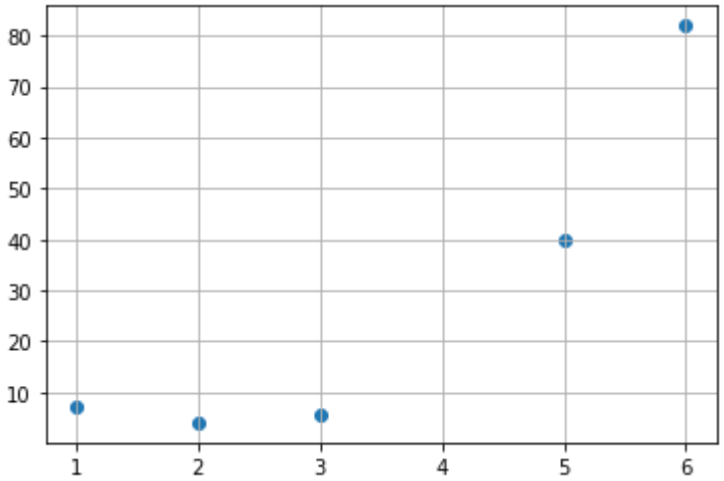
O mesmo problema de interpolação usando todas as técnicas

In [ ]:
```
O mesmo exercício de interpolação resolvido por todas as técnicas:
Considere os dados ([Cha13], pág. 425)
```

```
x        1       2        3          5        6
-------------------------------------------------
f(x)     7       4        5.5       40       82
```

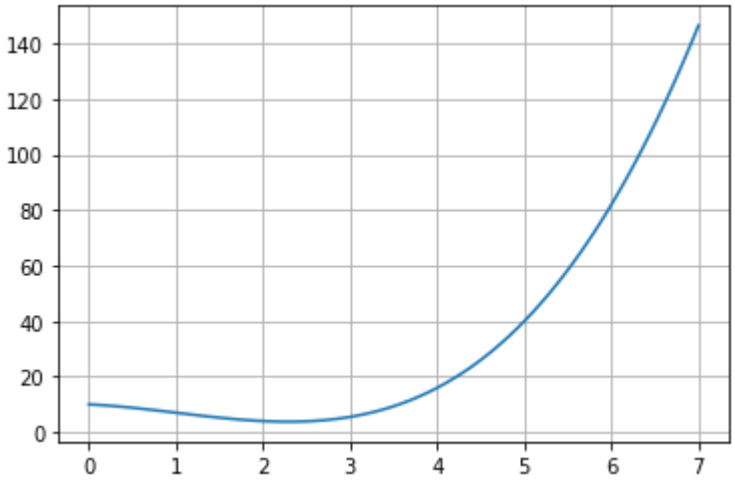
In [15]:
```
import matplotlib.pyplot as plt
x=[1,2,3,5,6]
y=[7,4,5.5,40,82]
plt.scatter(x,y)
plt.grid()
plt.show()
```



Padrão ouro

In [20]:
```
import numpy as np
print('padrao ouro ')
p=np.polyfit([1,2,3,5,6],[7,4,5.5,40,82],4)
print (p.round(4))
print('para x=4. f(x)=',np.polyval(p,4))
x=np.linspace(0,7,100)
y=np.polyval(p,x)
plt.plot(x,y)
plt.grid()
plt.show()
```

```
padrao ouro
[-0.    0.75 -2.25 -1.5  10.  ]
para x=4. f(x)= 15.999999999999945
```



Método bruto

In [3]:
```
a=np.array([[1,1,1,1,1],[1,2,4,8,16],[1,3,9,27,81],[1,5,25,125,625],
            [1,6,36,216,1296.0]])
b=np.array([7,4,5.5,40,82])
print (np.linalg.solve(a,b))  # invertido
```

```
[10.   -1.5  -2.25  0.75 -0.  ]
```

In [ ]:
```
Vamos lá: P(x)=0*x^4+0.75*x^3-2.25*x^2-1.5*x+10 ou
    p(4) = 0 + 0.75*64 - 2.25*16 - 1.5*4 + 10 ou
    p(4) = 48 - 36 - 6 + 10 = 16
```

Lagrange

Lagrange

x	1	2	3	5	6
$f(x)$	7	4	5.5	40	80

	1	2	3	5	6	π
1	—	-1	-2	-4	-5	40
2	1	—	-1	-3	-4	-12
3	2	1	—	-2	-3	12
5	4	3	2	—	-1	-24
6	5	4	3	1	—	60

~~4~~	3	2	1	-1	-2

$$P(4) = 7 \times \frac{4}{40} + 4\frac{6}{-12} + 5.5\frac{12}{12} + 40\frac{-12}{-24} + 82\frac{-6}{60}$$

$$= 0{,}7 - 2 + 5.5 + 20 - 8.2$$

$$= 26.2 - 10.2 = 16$$

Newton

1	7	$\frac{4-7}{2-1} = -3$	$\frac{1.5-(-3)}{3-1} = 2.25$	$\frac{5.25-2.25}{5-1} = 0.75$	$\frac{0.75-0.75}{?} = 0$
2	4	$\frac{5.5-4}{3-2} = 1.5$	$\frac{17.25-1.5}{5-2} = 5.25$	$\frac{8.25-5.25}{6-2} = 0.75$	
3	5.5	$\frac{40-5.5}{5-3} = 17.25$	$\frac{42-17.25}{6-3} = 8.25$		
5	40	$\frac{82-40}{6-5} = 42$			
6	82				

$$P(x) = 7 + (-3)(x-1) + 2.25(x-1)(x-2) +$$

$$0.75(x-1)(x-2)(x-3) + 0 \times ? ? ?$$

$$P(4) = 7 + (-3)(3) + 2.25(3)(2) + 0.75(3)(2)(1) =$$

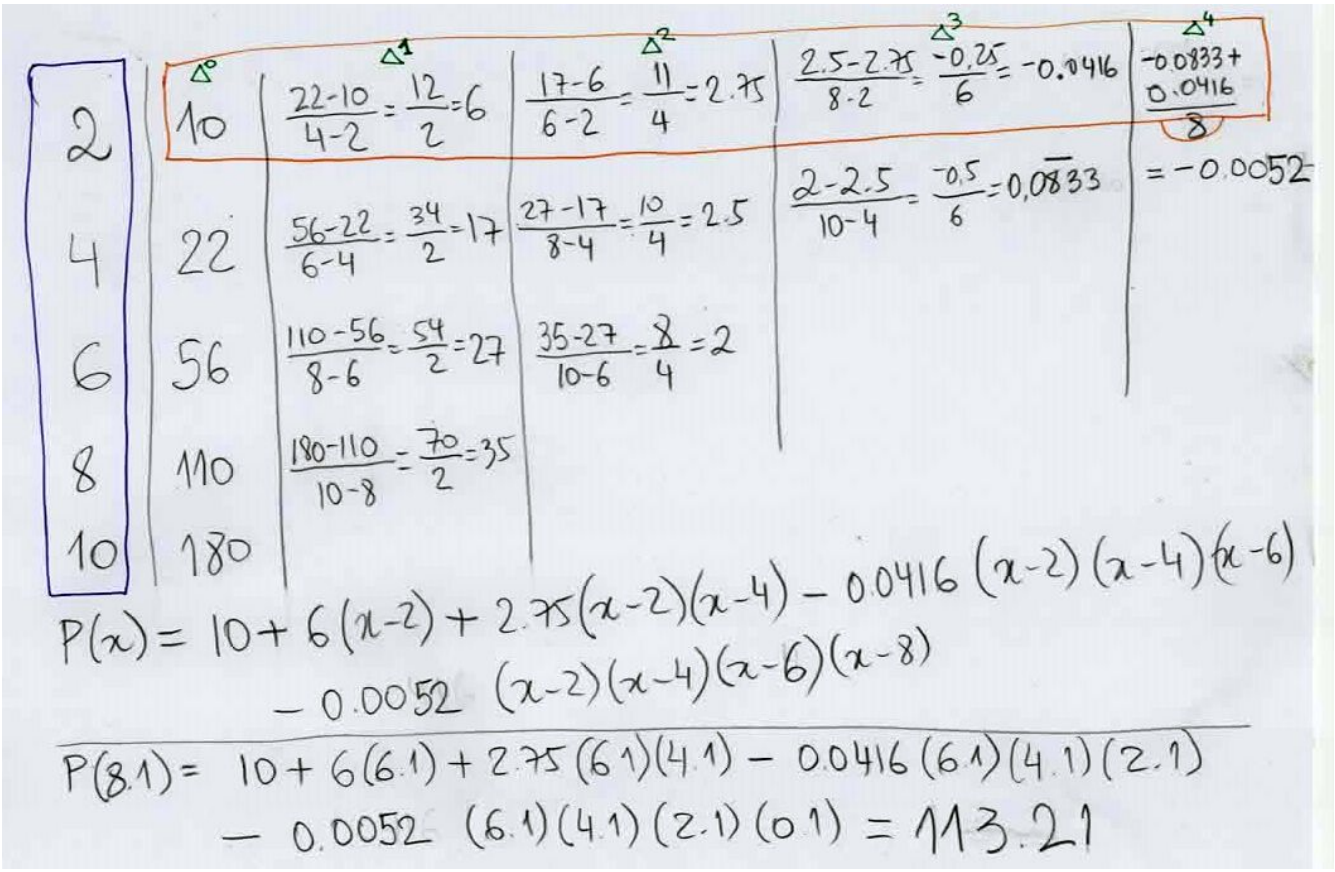
$$= 7 - 9 + 13.5 + 4.5 = 25 - 9 = 16$$

In [23]:
```
import sympy as sp
import numpy as np
x=sp.symbols('x')
def impmat(a,t,d):
    return ('\n'.join([''.join([('{:'+str(t)+'.'+str(d)+'f}').format(item) \
                                for item in row])
                       for row in a]))
def tdifdiv(x,fx):
  t=len(x)
  tb=np.zeros((t,t+1),float)
  for i in range(t):
    tb[i][0]=x[i]
    tb[i][1]=fx[i]
  j=2
  while j<t+1:
    for i in range(t-j+1):
      tb[i][j]=(tb[i+1][j-1]-tb[i][j-1])/(tb[i+j-1][0]-tb[i][0])
    j=j+1
  return tb

def polnew(x,fx):
  tb=tdifdiv(x,fx)
  t=len(x)
  print('------ o polinomio -------')
  print(str(tb[0][1]),end='')
  for i in range(1,t):
    aa=round(tb[0][i+1],4)
    if aa<0:
      bb='('+str(aa)+')'
    else:
      bb=str(aa).strip()
    print('+'+bb+'[',end='')
    for j in range(i):
      aa=tb[j,0]
      if aa<0:
        bb='('+str(aa)+')'
      else:
        bb=str(aa)
      print('(x-'+bb+')',end='')
    print(']')
  return tb
xx=polnew([1,2,3,5,6],[7,4,5.5,40,82])
print(impmat(xx,10,5))
```

```
------ o polinomio -------
7.0+(-3.0)[(x-1.0)]
+2.25[(x-1.0)(x-2.0)]
+0.75[(x-1.0)(x-2.0)(x-3.0)]
+0.0[(x-1.0)(x-2.0)(x-3.0)(x-5.0)]
   1.00000   7.00000  -3.00000   2.25000   0.75000   0.00000
   2.00000   4.00000   1.50000   5.25000   0.75000   0.00000
   3.00000   5.50000  17.25000   8.25000   0.00000   0.00000
   5.00000  40.00000  42.00000   0.00000   0.00000   0.00000
   6.00000  82.00000   0.00000   0.00000   0.00000   0.00000
```

In [ ]:
```
Para estudarmos Newton Gregory, antes vamos fazer um novo problema através de Newton. Se
x=[2,4,6,8,10] e fx=10,22,56,110,180] e estamos interessados em P(8.1) = ?
```



Newton-Gregory

Toma proveito de que os pontos de entrada devem estar igualmente espaçados por um fator h. Isto simplifica a tabela de diferenças divididas, que agora passa a se chamar de tabela de diferenças finitas.

```
Eis como se calcula:
    x       f(x)      ordem0     ordem1     ordem2     .....
    a       b         d-b>i      j-i>l      m-l>n
    c       d         f-d>j      k-j>m
    e       f         h-f>k
    g       h
```

Agora o polinômio interpolador é

$$P(x) = b + \frac{i}{1!\times h^1} \times (x-a) + \frac{l}{2!\times h^2} \times (x-a)(x-c) + \frac{n}{3!\times h^3} \times (x-a)(x-c)(x-e)$$

In [ ]:

Chamando a primeira linha da tabela de diferenças finitas de Δ^0, Δ^1 e assim por diante, a fórmula de Newton gregory fica

$$P_n(x) = \Delta^0 + \sum_{i=1}^{n} \left(\prod_{j=0}^{i-1} (x - x_j) \frac{\Delta^i}{i! \times h^i} \right)$$

Vamos a um exemplo numérico: Seja x=[2,4,6,8,10] e fx=[10,22,56,110,180] e deve-se achar $x_n = 8.1$

In [26]:
```
p=np.polyfit([2,4,6,8,10],[10,22,56,110,180],4)
print(np.polyval(p,8.1))
```

```
113.16177031249997
```

```
Tabela de diferenças finitas:

       2        10        22-10=12        34-12=22        20-22=-2        -4-(-2)=-2
```

```
 4        22        56-22=34        54-34=20        16-20=-4
 6        56        110-56=54       70-54=16
 8       110        180-110=70
10       180
```

$$P(8.1) = 10 + \frac{12}{2}(8.1-2) + \frac{22}{2! \times 2^2}(8.1-2)(8.1-4) + \frac{-2}{3! \times 2^3}(8.1-2)(8.1-4)(8.1-6) + \\ + \frac{-2}{4! \times 2^4}(8.1-2)(8.1-4)(8.1-6)(8.1-8) =$$

$$P(8.1) = 10 + \frac{12}{2}(6.1) + \frac{22}{2 \times 4}(6.1)(4.1) + \frac{-2}{6 \times 8}(6.1)(4.1)(2.1) + \\ + \frac{-2}{24 \times 16}(6.1)(4.1)(2.1)(0.1) = 113.1617703$$

In [ ]:
```
Para encerrar, resolver x=[1,11,21,31,41], fx=[-2,10,150,160,40] e depois achar f(25)?
```

In [40]:
```
p=np.polyfit([1,11,21,31,41],[-2,10,150,160,40],4)
print(p)
print(np.polyval(p,25))
def ngreg(x,fx):
    t=len(x)
    r=np.zeros((t,t+1),float)
    for i in range(t):
        r[i][0]=x[i]
        r[i][1]=fx[i]
    for j in range(2,t+1):
        i=t-j
        while i>=0:
            r[i][j]=r[i+1][j-1]-r[i][j-1]
            i=i-1
    return r
print(ngreg([1,11,21,31,41],[-2,10,150,160,40]))
```

```
[ 1.0750000e-03 -1.1180000e-01  3.4414500e+00 -2.6801800e+01
  2.1471075e+01]
175.37920000000082
[[   1.   -2.   12.  128. -258.  258.]
 [  11.   10.  140. -130.    0.    0.]
 [  21.  150.   10. -130.    0.    0.]
 [  31.  160. -120.    0.    0.    0.]
 [  41.   40.    0.    0.    0.    0.]]
```

In [46]:
```
import sympy as sp
x=sp.symbols('x')
p=(-2)+(12*(x-1)/10)+(128*(x-1)*(x-11)/(2*10**2))+(-258*(x-1)*(x-11)*(x-21)/(6*10**3))+
  (258*(x-1)*(x-11)*(x-21)*(x-31)/(24*10**4))
print(sp.expand(p))
print(p.subs(x,25).n())
```

```
43*x**4/40000 - 559*x**3/5000 + 68829*x**2/20000 - 134009*x/5000 + 858843/40000
175.379200000000
```