INSTITUTO DO EMPREGO
E FORMAÇÃO PROFISSIONAL

INTRODUÇÃO À PROGRAMAÇÃO

UFCD(s) 5117, 5091

GUIA DE LABORATÓRIO 2.3
RESOLUÇÃO DOS EXERCÍCIOS DE REVISÃO

EXERCÍCIOS DE REVISÃO

1. Conceptualmente, uma função é um bloco de código com um nome e um propósito bem definido. Pelo facto de associarmos um nome a este bloco de código, podemos nos referir a ele várias vezes ao longo do programa através desse nome. Em geral, uma função processa a informação de entrada que lhe fornecemos e de seguida produz e devolve um resultado com essa informação. Em determinadas linguagens, quando uma função não devolve um resultado é designada de procedimento *(procedure*) e/ou sub-rotina. Seja como for, uma função é uma "espécie" de sub-programa e constitui o mecanismo por excelência de decomposição de um programa em sub-problemas, isto é, em sub-programas.

De forma mais rigorosa, em Python uma função é um bloco de instruções com um nome, introduzido através da palavra-reservada **`def`**, com uma lista com zero ou mais parâmetros, e que pode (mas não é obrigada a) devolver um ou mais resultados através da palavra-reservada **`return`**.

2. A variável global **`__name__`** indica para cada módulo o seu nome completo. Se importarmos o módulo `xpto`, definido no ficheiro `xpto.py`, a variável **`__name__`** vai ter o nome `'xpto'` ao passo que a variável **`__file__`** deverá ter o caminho completo para o ficheiro `xpto.py`. No entanto, se "executarmos" o módulo `xpto` através do interpretador (eg, na linha de comandos: `$ python3 xpto.py`) então esta variável vai ter o nome `'__main__'` que é o nome do módulo que iniciou o interpretador. Deste modo podemos distiguir se um determinado módulo foi "arrancado" na linha de comandos ou se foi importado através da instrução **`import`**.

3. Pesquisar na net.

4. Pesquisar na documentação de referência do Python, na parte dedicada à biblioteca padrão.

5.

5.1 `txt[3]` = `' '`

5.2 `txt[3:]` = `' dia, Alberto!'`

5.3 `txt[-len(txt)]` = `'B'`

5.4 `txt[-len(txt) + 4]` = `'d'`

5.5 `txt[4:-5]` = `'dia, Alb'`

6.

`vals = [1 2 3 4]` `for val em vals:` `  print(val)`	*`vals = [1, 2, 3, 4]`* *`for val in vals:`* *`  print(val)`*
`if __name__ = __main__:` `  print("Bem vindo")`	*`if __name__ = '__main__':`* *`  print("Bem vindo")`*
`def formula(x, y):` `  z = 2*x + y/3` `def mostraResultado(x, y):` `  print(formula x, y)`	*`def formula(x, y):`* *`  return 2*x + y/3`* *`def mostraResultado(x, y):`* *`  print(formula(x, y))`*
`import Decimal from decimal` `x = decimal['0.1']`	*`from decimal import Decimal`* *`x = Decimal('0.1')`*

7.

`"{}{}".format(y, x - 10)`	*4*	*1*	*0*	*.*	*1*	*8*													

`"{:5}\n".format(y)`				*4*	*1*	*\n*													
`"{1:>6}{0:<6}".format(x, x+2)`		*1*	*2*	*.*	*1*	*8*	*1*	*0*	*.*	*1*	*8*								

8.

`print('\n'.join("Alberto"))`	*`A`* *`l`* *`b`* *`e`* *`r`* *`t`* *`o`*
`print('\n\t'.join(("abc", "def", "ghi")))`	*`abc`* *`  def`* *`  ghi`*

DIM = 7 v = [10, 90, 8, 17, 16, 10, 4] i = 1 v[i] = 19; i += 1; v[i] = 10; i+=3; v[i] = 6 print(v[v[DIM*2-10] - v[2] - 3] + v[i*2-4])	*v -> {10, 90, 8, 17, 16, 10, 4} ->* *{10, 19, 10, 17, 16, 6, 4}* *i -> 5* *v[v[DIM*2-10] - v[2] - 3] + v[i*2-4]* *= v[v[7*2-10] – 10 – 3] + v[5*2-4]* *= v[v[4] – 13] + v[6]* *= v[16 – 13] + 4* *= v[3] + 4* *= 17 + 4* *= 21* *---* *R: 21*
v = [8, 9, 8, 1, 1, 8] DIM = len(v) i = 1 v[DIM-i] = 14; i-=1; v[i] = 10; i+=2; v[DIM-2*i] = 6 print(v[i] + v[5]*2 + v[v[0]-v[DIM-1]+5])	*43*