Folha prática 2 Introdução à Programação, DCC/FCUP, 2009/2010

1. Interpretador de comandos
   Abra um terminal do Linux. Escreva `python`. Deve aparecer o "prompt" >>>. [Em alternativa use um 'buffer' do `emacs` com o interpretador de `python`]. O interpretador de comandos do python está à espera das suas ordens. Faça-o calcular os seguintes valores; antes de obter cada uma das respostas, tente prever qual será! Cada previsão correcta vale 1 valor (numa escala de 0 a 24); que classificação teve? Se ocorrer um erro, explique-o. Nota. Para sair do interpretador escreva `quit()` ou Ctl-`D`.

   (a) `3*5` (b) `2*(4+2)` (c) `2*4+2`
   (d) `11/3` (e) `11%3` (f) `11/3.0`
   (g) `2**3` (h) `1+3*2**4` (i) `(1+3)*2**4`
   (j) `2**2**3` (k) `"Ai" + "Ui!"` (l) `"Ai!" + ""`
   (m) `2>3` (n) `3>=3` (o) `1>2 or 2>1`
   (p) `not False` (q) `0/2` (r) `2/0`
   (s) `"not False"` (t) `"2>2"` (u) `2=3`
   (v) `2==2` (w) `2!=2` (x) `2+2==5`

2. Tipos dos valores
   As expressões têm valores; os valores têm tipos; os tipos podem ser `int` (inteiro), `float`, `str` (palavra, sequência de caracteres), `bool` (booleano), etc. Para cada alínea do exercício anterior (que não tenha originado um erro!) indique o tipo do *resultado*. Por exemplo, para a alínea (a), a resposta deve ser `int`. Confirme as suas respostas interpretador de comandos usando a função `type()`; por exemplo `type(5555)` deve ser `int`.

3. Variáveis e atribuições
   Uma variável tem um nome e tem (ou pode ter) um valor. A instrução (de atribuição) `x=5+3` coloca o valor 8 na variável com o nome `x`. Com vista a encontrar as raízes da equação $ax^2 + bx + c = 0$ para $a = 1$, $b = 3$, $c = 1.5$, escreva *no interpretador de comandos* uma sequência de comandos com os objectivos indicados. Note que pode em qualquer altura ver o conteúdo de uma variável, escrevendo o seu nome.

   (a) Coloque 1 na variável `a`.
   (b) Coloque 3 na variável `b`.
   (c) Coloque 1.5 na variável `c`.
   (d) Coloque $\sqrt{b^2 - 4ac}$ na variável `d`; note que $\sqrt{x} = x^{0.5}$.
   (e) Coloque a raíz $(-b + d)/2$ em `r1`.
   (f) Coloque a raíz $(-b - d)/2$ em `r2`.
   (g) Escreva `r1,r2` para saber as raízes.
   (h) Verifique que `r1` satisfaz (aproximadamente, isto é, a menos de um erro muito pequeno) a equação.

4. Funções matemáticas
   Existem muitos módulos incluídos na distribuição do python. O módulo `math` disponibiliza muitas funções "matemáticas", como o logaritmo, o seno, etc. Para utilizar estas funções é necessário importar o módulo `math`, usando o comando `import math`. A função seno(x) passa a chamar-se math.sin(x). Em alternativa pode usar `from math import *`; mas neste caso a função seno(x) passa a chamar-se simplesmente sin(x). No interpretador de comandos faça

   (a) Importe o módulo `math`.
   (b) Coloque 2 na variável `x`.
   (c) Calcule $\log(x)$.
   (d) Calcule $\sin(x)$.
   (e) Calcule $\sin^2(x) + \cos^2(x)$.

5. A função misteriosa...
   Abra no `emacs` um novo ficheiro `misterio.py` onde deverá escrever o seguinte

```
t=100
def funcao(x):
    t=x*x
    t=t*t
    t=t*t
    return t*x
print funcao(1), funcao(2), funcao(10)
print t
```

   (a) Sem executar o programa diga quanto vale, no caso geral, `funcao(x)`

   (b) Sem executar o programa diga quais os valores impressos (incluindo o valor de `t`)

   .

   (c) Corra o programa, confirmando (ou não!) as suas conjecturas.

   (d) No interpretador de comandos, e depois de executar `funcao(2)`, tente determinar o valor de `t`. Que conclui?

6. Distância entre 2 pontos
   A distância no plano entre dois pontos de coordenadas (x_0, y_0) e (x_1, y_1) é dada por $|(x_1, y_1) - (x_0, y_0)| = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$. Implemente uma função `dist(x0, y0, x1, y1)` para calcular esta distância.

7. Área de um triângulo
   A área de um triângulo cujos lados medem a, b e c pode ser calculada usando a *fórmula de Heron*:

$$\text{área}_{\triangle} = \sqrt{s(s-a)(s-b)(s-c)}$$

   onde $s = (a+b+c)/2$ é o semi-perímetro do triângulo. Implemente uma função `area_triang(a,b,c)` que calcule a área de um triângulo usando esta fórmula.

8. Área de um polígono
   A área de um polígono regular com n lados inscrito num círculo de raio r é:

$$\text{área}(n, r) = \frac{1}{2} n \sin(2\pi/n) r^2$$

   Implemente uma função `area_poligono(n,r)` que calcule a área dum polígono regular usando esta fórmula. Mostre que esta fórmula é "compatível" com a fórmula da área de um círculo, no sentido em que, quando $n \to \infty$, a área do polígono converge para a área do círculo em que está inscrito.

9. O que calcula?
   Indique a função de `x` e `y` calculada pela função `f(x,y)` (ver fundo da página). Mostre que as funções `f` e `g` são *equivalentes*. Para isso pode interessar considerar os casos: $\texttt{x} < \texttt{x}$, $\texttt{x} = \texttt{x}$ e $\texttt{x} > \texttt{x}$.

10. Potência obtida com multiplicações
    Escreva uma função `f` que receba `x` e `y` (supõe-se que `y` é um inteiro não negativo) e retorne $\texttt{x}^{\texttt{y}}$; esse valor deverá ser obtido multiplicando sucessivamente uma variável `p` (com valor inicial 1) por `x` . Quantas vezes deverá `p` ser multiplicado por `x` para se obter $\texttt{x}^{\texttt{y}}$?
    Notas. A primeira linha deverá obviamente ser "`def f(x,y):`". É suficiente a utilização das instruções de atribuição, `while` e `return`. Teste, e eventualmente corriga, a função que escreveu.

Nota 1. Como se explicou nas aulas teóricas, a instrução "`while`" executa o conjunto de comandos que lhe está associado enquanto a condição for verdadeira (em particular, esses comandos podem ser executados 0 vezes; em particular, a execução da instrução pode demorar um tempo infinito). Exemplo de uma função que calcula $1+2+\ldots+n$

```
def func(n):
    s=0
    i=1
    while i<= n:
        s = s+i
        i = i+1
    return s
```

Nota 2. Funções `f(x,y)` e `g(x,y)` usadas no problema 9.

```
def f(x,y):
  if x<=y:
     return x
  else:
     return y
```

```
def g(x,y):
  if x>y:
     return y
  return x
```