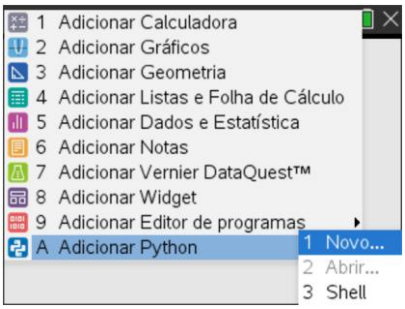
```
codigo=input("Escreve o código a desencriptar: ")
def rotn13(codigo):
  l=len(codigo)
  texto=""
  for i in range(l):
    c=ord(codigo[i])
    c=c-13
    if c<97:
      c=c+26
    if c==45:
      c=c-13
    m=chr(c)
    texto=texto+m
  return texto
print(rotn13(codigo))
```

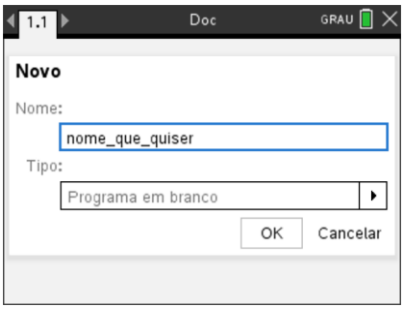
Editar Python na TI-nspire CX II-T

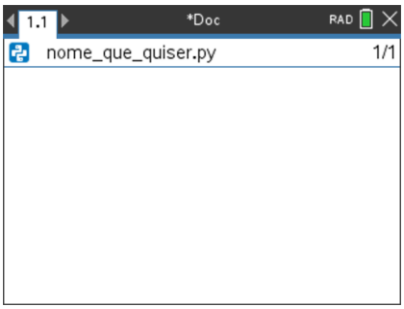
Ligue a sua calculadora e crie um novo documento.
Escolha uma página de *Python*:

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.





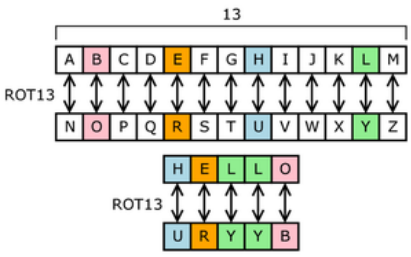


1. Como desencriptar um código previamente encriptado, recorrendo à sua conversão em código ASCII, seguido da subtração de um valor de rotação e no fim convertendo de novo o código ASCII em texto?

O programa seguinte tem como objetivo desencriptar mensagens utilizando uma cifra do tipo César – conhecida como ROT13 (*Rotation 13*). Mas, ao contrário do algoritmo de encriptação, agora subtrai-se 13.
Por exemplo, se o utilizador colocar "byn" para desencriptar, segundo a Cifra Rot-13:

b → o **y → l** **n → a** ⇉ **byn → ola**

No programa, o utilizador introduz um código, e o algoritmo aplica ROT13 a cada caráter individualmente, com a particularidade de que subtrai 13 ao respetivo número. Para isso, utiliza-se a tabela ASCII (*American Standard Code for Information Interchange*) — uma codificação padrão que atribui um número a cada caráter, como letras, números, pontuação e símbolos. No caso em questão, em que só se utilizam letras minúsculas:



Fonte: https://pt.wikipedia.org/wiki/ROT13

Código ASCII																								
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	121

I. Pretende-se fazer um programa que, depois de executado, comece por pedir ao utilizador o código para desencriptar.

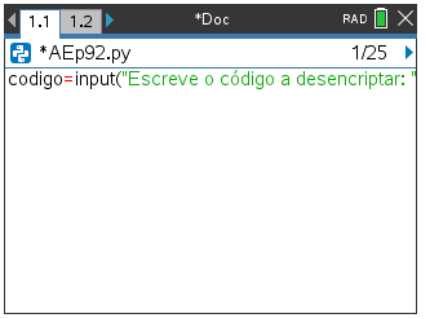
Em *Python*, a função que permite receber um dado por parte do utilizador é **input()**.

O pedido para o utilizador introduzir o código é feito através de uma mensagem:

input("Escreve o código a desencriptar:")

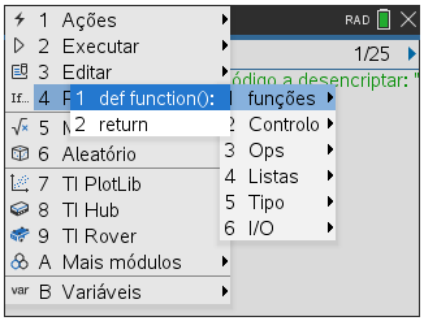
Após o utilizador introduzir o código, este será guardado na variável de nome **codigo**.

codigo=input("Escreve o código a desencriptar:")



II. Com o objetivo de se desencriptar vários códigos diferentes, usando a cifra ROT13. Convém definir a função **rotn13**, a qual irá receber o código introduzido pelo utilizador e, no fim, devolve a mensagem original.

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 das linhas de código do programa 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 **rotn13**, é uma função que recebe uma *string* (uma cadeia de caracteres) (**codigo**) como argumento:

```
def rotn13(texto):
    bloco
```

Este código pode ser facilmente escrito com o teclado, ou obtida no menu, na mesma lista pendente onde se obteve a função **def function():**. Da estrutura base, deve apagar-se o que não for necessário para esta secção funcionar conforme o pretendido.

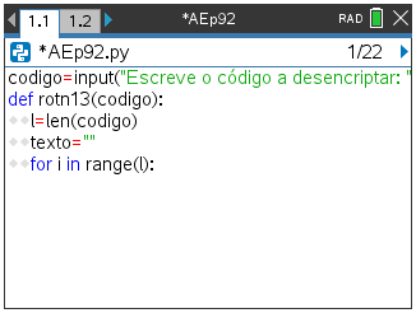
III. No programa, a cifra ROT13 irá desencriptar, um a um, todos os caracteres do código introduzido. Para tal, será necessário utilizar um ciclo de repetição (**for**) que percorra todos as posições do código, de 0 até ao comprimento do código menos 1.

Em *Python*, para saber o comprimento do código utiliza-se a função **len()**.

Esta pode ser escrita com o teclado, mas também pode ser obtida através do menu:

menu → 4 **Planos integrados** → 4 **Listas** → 3 **len()**

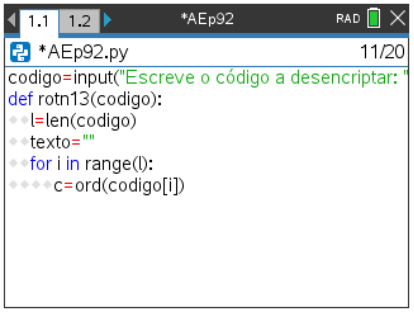
Para além disto, à medida que os caracteres forem sendo desencriptados, estes vão sendo adicionados na variável, designada por **code**, que inicializa vazia, sem espaços.




```
def rot13(texto):
    l=len(texto)          # o comprimento da frase é l
    code=""               # a variável começa vazia, sem espaços
    for i in range(l):    # para cada caráter do código da posição 0 até
                          à posição l-1
```

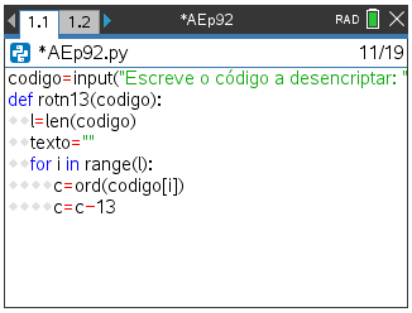
IV. Como já foi referido, o programa vai desencriptar cada letra do código introduzido pelo utilizador. Assim, na letra do código que está na posição i, **codigo[i]**, o programa vai convertê-la para o seu valor ASCII (por exemplo, como se pode visualizar na tabela "Código ASCII", a letra a é convertida no valor 97.)
Em Python, esta conversão é feita usando a função **ord()**. Esta função não se encontra no menu, tendo que ser escrita com o teclado.
Assim, deve-se inserir, dentro do ciclo **for**, a seguinte linha de código:

```
c=ord(codigo[i])
```



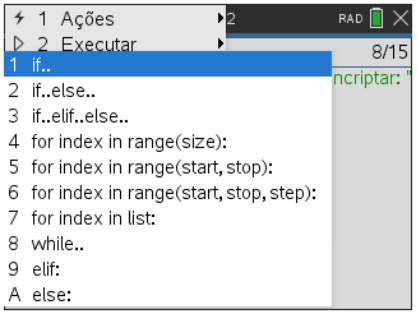
V. Após a letra ter sido convertida para ao seu valor ASCII, é feito o passo que "reverte" a cifra ROT13, subtraindo-se 13 a este valor.
Portanto, acrescenta-se a linha de código seguinte, dentro do ciclo **for**:

```
c=c-13
```



VI. **Se**, ao efetuar esta subtração, o valor for menor que 97, que é o código ASCII da letra "a", então soma-se 26 para que a contagem volte ao início do alfabeto, respeitando o comportamento circular deste.
Por exemplo, se no meio do código se encontrar um "a" o algoritmo é o seguinte:

- **codigo[i]="a" → c=ord("a")=97** ↓
- **c-13=97-13=84 → c<97** ↓
- **c=84+26=110 → m=chr(110) = "n"**

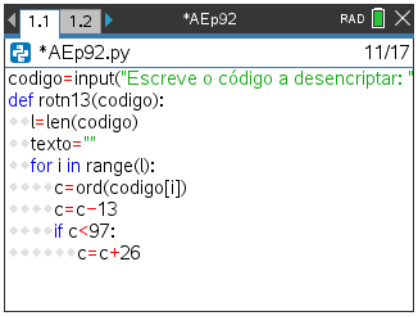


Para tal, em *Python*, é necessário utilizar uma estrutura de condição, **if**, que se pode escrever com o teclado ou obter-se no menu:

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

Assim, tem-se de acrescentar as seguintes linhas de código (ainda dentro do ciclo **for**):

```
if c<97:
    c=c+26
```

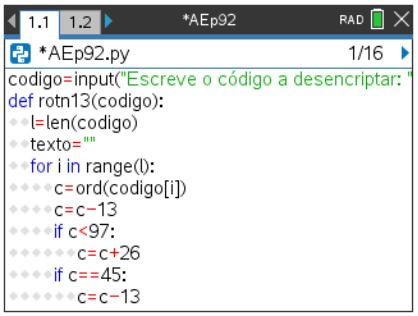


VII. **Se**, no código introduzido pelo utilizador, aparecer um espaço, que em código ASCII é o 32, o programa irá proceder do seguinte modo:

- **codigo[i]=" " → c=ord(" ")=32** ↓
- **c-13=32-13=19 → c<97** ↓
- **c=19+26=45 → m=chr(45) = "-"**

De modo a aparecer no texto decifrado os espaços colocados no código, tem de se subtrair 13 ao 45. (Atenção ao facto de isto só acontecer quando há espaços.)
Assim, deve-se inserir as seguintes linhas de código (ainda dentro do ciclo **for**):

```
if c==45:          # Se c for 45,
    c=c-13         # então subtrai-se 13.
```

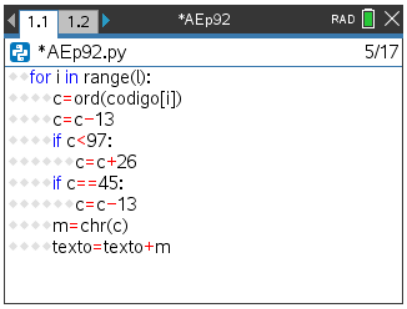


VIII. Feito o deslocamento do caracter do código para "trás", o programa precisa de converter o novo valor numérico de volta para uma letra. Esta conversão é feita através da função **chr()**. Esta é outra função que não se encontra no menu, tendo que ser escrita com o teclado.
Assim, ainda dentro do ciclo **for**:

m=chr(c)

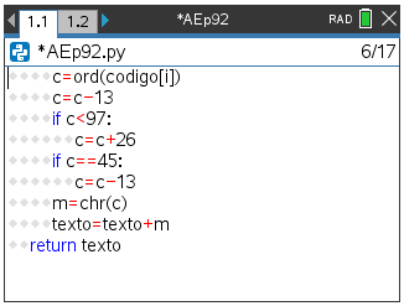
Esta letra é a nova letra encriptada que será adicionada à mensagem original, **texto**.
Assim, a linha de código que se segue:

texto=texto+m



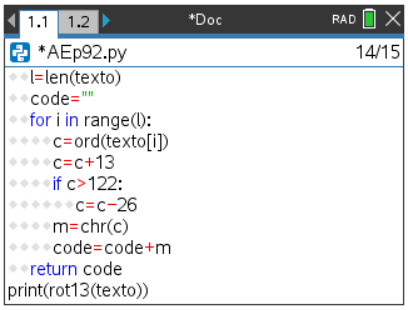
IX. Ao fim do programa acabar o ciclo de repetição, ou seja, quando acabar de percorrer toda a frase introduzida pelo utilizador, espera-se que a função retorne o código descodificado, **texto** (o texto original). Portanto, já fora do ciclo, deve-se escrever **return** com o que se pretende que seja o resultado, a saída desta secção de código:

return texto

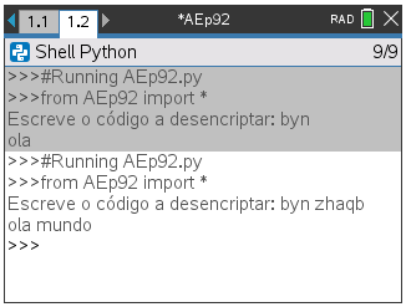


X. De maneira a concluir o programa é necessário colocar as instruções que permitem a observação ou a impressão da informação pretendida, ou seja, do código, **codigo**, desencriptado pela função construída, **rotn13**, na página Shell Python, que se abre automaticamente logo após a execução do programa.
Deste modo, insere-se a seguinte linha de código:

print(rotn13(codigo))



XI. 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 se podem fazer operações e pequenos 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, mas agora com o botão lateral direito.
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