



```
import math
idades=[14,16,14,15,17,23]
idades.sort()
n=len(idades)
soma=0
for i in range(n):
    soma=soma+idades[i]
media=soma/n
desvios=0
for i in range(n):
    desvios = desvios + (idades[i]-media)**2
desvio_padrao=math.sqrt(desvios/n)
amplitude=idades[n-1]-idades[0]
print('Desvio padrao:', desvio_padrao)
print('Amplitude:', amplitude)
```

Editar Python na TI-nspire CX II-T

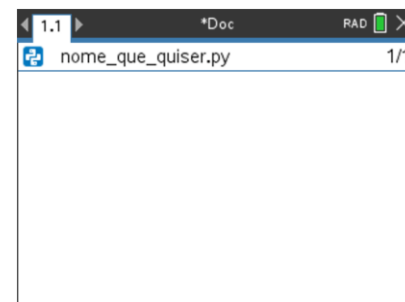
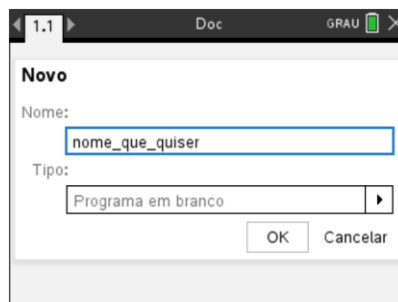
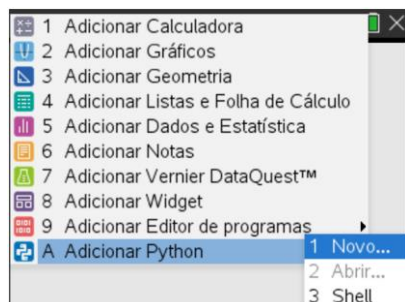
Ligue a sua calculadora e crie um novo documento.

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.



I. Dada uma lista de idades com um número de elementos variável, como determinar o desvio-padrão e a amplitude dos dados da lista?

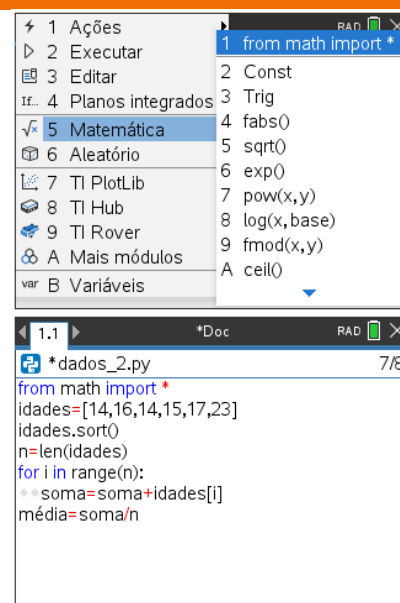
- I. Na sequência do programa anterior, também na página 27 das AEs, o número de elementos da lista terá de ser agora determinado com uma função, **len()**. Esta instrução pode ser escrita com o teclado, ou obtida no menu.

[menu] [4] Planos integrados → [4] Listas → [3] len()

Além desta alteração, note que neste programa aparece a função **sqrt()**, a qual se destina à determinação do valor da raiz quadrado do número que estiver em argumento (entre parênteses). Pode escrever-se **sqrt()** no teclado ou obter-se a partir do menu. Não é uma função que esteja nas operações matemáticas básicas, pelo que é necessário importar o **módulo math**, logo no início do programa. Isso pode ser feito escrevendo no teclado, se conhecer a sintaxe, ou no menu.

[B] [5] Matemática → [1] from math import*

Pode agora começar a escrever a primeira parte do programa, analogamente ao que foi feito para a situação do início da página 27 das AEs, considerando estas primeiras alterações.



II. Para continuar a escrita do programa, de modo a imprimir-se o desvio padrão e a amplitude após a execução, vamos considerar as seguintes linhas de código:

desvio = 0 # inicialização da variável desvio, com 0, para acumular valores posteriores.

for i in range(n):

◆◆ **desvio = desvio + (idades[i]-media)**2** # acumula em desvio a soma dos quadrados dos desvios em relação à média um dos elementos da lista

desvio_padrao = sqrt(desvio/(n-1)) # raiz quadrada da variância para amostra

amplitude=idades[n-1]-idades[0] # último dado da lista, subtraído do primeiro

print("Desvio padrão: ",desvio_padrao)

print("Amplitude: ",amplitude)

Nota: Para executar o programa pode utilizar-se uma instrução do menu (**menu** **2** **1**), mas é claramente mais simples utilizar um atalho, uma combinação de teclas (**ctrl** + **R**).

```
1.1 1.2 *Doc RAD 13/17
dados_2.py
**soma=soma+idades[i]
média=soma/n
# desvio-padrão
desvios=0
for i in range(n):
**desvios=desvios+(idades[i]-média)**2
desvio_padrao=sqrt(desvios/(n-1))
# amplitude
amplitude=idades[n-1]-idades[0]
print("Desvio padrão: ",desvio_padrao)
print("Amplitude: ",amplitude)
```

```
1.1 1.2 *Doc RAD 5/5
Shell Python
>>>#Running dados_2.py
>>>from dados_2 import *
Desvio padrão: 3.391164991562634
Amplitude: 9
>>>
```

Um pouco mais longe:

Em Python há diversas funções relacionadas com a estatística e no que consta na tecnologia TI-Nspire CX II-T também há mais algumas. Uma delas, **sum()**, permite obter a soma dos elementos de uma lista, prescindindo-se assim do ciclo de repetição.

sum() pode ser escrita com o teclado ou acedida no menu

menu **4** **Planos integrados** → **4** **Listas** → **C** **sum()**

Com ligeiras adaptações, como as que se percebe na figura ao lado, o programa pode ser um pouco simplificado, devolvendo exatamente os mesmos resultados.

```
1.2 2.1 2.2 *Doc RAD 4/15
dados_1.py
from math import *
idades=[14,16,14,15,17,23]
idades.sort()

média=sum(idades)/len(idades)

# desvio-padrão
desvios=0
for i in range(len(idades)):
+ desvios=desvios+(idades[i]-média)**2
desvio_padrao=sqrt(desvios/(len(idades)-1))
```



Algumas ideias sobre programação, relacionadas com o contexto

