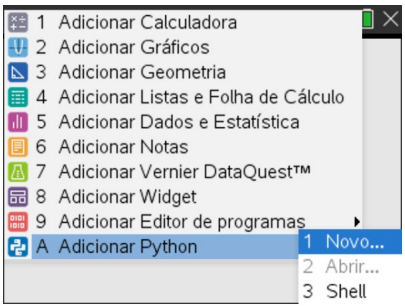
```
def integral_aproximado(f,a,b,N):
  dt=(b-a)/N
  soma=0
  for k in range(N):
    soma+=f(a+k*dt+dt/2)
  return dt*soma

def f(x):
  import math
  return math.exp(x)/x

a=1
b=5
N=20
for d in range(10):
  N=2*N
  I=integral_aproximado(f,a,b,N)
  print('Com N=',N,' o integral aproximado dá: ',I)
```

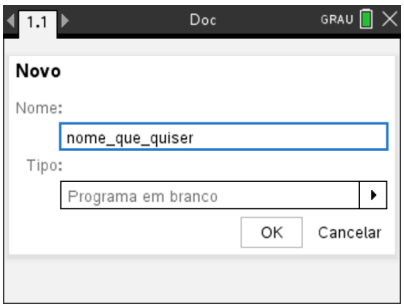
Editar Python na TI-nspire CX II-T

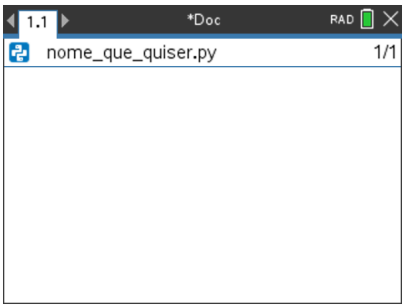
Ligue a sua calculadora e crie um novo documento.
Escolha uma página de *Python*:

menu A **Adicionar Python** → 1 **Novo.**

Coloque um nome à sua escolha, de seguida, prime em **OK**.
Abre-se uma página vazia, que é o editor de *Python* da calculadora/tecnologia TI-Nspire CX II-T, onde deve escrever o código.







1. Dada uma função, como aproximar o valor do integral definido num intervalo $[a, b]$?

Para calcular a aproximação do integral de uma função num intervalo pretendido, um dos métodos numéricos usados é o que aproxima o valor de um integral por áreas de retângulos, a Regra do Ponto Médio, que segue o seguinte algoritmo:

(1) Recebe a função **f**, os extremos do intervalo pretendido, **a** e **b**, e o número **N** de vezes que se pretende dividir o intervalo inicial em intervalos adjacentes com a mesma amplitude;
(2) Divide o intervalo inicial em **N** subintervalos iguais e adjacentes, com a mesma amplitude, **(b-a)/N=dt**;
(3) Em cada subintervalo, calcula o valor central ("ponto médio");
(4) Calcula a imagem, pela função, de cada um desses valores centrais, obtidos anteriormente.
(5) Calcula a área de cada retângulo multiplicando o valor obtido em **(4)** pela largura do subintervalo, **dt**.
(6) A soma dos valores obtidos em **(5)** , que calcularás de forma aditiva. é um valor aproximado do integral em questão, o qual será imprimido.

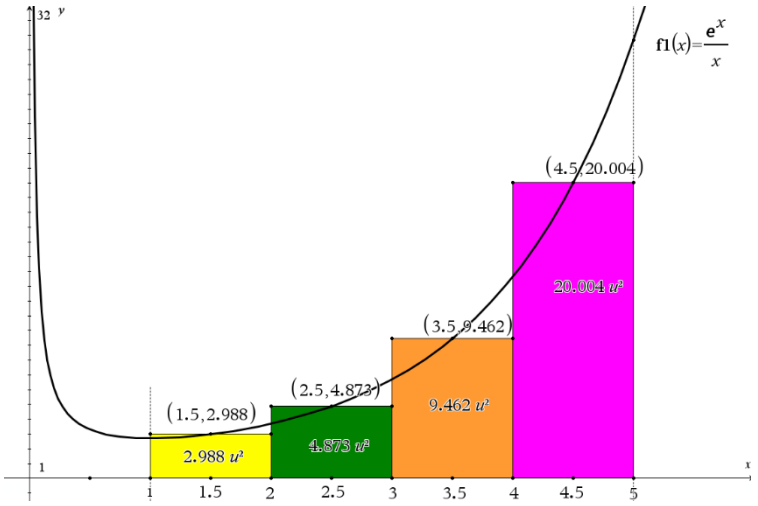
Obviamente, que à medida que se aumenta o número de subintervalos (**N**), melhor será a aproximação obtida. Por exemplo, ao duplicar sucessivamente o número de subintervalos (que é o que se sucede no programa *Python* apresentado), a largura de cada divisão diminui, permitindo obter aproximações cada vez melhores do integral definido.

Antes de se apresentar a construção do programa em linguagem *Python*, observe-se uma ideia gráfica do programa, que favorece a compreensão do mesmo.

No programa apresentado nas Aprendizagens Essenciais, o integral a ser aproximado, **I**, $\int_1^5 \frac{e^x}{x}\,dx$. Com isto, pretende-se calcular a área sob o gráfico da função **f**, definida por $f(x) = \frac{e^x}{x}$, no 1º quadrante, entre **a** e **b**, ou seja, em $[1,5]$.

Para o exemplo, considere-se que o integral a calcular é $\int_1^5 \frac{e^x}{x}\,dx$, com $N = 5$:

Assim, $f(x) = \frac{e^x}{x}$, $a = 1$ e $b = 5$ e $N = 4$:



(1) O intervalo $[1,5]$ é divido em 4 subintervalos, cada um com um comprimento $dt = \frac{5-1}{4} = \frac{4}{4} = 1$;

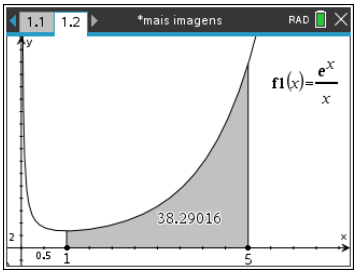
(2) Em cada intervalo, é calculado o seu valor central;

(3) Em cada ponto médio, é calculado o valor na função;

(4) Para cada valor calculado no passo anterior, multiplica-se por dt, de modo a calcular a área de cada um dos retângulos;

(5) Adicionam-se as medidas das áreas obtidas anteriormente e obtém-se, assim, um valor aproximado do integral em questão:

$$I \approx 2{,}988 + 4{,}873 + 9{,}462 + 20{,}004 = 37{,}327$$



De facto, o valor do integral em questão é: $\int_1^5 \frac{e^x}{x}\,dx \approx 38{,}2902$.

Uma melhor aproximação ocorre quando se aumenta o número de subintervalos para, por exemplo, o dobro:

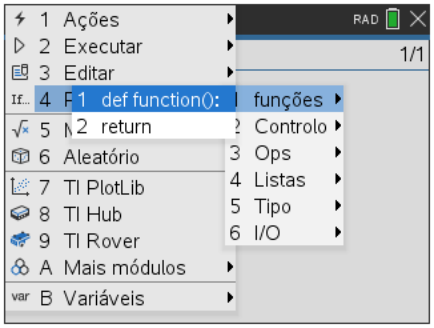
$$N = 8 \rightarrow dt = \frac{5-1}{8} = \frac{4}{8} = 0.5$$

Intervalo	**Ponto médio**	**Função** f (3 c.d.)	**Área do retângulo**
$[1;1{,}5]$	$1{,}25$	$2{,}792$	$2{,}792 \times dt = 2{,}792 \times 0{,}5 = 1{,}396$
$[1{,}5;2]$	$1{,}75$	$3{,}288$	$3{,}288 \times dt = 3{,}288 \times 0{,}5 = 1{,}644$
$[2;2{,}5]$	$2{,}25$	$4{,}217$	$4{,}217 \times dt = 4{,}217 \times 0{,}5 = 2{,}109$
$[2{,}5;3]$	$2{,}75$	$5{,}688$	$5{,}688 \times dt = 5{,}688 \times 0{,}5 = 2.844$
$[3;3{,}5]$	$3{,}25$	$7{,}935$	$7{,}935 \times dt = 7{,}935 \times 0{,}5 = 3{,}968$
$[3{,}5;4]$	$3{,}75$	$11{,}339$	$11{,}339 \times dt = 11{,}339 \times 0{,}5 = 5{,}670$
$[4;4{,}5]$	$4{,}25$	$16{,}495$	$16{,}495 \times dt = 16{,}495 \times 0{,}5 = 8{,}243$
$[4{,}5;5]$	$4{,}75$	$24{,}334$	$24{,}334 \times dt = 24{,}334 \times 0{,}5 = 12{,}167$
Valor aproximado do integral			$38{,}039$

I. Para se conseguir obter um valor aproximado de um integral definido para qualquer função e em qualquer intervalo, pode definir-se uma função em *Python*, designando-se, por exemplo, por **`integral_aproximado`**.
Esta função irá receber quatro parâmetros:

- **`f`** – a função que se pretende integrar;
- **`a`** – limite inferior do intervalo;
- **`b`** – limite superior do intervalo;
- **`N`** – número de subintervalos adjacentes e de igual amplitude em que será dividido o intervalo **`[a,b]`**.

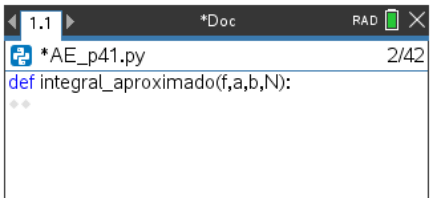
Esta "função" é semelhante ao que noutras linguagens de programação se chama de subrotina, ou seja, uma secção de código que é ativada a partir de linhas de código subsequentes. Esta estrutura é bastante útil, por exemplo, quando no programa se precisa de fazer algumas vezes as ações que nela estão previstas



Para obter esta estrutura, caso conheça a sintaxe, poderá escrever com o teclado, mas pode também obter através do menu:

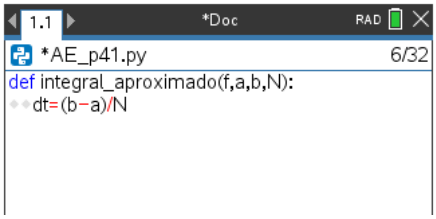
menu 4 **Planos integrados** → 1 **funções** → 1 **def function():**

Obtida a estrutura, é necessário preencher os campos em falta. A função, chamada de **integral_aproximado**, é uma função que recebe quatro argumentos:



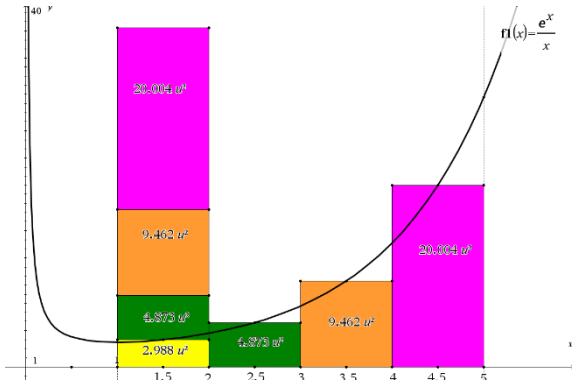

```
def integral_aproximado (f,a,b,N):
    bloco
```

II. Perante o intervalo recebido, **`[a,b]`**, este é dividido em **`N`** subintervalos com a mesma amplitude.
Cada subintervalo possui a amplitude, designada por **`dt`**: $dt = \frac{b-a}{N}$
Assim, é necessário, acrescentar a seguinte linha de código:

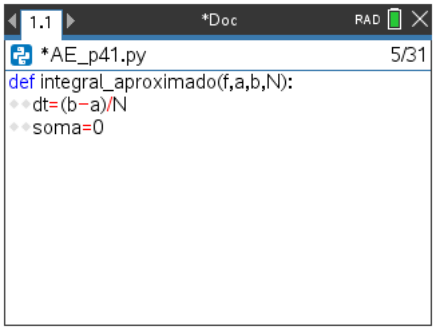



```
dt=(b-a)/N
```

Anteriormente, foi referido que as medidas das áreas dos retângulos são adicionadas no final. No entanto, uma vez que todos os retângulos têm a mesma "base", de comprimento (dt), não é necessário calcular separadamente a área de cada um.
Assim, o programa começa por adicionar as "alturas" dos retângulos, que correspondem aos valores da função nos valores centrais de cada subintervalo. No fim, essa soma é multiplicada por (dt), obtendo-se, desta forma, a aproximação do integral definido.



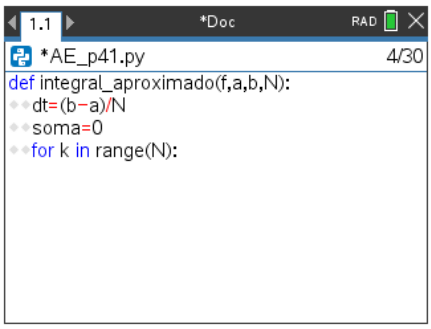
III. Deste modo, é criada uma variável designada por **soma**, que irá armazenar a soma dos valores da função avaliados nos valores centrais de cada subintervalo (ou seja, onde são acumuladas as alturas dos retângulos).
Inicialmente a variável soma começa em zero para que ocorra um processo aditivo recorrente.




```
soma=0
```

IV. Inicialmente, o intervalo **[a,b]** foi dividido em **N** subintervalos com igual amplitude.

Perante isto, é necessário calcular o valor da função no ponto médio de cada um desses subintervalos. Para evitar escrever o mesmo conjunto de instruções N vezes, utiliza-se um ciclo de repetição **for**:



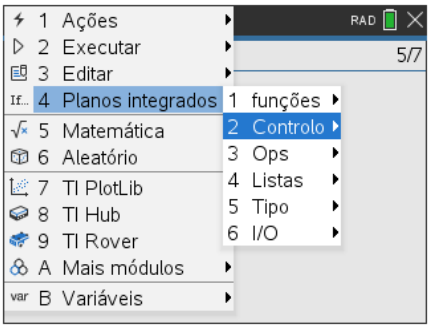
for k in range(N):

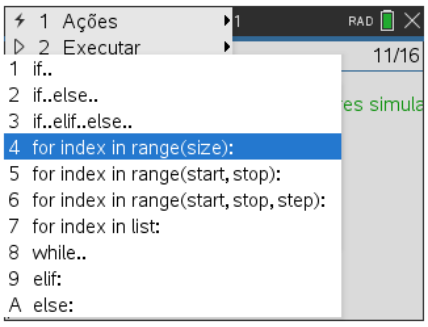
◆◆ **bloco**

onde k irá identificar uma espécie de ordem do subintervalo, iniciada em 0, que está a ser analisado, assumindo sucessivamente os valores $0,1,2,\ldots,N-1$

Para obter esta linha de código pode-se utilizar o teclado. Contudo, pode-se também recorrer ao menu para obter as linhas de código, a completar com os elementos específicos.

menu 4 Planos integrados → 2 Controlo → 4 for index in range(size):





V. Para se determinar o valor central de cada um dos **N** subintervalos em que o intervalo $[a,b]$ foi dividido, utiliza-se a seguinte expressão:

$$a + k \times dt + \frac{dt}{2}$$

Isto é, para se determinar o valor central do $(k+1)$-ésimo subintervalo, "parte-se" do extremo inferior do intervalo, a, e adiciona-se metade da amplitude dos subintervalos para obter o valor central do primeiro. A partir dai, obtêm-se os restantes valores centrais adicionando-se sucessivamente a amplitude de cada subintervalo (dt), $k-1$ vezes, até ao valor central do intervalo $[b-dt,b]$.

No final do ciclo, a variável **soma** contém

$$f(1{,}5) + f(2{,}5) + f(3{,}5) + f(4{,}5)$$

isto é, a soma dos valores da função em todos os pontos médios.

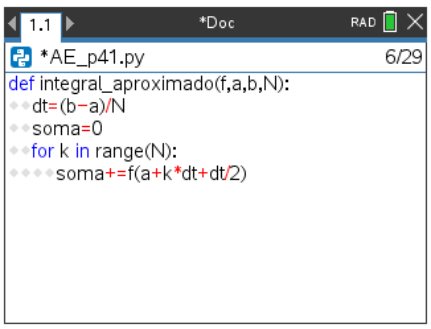
Assim, a linha de código que permitirá obter a soma dos valores da função nos valores centrais referidos é **soma = soma + f(a+k*dt+dt/2).**

Em Python, esta soma pode ser escrita recorrendo ao operador de atribuição com adição (**+=**), que permite, em cada passagem de ciclo, adicionar um novo valor ao conteúdo atual da variável **soma**, substituindo o seu valor pelo resultado dessa adição:

soma += f(a+k*dt+dt/2)

No exemplo anterior:

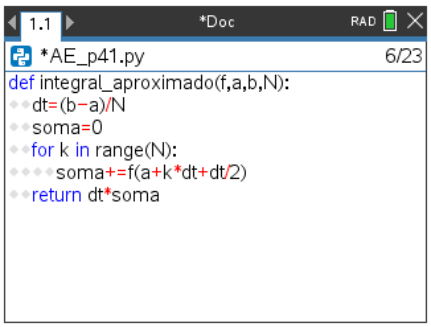
- Se $k=0$, então o ponto médio a considerar é
$$1 + 0 \times 1 + \frac{1}{2} = \frac{3}{2} = 1{,}5$$
- Se $k=1$, então o ponto médio a considerar é
$$1 + 1 \times 1 + \frac{1}{2} = \frac{5}{2} = 2{,}5$$
...
- Se $k=3$, então o ponto médio a considerar é
$$1 + 3 \times 1 + \frac{1}{2} = \frac{9}{2} = 4{,}5$$



VI. Depois de concluído o ciclo, a variável **soma** contém a soma de todas as "alturas" dos retângulos.

Como referido anteriormente, todos os retângulos possuem uma dimensão comum, de comprimento dt. Por isso, não é necessário calcular individualmente a área de cada um, bastando multiplicar a soma das alturas pela dimensão comum, **dt**. Como esta área é a aproximação pretendida, tem-se que incluir a instrução return, para devolver o valor em questão:

return dt*soma



VII. O próximo passo é definir a função f.

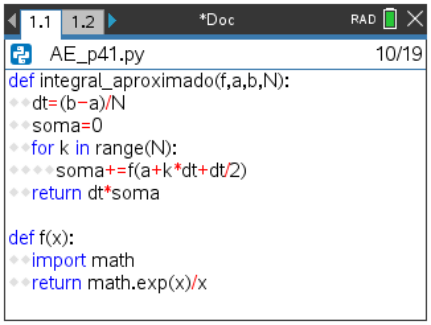
Para tal, devem escrever-se as seguintes linhas de código:

def f (x):

import math

return math.exp(x)/x

Utiliza-se a instrução **math.** porque a função exponencial não é uma operação ou função básica. Poderia evitar-se esta expressão se se iniciasse o programa com a biblioteca **math**.



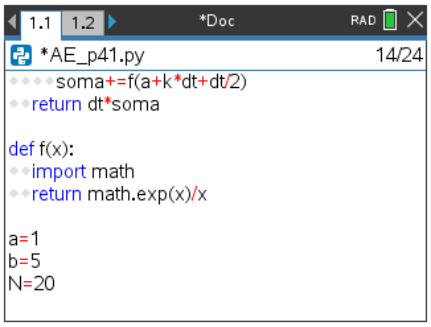
VIII. De modo a observar como a aproximação evolui à medida que o intervalo é dividido em partes cada vez menores, surge a terceira, e última, parte do programa.

Para tal, começa-se por definir os valores iniciais necessários: a, b e N:

a = 1

b = 5

N = 20

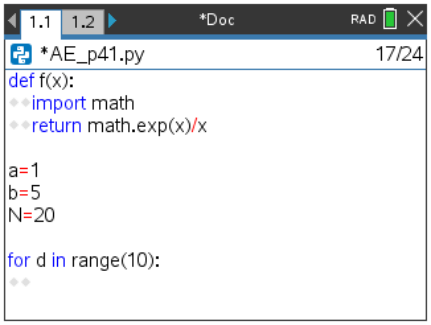


IX. Depois de definidos os valores iniciais, inicia-se um ciclo **for**, que será executado 10 vezes. O objetivo deste ciclo é calcular várias aproximações do mesmo integral, utilizando um número crescente de subintervalos. Reduzindo a amplitude de cada um a metade de cada vez que é considerado um valor de N duplo do anterior.

for d in range(10):

bloco

A variável **d**, neste caso, funciona apenas como um contador do número de iterações realizadas e serve apenas para garantir que o processo é repetido dez vezes.



X. No início de cada iteração, o número de subintervalos é duplicado.

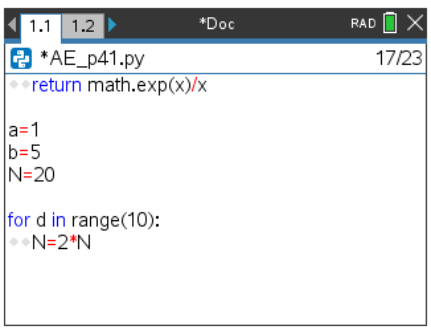
Embora inicialmente tenha sido definido que $N = 20$, na primeira passagem pelo ciclo passa a $N = 40$.

Na segunda passagem, $N = 80$, seguindo-se $160, 320, 640, \ldots$

Assim, deve-se incluir a seguinte linha de código:

N = 2N

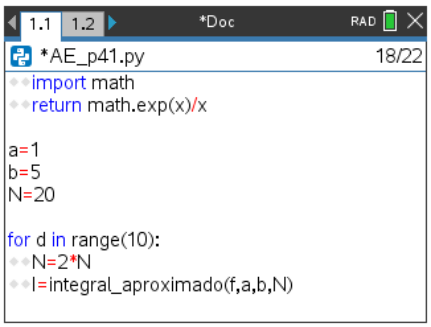
À medida que N aumenta, cada subintervalo torna-se mais pequeno. Consequentemente, os retângulos ajustam-se cada vez melhor `região sob o gráfico da qual se pretende obter a área, pelo que se vão obtendo valores do integral cada vez mais precisas.



XI. Depois de definidos os valores iniciais e a expressão analítica da função a integrar, depois de atualizado o número de subintervalos, procede-se à "chamada" da função definida anteriormente **`integral_aproximado`**:

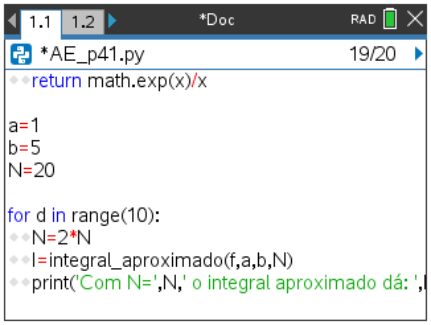
I = integral_aproximado(f,a,b,N)

A função em Python será executada e devolverá a aproximação do integral definido calculado. Este resultado irá ser armazenado na variável **I**, que representa o valor aproximado do integral obtido para o número de subintervalos utilizado nessa iteração.



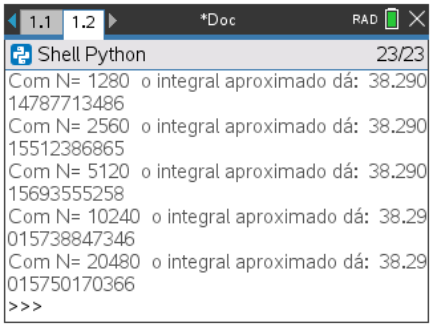
XII. Por fim, é apresentada uma mensagem, no ecrã, a indicar o número de subintervalos utilizados e o respetivo valor aproximado do integral:

print("Com N=", N, "o integral aproximado dá:", I)



XIII. Escrito o programa, falta executá-lo. Pode utilizar-se uma instrução do menu (menu 2 1), mas é claramente mais simples utilizar um atalho, uma combinação de teclas (ctrl + R).

O resultado aparece numa nova página destinada a mostrar o resultado da execução do programa, ***Shell Python***, na qual também e podem fazer operações e programas, mas que não permanecerão gravados após o fecho da aplicação.



Para voltar ao editor de *Python*, onde poderá alterar os dados de entrada, por exemplo, há mais do que um procedimento à escolha, baseados no botão do touchpad. Pode fazer deslocar o cursor com o dedo até o sobrepor ao retângulo com a designação da página, 1.1, neste caso, e premir o touchpad na parte central (). Pode também utilizar os botões laterais do touchpad após premir a tecla ctrl. Neste caso, ao premir o botão lateral esquerdo, vai para a página anterior, a do editor. Pode voltar à página de *Shell Python* utilizando o mesmo tipo de procedimento.

Na parte superior do ecrã apenas se pode observar a designação e 3 páginas consecutivas, pelo que se o documento tiver mais páginas terá de conjugar os dois procedimentos referidos ou simplesmente o que recorre às teclas laterais do touchpad.

Algumas ideias sobre programação, relacionadas com o contexto