1-1 利用 Newton 法编程计算 x*e^x - 1 = 0 的根的近似值（P125例1）

In [33]:

```
import math

x = [0.5]
i = 0

def g(x):
    return x - ((x - math.exp(-1 * x)) / (1 + x))

def f_(x):
    return math.exp(x) * (1.0 + x)

while f_(x[i]) != 0.0 and math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [0.5, 0.5710204398084223, 0.5671555687441144, 0.5671432905332611]
精确值: 0.567143290533

1-2 P129例2的方法编程计算

In [34]:

```
import math

x = [0.5]
i = 0

def g(x):
    return math.exp(-1 * x)

def f_(x):
    return -1 * math.exp(-1 * x)

while f_(x[i]) != 0.0 and math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [0.5, 0.6065306597126334, 0.545239211892605, 0.5797030948780683, 0.5600646279389019, 0.5711721489772151, 0.5648629469803235, 0.5684380475700662, 0.5664094527469208, 0.5675596342622424, 0.5669072129354714, 0.5672771959707785, 0.5670673518537281, 0.5671863600876381, 0.5671188642569858, 0.5671571437076446, 0.5671354336592732, 0.5671477463306249]
精确值: 0.567147746331

2-1 P135 2

In [35]:

```
import math

x = [0.5]
i = 0

def g(x):
    return x - (f(x) / f_(x))

def f(x):
    return x*x*x + 2*x*x + 10*x - 20

def f_(x):
    return 3*x*x + 4*x + 10

while f_(x[i]) != 0.0 and math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [0.5, 1.6274509803921569, 1.3869266865984866, 1.3689026988957766, 1.368808
1104112189]
精确值: 1.36880811041

2-2 P136 3

3(1)

In [36]:

```
import math

x = [2.0]
i = 0

def g(x):
    return x - (f(x) / f_(x))

def f(x):
    return x*x*x - 3*x - 1

def f_(x):
    return 3*x*x - 3

while f_(x[i]) != 0.0 and math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [2.0, 1.8888888888888888, 1.879451566951567, 1.8793852448366712]
精确值: 1.87938524484

3(2)

In [37]:

```
import math

x = [1.0]
i = 0

def g(x):
    return x - (f(x) / f_(x))

def f(x):
    return x*x*x - 3*x - math.exp(x) + 2

def f_(x):
    return 3*x*x - 3 - math.exp(x)

while f_(x[i]) != 0.0 and math.fabs(x[i] - g(x[i])) >= 1e-5:
    x.append(g(x[i]))
    i += 1

print '迭代过程:', x
print '精确值:', x[-1]
```

迭代过程: [1.0, 1.1102230246251565e-16, 0.25, 0.24550828694659507]
精确值: 0.245508286947

P136 8

In [ ]:

```
```