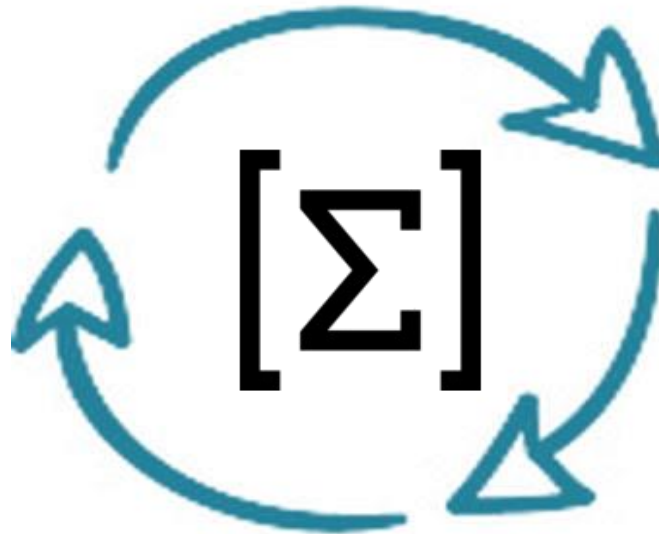




Escola de Engenharia Linguagem de Programação



LAB 8: Tarefa Mínima e Complementar



Somatória e Somatória com troca de sinal

Prof. Bira Carnevale

Se prepare para os desafios deste LAB !

- Vamos treinar o uso da somatória (comando for, sequência, troca de sinal, etc).
- Para a somatória, sugiro que vcs abram a apresentação da aula de Teoria e consultem os slides 46, 47, 48, 50 e 51.





Tarefa Mínima

(Lembre-se: codificação em Python)

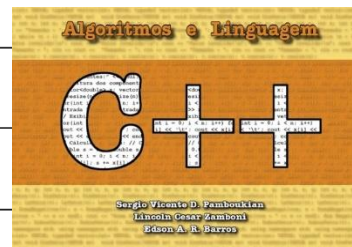
Entrega hoje até às 23h59

1) Elaborar um programa para calcular a soma abaixo que representa, matematicamente, o cálculo do valor de π .

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \dots$$

Neste exercício, a expressão calcula o valor de π , por isso use como acumulador “P”

9.27	Dado: $n = 1000$	Resposta: $\pi = 3.14059$
	Dado: $n = 100000$	Resposta: $\pi = 3.14158$



2) Elaborar um programa para calcular a soma das n primeiras parcelas da sequência abaixo. Fazer a validação de N:



$$S = e^x + \pi + 2^x + 3^x + 4^x + \dots$$

<https://cdn-icons-png.flaticon.com/512/5090/5090084.png>

8.24	Dados: $n = 5$ e $x = 2.5$	Resposta: $S = 68.5694$
	Dado: $n = 1$	Resposta: Valor de n é inválido.



3) Elaborar um programa para calcular o valor do seno de um valor x em radianos(*), usando a seguinte série:

$$\text{sen}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

<https://engenharia360.com/wp-content/uploads/2021/04/image-8.png>

- Obs: Nas séries de potência, quanto maior a quantidade de parcelas (N) utilizadas no cálculo, maior será a precisão do resultado.



Use a calculadora para testar este programa.

() Para 30° digite o corresponde em radianos, ou seja, 0.5235987 e o resultado de seu seno será próximo a 0.5 (Lembre-se que o seno de 30° é 0,5)*

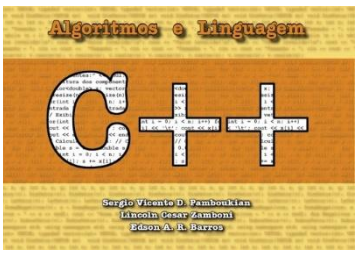
4) Elaborar um programa para calcular a soma abaixo que representa, matematicamente, o cálculo do logaritmo decimal de x. Leia as instruções abaixo. Essa sequência é também conhecida como série.

8.25 O valor do logaritmo de x na base e pode ser calculado através da série a seguir. Neste tipo de série, quanto maior a quantidade de parcelas, maior é a precisão do cálculo. Dados os valores de n e x , elaborar um programa para calcular e exibir o valor do logaritmo de x utilizando as n primeiras parcelas desta série. O programa deve exibir uma mensagem de erro se o valor de x fornecido pelo usuário não for positivo. O programa também deve comparar o resultado obtido pela série com o valor obtido através da função `log` declarada no arquivo de cabeçalhos `cmath`.

Neste exercício, a expressão calcula o logaritmo, por isso use como acumulador “L” e não `log(x)`.

$$\log(x) = 2 \cdot \left\{ \frac{x-1}{x+1} + \frac{1}{3} \left(\frac{x-1}{x+1} \right)^3 + \frac{1}{5} \left(\frac{x-1}{x+1} \right)^5 + \dots \right\}$$

8.25	Dados: $x = 2$ e $n = 5$	Resposta: $\log(x)$ usando a serie = 0.6931460474 $\log(x)$ usando <code>cmath</code> = 0.6931471806
	Dados: $x = 2$ e $n = 100$	Resposta: $\log(x)$ usando a serie = 0.6931471806 $\log(x)$ usando <code>cmath</code> = 0.6931471806
	Dado: $x = 0$	Resposta: Valor de x inválido
	Dado: $x = -2$	Resposta: Valor de x inválido






Tarefa Complementar

(entregar em uma semana)

1) Elaborar um programa para calcular o logaritmo de $(1+x)$ a partir da soma das N primeiras parcelas da série:

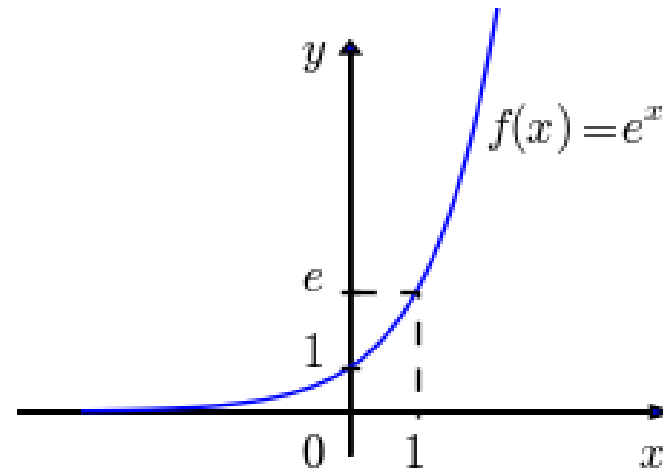
$$\log(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots \quad \text{para } -1 < x \leq 1$$

- Obs: Desta vez, temos uma restrição para o valor de x . Fazer a validação da digitação de x .

9.35	Dados: $x = 0.5$ e $n = 100$	Resposta: $\log(1+x) = 0.405465$	
	Dados: $x = 1$ e $n = 100000$	Resposta: $\log(1+x) = 0.693142$	
	Dado: $x = 2$	Resposta: Valor de x inválido	
	Dado: $x = -1$	Resposta: Valor de x inválido	

2) Elaborar um programa para calcular o valor do exponencial de x a partir da soma das N primeiras parcelas da série:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

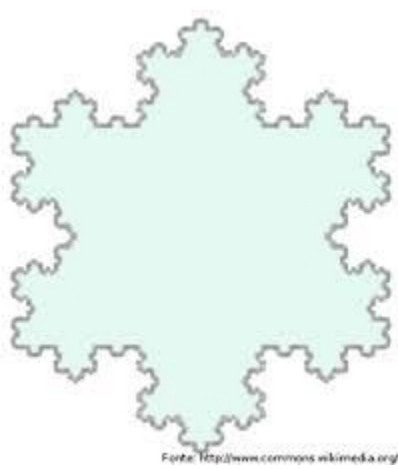


Use a calculadora para teste este programa.

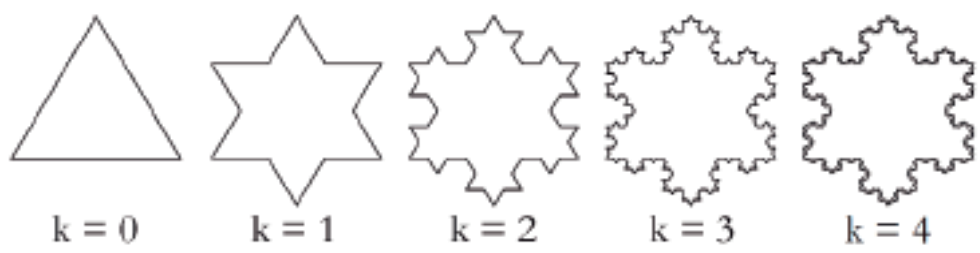
3)

8.30

A curva “Floco de Neve” é atribuída ao matemático sueco Niels Fabian Helge von Koch (1870 - 1924). A curva, ilustrada na figura a seguir, pode ser obtida com o seguinte algoritmo: a partir de um triângulo equilátero ($k = 0$) de lado L , dividimos cada um dos seus lados em três partes iguais e substituímos a parte central por dois segmentos de mesmo tamanho que a parte central, formando um novo polígono ($k = 1$); aos lados desse novo polígono ($k = 1$) é reaplicada a divisão em três partes iguais e a substituição da parte central por dois segmentos de mesmo tamanho que a parte central, formando outro novo polígono ($k = 2$); tal processo é repetido indefinidamente ($k = 3, 4, \dots$).



Fonte: <http://www.commonswiki.org>

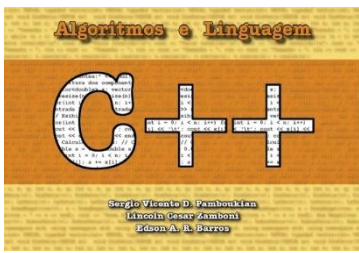


A área do “Floco de Neve” de ordem k é calculada pelas fórmulas a seguir.

$$A_k = A_0 \cdot \left(1 + \frac{s}{3}\right) \quad A_0 = L^2 \cdot \frac{\sqrt{3}}{4}$$
$$s = 1 + \frac{4}{9} + \left(\frac{4}{9}\right)^2 + \left(\frac{4}{9}\right)^3 + \dots + \left(\frac{4}{9}\right)^{k-1}$$

Dados: $L = 2$ e $k = 10$ Resposta: $A_k = 2.77097$

Elaborar um programa para calcular e exibir a área A_k do floco de neve de ordem k , a partir do lado L . O usuário deve entrar com os valores de k e L e o programa deve exibir o valor da área A_k .



Material para consulta deste LAB



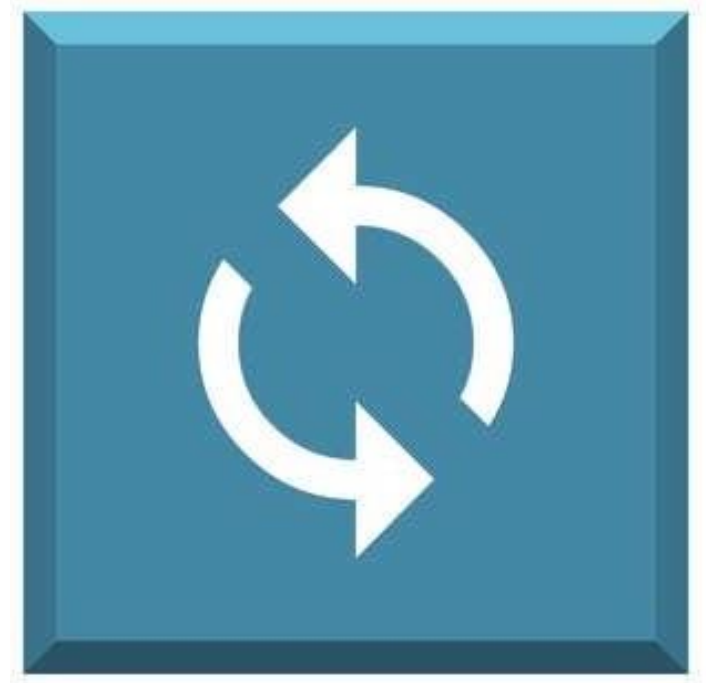
Biblioteca MATH

<u>math.função</u>	<u>Significado</u>
<u>cos(x)</u>	Cosseno, x em radianos
e	A constante e (Número de Euler= 2.718281828459045235360287)
<u>exp(x)</u>	e^{**x} , ou seja, função <u>e^x</u>
<u>factorial(x)</u>	Fatorial de x de tipo <u>int</u> , resultado <u>int</u>
<u>fabs(x)</u>	Valor absoluto de x , ou seja, $ x $
<u>log(x)</u>	Logaritmo neperiano de x, ou seja, <u>$\ln x$</u> ou <u>$\log_e x$</u>
<u>log10()</u>	Logaritmo na base 10 (decimal)
<u>log2()</u>	Logaritmo na base 2
pi	A constante pi (π)
<u>pow (b,ex)</u>	Potenciação: b=base, <u>ez</u> =expoente ou seja, (<u>b^{ex}</u>)
<u>radians(x)</u>	Converte x de graus para radianos <u>degrees (x)</u> , o inverso
<u>sin(x)</u>	Seno, x em radianos
<u>sqrt()</u>	Raiz quadrada
<u>tan(x)</u>	Tangente, x em radianos
<u>acos (x)</u> , <u>asin(x)</u> , <u>atan(x)</u>	Arco seno, arco cosseno e arco tangente de x (resultado em radianos)

	30°	45°	60°
Seno	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$
Cosseno	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$
Tangente	$\frac{\sqrt{3}}{3}$	1	$\sqrt{3}$

Comandos de Looping

- **while ()**
 - ✓ Validação
- **for ()**
 - ✓ Intervalo numérico
 - ✓ Repetição de tarefas
 - ✓ Sequências
 - ✓ Somatórias

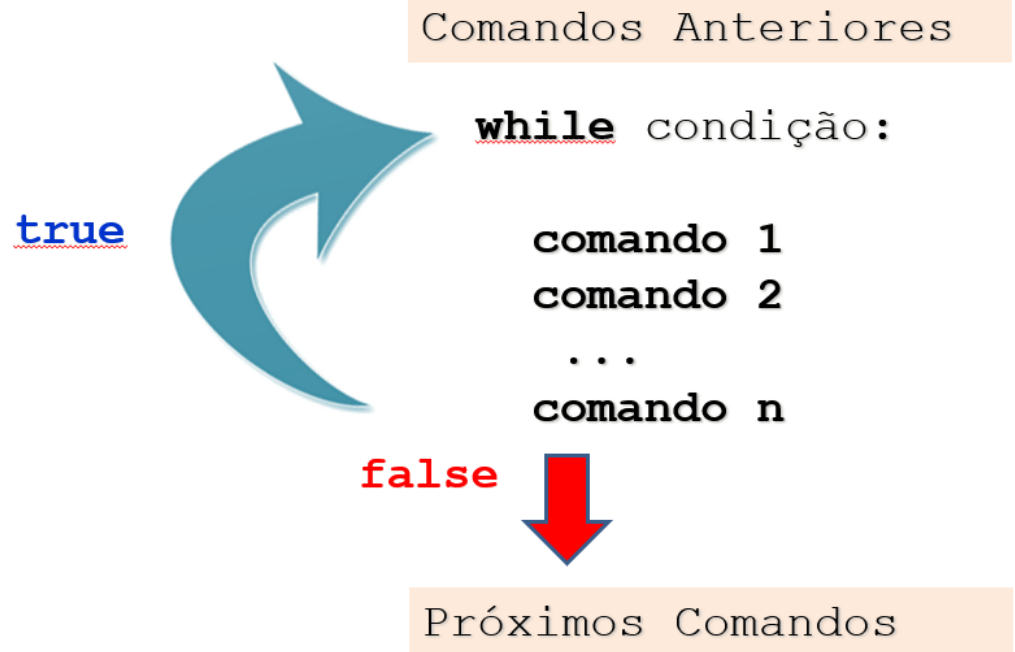


Com Repetição (Loop)

Elaborar um programa para exibir os **números inteiros entre 1 e 15**. Depois, exibir a palavra “FIM”

```
n=1
while n<=15:
    print(n)
    n=n+1
print("FIM")
```

Sintaxe do comando WHILE



Aplicação do Looping while

Consistência de Dados (Validação) com mensagem de erro !!



<https://img.freepik.com/>

```
import math as m
```

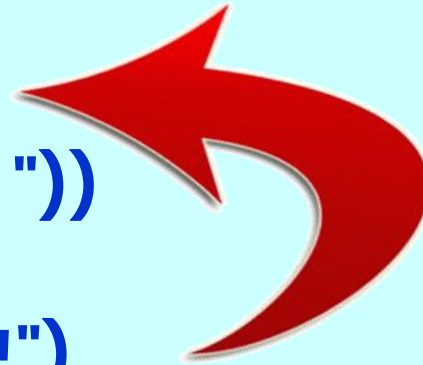
```
R=0
```

```
while R<=0:
```

```
    R= float (input ("Raio: "))
```

```
    if R<=0:
```

```
        print(" Raio inválido!")
```



Versão 1

```
h= float (input ("Altura: "))
```

```
while h<=0:
```

```
    print(" Altura inválida!")
```

```
    h= float (input ("Altura: "))
```



Versão 2

```
S= 2*m.pi*R*(R+h)
```

```
V= m.pi*m.pow(R,2)*h
```

```
print(" Área Total= ", S)
```

```
print(" Volume= ", V)
```

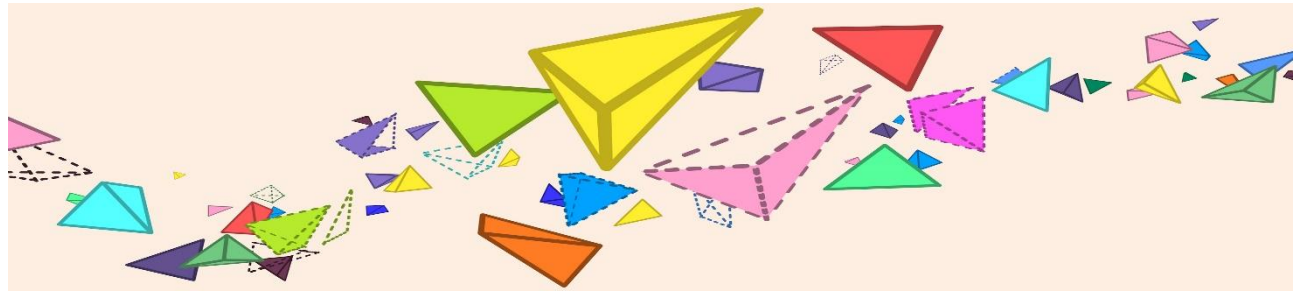
Material para consulta



Aplicações do comando for ()

1) **Gerar intervalos de números:** 1, 2, 3, 4 ou
20, 22, 24, 26 ou 0, -1, -2

2) **Repetição de tarefas:** calcular a área de 20 triângulos



3) **Exibir uma sequência com n elementos:** 3, 6, 9,...

4) **Calcular uma somatória:** $3 + 6 + 9 + 12 + \dots$

Sintaxe do comando for

```
for variável in range(valor inicial, limite final) :
```

```
comando 1
```

```
comando 2
```

```
...
```

```
comando n
```

```
comando n+1
```

Inicia nesse
valor

Vai até o
valor
anterior

Após executar o
for, continua
para as linhas
seguintes a ele



Sintaxe do comando for ()

Inicia nesse
valor

Vai até o valor
anterior

Razão. Quando
ausente, o default
é 1 (padrão)

```
for variável  
in range(valor inicial, limite final, incremento) :
```

```
comando 1
```

```
comando 2
```

```
...
```

```
comando n
```

Após executar o
for, continua
para as linhas
seguintes a ele



```
1 # Área de N Triângulos
2 # Validação de N
3 N=-1
4 ▼ while N<=0:
5     N = int (input("Digite o número de triângulos:"))
6 ▼     if N<=0:
7         print ("Valor de N inválido\n")
8
9 ▼ for i in range (1, N+1):
10     b=-1
11 ▼     while b<=0:
12         b= float (input ("Base: "))
13 ▼         if b<=0:
14             print ("Valor da base incorreto\n")
15     h=-1
16 ▼     while h<=0:
17         h= float (input ("Altura: "))
18 ▼         if h<=0:
19             print ("Valor da altura incorreto")
20     A= b*h/2
21     print(" Área do Triângulo= ", A, "\n")
22 print ("Fim")
```

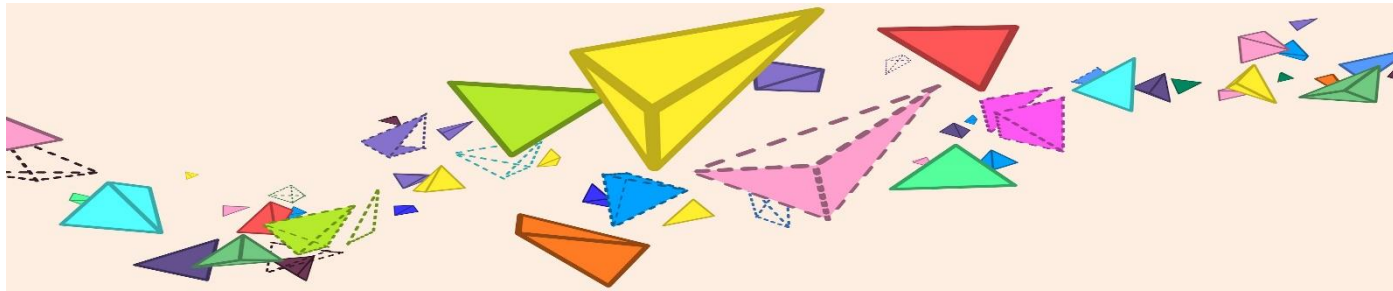

Observe na codificação anterior ...

- 1 Iniciar a variável com um valor errado e o while
- 2 Fazer a entrada para o valor de N
- 3 Exibir a mensagem de erro com o if
- 4 Abrir o “for” para a repetição de N vezes
- 5 Fazer a validação com while (condição de erro)
- 6 Mais de um campo de entrada, repetir o while
- 7 Depois da validação dos campos, calcular.
- 8 Após o “for” continuar o programa ou finalizar.

Aplicações do comando for ()

1) **Gerar intervalos de números:** 1, 2, 3, 4 ou
20, 22, 24, 26 ou 0, -1, -2

2) **Repetição de tarefas:** calcular a área de 20 triângulos



3) **Exibir uma sequência com n elementos:** 3, 6, 9,...

4) **Calcular uma somatória:** $3 + 6 + 9 + 12 + \dots$

Criar uma sequencia com N elementos:



- Obs.: não sabemos o último valor. Mas sabemos o valor de N e também podemos identificar a razão da sequência (cinco).

Ex1: Exibir na tela a sequência 3 5 7 9 11... para N elementos

Neste caso, precisaremos verificar o **termo geral** da sequência.

Observar atentamente a sequência.

Veja qual é a **razão entre os elementos**. A partir da razão conseguiremos definir o termo geral que neste caso é:

$$2 * K + 1$$

Mas de onde tiramos esses valores?

A sequência **3 5 7 9 11...** para **n elementos**

Termo Geral: **$2*K+1$**

O termo geral em azul foi obtido da seguinte forma:

Variável usada no comando “for” (K) multiplicada pela razão da sequência (2) +/- ajustes (1)

- Realmente a razão é 2 que multiplicamos pela variável que será usada no “for”, ou seja $2*K$, porém $2*k$, com o K iniciando de 1 (1º valor) resulta em $2*1=2$. Porém o primeiro número é 3. Então é necessário somar 1 nos ajustes para se chegar ao 3.
- É possível testar, substituindo o K pelos valores que ele ira assumir (1, 2, 3, ...n).... Os valores exibidos na tela serão $2*1+1$, $2*2+1$, $2*3+1$, etc, ou seja, 3, 5, 7 e assim por diante.

Exibir na tela a sequência **3 5 7 9 11...** para **n** elementos.

Usaremos então o Looping “for”. Assim:

```
for K in range (1, N+1):  
    print (2*K+1)
```


Termo Geral





Codificação completa com a digitação validada de N:

```
N=-1
while N<=0:
    N = int(input("Digite o valor de N:"))
    if N<=0:
        print("Valor de N inválido")
for K in range(1, N+1):
    print(2*K+1)
```

A sequência **3 5 7 9 11...** para **n elementos** será exibida na tela:

```
Digite o valor de N:5
3
5
7
9
11
> □
```

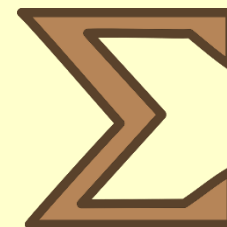

Exibir resultados na mesma linha com o "end"

```
1  # Sequência 1, 5, 9, 13, ...
2
3  import math
4  N=-1
5  ▼ while N<=1:
6      N = int (input("Digite o valor de N:"))
7  ▼   if N<=1:
8      print ("\033[0;31;40mValor de N inválido\033[m")
9
10 ▼ for K in range (1, N+1):
11 ▼     if K<N:
12         print (4*K-3,end=",")
13 ▼     else:
14         print (4*K-3)
```

```
Digite o valor de N:4
1,5,9,13
> 
```

Lembrar das regras para a Somatória:

- 1) Criar um acumulador (variável onde ficarão as somas e, no final, o resultado);
- 2) Zerar o acumulador (atribuir à essa variável, o valor zero);
- 3) Dentro do “for”, somar as parcelas, utilizando o Termo Geral e armazenar o cálculo no acumulador;
- 4) Fora do “for”, exibir o resultado que estará no acumulador.



Lembrar das regras para Troca de Sinal:

- Criar uma variável para o sinal (C);
- Colocar nessa variável, o valor 1 (se a primeira parcela for positiva);
Colocar nessa variável, o valor -1 (se a primeira parcela for negativa);
- Dentro do “for” trocamos o sinal, multiplicando o termo geral por C e fazendo o $C = -C$;
- Com isso, colocamos as chaves no “for”, já que são duas ações (calcular S e trocar o sinal).



Codificação:

```
1  # Somatória para calcular o cosseno
2  import math as m 1
3  # Digitar o valor de N com validação
4  N=0
5  while N<=1:
6      N= int(input("Digite o número de parcelas: "))
7  if N<=1: 2
8      print ("Número de parcelas inválido\n")
9  # Digitar o valor de x 3
10 X= float(input("X em graus= "))
11 R= m.radians(X) 4
12 S=0; C=1 5
13 for K in range (1, N+1): 6
14     S=S+m.pow(R, 2*K-2)/m.factorial(2*K-2) *C 7
15     C=-C 8
16 print (" Cosseno de ", X, " graus vale=", S) 9
```



Universidade Presbiteriana Mackenzie

Escola de Engenharia



Ótima semana !!