

INF100 – Introdução à Programação I  
**Roteiro da Aula Prática 06 - 26 a 29 de outubro de 2020**  
**Comandos repetitivos while e for**  
**Valor: 2 pontos**

## Instruções

Antes de realizar esta prática é necessário assistir o vídeo sobre comando repetitivo for disponibilizado no PVANet e Google Classroom (Turma Teórica), cujo link é: [https://youtu.be/mIR\\_b-P3Otg](https://youtu.be/mIR_b-P3Otg).


Nesta prática SEMPRE que o número de repetições for conhecido, deverá ser utilizado o comando repetitivo *for*. Caso contrário, deve ser usado o comando repetitivo *while*.


Nome do arquivo a ser entregue: **p06.py**

**Importante:** Como qualquer outra prática de INF100 você deve:

1. Criar o cabeçalho obrigatório.
2. Após finalizar o cabeçalho salve o arquivo com o nome correto
3. Leia as instruções até o final e, após finalizar sua leitura, inicie sua programação.

**Obs.:** Recomenda-se salvar o arquivo com certa frequência para não perder a digitação já feita em caso de uma falha na rede elétrica.

 A saída do programa deve obedecer à formatação **exata** mostrada nos exemplos acima.

 Não esqueça de preencher o cabeçalho com seus dados e uma breve descrição do programa.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p06.py**) através do sistema do LBI.

## Questões a serem Resolvidas

- 1) Na matemática, a Sequência de Fibonacci, é uma sequência de números inteiros, começando por 0 e 1, na qual, cada termo subsequente corresponde à soma dos dois anteriores. A sequência recebeu o nome do matemático italiano Leonardo de Pisa, mais conhecido por Fibonacci, que descreveu, no ano de 1202, o crescimento de uma população de coelhos, a partir desta sequência.

Formalmente, a série de Fibonacci pode ser definida como:

$$\begin{aligned}F_0 &= 0 \\F_1 &= 1 \\F_n &= F_{n-1} + F_{n-2}, \text{ para } n > 1\end{aligned}$$

Ou seja, usando esta definição formal, podemos calcular a sequência de Fibonacci de qualquer posição, conhecendo apenas  $F_0$  e  $F_1$  que são os elementos base da série, como por exemplo, para definir o valor de  $F_4$  (o quinto elemento da série) basta usar a definição, expandindo os termos até que eles estejam expressos apenas em função de  $F_0$  e  $F_1$ , como mostrado a seguir:

$$F_4 = F_3 + F_2 = (F_2 + F_1) + (F_1 + F_0) = ((F_1 + F_0) + F_1) + (F_1 + F_0) = ((1+0)+1) + (1+0) = 3$$

Para calcular um elemento da série de Fibonacci usando programação, podemos usar o mesmo algoritmo já utilizado anteriormente: gerar os dois elementos base; usar a regra de geração do novo elemento, que consiste de somar os dois anteriores a ele, em um comando repetitivo para gerar os N-2 elementos seguintes.

Escreva um programa para calcular a soma dos N primeiros elementos da série de Fibonacci. O programa deverá, obrigatoriamente, atender a todos os requisitos abaixo:

- O número N de elementos deve ser determinado pelo usuário do programa;
- O valor de N deve ser maior que 2, ou seja, a menor série gerada pelo programa seria 1, 1, 2. Esta validação pode ser feita com *while*.
- O usuário pode optar por ver todos os elementos da série. O padrão será NÃO mostrar. Apenas de o usuário responder sim é que os elementos deverão ser mostrados.
- Não pode ser usado arranjo (array) e nem lista (list). Não há necessidade de armazenar os elementos da série.

A figura a seguir mostra dois exemplos de execução do programa.

```
>>> q1()
Este programa calcula a soma dos N primeiros termos da série de Fibonacci.

Informe o número de elementos (N >= 3): 2
N deve ser >= 3
Informe o número de elementos (N >= 3): 15
Deseja ver todos os elementos (S/N): s

Fibonacci: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
A soma dos 15 primeiros elementos da série de Fibonacci é: 986.
>>> q1()
Este programa calcula a soma dos N primeiros termos da série de Fibonacci.

Informe o número de elementos (N >= 3): 20
Deseja ver todos os elementos (S/N): nao

A soma dos 20 primeiros elementos da série de Fibonacci é: 10945.
>>>
```

2) Uma forma conhecida para o cálculo do valor da constante  $\pi$  foi desenvolvida por Nilakantha por volta de 1500. O cálculo é feito através da série abaixo:

$$\pi = 3 + \sum_{i=1}^{\infty} \frac{(-1)^{i+1} \times 4}{2i \times (2i+1) \times (2i+2)}$$

A série com 6 termos seria:

$$\pi = 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \frac{4}{8 \times 9 \times 10} + \frac{4}{10 \times 11 \times 12}$$

Faça um programa para calcular o valor de  $\pi$  usando o somatório dos N primeiros da série acima. Observe que o primeiro termo não tem a forma geral dos demais elementos da série. E representa o termo zero. Do termo 1 em diante, a expressão é dada pelo termo  $i$  do somatório (para  $i \geq 1$ ).

O programa deverá informar em uma tabela os valores parciais de  $\pi$  calculado com um elemento, dois elementos e assim por diante, até o valor final, com os N elementos da série solicitados pelo usuário. O objetivo é que o usuário do programa verifique como a precisão vai aumentando à medida que novos elementos vão sendo inseridos no somatório.

Para facilitar ainda mais esta percepção, o programa deverá informar o erro, considerando o valor conhecido de  $\pi$  com 8 casas decimais (3.14159265). Este valor está no arquivo p06.py atribuído para a variável `Plc8`.

O erro é definido como o módulo da diferença entre o valor calculado e o valor exato. Em Python, a função `abs()` retorna o módulo de um número. Ex:

```
x = -10
y = abs(x) # y receberá o valor 10
```

O programa deverá atender aos seguintes requisitos:

- O número N de elementos deve ser determinado pelo usuário do programa;
- O valor de N deve ser maior que 0, ou seja, a menor série gerada pelo programa apresentaria o valor de  $\pi$  igual a 3.
- Caso o valor informado para N seja inválido, o programa deve gerar uma mensagem de erro e solicitar um novo valor.
- Caso N seja válido, o programa deverá exibir os valores calculados para  $\pi$  e o erro de 1 a N elementos na série (veja exemplos de execução). Este trecho de código deverá ser OBRIGATORIAMENTE implementado com o uso do comando repetitivo `for`. Note que a forma de expressar estes elementos é em função de uma variável  $i$ . Aproveite esta característica para gerar todos os elementos que seguem esta regra de geração, usando a variável de controle do número de repetições do próprio comando `for`.
- Ao final da execução com exibição de valores, o usuário deverá ser perguntado se deseja repetir o processamento (veja exemplo de execução).
- A implementação não pode ser feita usando arranjo (array) e nem lista (list). Não há necessidade de armazenar os elementos da série.

Exemplo de execução.

```
>>> q2()
Cálculo de PI usando a série de Nilakantha

Informe o número de elementos da série (N >= 1): 0
N deve ser >= 1

Informe o número de elementos da série (N >= 1): 2

  N    PI calculado      Erro
  1    3.00000000      0.14159265
  2    3.16666667      0.02507402
Deseja executar novamente ? (S/N): s

Informe o número de elementos da série (N >= 1): 5


  N    PI calculado      Erro
  1    3.00000000      0.14159265
  2    3.16666667      0.02507402
  3    3.13333333      0.00825932
  4    3.14523810      0.00364545
  5    3.13968254      0.00191011
Deseja executar novamente ? (S/N): N
>>>
```

Outro exemplo:

```
>>> q2()
Cálculo de PI usando a série de Nilakantha

Informe o número de elementos da série (N >= 1): 10

  N    PI calculado      Erro
  1    3.00000000      0.14159265
  2    3.16666667      0.02507402
  3    3.13333333      0.00825932
  4    3.14523810      0.00364545
  5    3.13968254      0.00191011
  6    3.14271284      0.00112019
  7    3.14088134      0.00071131
  8    3.14207182      0.00047917
  9    3.14125482      0.00033783
 10    3.14183962      0.00024697
Deseja executar novamente ? (S/N): não
>>>
```

 A saída do seu programa (em ambas as questões) deve obedecer à formatação **exata** mostrada nos exemplos.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p06.py**) através do sistema do LBI.

**A entrega deverá ser feita até às 23h59 do dia 29/10/2020 (5ª. feira)**