Unidade 8: micro:bit com Python **Lição 2: Botões e Gestos**

Nesta lição irá aprender a usar os **botões e gestos** do micro:bit para conseguir criar um programa que permita simular o lançamento de um dado, registando os valores numa lista que será transferida para um gráfico.
Esta lição tem duas partes:

Parte 1: Investigar os botões e gestos
Parte 2: Usar um botão ou um gesto para gerar um conjunto de dados

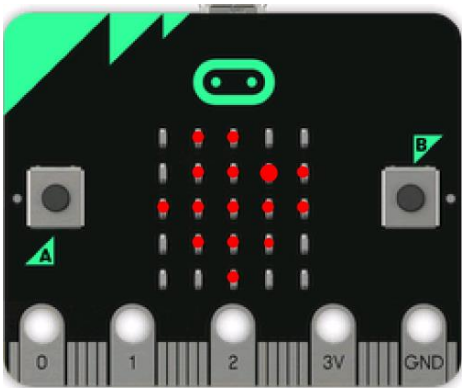
Objetivos:

- Ler e processar as teclas A e B no micro:bit
- Observar a diferença entre **.was** e **.is**
- Transferir os dados do python para a TI-Nspire
- Avaliar os dados recolhidos do micro:bit
- Utilizar gestos para controlar o *display*

Dica para o Professor: Tal como na Lição 1, esta lição é projetada de dentro para fora. O código não é introduzido sequencialmente, sendo este desenvolvido de modo a focar-se primeiramente nas funcionalidades de botões e gestos do micro:bit, terminando com o estabelecimento de uma conexão entre as listas python e as listas TI-Nspire.

1. O micro:bit tem dois botões, A e B, em cada lado do visor. O módulo python micro:bit tem dois métodos semelhantes para ler os botões e depois executar tarefas com base nesses botões. Primeiro testará os métodos e depois escreverá um programa que permita recolher dados e analisá-los noutro local na TI-Nspire CX II.

   Há também um acelerómetro/compasso de 3 eixos no verso do micro:bit e métodos para interpretar o seu movimento e orientação.



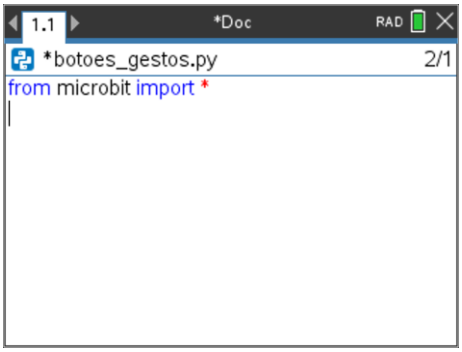
Dica para o Professor: Na versão micro:bit 2 existe um botão acima do visor (a oval dourada). No módulo, este é referido como 'Logo Touch' e utiliza o método 'is_touched()'. Este botão não é discutido nestas lições, mas é fácil de incorporar tendo em conta os conhecimentos adquiridos durante esta lição. Note a diferença entre o comportamento de .is_ e .was_ below...

2. **Parte 1: Investigar os Gestos e Botões**
   Crie um programa python num novo documento.

   Pressione a tecla **[home]** e selecione **Novo,** e depois **Adicionar Python >Novo…**
   O nome que demos ao programa foi **botoes_gestos.**

   Do **[menu] > Mais módulos > BBC micro:bit** selecione o comando **import** no topo da lista:

   from microbit import *



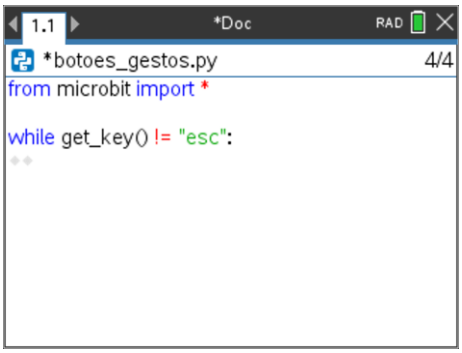
3. Adicione o ciclo **while**:

   while get_key()! = 'esc':

   a partir de

   [menu] > Mais módulos > BBC micro:bit > Commands

   Quase todos os teus programas micro:bit serão designados desta forma.



4. Para testar o botão A, adicione uma estrutura **if:**

   ♦♦**if button_a.was_pressed():**
   ♦♦♦♦**print("Botão A")**

   if *fica identado por fazer parte do ciclo* ***while*** *e* ***print( )*** *fica ainda mais identado por fazer parte do bloco* ***if****. Nota que a identação apropriada é muito importante em python. Uma identação errada provoca erros de sintaxe ou de execução não esperada do código. Note que os losangos cinzentos (♦♦) indicam o espaço de identação.*

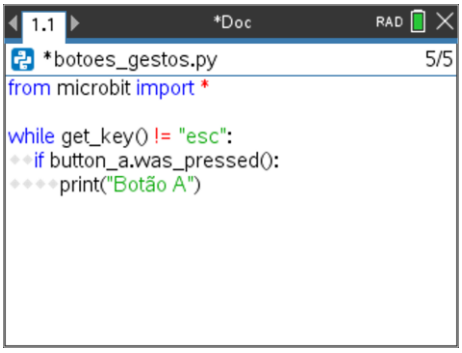
   if pode ser encontrado em **[menu] > Planos integrados > Controlo**.

   A condição **button_a.was_pressed()** pode ser encontrada em **[menu] > Mais módulos> BBC micro:bit > Buttons and Logo Touch**

   print( ) encontra-se em **[menu] > Planos integrados > I/O**

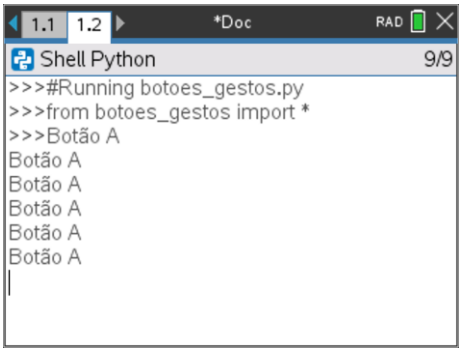
   Escreva "Botão A" dentro da função **print( )**.

   Nota: **is_pressed()** *será discutido posteriormente.*



5. Estás pronto para testar o programa. Prime **[ctrl] [R]** para executar o programa. Parece que nada está a acontecer. Prime e solta o botão A no micro:bit. Podes verificar a mensagem 'Botão A' no ecrã da calculadora. Cada vez que premires o botão, o texto irá aparecer como é representado na imagem.

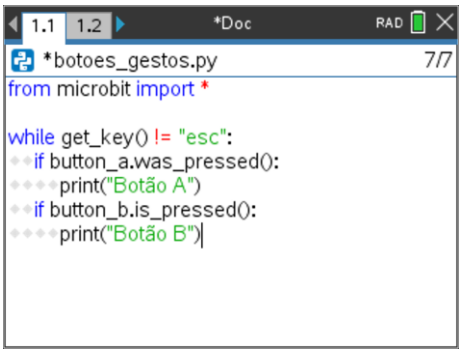
   Prime **[esc]** e volta para o editor Python.



6. Adicione outro comando **if** para verificar o botão B utilizando a condição **button_b.is pressed()**. Note que 'IS' é diferente de 'WAS'. Brevemente irá perceber a diferença.

```
if button_b.is_pressed():
    print("Botão B")
```

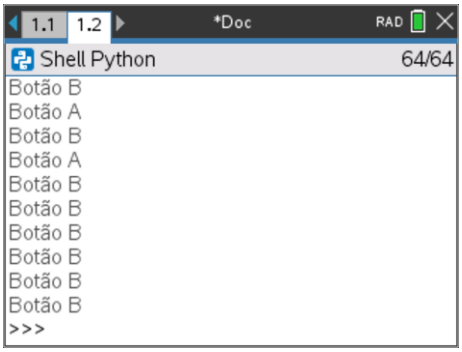
Dica: novamente, preste atenção às identações!



Dica para o Professor: As duas funções comportam-se de forma diferente. Existem circunstâncias onde a escolha entre os dois tipos de comportamentos pode ser relevante.

7. Corra o programa novamente (prime **[ctrl] [R]**). Experimente os botões A e B. **Prima** cada botão *e* **prima-e-mantenha-premido** cada botão. Poderá ver a mensagem 'Botão B' exibida repetidamente enquanto este botão estiver a ser premido, mas não verás a mensagem 'Botão A'. Existe uma diferença entre **.was_pressed()** (que precisa que solte o botão para que ocorra o *reset*) e **.is_pressed(),** que apenas verifica se o botão se encontra premido no momento exato em que o comando é processado.

Nota: se carregar no botão B muito rapidamente, o programa poderá não exibir a mensagem "Botão B", visto que o botão não se encontra premido no momento exato em que o comando **if** *é processado.*



Dica para o Professor:

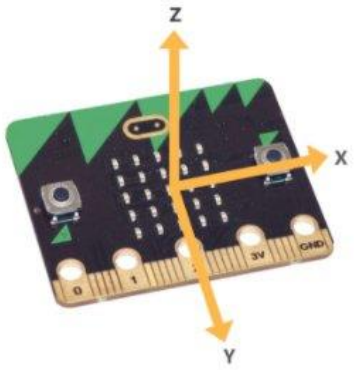
.was_pressed() requer um ação 'premir-soltar" para detetar cliques individuais.

.was_pressed() deteta um evento premir-soltar que pode acontecer mesmo quando o botão é pressionado noutro momento durante a execução do ciclo. O botão deve ser solto para que outro clique possa ocorrer. O micro:bit 'lembra-se' que o botão foi premido.

.is_pressed() funciona mais como um evento 'está premido?'. Algumas linguagens de programação orientadas para eventos possuem o mesmo tipo de funcionalidade, como por exemplo 'mouse_down'. **.is_pressed()** produz um output **True se o botão estiver para baixo (premido) no momento exato em que o comando é processado.**

*Durante o resto desta lição, o código para o botão B será ignorado***.**

8. **Introduzindo gestos**. O micro:bit tem um acelerómetro eletrónico que pode medir forças de aceleração em três direções diferentes (um acelerómetro de 3 eixos ou 3D). Além de fornecer valores numéricos para a aceleração em cada direção (ver imagem), o micro:bit também fornece 'gestos' simples e fáceis de compreender, tais como 'face para cima' e 'face para baixo' que se baseiam nesses valores direcionais.



   Esta lição explora estes gestos e demonstra como utilizá-los para programar a tua TI-Nspire CX II com o micro:bit.

9. Recolha o valor do gesto do micro:bit e armazene-o na variável **g:**

   ♦♦**g = accelerometer.current_gesture()**

   Dica: novamente, preste atenção às indentações!

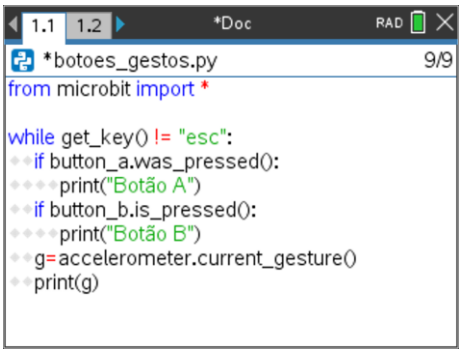
   Escreve ♦♦**g =** e depois

   Encontre **accelerometer.current_gesture()** em

   [menu] > Mais módulos > BBC micro:bit > Sensors > Gestures >

   Por fim, exiba o valor de **g**:

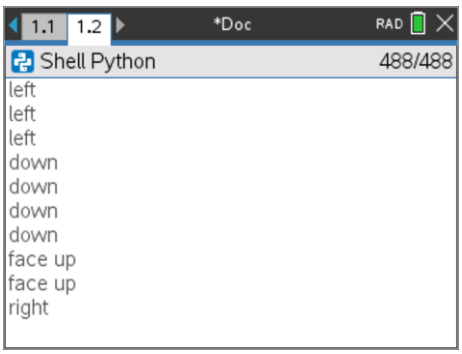
   ♦♦**print(g)**



   Nota: **print( )** encontra-se em **[menu] > Built ins > I/O**

   Note que estas duas linhas de commando estão identadas, um par apenas para ser parte do ciclo **while** *mas <u>não</u> parte do commando* **if** *abaixo. Lembre que a identação de cada linha determina o significado, pelo que ATENÇÂO!*

10. Execute o programa e observe o visor da calculadora para os vários valores, enquanto movimenta o micro:bit. Vire o micro:bit, equilibre-o em cada extremidade, abane, chocalhe, e role-o. Encontra representado na imagem um exemplo.



   Alguns dos gestos disponíveis são: **face up, face down, up, down, left, right, shake.** Estes são devolvidos como valores *string*. Consegue ver todos os gestos no seu ecrã?

11. #comentário os últimos dois comandos de gestos *(use **[ctrl] [T]** para adicionar um #comentário numa linha)*, como é demonstrado na figura e adicione esta nova função de gesto a partir do mesmo menu:

    ♦♦**if accelerometer.was_gesture("face down"):**
    ♦♦♦♦**print("face para baixo")**

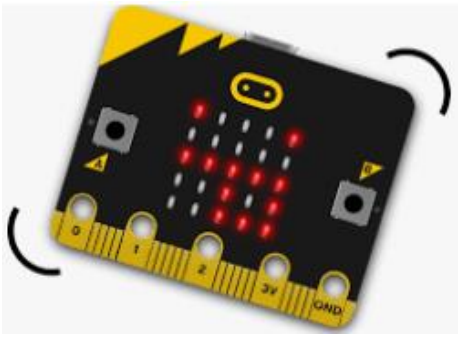
    Os métodos dos gestos e as suas "strings" encontram-se em:
    [menu] > Mais módulos > BBC micro:bit > Sensors >-Gestures

    Sim, pode simplesmente escrever as strings dos gestos mas no método **.was_gesture()** *deve corresponder exatamente ao item do menu.*

    Quando correr este programa, irá reparar numa mudança no output:
    .current_gesture() exibe constantemente o gesto.
    .was_gesture() apenas exibe quando o gesto muda.

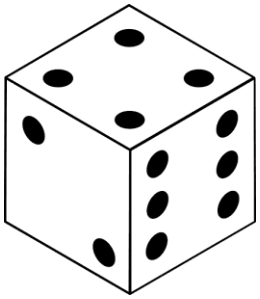
```
1.1 1.2  *Doc  RAD
botoes_gestos.py  10/11
from microbit import *

while get_key() != "esc":
    if button_a.was_pressed():
        print("Botão A")
    if button_b.is_pressed():
        print("Botão B")
# g=accelerometer.current_gesture()
# print(g)
    if accelerometer.was_gesture("face down"):
        print("face para baixo")
```

12. Botões e gestos são duas formas de obter input do micro:bit e produzir resultados no ecrã da TI-Nspire CX II ou do micro:bit... ou ambos.
    A parte seguinte desta lição tem como objetivo criar uma atividade que produz um conjunto de dados utilizando o micro:bit para uma avaliação aprofundada posterior, no ecrã TI-Nspire CX II.



13. **Parte 2:** Vamos lançar um dado (um cubo numerado de 1 a 6, com um valor em cada face). Quando o **botão A é premido,** um número inteiro aleatório entre 1 e 6 é atribuído a uma variável. Pode usar o botão A, o B ou um gesto à sua escolha.
    Exibe o valor apenas no micro:bit. Tente fazê-lo antes de passar ao passo seguinte. Utilizaremos o programa atual e adicionaremos linhas de código para simular o lançamento do dado.

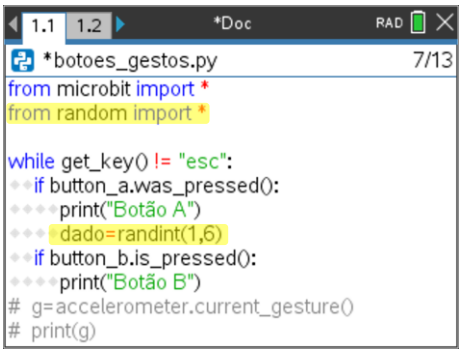
    Consegue determinar o número de pintas na face inferior do dado apresentado na fotografia?



Dica para o Professor:
Resposta*: O valor é 3. As soma das pintas de faces opostas de um dado é sempre 7.*

14. Adicione os dois comandos realçados, tal como representado na imagem:

    from random import * e **randint()** podem ser ambos encontrados em
    [**menu] > Aleatório**



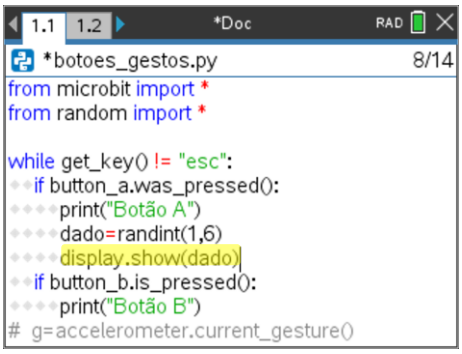
    O resto do comando ♦♦♦♦**dado = randint(1, 6)** é escrito manualmente e cuidadosamente indentado porque faz parte do bloco **if button_a…**

    Novamente, cuidado com as indentações.

15. Depois do valor do dado ser determinado, adicione o comando:

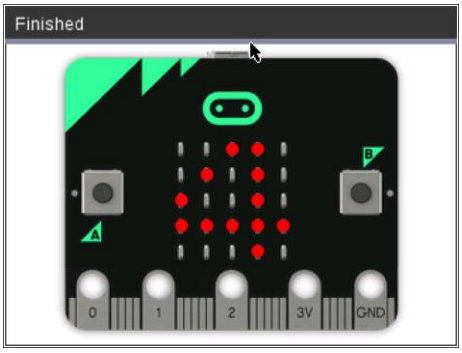
    display.show(dado)

    para exibir o valor do dado no *display* do micro:bit.



    Execute o programa novamente. Quando prime o botão A, pode observar a mensagem 'Botão A' no ecrã e o número no visor do micro:bit muda... mas nem sempre! Por vezes o número aleatório selecionado é o mesmo que o último número. Cada premir do botão é um 'evento independente'.

16. **Recolha de dados:** Lançar todos estes dados com o premir de um botão é agradável mas para prosseguir o estudo seria interessante armazenar esses valores de modo a possibilitar posterior análise. Qual é a face do dado que sai mais frequentemente? Qual é a média? E por ai fora…



    Adicione comandos ao programa para:
    - Criar uma lista em branco
    - Adicionar (.append) o valor do dado à lista
    - Armazenar a lista do python para a TI-Nspire CX II

    Cada uma destas tarefas pode ser descrita sob a forma de comandos localizados em lugares particulares no programa. Tenta sozinho antes de passares ao próximo passo.

17. A lista em branco deve ser criada no início do programa:

    lançamentos = [ ]

    Os parêntesis retos encontram-se no teclado e em **[menu] > Planos integrados > Listas**

    Depois de definido valor do dado, este é adicionado à lista:

    lançamentos.append(dado)

    .append() pode ser encontrado em **[menu] > Planos integrados > Listas**

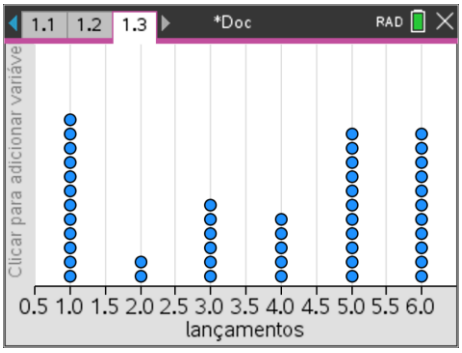
No final do programa, a lista definitiva é guardada numa lista TI-Nspire:

store_list("lançamentos", lançamentos)

a função **store_list** encontra-se em **[menu] >BBC micro:bit >Commands**

Nota: os dois comentários foram removidos de modo a que todo o código possa caber no ecrã. Podes deixar esses comentários no programa.

18. Quando executar o programa, pressione o botão A várias vezes, depois prima **[esc]** para terminar o programa. O seu programa python tem uma lista chamada 'lançamentos' e agora a sua TI-Nspire também possui uma lista chamada 'lançamentos'. Estas constituem duas listas separadas.



Prima **[ctrl]-[doc]** ou **[ctrl]-[l]** para inserir uma nova página e selecione a aplicação de **Dados & Estatística**. Um conjunto de pontos encontra-se disperso no ecrã. Clique na mensagem '*Clicar para adicionar variável*' no fundo do ecrã e selecione a lista lançamentos. Consegue ver o ecrã à direita? Cada ponto corresponde a um lançamento do dado. Que informação consegue retirar do gráfico? É capaz de transformar o gráfico de pontos num histograma? Dica: observe o **[menu]**.

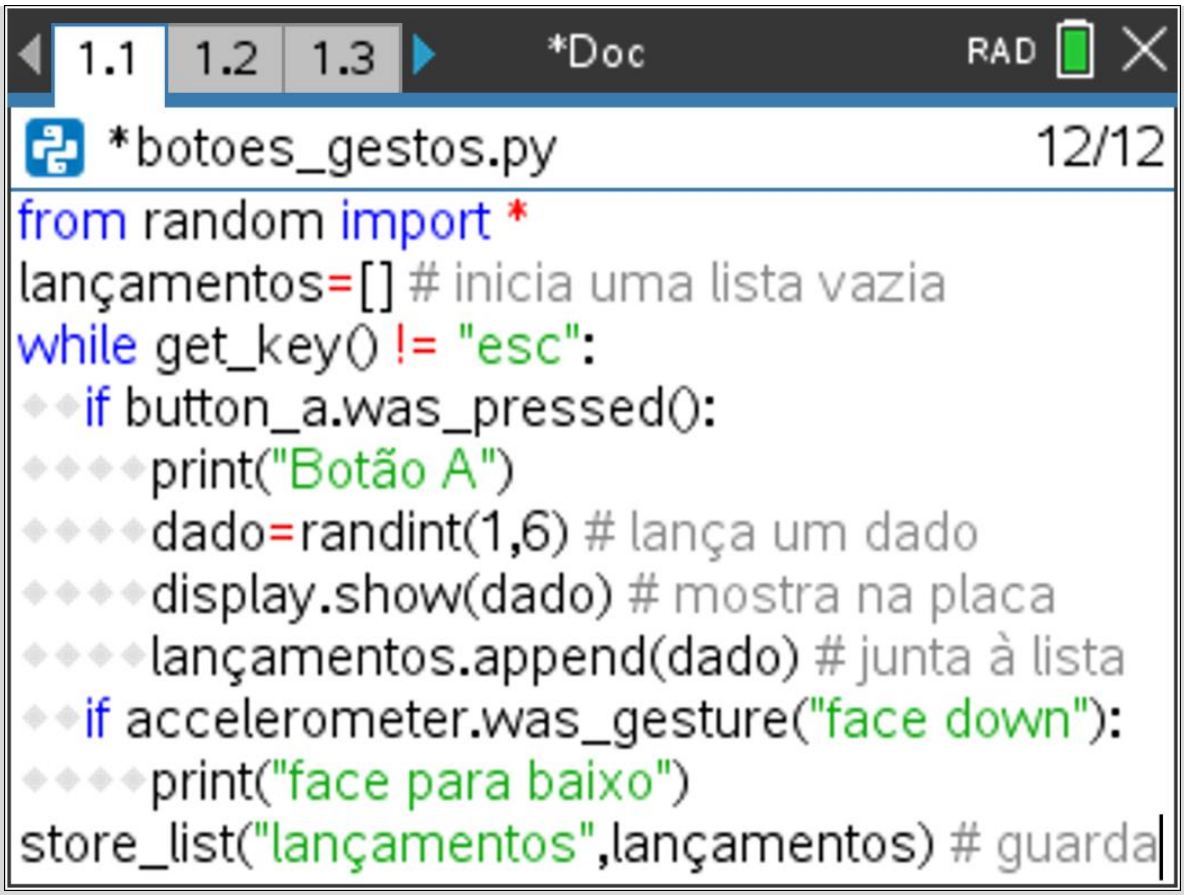
Podemos modificar o programa de diversas formas; por exemplo: mostrar (**print**) o número de lançamentos executados, à medida que estes vão sendo simulados.

*Nota: O botão B e o gesto não têm efeito sobre o dado. Tente modificar o código de modo a que o dado seja lançado **apenas** quando "agitar" o micro:bit.*

Lembre-se de gravar o documento!

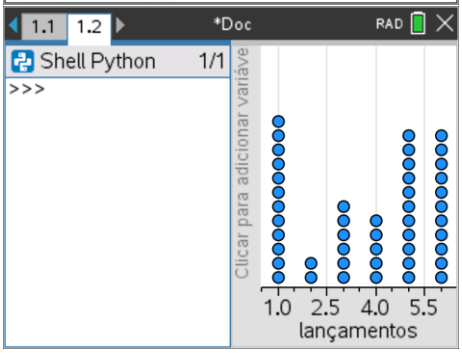
Dica para o Professor:
O **Botão B** pode ser usado como um botão de 'reset', limpando todos os dados e começando de novo com uma lista em branco.

O programa até agora com #comentários:



19. **Extensão:**
    - **print("terminado.")** no final do programa para saber que este terminou.

20. Direcione-se para a aplicação Shell e prima **[ctrl] [4]** para combinar a página Shell com a aplicação de Dados & Estatística, tal como é demonstrado na figura.



Executa o programa premindo **[ctrl] [R]** no Shell. Pressiona o botão A no micro:bit para dar início aos lançamentos do dado e observa o gráfico de pontos!

O gráfico irá exibir a mensagem 'No numeric data' *inicialmente, porque a lista de lançamentos (lançamentos***)** *foi limpa pelo programa, mas não se preocupes: se premir o botão A inicia a marcação dos dados.*

21. Em vez de exibir apenas a mensagem ‘Botão A’ com cada premir do botão, pode criar um **print** da lista de lançamentos. Onde irá incluir o comentário **print(lançamentos)** no seu código?