```
#-- 1 -------------------
#  preferl trabalharmos com inteiros do que
#  com flot: os arredondamentos podem dar origem
#  a erros graves - por exemplo, ciclos efectuados
#  mais ums vez que o previsto...
#

from math import *
for i in range(2,21): # i = 2x
    x=i/2.0
    print x,log10(x)

# 2 ---------------------
def newton(f,df,x0,n):
    x=x0
    for i in range(n):
        x = x-f(x)/df(x)
    return x

print "--- 2 ---"
print newton(sin,cos,2.5,5)
#  pi  -> 3.14159265359
#  Com 6 iteras obtemos exactamente (dentro da
#  precisdo computador) pi!


#-- 3 -------------------
```

```
def binom(n,k):
    return fact(n)/(fact(k)*fact(n-k))

def fact(n):
    prod=1
    for i in range(2,n+1):
        prod = prod*i
    return prod

#  [binom(4,i) for i in range(5)] -> [1,4,6,4,1]

#-- 4 ------------------
# a) Por sucessivas divises a meio do intervalo [a,b]
# determina-se a tal que a raestm [a,a+erro]
#
# b) Apividir o intervalo n vezes a meio, este tem
# o comprimento (b-a)/2^n. Logo deve ser (b-a)/2^n < erro
# ou seja, n = primeiro inteiro >= log2((b-a)/erro)

#-- 5 ------------------

def maximo(f,a,b,erro):
    while b-a>erro:
        d=(b-a)/3.0
        x1 = a+d
        x2 = b-d
        if f(x1)>f(x2):
            b=x2
        else:
            a=x1
```

```
    return a

print "--- 5 --------------"
print maximo(sin,0,5,0.01)

#-- 6 ----------------------

def ind_max(a):                #  1
    # Assume-se len(a)>=1      #  2
    m=0                        #  3
    for i in range(1,len(a)): #  4
        if a[i]>a[m]:          #  5
            m=i                #  6
    return m                   #  7

print "--- 6 --------------"
print ind_max([5,-1,8,1,8,1,6])

# para obter o maior ice i onde ocorre o mmo,
# basta na linha 5 mudar > => >=
```