6. Fun with square-roots

In [1]:

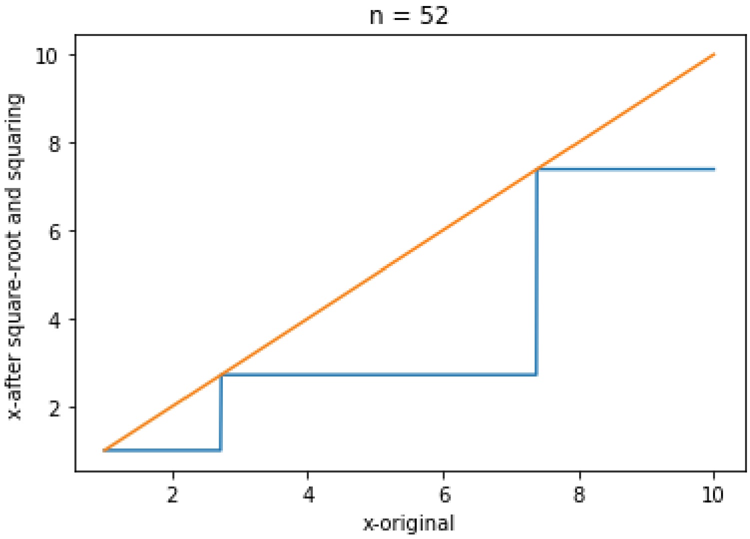
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib notebook

x = np.linspace(1,10,1001)
y = x
y2 = np.zeros(len(x))

n = 52

#carrying out square-roots

for i in range(n):
    for j in range(len(x)):
        y2[j] = np.sqrt(y[j])

    y = y2

#carrying out squares

for i in range(n):
    for j in range(len(x)):
        y2[j] = y[j]*y[j]

    y = y2
```

In [2]:

```
plt.figure()
plt.xlabel('x-original')
plt.ylabel('x-after square-root and squaring')
plt.title('n = 52')
plt.plot(x,y)
plt.plot(x,x)
```

Out[2]:

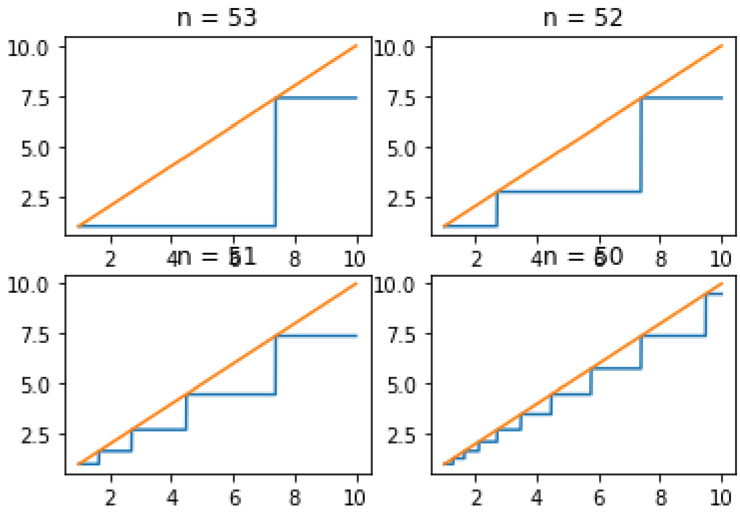
```
[<matplotlib.lines.Line2D at 0x7f2ea2b12d68>]
```



Here on inspecting the plot, we find that the values that stay what we expect it to be are: e^0, e^1 and e^2.

In [3]:

```
plt.figure()
n_steps = [50,51,52,53]

for n in n_steps:

    x = np.linspace(1,10,1001)
    y = x
    y2 = np.zeros(len(x))

    #carrying out square-roots

    for i in range(n):
        for j in range(len(x)):
            y2[j] = np.sqrt(y[j])

        y = y2

    #carrying out squares

    for i in range(n):
        for j in range(len(x)):
            y2[j] = y[j]*y[j]

        y = y2

    plt.subplot(2,2,54-n)
    #plt.xlabel('x-original')
    #plt.ylabel('x-after square-root and squaring')
    plt.title('n = %d'%n)
    plt.plot(x,y)
    plt.plot(x,x)
```



So from the above plots we find that each time we perform one step less square-root and squaring, the number of points that stay unchanged increases and these new values appear between the values that persisted when n was 1 more than it is at that step. The values that persist can be computed very similarly to the way we calculated the values for n = 53.