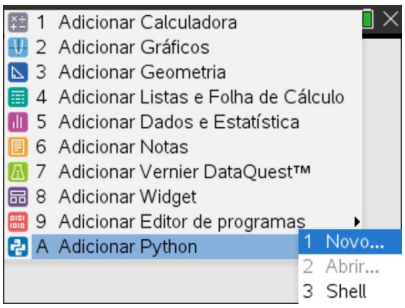
```
def bissecao(f,a,b,erro):
    if f(a)*f(b) > 0:
       print("A função tem imagens com o mesmo sinal nos extremos
       deste intervalo")
       exit(0)
    while b-a > 2*erro:
       c = (a+b)/2
       if f(a)*f(c) < 0:
          b = c
       elif f(a)*f(c) > 0:
          a = c
       else:
       return c #chamada no caso em que f(c)=0
    return c

def f(x):
    return x**3-5*x**2+2*x-3
print(bissecção(f,0,5,0.000001))
```

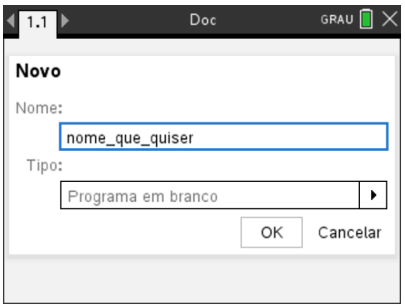
Editar Python na TI-nspire CX II-T

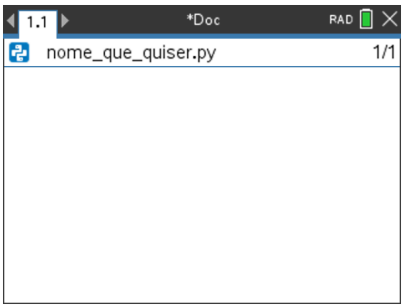
Ligue a sua calculadora e crie um novo documento.
Escolha uma página de *Python*:

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 calcular a aproximação de uma raiz num determinado intervalo através do método da Bisseção?

Utilizando o método da bisseção para encontrar um valor aproximado do zero de uma função com um erro máximo admissível, definido à partida, é necessário ter em consideração previamente:

- a expressão algébrica da função (referenciada com **`f`**);
- intervalo inicial que contenha o zero com o auxílio do Teorema de Bolzano-Cauchy (designe-se por **`[a,b]`**);
- o erro máximo tolerado para a estimativa do zero.

O funcionamento deste método baseia-se na divisão do intervalo ao meio (o intervalo é bissetado) e escolher aquele onde o Teorema de Bolzano-Cauchy garanta a existência do zero. O processo repete-se quantas vezes for necessário até que a amplitude do intervale chegue a um valor inferior ao erro admissível **ou**, caso a estimativa seja feita com os sucessivos valores centrais do intervalo, até que a **amplitude seja inferior ao dobro do erro admissível**.

No exemplo do programa apresentado nas Aprendizagens Essenciais, a função **f** é $f(x) = x^3 - 5x^2 + 2x - 3$, e pretende-se encontrar uma estimativa do zero da função com um erro máximo de $0{,}000001$, efetuando bisseções sucessivas de intervalos de números reais, iniciando com o intervalo $[0,5]$.

Antes de se apresentar a construção do programa *Python*, com a tecnologia TI-Nspire CX II-T, observe-se uma ideia gráfica do método, que favorece a compreensão do algoritmo.

1) Intervalo inicial: $[0,5]$.

Começa-se por analisar a função **f** nos extremos do intervalo:

(a) se `f(0)*f(5)>0` então não é garantida a existência de zero no intervalo, pelo que terá de se considerar outro intervalo inicial.

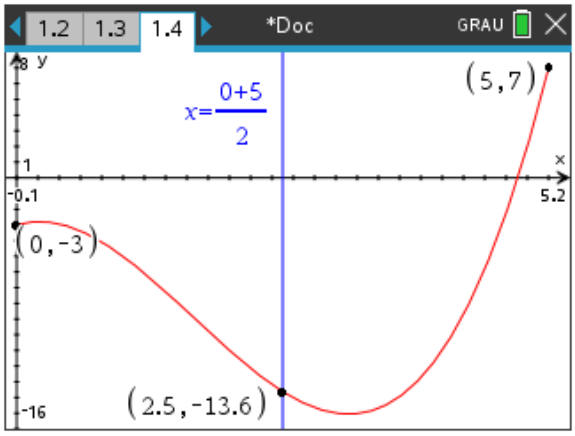
(b) se `f(0)*f(5)<0` então tem-se a garantia que a função tenha uma raiz no intervalo em questão e avança-se.

Neste caso, o erro máximo, ao considerar, como estimativa, qualquer valor do intervalo [0,5], é igual à sua amplitude. Mas considerando o valor central do intervalo como estimativa, o erro máximo é igual a metade da amplitude do intervalo, ou seja, $\frac{5-0}{2} = 2{,}5$.

O objetivo é que o erro máximo seja menor do que $0{,}00001$, considerando, como estimativa, o valor central do intervalo.

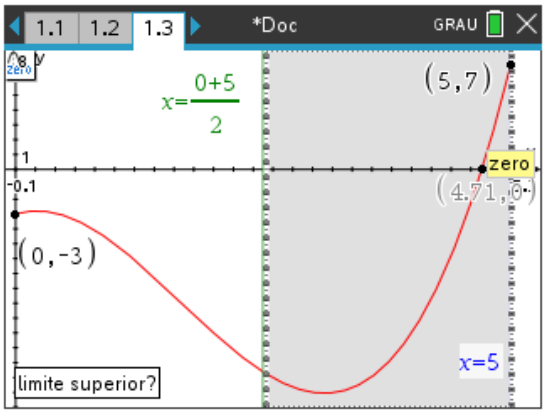
Enquanto a semi-amplitude do intervalo for superior ao erro admissível ou, equivalentemente, enquanto a amplitude do intervalo for superior ao dobro do erro máximo admissível, o método avança com as bisseções que garantam, pelo Teorema de Bolzano-Cauchy, a existência do zero no seu interior.

2) Iteração 1 – Uma vez que `f(0)*f(5)<0` e a amplitude do intervalo (2,5) é superior ao dobro do erro máximo admissível, considera-se o ponto médio c e cada um dos intervalos que resultam da bisseção do primeiro, $[0; 2.5]$ e $[2.5; 5]$.



De seguida, analisa-se a função **f** nos extremos do em cada um desses intervalos e escolhe-se aquele onde a função muda de sinal.

- `f(2,5)*f(5)<0` , logo o zero está em $[2.5; 5]$.



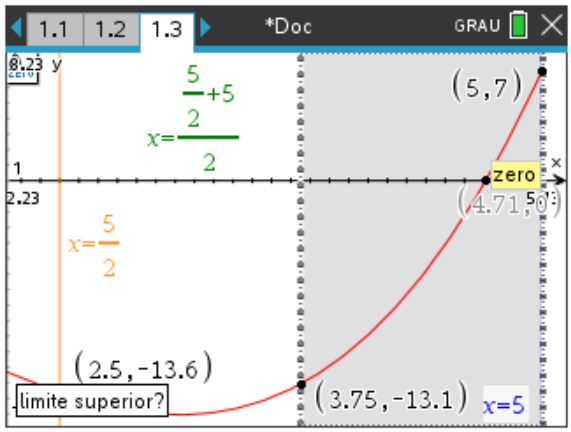
É este intervalo, $[2.5; 5]$, o resultado da 1ª iteração do método da bisseção, embora se considere especialmente o seu valor central como estimativa.

O erro máximo associado ao valor central $\frac{2.5+5}{2} = 3.75$ será, portanto, $\frac{5-2.5}{2} = 1.25$, ainda superior ao pretendido.

Há que proceder, sucessivamente, a iterações até que o erro máximo deixe de ser superior a 0.000001.

Como $1.25 > 0{,}00001$, então o método avança.

3) Iteração 2 – Na 1ª iteração, a passou a valer 2,5 e b continuou a valer 5. E obteve-se $c = 3{,}75$.



- `f(2,5)*f(3,75)>0` → no intervalo $[2.5; 3.75]$, não há garantia que a função tenha zero.
- `f(3,75)*f(5)<0` → no intervalo $[3.75; 5]$, há garantia de que aí estão zero da função.

Neste caso, o erro máximo ao considerar o seu valor central como estimativa, é $\frac{1{,}25}{2} = 0{,}625$.

Como $0{,}625 > 0{,}00001$, então o método avança e segue enquanto não se encontrar um intervalo que tenha uma amplitude inferior ao dobro do erro máximo.

Quando se obtiver esse intervalo, o seu valor central é, garantidamente a estimativa pretendida.

Seguindo este procedimento, não é difícil concluir quantas iterações o método tem de fazer até que se encontre **o intervalo pretendido e o respetivo valor central como estimativa do zero** com erro inferior a 0.000001.
Na tabela abaixo estão os resultados de cada bisseção.

#iteração	a	b	c	Erro máximo
	0	**5**	2.5	$2.5 = \frac{5-0}{2}$
1	2.5	5	3.75	$1.25 = \frac{5-0}{2^2}$
2	3.75	5	4.375	$0.0625 = \frac{5-0}{2^3}$
3	4.375	5	4.6875	0.3125
4	4.6875	5	4.84375	0.15625
5	4.6875	4.84375	4.765625	0.078125
6	4.6875	4.765625	4.7265625	0.0390625
Note-se que à n-ésima iteração, o erro máximo é de $\frac{5-0}{2^n}$ De $\frac{5-0}{2^{n+1}} < 0.000001$, vem que $2^{n+1} > \frac{5}{10^{-6}}$ ou $n > 21$.				
21	4.7106218338	4.71062421799	4.71062302589	$\approx 1.192 \times 10^{-6}$
22	4.71062302589	4.71062421799	4.71062362194	$\approx 5.960 \times 10^{-7}$

Note-se que, a precisão pretendida é garantida com o valor central da 22ª iteração, o que não significa que algum dos valores centrais anteriores não tenha sido mais aproximado do zero do que este.

Se em alguma iteração, **`f(a)*f(c)=0`**, então tinha-se alcançado a raiz do polinómio e o programa parava.

Em linguagem natural, pode escrever-se o seguinte:

Variáveis: a função `f`, os valores `a`, `b` e o erro máximo, `erro`

Início

Se `f(a)*f(b)>0`:

então escreve "A função tem imagens com o mesmo sinal" e termina imediatamente o programa

Caso contrário, enquanto o erro não for maior que o erro máximo indicado:

Define o ponto médio `c=(a+b)/2`

Se `f(a)*f(c)<0`:

Então o extremo máximo de valor `b` passará a ser o valor `c`

Já se `f(a)*f(c)>0`:

Então o extremo mínimo de valor `a` passará a ser o valor `c`

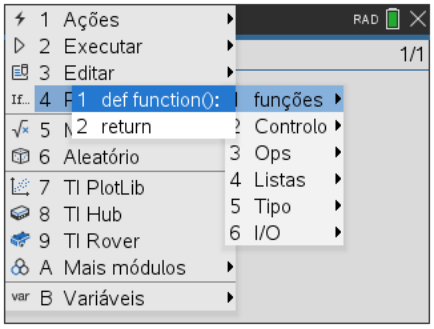
Caso contrário:

o valor pretendido é o `c` e devolve o valor final `c`

Devolve o valor `c`

Fim

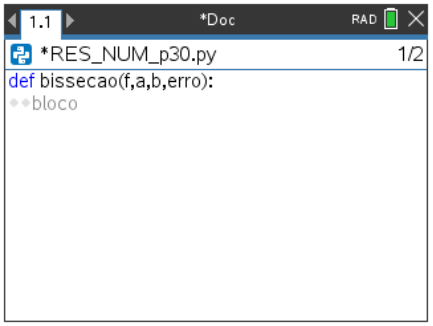
I. De modo a possibilitar a aplicação do método da Bisseção a várias funções, em diferentes intervalos e com distintas precisões, define-se uma função em *Python*, denominada **bissecao**. Esta função em *Python* recebe como argumentos a função a analisar, os extremos do intervalo considerado e o erro máximo admissível, devolvendo uma aproximação da raiz da função nesse intervalo (valor central).



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. A ideia desta estrutura é oportuna num contexto de funções matemáticas e, por isso, surge aqui como exemplo de um passo mais à frente em relação à simples utilização de uma estrutura condicional.

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():**



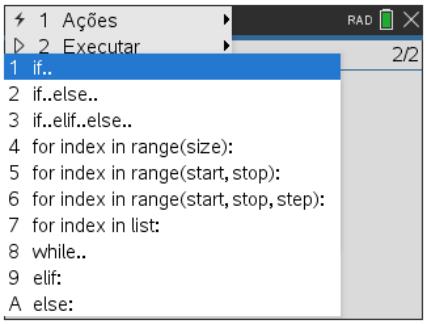
Obtida a estrutura, é agora necessário preencher os campos em falta.

A função, chamada de **bissecao**, é uma função que recebe quatro argumentos:

```
def bissecao(f,a,b,erro):
    bloco
```

II. Perante o intervalo inicial $[a,b]$, se $f(a) \times f(b) < 0$, garante-se a existência da raiz neste intervalo. Caso contrário, não se pode garantir tal existência.

Para colocar esta condição no intervalo inicial, em linguagem *Python*, é necessário utilizar uma estrutura condicional, **if**, que pode escrever com o teclado ou obter no menu:

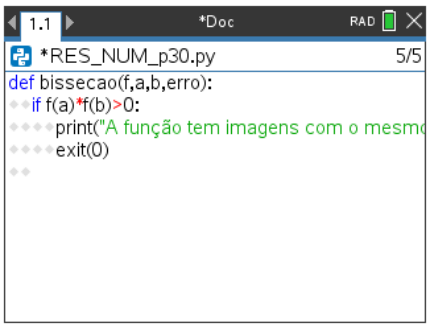


menu → 4 **Planos integrados** → 2 **Controlo** → 1 **if..**

Deste modo, tem-se de acrescentar as seguintes linhas de código, dentro da função:

```
if f(a)*f(b)>0:
    bloco
```

Se este último cenário acontecer, o programa irá parar e mostrará, na página Shell Python, uma mensagem acerca da razão de ter parado o programa. Para tal é necessário colocar as instruções que permitem a observação ou a impressão da informação pretendida, através da função **print(),** a qual pode ser escrita diretamente no editor, com o teclado, ou obtida no menu:



menu → 4 **Planos integrados** → 5 **Tipo** → 6 **I/O** → 1 **print()**

E, posteriomente, coloca-se a mensagem, sempre entre aspas, dentro da função:

```
print("A função tem imagens com o mesmo sinal nos extremos deste intervalo")
```

Dado o sucedido, o programa deve terminar imediatamente. Esta instrução é feita através do comando **exit(0)**, a qual deve ser inserida, com o teclado, logo após ser apresentada a mensagem.

III. Caso o intervalo inicial seja admissível para continuar o método, começa-se a bisseção do intervalo inicial, utilizando-se o ponto médio c do intervalo em questão. Para além disso, enquanto o erro máximo não for alcançado, o programa avança. Para este processo, em linguagem *Python*, é necessário utilizar uma estrutura de repetição do tipo **while…**, que pode escrever com o teclado ou obter no menu:

menu → 4 **Planos integrados** → 2 **Controlo** → 8 **while..**

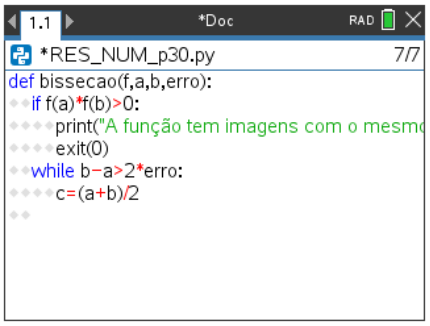
De seguida, coloca-se a seguinte linha de código:

```
while b-a>2*erro:
    bloco
```

A cada iteração, o intervalo é bissetado pelo respetivo valor médio c, que faz com que surjam dois subintervalos de igual amplitude.

Para tal, dentro da estrutura de controlo, insere-se a seguinte linha de código:

```
c=(a+b)/2
```

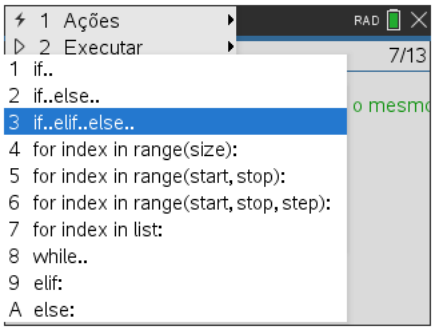


IV. Como do intervalo inicial, surgiram dois subintervalos: $[a, c]$ e $[c, b]$, no qual é necessário identificar em qual deles ocorre mudança de sinal, pois é nesse subintervalo que se encontra a raiz.

Deste modo:

- **se** $f(a) \times f(c) < 0$, então o zero pertence ao intervalo $[a, c]$ e este passa a ser o novo intervalo onde se vai avaliar a existência de raiz na iteração seguinte. Assim, o extremo superior, b, passa a ser c.
- **já se** $f(a) \times f(c) > 0$, então o zero pertence ao intervalo $[c, b]$ e este passa a ser o novo intervalo onde se vai avaliar a existência de raiz na iteração seguinte. Assim, o extremo inferior, a, passa a ser c.
- **caso contrário**, ou seja, $f(a) \times f(c) = 0$, então $f(c) = 0$, uma vez que $f(a) \neq 0$ e, portanto, o valor central c é exatamente um zero da função.

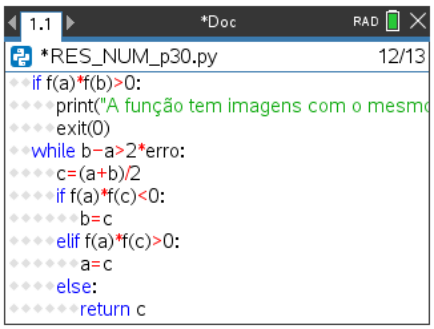
Esta linha de condições, pode ser escrita, utilizando a estrutura condicional, **if..elif..else..**, que pode escrever com o teclado ou obter no menu:



menu → 4 **Planos integrados** → 2 **Controlo** → 1 **if..**

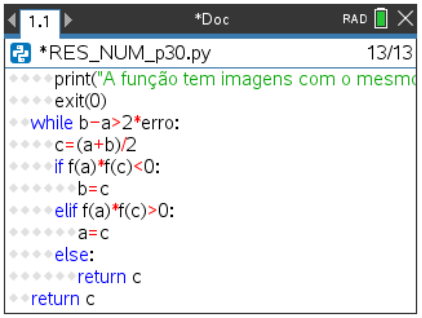
Deste modo, tem-se de acrescentar as seguintes linhas de código, dentro, também, da estrutura de controlo:

```
if f(a)*f(c)<0:
    b=c
elif f(a)*f(c)>0:
    b=c
else:
    return c
```



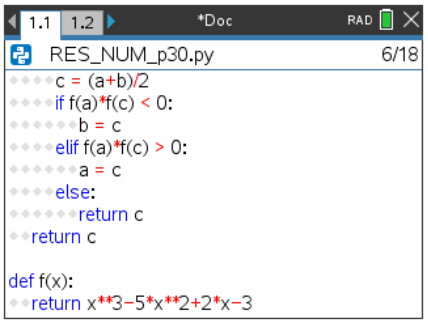
V. O ciclo **while..** termina, isto é, quando a amplitude do intervalo já é suficientemente pequena para garantir que o erro máximo seja inferior ao erro pretendido. Assim, o valor devolvido é o último valor central calculado. Esse valor central será o zero se nenhuma das condições anteriores se veerificar, pelo que é necessário incluir uma última instrução para devolver o valor c, a mesma que deverá aparecer fora da estrutura while, esta com o tresultado da execução da função:

```
return c
```



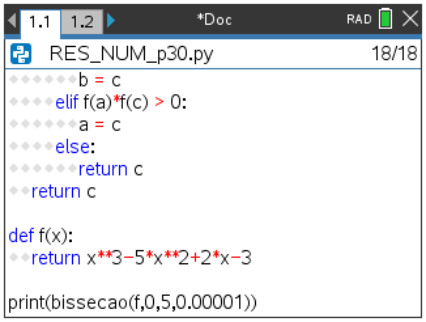
VI. Depois de definida a função *bissecao*, define-se a função matemática que se pretende estudar. A função em questão é $f(x) = x^3 - 5x^2 + 2x - 3$.
Assim, de modo análogo ao que foi feito anteriormente, deve-se escrever as seguintes linhas de código:

```
def  f(x):
    return x**3−5*x**2+2*x−3
```

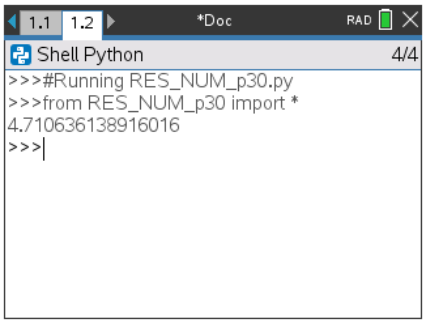


VII. Para se concluir o programa é necessário colocar as instruções que permitem a observação ou a impressão da informação pretendida, na página Shell Python, que se abre automaticamente logo após a execução do programa.
A linha de código, a colocar agora, é a que se segue:

```
print(bissecao(f,0,5,0.00001))
```



VIII. 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