos usar uma condicional else:

```
while(condição):
    bloco de código
else:
    bloco de código
```

Exemplo: Entre com um número menor do que 10. Imprima a sequência de sucessores até 20 e no final imprima a quantidade de números entre o valor digitado e 20 utilizando while ... else.

Alternativas para uso de while

O uso normal do `while` é:

```
while(condição):
    bloco de código
```

Todavia há alternativas. Podemos usar uma condicional else:

```
while(condição):
    bloco de código
else:
    bloco de código
```

Exemplo: Entre com um número menor do que 10. Imprima a sequência de sucessores até 20 e no final imprima a quantidade de números entre o valor digitado e 20 utilizando while ... else.

Alternativas para uso de while

O uso normal do `while` é:

```
while(condição):
    bloco de código
```

Todavia há alternativas. Podemos usar uma condicional else:

```
while(condição):
    bloco de código
else:
    bloco de código
```

Exemplo: Entre com um número menor do que 10. Imprima a sequência de sucessores até 20 e no final imprima a quantidade de números entre o valor digitado e 20 utilizando while ... else.

Alternativas para uso de while

O uso normal do `while` é:

```
while(condição):
    bloco de código
```

Todavia há alternativas. Podemos usar uma condicional else:

```
while(condição):
    bloco de código
else:
    bloco de código
```

Exemplo: Entre com um número menor do que 10. Imprima a sequência de sucessores até 20 e no final imprima a quantidade de números entre o valor digitado e 20 utilizando while ... else.

Laços com controle infinito

O uso do while como um laço infinito é feito com:

```
while True:
    Bloco de codigo
    if (condição):
        break
```

Deve-se perceber que ao receber a condição True o laço será executado indefinidamente até que uma condição de parada seja chamada.

Como exemplo vamos refazer o problema 1059 do Beecrowd utilizando esta técnica.

```
#-*- coding: utf-8 -*-
m = 2
while True:
    print(m)
    m+=2
    if (m>100):
        break
```

Laços com controle infinito

O uso do while como um laço infinito é feito com:

```
while True:
    Bloco de codigo
    if (condição):
        break
```

Deve-se perceber que ao receber a condição True o laço será executado indefinidamente até que uma condição de parada seja chamada.

Como exemplo vamos refazer o problema 1059 do Beecrowd utilizando esta técnica.

```
#-*- coding: utf-8 -*-
m = 2
while True:
    print(m)
    m+=2
    if (m>100):
        break
```

Laços com controle infinito

O uso do while como um laço infinito é feito com:
while True:
    Bloco de codigo
    if (condição):
        break

Deve-se perceber que ao receber a condição True o laço será executado indefinidamente até que uma condição de parada seja chamada.

Como exemplo vamos refazer o problema 1059 do Beecrowd utilizando esta técnica.

```
#-*- coding: utf-8 -*-
m = 2
while True:
    print(m)
    m+=2
    if (m>100):
        break
```

Laços com controle infinito

O uso do while como um laço infinito é feito com:
while True:
    Bloco de codigo
    if (condição):
        break

Deve-se perceber que ao receber a condição True o laço será executado indefinidamente até que uma condição de parada seja chamada.

Como exemplo vamos refazer o problema 1059 do Beecrowd utilizando esta técnica.

```
#-*- coding: utf-8 -*-
m = 2
while True:
    print(m)
    m+=2
    if (m>100):
        break
```

Sumário

Comando for

beecrowd | 1157

Divisores I

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Ler um número inteiro N e calcular todos os seus divisores.

Entrada

O arquivo de entrada contém um valor inteiro.

Saída

Escreva todos os divisores positivos de N, um valor por linha.

Exemplo de Entrada	Exemplo de Saída
6	1 2 3 6

Sumário

Comando while

Estudar os problemas 1157, 1387, 1546

beecrowd | 1157

Divisores I

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Ler um número inteiro N e calcular todos os seus divisores.

Entrada

O arquivo de entrada contém um valor inteiro.

Saída

Escreva todos os divisores positivos de N, um valor por linha.

Exemplo de Entrada	Exemplo de Saída
6	1 2 3 6