Gyakorló feladatok megoldásai

**Ha valaki az igazságot és a törvényszerűséget keresi, nem tehet különbséget kicsiny és nagy problémák között.
Aki a kicsiny dolgokban nem veszi komolyan az igazságot, abban az emberben nagy dolgokkal kapcsolatban sem bízhatunk meg.**
Albert Einstein

In [1]:

```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Több megoldást is közzéteszünk, mindegyik helyes, és tökéletesen elvégzi a kiadott feladatot.

Először mindig futtasuk le a "%pylab inline" parancsot

Generálj egy egész számokból álló 1000 tagú véletlen számsorozatot (0-100-ig vannak elemek). Majd írasd ki a 100.-tól a 200.-ig elmeig 7-sével az értékeket. Ehhez használd a "randint()" függvényt.

Megoldás

In [2]:

```
A=[]   # Létrehozunk egy üres listát, amit majd később lehet növelni
for i in range(0,1000):  # Elindítunk egy ciklust, ami 1000-szer fut le
    B=randint(0,100)  # randint választ egy számot 0 és 100 között
    A.append(B) # hozzáírjuk az A listához a B számot
A[100:200:7] # kiírjuk a lista elemeit 100 és 200 között 7-sével
```

Out[2]:

```
[76, 13, 67, 47, 57, 2, 79, 58, 14, 49, 38, 75, 26, 67, 73]
```

Megoldás

In [3]:

```
B=randint(0,100,1000) # Generálunk egy 1000 tagú listát a randint-tel
C=arange(100,200,7) # megalkotjuk azoknak a számoknak a listáját, amilyen indexű elm
eet kiíratni szeretnénk
B[C] #kiírjuk a megfelelő indexű számokat
```

Out[3]:

```
array([78, 82, 24, 26, 70, 12, 56, 58, 38, 68, 87, 89,  3, 85, 86])
```

Megoldás

In [4]:

```
B=randint(0,100,1000) # Generálunk egy 1000 tagú listát a randint-tel
B[100:200:7] #Lekérjük a megfelelő elemeket a listából
```

Out[4]:

```
array([29, 65, 46, 11, 79, 48, 75, 93, 22, 20, 86, 32, 35,  7, 49])
```

Megoldás (1 sorban megoldva)

In [5]:

```
randint(0,100,1000)[100:200:7] #A generált lista megfelelő elemeét hívjuk meg
#Magyarázat:  A randint egy listát ad, aminek hívatkozhatunk az elemeire
```

Out[5]:

```
array([55, 51, 68, 43, 47, 90, 55, 87, 66,  1, 62, 44, 97, 80,  8])
```

Írjunk egy olyan programot, ami megéri a felhasználót, hogy adjon egy alsó és egy felső határt, majd kirajzol egy $log10(a * x) + b$ függvényt a definiált tartományban. (Itt a és b is felhasználótól kért paraméter)

Megoldás

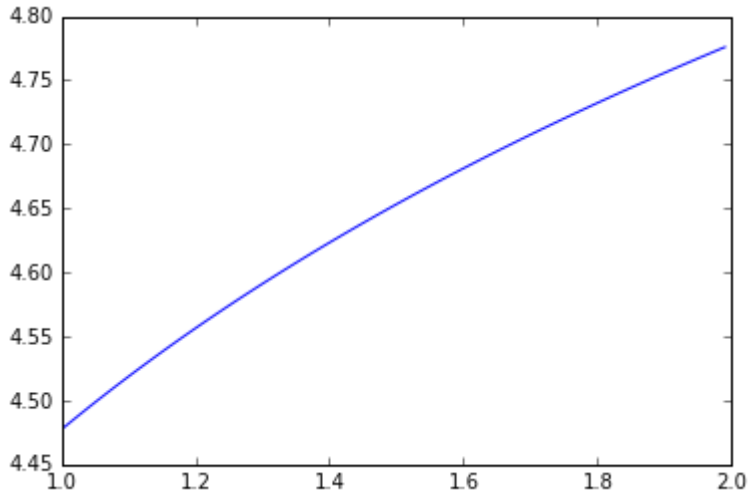
In [6]:

```
# Az input mindig szöveget olvas be, és nekünk át kell konvertálni számmá!
mini=float(input("Kezdőpont")) #bekérjük a kezdő értéket
maxi=float(input("Végpont")) #bekérjük a végső értéket
a=float(input("a")) # bekérjük az "a" értéket
b=float(input("b")) # bekérjük a "b" értéket
xlim(mini, maxi) # megadjuk az ábrázolási tartományt (x tengelyen)
x=arange(mini,maxi, 0.01) # 0,01-s lépésközzel megadom az x tengely értékeit
plot(x,(log10(a*x)+b)) # x-et és az egyenletet ábrázoljuk
```

```
Kezdőpont1
Végpont2
a3
b4
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x7effed40b048>]
```



Megoldás

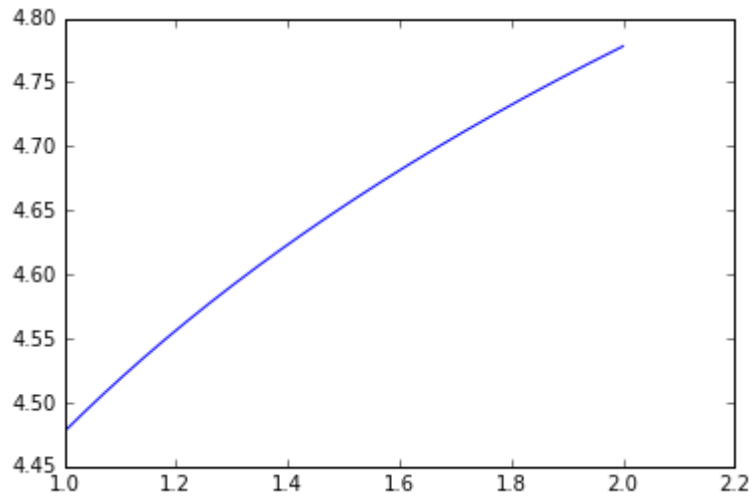
In [7]:

```
mini=float(input("Kezdőpont"))
maxi=float(input("Végpont"))
a=float(input("a"))
b=float(input("b"))
x=linspace(mini, maxi,1000) # Generálunk 1000 számot a tartományban egyenlő lépésköz
zel
plot(x,log10(a*x)+b) # x-et és az egyenletet ábrázoljuk
```

```
Kezdőpont1
Végpont2
a3
b4
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x7effd40d8e48>]
```



Megoldás

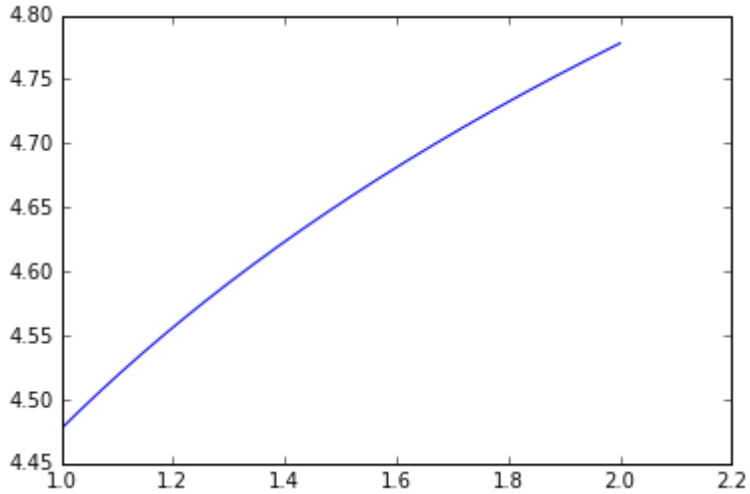
In [8]:

```
mini=float(input("Kezdőpont"))
maxi=float(input("Végpont"))
a=float(input("a"))
b=float(input("b"))
plot(linspace(mini,maxi, 1000),log10(a*x)+b) # Beírható a generálás közvetlen a plot
-ba
```

```
Kezdőpont1
Végpont2
a3
b4
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x7effcd151ef0>]
```



Megoldás

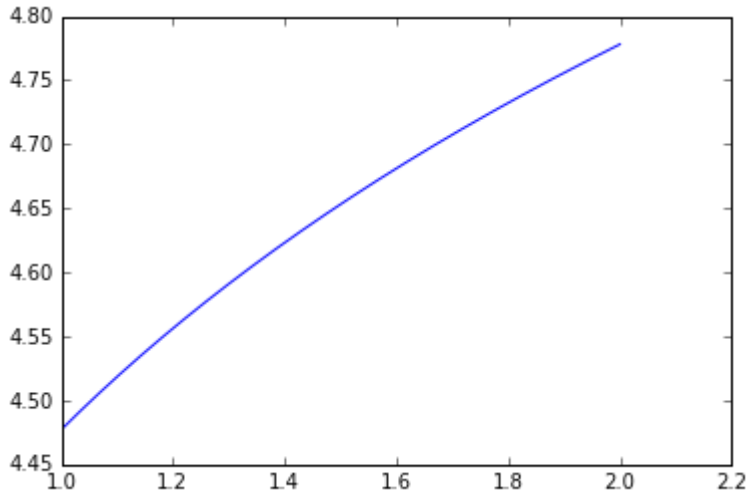
In [9]:

```
mini=float(input("Kezdőpont"))
maxi=float(input("Végpont"))
a=float(input("a"))
b=float(input("b"))
def func(x,a,b): # létrehozok egy függvényt, ami elvégzi a kért műveletet
# A függvény megkapja az "x", "a" és "b" értékeket!
    y=log10(a*x)+b # kiszámoljuk az y-t
    return y # a függvény visszatér az ny értékkel
x=linspace(mini,maxi, 1000)
plot(x,func(x,a,b)) #ábárolzjuk az x-et és a függvényt
```

```
Kezdőpont1
Végpont2
a3
b4
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x7effcd1897b8>]
```



Megoldás

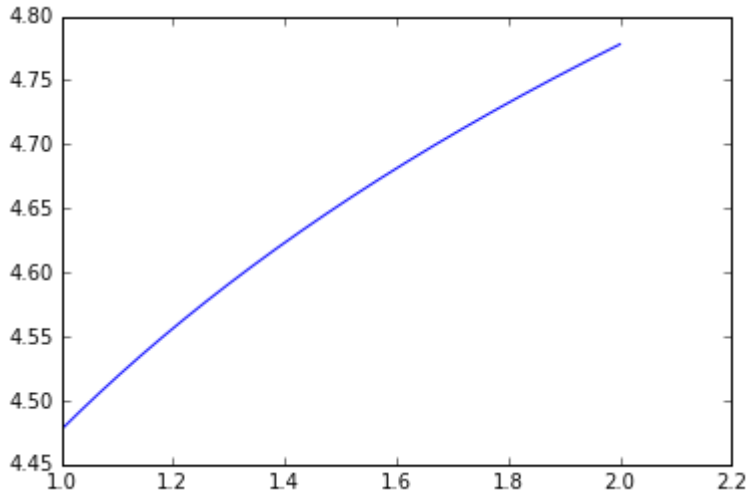
In [10]:

```
mini=float(input("Kezdőpont"))
maxi=float(input("Végpont"))
a=float(input("a"))
b=float(input("b"))
def func(x,a,b):
    y=log10(a*x)+b
    return y
x=linspace(mini,maxi, 1000)
adat=(a,b) #létrehozok egy "tuple" listát az a,b paraméterekből
plot(x,func(x,*adat)) # a paraméterekkel hívjuk mega  függvényt
```

```
Kezdőpont1
Végpont2
a3
b4
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x7effcebbd278>]
```



Írjunk olyan programot, ahol egy 2 dimenziós rácson haladunk előre (0,0-ból indulva). X vagy Y koordináta változhat, 0 vagy +1 értékkel. Használjuk a random.choice függvényt a megoldáshoz. Tegyünk meg 100 lépést. Minden lépést mentsünk egy 100 x 2-es tömbbe, és ábrázoljuk a mozgást.

Megoldás

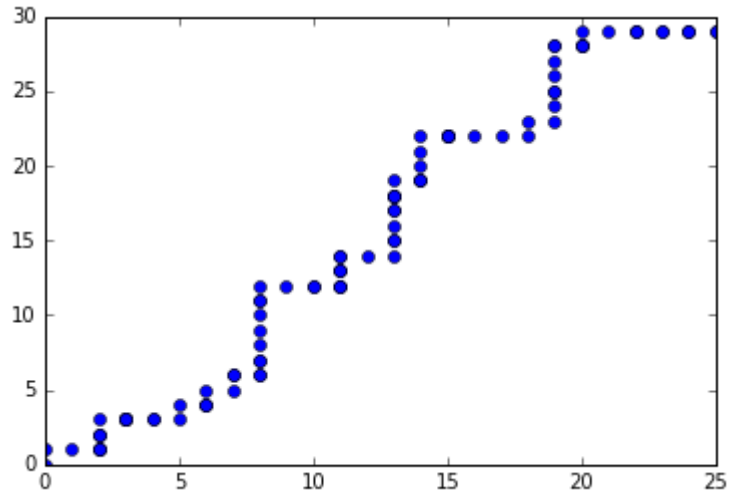
- Létrehozunk egy tömböt, melynek van 100 sor és 2 oszlopa (csupa nullával töltjük fel)
- Végigmegyünk a tömb elemein (ciklussal)
- Először az adott sorba beírjuk az előző sor értékeit, azaz megmondjuk, hogy honnan fogunk továbblépni (emlékezzünk mindenhova alapból 0 van beírva!) !! Ezért kell 1-től indítani a cilust és nem nullától !!
- Kiválasztjuk a lépés irányát és nagyságát
- Csinálunk egy elágazást, ahol megadjuk, hogy merre lépünk
- Az adott lépést elvégezzük a megadott irányban (itt a 0. oszlop az x irány és az 1. oszlop az y irányt jelenti)
- Ábrázoljuk a mozgást

In [11]:

```
b=zeros((100,2))
for i in arange(1,100):
    b[i,:]=b[i-1,:]
    irany=random.choice(['X','Y'])
    lepes=random.choice([0,1])
    if irany == "X":
        b[i,0]=b[i,0]+lepes
    else:
        b[i,1]=b[i,1]+lepes
plot(b[:,0], b[:,1], "o")
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x7effce676908>]
```



Megoldás

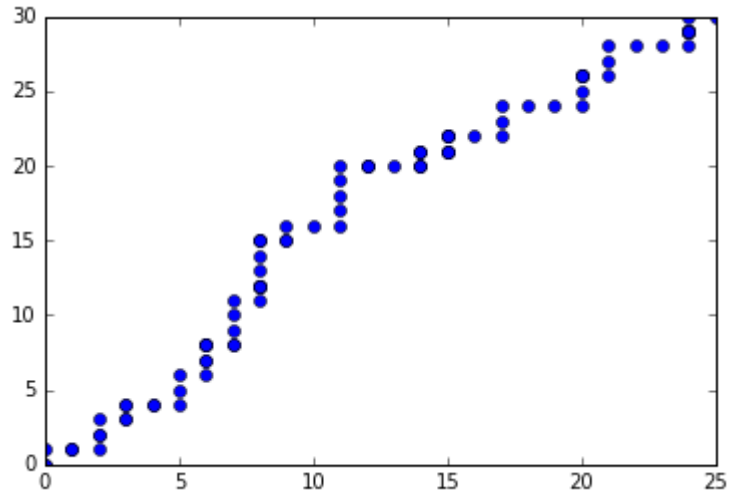
- Létrehozok egy 2 elemű vektort, 0-0 elmekkel
- Létrehozunk egy tömböt, melynek van 100 sor és 2 oszlopa (csupa nullával töltjük fel)
- Végigmegyünk a tömb elemein (ciklussal)
- Kiválasztjuk a lépés irányát
- Csinálunk egy elágazást, ahol megadjuk, hogy merre lépünk
- A vektor kiválasztott elemének helyére véletlenül választunk lépésnagyságot és ezzel megnöveljük a nagyságát. (itt a 0. oszlop az x irány és az 1. oszlop az y irányt jelenti)
- Beírjuk a tömb adott sorába az aktuális kordinétapárost
- Ábrázoljuk a mozgást

In [12]:

```
a=[0,0]
b=zeros((100,2))
for i in arange(0,100):
    irany=random.choice(['X','Y'])
    if irany == "X":
        a[0]=a[0]+random.choice([0,1])
    else:
        a[1]=a[1]+random.choice([0,1])
    b[i,:]=a[:]
plot(b[:,0], b[:,1], "o")
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x7effcd1b8ac8>]
```



Megoldás

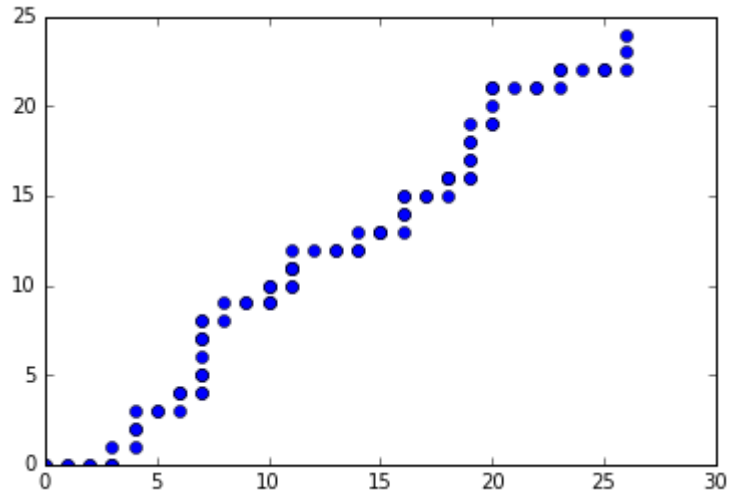
- Létrehozunk egy tömböt, melynek van 100 sor és 2 oszlopa (csupa nullával töltjük fel)
- Végigmegyünk a tömb elemein (ciklussal)
- Először az adott sorba beírjuk az előző sor értékeit, azaz megmondjuk, hogy honnan fogunk továbblépni (emlékezzünk mindenhova alapból 0 van beírva!) !! Ezért kell 1-től indítani a cilust és nem nullától !!
- Kiválasztjuk a lépés irányát
- Csinálunk egy elágazást, ahol megadjuk, hogy merre lépünk
- A megadott irányban választunk egy lépést, és el is végezzük (itt a 0. oszlop az x irány és az 1. oszlop az y irányt jelenti)
- Ábrázoljuk a mozgást

In [13]:

```
b=zeros((100,2))
for i in arange(1,100):
    b[i,:]=b[i-1,:]
    irany=random.choice(['X','Y'])
    if irany == "X":
        b[i,0]=b[i,0]+random.choice([0,1])
    else:
        b[i,1]=b[i,1]+random.choice([0,1])
plot(b[:,0], b[:,1], "o")
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x7effcceb6da0>]
```



Megoldás

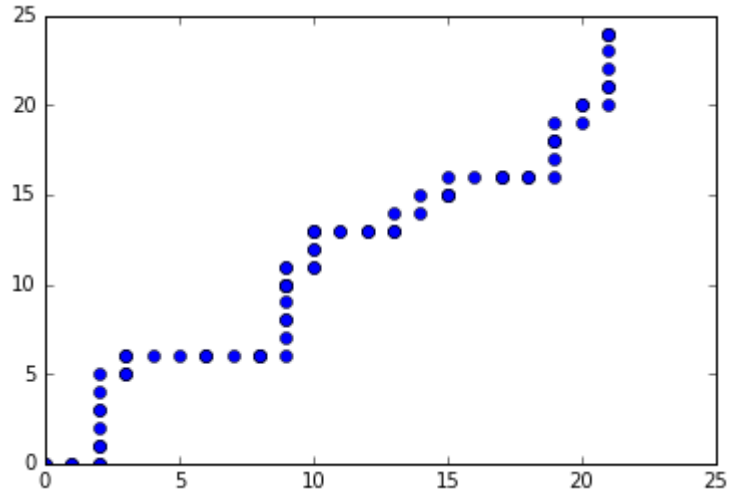
- Létrehozunk egy tömböt, melynek van 100 sor és 2 oszlopa (csupa nullával töltjük fel)
- Végigmegyünk a tömb elemein (ciklussal)
- Először az adott sorba beírjuk az előző sor értékeit, azaz megmondjuk, hogy honnan fogunk továbblépni (emlékezzünk mindenhova alapból 0 van beírva!) !! Ezért kell 1-től indítani a cilust és nem nullától !!
- Kiválasztjuk a lépés irányát, és nagysagat egyben, úgy hogy felismerjük, hogy az irány is lehet 0-1, és így 2-szer kell ugyanaból választani (itt a 0. oszlop az x irány és az 1. oszlop az y irányt jelenti)
- A mátrix megadott eleméhez hozzáadjuk a választott lépés nagyságát (itt szintén a 0. oszlop az x irány és az 1. oszlop az y irányt jelenti)
- Ábrázoljuk a mozgást

In [14]:

```
xy=zeros((100,2))
for i in range(1,100):
    xy[i]=xy[i-1]
    irany,lepes=random.choice([0,1],2)
    xy[i,irany]+=lepes
plot(xy[:,0], xy[:,1], "o")
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x7effca7d80b8>]
```



Megoldás

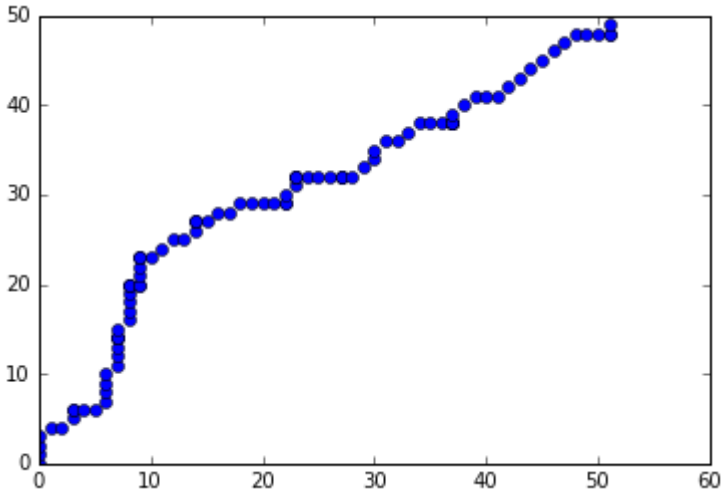
- Létrehozok egy 2 elemű vektort, 0-0 elmekkel
- Létrehozunk egy tömböt, melynek van 100 sor és 2 oszlopa (csupa nullával töltjük fel)
- Végigmegyünk a tömb elemein (ciklussal)
- Módósítjuk a "a" vektor elemét a választott helyen a választott mértékben
- A mátrix megadott sorába beírjuk az előző sor plusz az "a" vektor értékét
- Ábrázoljuk a mozgást

In [15]:

```
a=[0,0]
xy=zeros((100,2))
for i in range(1,100):
    a[random.choice([0,1])]=random.choice([0,1])
    xy[i,:]=xy[i-1,:]+a
plot(xy[:,0], xy[:,1], "o")
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x7effca7b6550>]
```



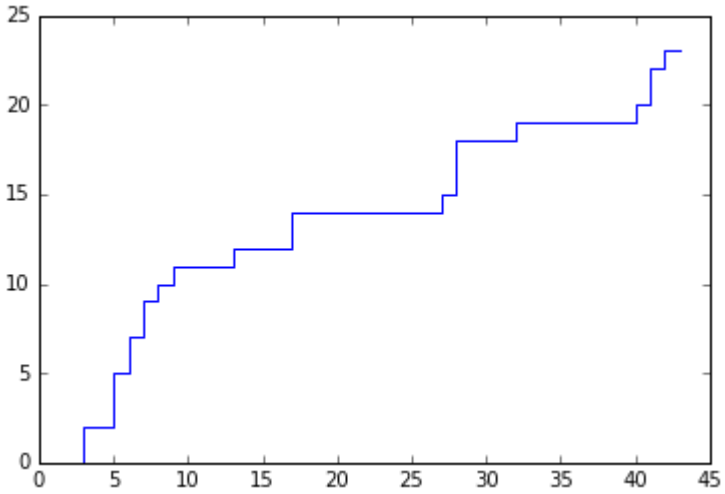
Megoldás

In [16]:

```
x,y=cumsum(list(map(eval,random.choice(('[0,0]','[0,1]','[1,0]'),100))),axis=0).T
plot(x,y)
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0x7effca7135c0>]
```



Aranymetszés aránya (aranyarány) előállítása Fibonacci-számokkal. A Fibonacci-sorozat egymást követő tagjainak hányadosából képzett sorozat (1/1, 2/1, 3/2, 5/3, …) határértéke éppen az aranymetszés arány. Határozzuk meg, hanyadik tagtól vállik a különbség 1e16-nál kisebbé (azaz 0.00000000e+00-at ír a python). Az aranymetszés pontos értéke: $\frac{1+\sqrt{5}}{2}$

Megoldás

In [17]:

```
fiblist = [0,1] # Megadjuk a sorozat első kettő elemét
for i in range(50):  # létrehozunk egy ciklust, ami kiszámolja a sorozatot
    #a sorozathoz hozzáírjuk az előző két tag összegét:
    fiblist.append(fiblist[i] + fiblist[i+1])

hanyados=[] # létrehozok egy üres listát
for i in range(2,len(fiblist)):
    # az arány listába beírom a sorozat tagjainak hányadoását
    hanyados.append((fiblist[i]) / (fiblist[i-1]))
    print(i,hanyados[i-2]-(1+sqrt(5))/2) #kiíratom a hanyados és az aranymetszés kül
önbségét
# Vegyük észre, hogy a ciklus 2-től indul (így nincs zéróval való osztás), de a
# hányadost ugyanúgy nullától kell indexelni, ezért van az "i-2" benne
```

2 -0.61803398875
3 0.38196601125
4 -0.11803398875
5 0.0486326779168
6 -0.0180339887499
7 0.00696601125011
8 -0.00264937336528
9 0.00101363029772
10 -0.000386929926365
11 0.000147829431923
12 -5.64606600073e-05
13 2.15668056607e-05
14 -8.23767693348e-06
15 3.14652861966e-06
16 -1.20186464891e-06
17 4.59071787029e-07
18 -1.75349769593e-07
19 6.69776591966e-08
20 -2.55831884566e-08
21 9.77190839357e-09
22 -3.73253694619e-09
23 1.42570222295e-09
24 -5.44569944694e-10
25 2.08007167046e-10
26 -7.94517784897e-11
27 3.03477243335e-11
28 -1.15918386001e-11
29 4.42756942221e-12
30 -1.69131375571e-12
31 6.45927755727e-13
32 -2.46691556072e-13
33 9.41469124882e-14
34 -3.59712259979e-14
35 1.37667655054e-14
36 -5.3290705182e-15
37 1.99840144433e-15
38 -8.881784197e-16
39 2.22044604925e-16
40 -2.22044604925e-16
41 0.0
42 0.0
43 0.0
44 0.0
45 0.0
46 0.0
47 0.0
48 0.0
49 0.0
50 0.0
51 0.0

Megoldás

In [18]:

```
fiblist = [0,1] # megadjuk az első 2 tagot
for i in range(2,100):
    #a sorozathoz hozzáírjuk az előző két tag összegét:
    fiblist.append(fiblist[-1] + fiblist[-2])
    hanyados = (fiblist[-1]) / (fiblist[-2]) # kiszámoljuk a hányadost
    if hanyados == (1+sqrt(5))/2: # Megvizsgáljuk mikor egyezik meg a hányados az ar
anymetszéssel
        print('it:',i)
        break # Megállítjuk a program futását
```

```
it: 41
```

Teszteljük a random.choice() függvény eloszlását véletlenszerűségét. Készíts programot, ami rendre 20, 50, 100, 200, 500, 1000 elmet választ az [1,2,3,4] tömbből.

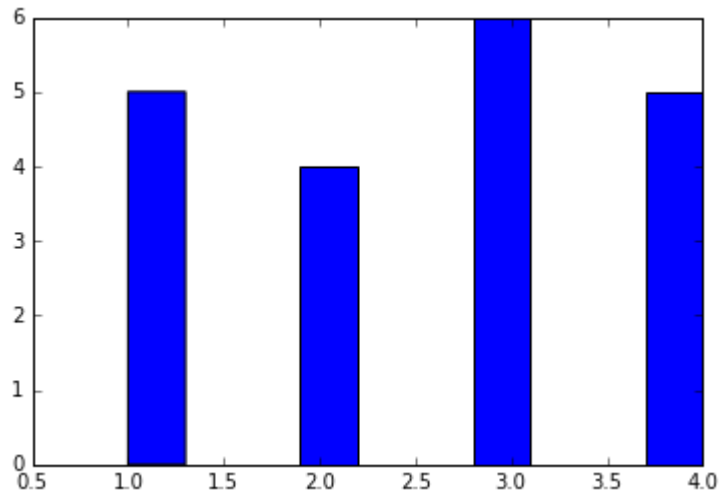
Megoldás: 5 külön kódcellában

In [19]:

```
hist(random.choice([1,2,3,4],20)) # generálok egy 20 elemű véletlen listát, és hiszt
ogrammal ábrázolom
```

Out[19]:

```
(array([ 5.,  0.,  0.,  4.,  0.,  0.,  6.,  0.,  0.,  5.]),
 array([ 1. ,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,  4.
 ]),
 <a list of 10 Patch objects>)
```

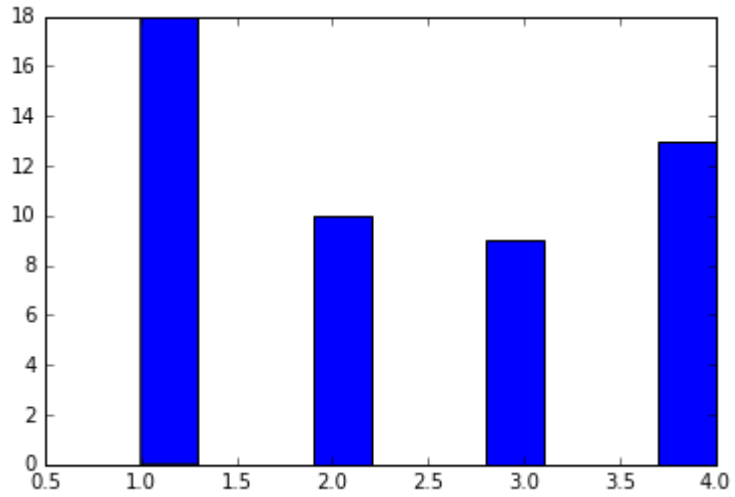


In [20]:

```
hist(random.choice([1,2,3,4],50))
```

Out[20]:

```
(array([ 18.,   0.,   0.,  10.,   0.,   0.,   9.,   0.,   0.,  13.]),
 array([ 1. ,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,  4.
 ]),
 <a list of 10 Patch objects>)
```

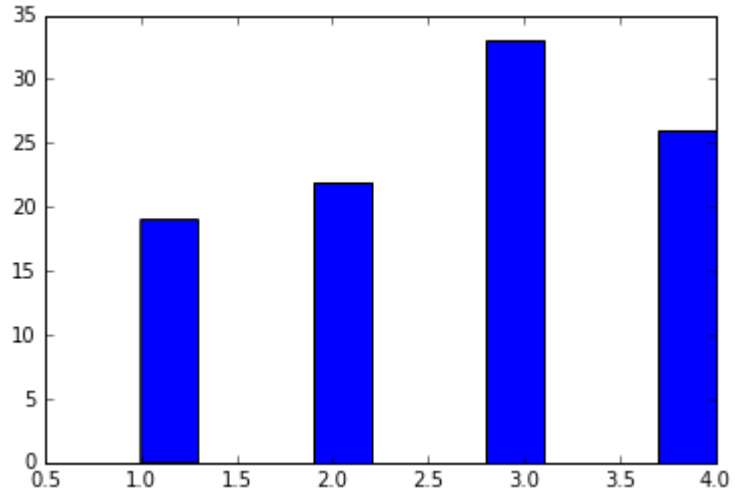


In [21]:

```
hist(random.choice([1,2,3,4],100))
```

Out[21]:

```
(array([ 19.,   0.,   0.,  22.,   0.,   0.,  33.,   0.,   0.,  26.]),
 array([ 1. ,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,  4.
 ]),
 <a list of 10 Patch objects>)
```

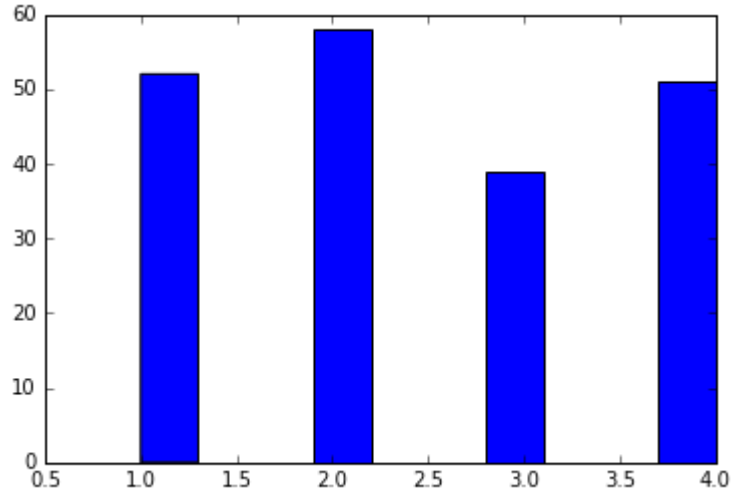


In [22]:

```
hist(random.choice([1,2,3,4],200))
```

Out[22]:

```
(array([ 52.,   0.,   0.,  58.,   0.,   0.,  39.,   0.,   0.,  51.]),
 array([ 1. ,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,  4.
 ]),
 <a list of 10 Patch objects>)
```

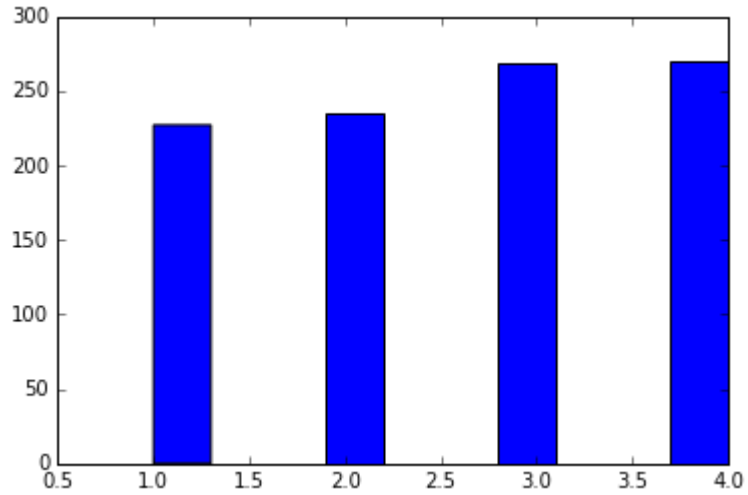


In [23]:

```
hist(random.choice([1,2,3,4],1000))
```

Out[23]:

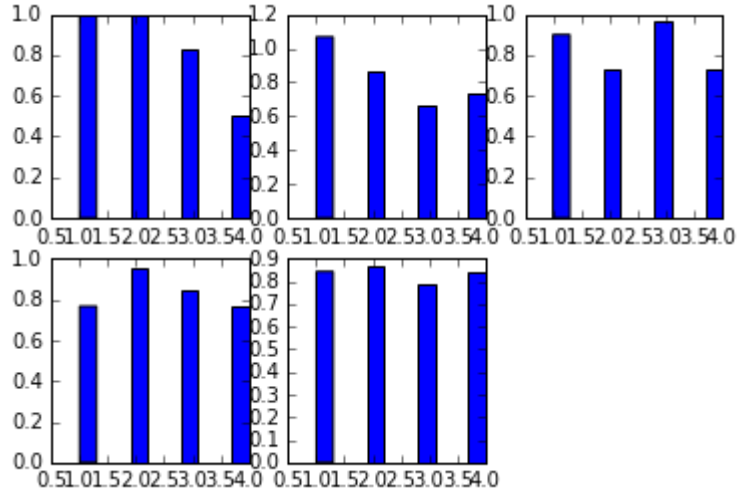
```
(array([ 227.,    0.,    0.,  235.,    0.,    0.,  268.,    0.,    0.,
 270.]),
 array([ 1. ,  1.3,  1.6,  1.9,  2.2,  2.5,  2.8,  3.1,  3.4,  3.7,  4.
 ]),
 <a list of 10 Patch objects>)
```



Megoldás több ciklussal

In [24]:

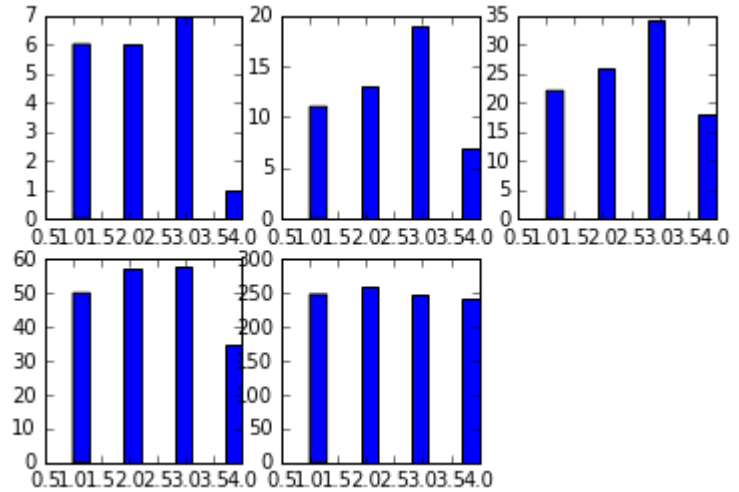
```
j=1 # legyen j változó 1
for n in [20,50,100,200,1000]: # menjen egy ciklus végig a az elemszámok listáján
    a=[] # legyen "a" egy üres vektor
    for i in range(n): # egy ciklus végigmegy az adott elemszámig
        a.append(random.choice([1, 2, 3, 4])) # folyamatosan hozzáírok az "a" vektor
hoz
    subplot(2,3,j) # suboplotban beállítom, hogy hanyadik részképet ábrázolom
    hist(a,normed=1) # elkészítem a hisztogramot
    j=j+1 # megnövelem j-t, hogy másik részképet válasszak
```



Megoldás 1 ciklussal

In [25]:

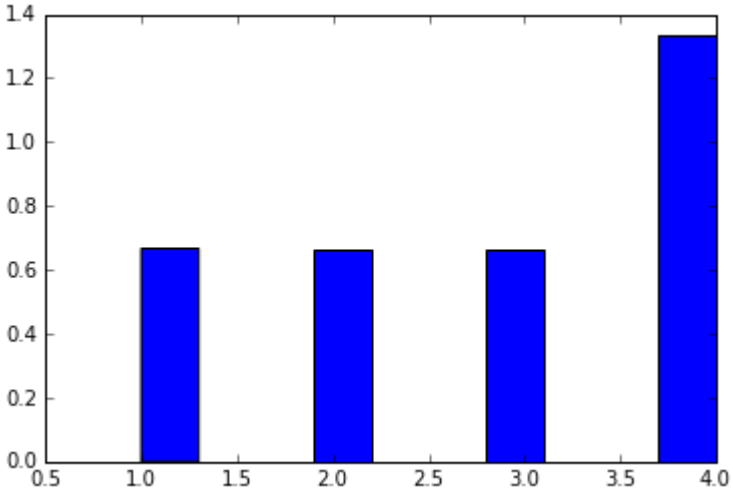
```
n=[20,50,100,200,1000] # létrehozom az elemszámok listáját
a=random.choice([1,2,3,4],1000) # generálok egy 1000 elemű véletlen listát
for j in range(0,len(n)): # egy ciklus anyiszor ismétlődik, aminnyi elemt tartalmaz
 az "n" lista
    subplot(2,3,j+1) # használjuk a subplot parancsot, és mindig új részegységben ra
jzolunk
    hist(a[0:n[j]]) # az adott darabszámú részlistáról csinálunk hisztogramot
```

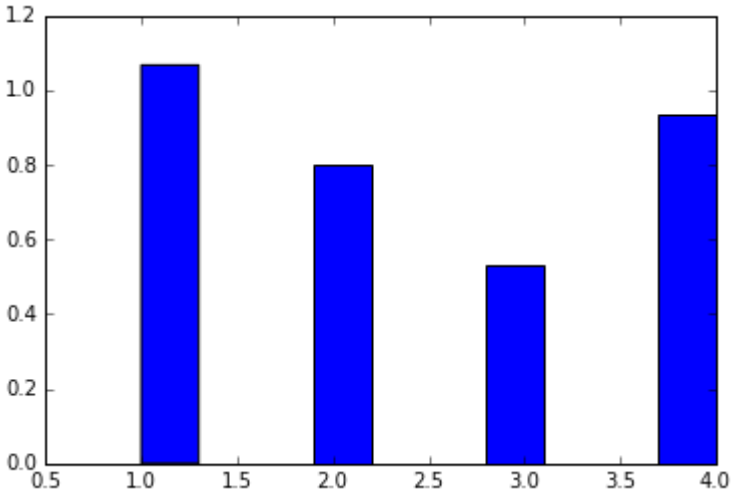


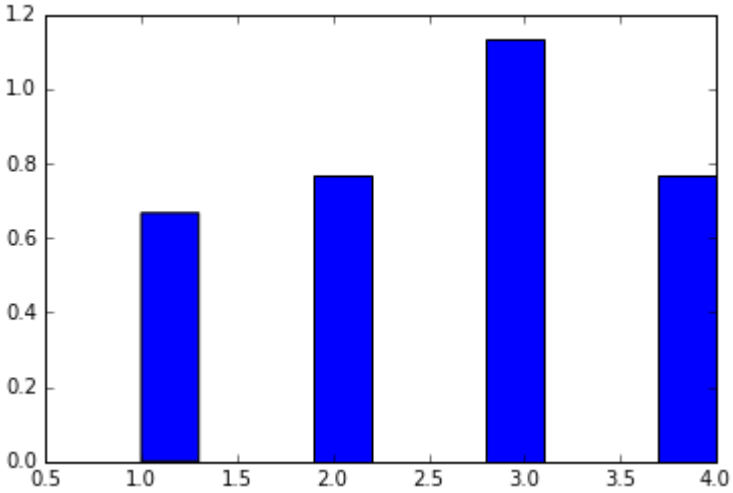
Megoldás elegánsan

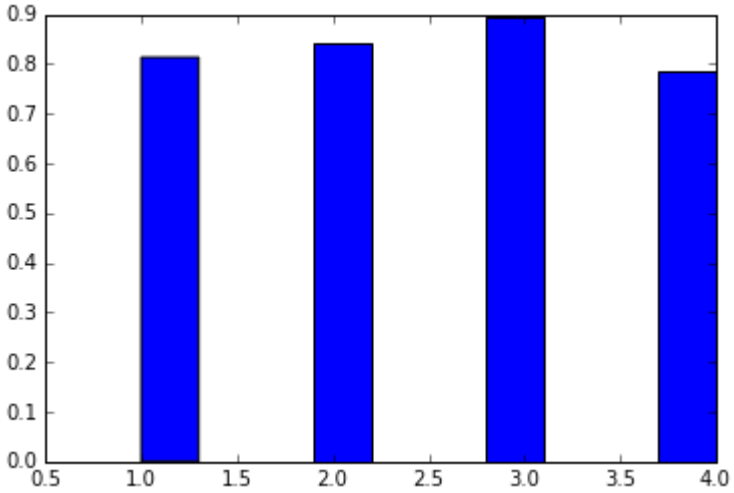
In [26]:

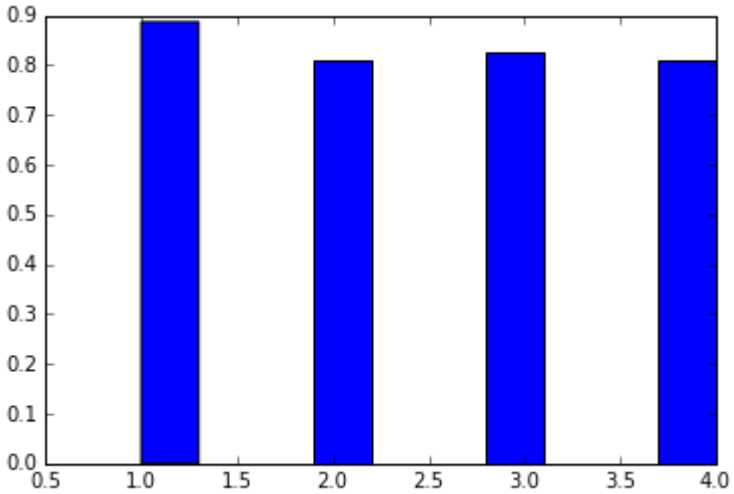
```
for n in [20,50,100,500,1000]: # egy ciklus végigmegy az elemszámok listáján
    figure() #mindig új ábrát fog nyitni
    # hasonlóan működik, mint a subplot, de itt nem egy képet osztok fel kis részekr
e,
    # hanem magát egy teljesen új képet használok (mintha egy új cellát használnék a
z ábrázolásra)
    hist(random.choice([1, 2, 3, 4],n) ,normed=True) #hisztogram készítése
```


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0