A atribuição de variáveis - Correção

NUMWORKS

1 Exercício

Escrever uma série de instruções que altere o conteúdo de duas variáveis.

1.1 Análise do enunciado

É importante compreender corretamente a questão colocada. Para tal, o melhor é usar um exemplo.

Vamos imaginar que dois envelopes contêm uma certa quantia de dinheiro: um envelope azul contém uma nota de 20 euros e um envelope vermelho contém uma nota de 50 euros.

Podemos definir imediatamente duas variáveis indicando a quantia do envelope azul e a quantia do envelope vermelho.

```
azul = 20
vermelho = 50
```

Assim, vamos guardar o valor 20 na variável `azul` e o valor 50 na variável `vermelho`.

Desejamos agora trocar o conteúdo dos dois envelopes. A nota de 50 euros estará, portanto, no envelope azul e a nota de 20 euros no envelope vermelho.

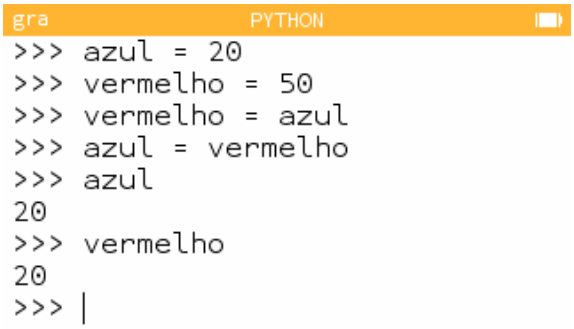
A ideia é escrever em Python as instruções que realizam esta operação.

1.2 Resolução ingénua que não funciona

Temos então duas variáveis `azul` e `vermelho` cujo conteúdo queremos trocar.

Como sabemos a a atribuição de variáveis é feita com o sinal `=`, é muito tentador escrever:

```
vermelho = azul
azul = vermelho
```



Estamos no interpretador interativo.
O resultado não é o esperado pois o conteúdo de **azul** ainda é 20...

O que aconteceu?

Uma vez que escrevemos **vermelho = azul**, o conteúdo de **azul**, 20, foi armazenado em **vermelho**. O novo conteúdo de **vermelho** é então 20. Estamos perante o seguinte:

```
>>> azul
20
>>> vermelho
20
```

Assim que escrevemos **azul = vermelho**, o conteúdo de **vermelho**, que é agora 20, é armazenado em **azul**. O novo conteúdo de **azul** é então 20. Estamos perante o seguinte:

```
>>> azul
20
>>> vermelho
20
```

Este método não funciona, portanto, uma vez que a primeira instrução foi gravada por cima do conteúdo da segunda variável. Portanto, a ideia é armazenar o conteúdo da segunda variável algures antes de gravar por cima alguma coisa.

1.3 Surgimento de uma terceira variável

Assim, vamos, em primeiro lugar, armazenar o conteúdo da segunda variável numa terceira variável, a que daremos o nome de **stock**, para evitar perder a informação. Um pouco como se tivéssemos um terceiro envelope vazio no qual mantivéssemos a nota de 50 euros temporariamente.

Instrução 1 :

```
stock = vermelho
```

Agora que a informação está guardada, podemos colocar o conteúdo da primeira variável na segunda variável. No nosso exemplo, colocamos a nota de 20 euros no envelope vermelho (agora vazio).

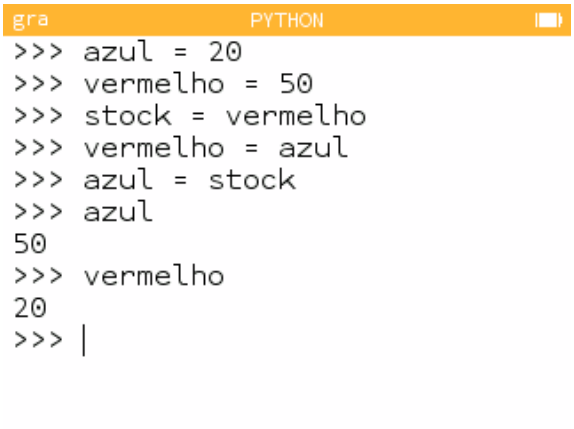
Instrução 2 :

```
vermelho = azul
```

Só falta colocar a nota de 50 euros no envelope azul, ou seja, o conteúdo de **stock** na primeira variável.

Instrução 3 :

```
azul = stock
```



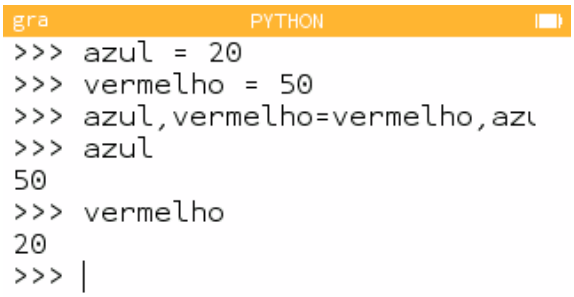
No fim destas instruções, o conteúdo dos dois envelopes deverá estar trocado.

1.4 Para terminar: um pequeno truque do próprio Python

De facto, a linguagem Python permite evitar passar por outra variável. É possível fazer a atribuição de duas variáveis na mesma instrução.

```
a,b = 2,3
```

Esta instrução permite armazenar 2 em `a` e 3 em `b`.



No nosso exemplo era então possível escrever o seguinte: `azul,vermelho=vermelho,azul`. Como as atribuições são feitas na mesma variável, o conteúdo de `vermelho` não é apagado, e o conteúdo das duas variáveis é trocado.

2 Um outro exercício

Escreva uma função `media` que tome uma lista de valores como argumento e que devolva a média aritmética desses valores.

2.1 Correção

```
gra                    PYTHON
>>> from exercicio2 import *
>>> media([10,15,13.5,16])
13.625
>>>
```

Este exercício mostra a utilização prática de variáveis concretas dentro de uma função.

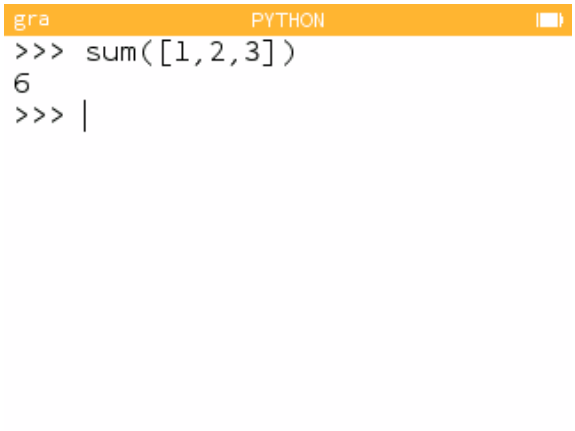
2.2 Algumas notas sobre a correção

```
gra                    PYTHON
 1 from math import *
 2 def media (lista_valores):
 3   numero = len(lista_valores)
 4   soma = sum(lista_valores)
 5   resultado = soma/numero
 6   return resultado
 7
 8
 9
10
11
12
```

Uma lista de números é uma sequência de números entre parênteses rectos separados por vírgulas. Assim, no script irá escrever:

```
gra                    PYTHON
>>> len ([1,2,4,6])
4
>>>
```

A função **`len`** permite calcular o comprimento de uma lista (número de elementos).



A função **`sum`** é utilizado para calcular a soma dos elementos de uma lista, neste caso a soma dos valores introduzidos.