第4章　循环结构

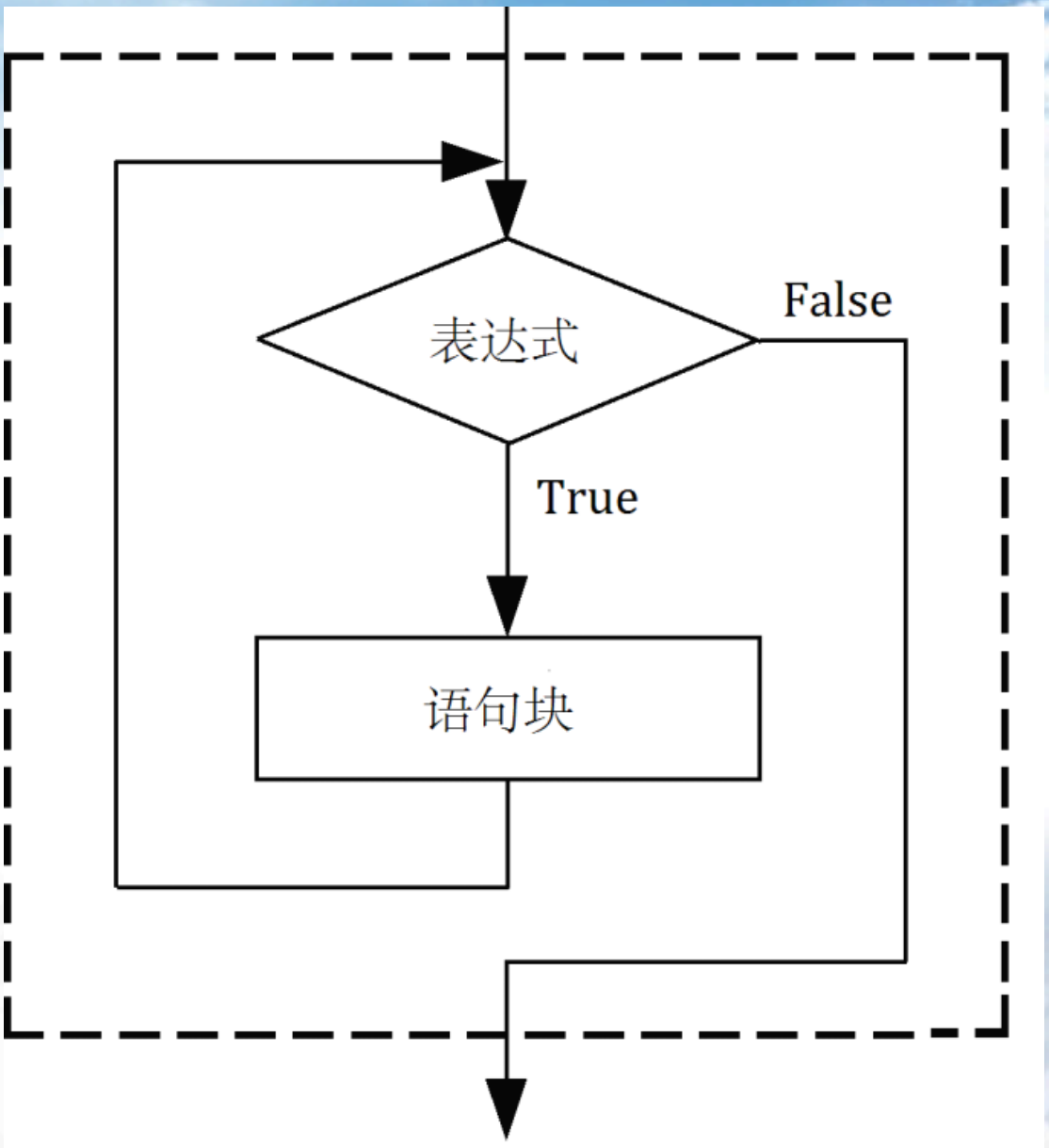
- while语句
- for语句
- 循环结构程序设计方法。

4.1 while循环结构

4.1.1 while语句

1. while语句的一般格式

while 表达式:
    语句块



while语法的执行过程

2．在while语句中使用else子句

在Python中，可以在循环语句中使用else子句，else中的语句会在循环正常执行完的情况下执行（不管是否执行循环体）。例如：

```
count=int(input())
while count<5:
    print(count,"is less han 5")
    count=count+1
else:
    print(count,"is not less than 5")
```

程序的一次运行结果如下：

```
8↙
8 is not less than 5
```

4.1.2 while循环的应用

例　计算1+2+3+……+100的值。

分析：累加问题可用递推式来描述：$s_i=s_{i-1}+n_i$（$s_0=0,n_1=1$），从循环的角度看即本次循环的累加和s等于上次循环的累加和加上本次的累加项n，可用赋值语句“s+=n”来实现。

此例的累加项n的递推式为：$n_i=n_{i-1}+1$（$n_1=1$），可用赋值语句“n+=1”来实现（n=1，2，3，……，100）。

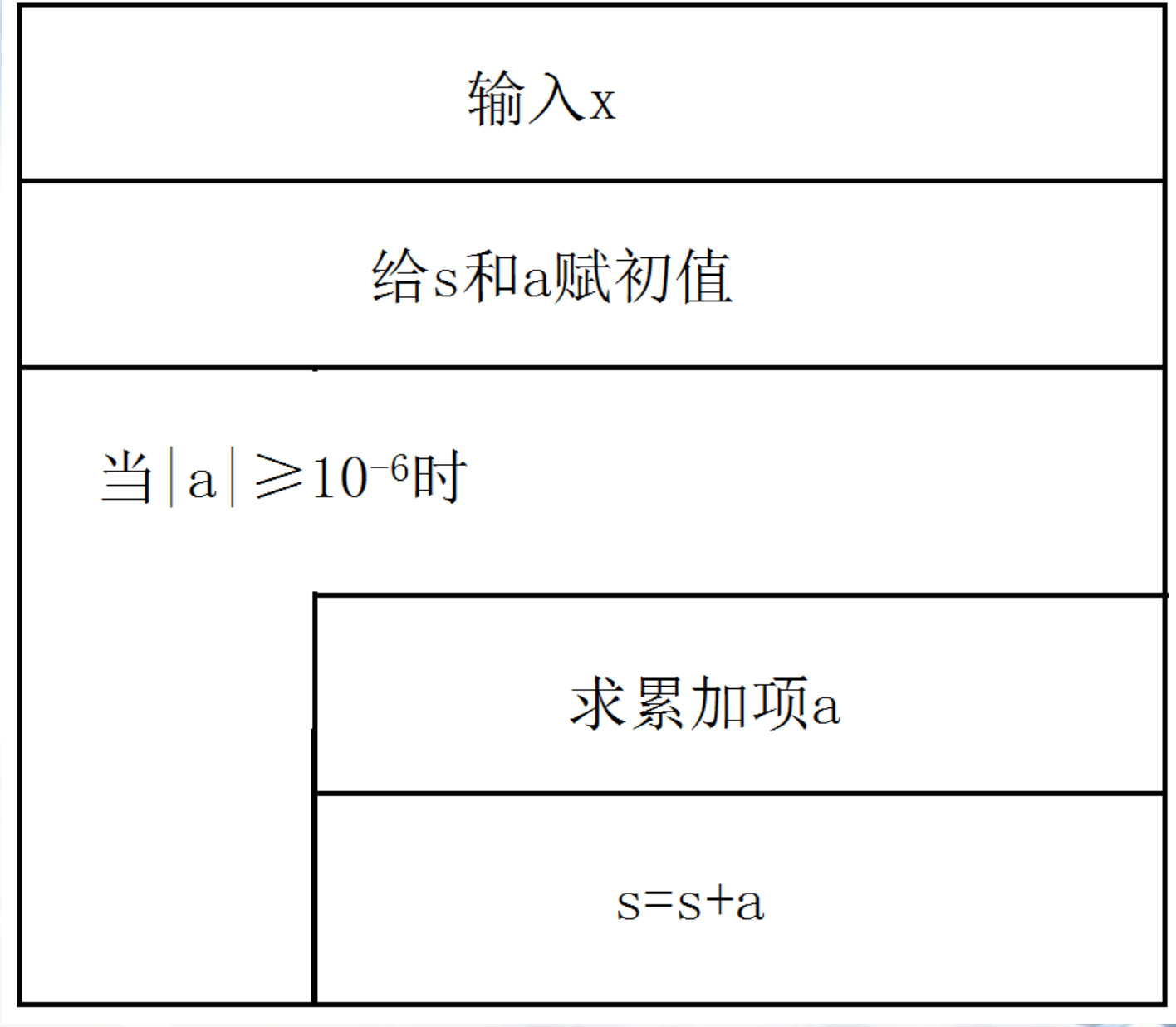
循环体要实现两种操作：s+=n和n+=1，并置s的初值为0，n的初值为1。

例　求sin*x*近似值，直到最后一项的绝对值小于10^{-6}时停止计算。其中*x*为弧度，但从键盘输入时以角度为单位。

分析：这是一个累加求和问题。关键是如何求累加项，较好的办法是利用前一项来求后一项，即用递推的办法来求累加项。第*i*项与第*i*-1项之间的递推关系为：

$a_1=x$

$$a_i=-\frac{x^2}{(2i-2)(2i-1)}a_{i-1}\ (i=2,3,4,\cdots)$$



求sin*x*值的算法

例　输入一个整数，输出其位数。

　分析：输入的整数存入变量n中，用变量k来统计n的位数，基本思路是每循环一次就去掉n的最低位数字（用Python的整除运算符实现），直到n为0。

4.2 for循环结构

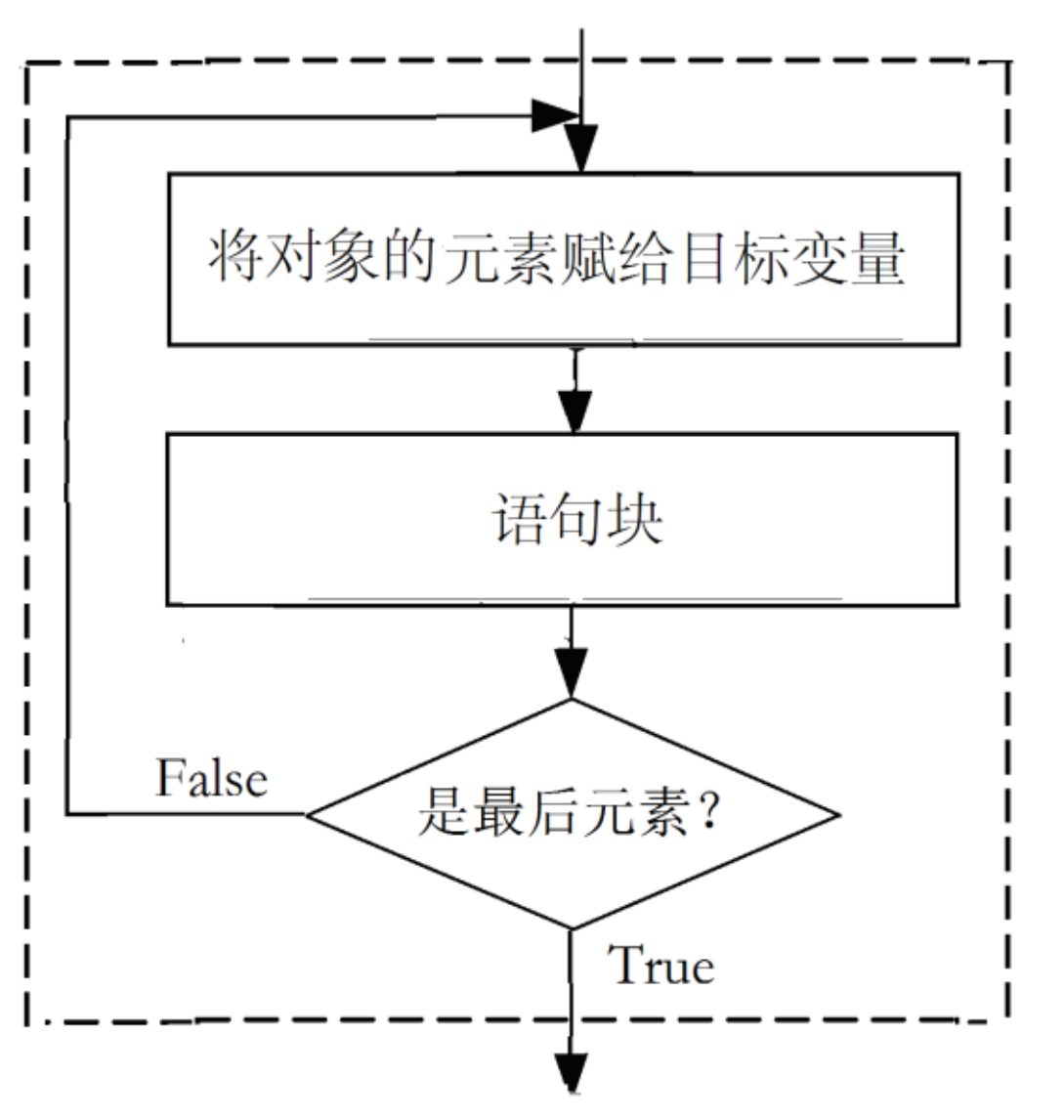
4.2.1 for语句

1. for语句的一般格式

for 目标变量 in 序列对象:

语句块

for语句的首行定义了目标变量和遍历的序列对象，后面是需要重复执行的语句块。语句块中的语句要向右缩进，且缩进量要一致。



for语句执行过程

注意：

（1）for语句是通过遍历任意序列的元素进行来建立循环的，针对序列的每一个元素执行一次循环体。列表、字符串、元组都是序列，可以利用它们来建立循环。

（2）for语句也支持一个可选的else块，它的功能就像在while循环中一样，如果循环离开时没有碰到break语句，就会执行else块。也就是序列所有元素都被访问过了之后，执行else块。

2. rang对象在for循环中的应用

在Python 3.x中，range()函数返回的是可迭代对象。Python专门为for语句设计了迭代器的处理方法。例如：

```
for i in range(5):
    print(i,end=' ')
```

程序输出结果如下：

```
0 1 2 3 4
```

首先Python对关键字in后的对象调用iter()函数获得迭代器，然后调用next()函数获得迭代器的元素，直到抛出StopIteration异常。

4.2.2 for循环的应用

例　输入20个数，求出其中的最大数与最小数。

分析：先假设第一个数就是最大数或最小数，再将剩下的19个数与到目前为止的最大数、最小数进行比较，比完19次后即可找出20个数中的最大数与最小数。

例　求Fibonacci数列的前30项。

分析：设待求项（即f_n）为f，待求项前面的第一项（即f_{n-1}）为f1，待求项前面的第二项（即f_{n-2}）为f2。首先根据f1和f2推出f，再将f1作为f2，f作为f1，为求下一项作准备。如此一直递推下去。

1 1 2 3 5

第一次：f2 + f1 →f

↓ ↓

第二次：f2 + f1 →f

↓ ↓

第三次：f2 + f1 →f

例4-6 输入一个整数m，判断是否为素数。

分析：素数是大于1，且除了1和它本身以外，不能被其他任何整数所整除的整数。为了判断整数M是否为素数，一个最简单的办法用2、3、4、5、……、M-1这些数逐个去除M，看能否整除，如果全都不能整除，则M是素数，否则，只要其中一个数能整除，则M不是素数。当M较大时，用这种方法，除的次数太多，可以有许多改进办法，以减少除的次数，提高运行效率。

可以设一个变量来作为是否素数的标志，用for语句实现程序。

4.3 循环控制语句

4.3.1 break语句

break语句用在循环体内，迫使所在循环立即终止，即跳出所在循环体，继续执行循环结构后面的语句。

例　求两个整数a与b的最大公约数。

分析：找出a与b中较小的一个，则最大公约数必在1与较小整数的范围内。使用for语句，循环变量i从较小整数变化到1。一旦循环控制变量i同时整除a与b，则i就是最大公约数，然后使用break语句强制退出循环。

4.3.2 continue语句

当在循环结构中执行continue语句时，立即结束本次循环，重新开始下一轮循环。

例　求1～100之间的全部奇数之和。

程序如下：

```
x=y=0
while True:
    x+=1
    if not(x%2):continue
    elif x>100:break
    else:y+=x
print("y=",y)
```

4.3.3 pass语句

pass语句是一个空语句，它不做任何操作，代表一个空操作。看下面的循环语句。

```
for x in range(10):
    pass
```

该语句的确会循环10次，但是除了循环本身之外，它什么也没做。

4.4 循环的嵌套

如果一个循环结构的循环体又包括一个循环结构，就称为循环的嵌套，或称为多重循环结构。

例　输入n，求表达式的值。

分析：这是求n项之和的问题。先求累加项a，再用语句“s=s+a”实现累加，共有n项，所以共循环n次。

求累加项a时，分母又是求和问题，也可以用一个循环来实现。因此整个程序构成一个二重循环结构。

例4 输出[100,1000]以内的全部素数。

分析：可分为以下两步。

（1）判断一个数是否为素数。

（2）将判断一个数是否为素数的程序段，对指定范围内的每一个数都执行一遍，即可求出某个范围内的全部素数。这种方法称为穷举法，也叫枚举法

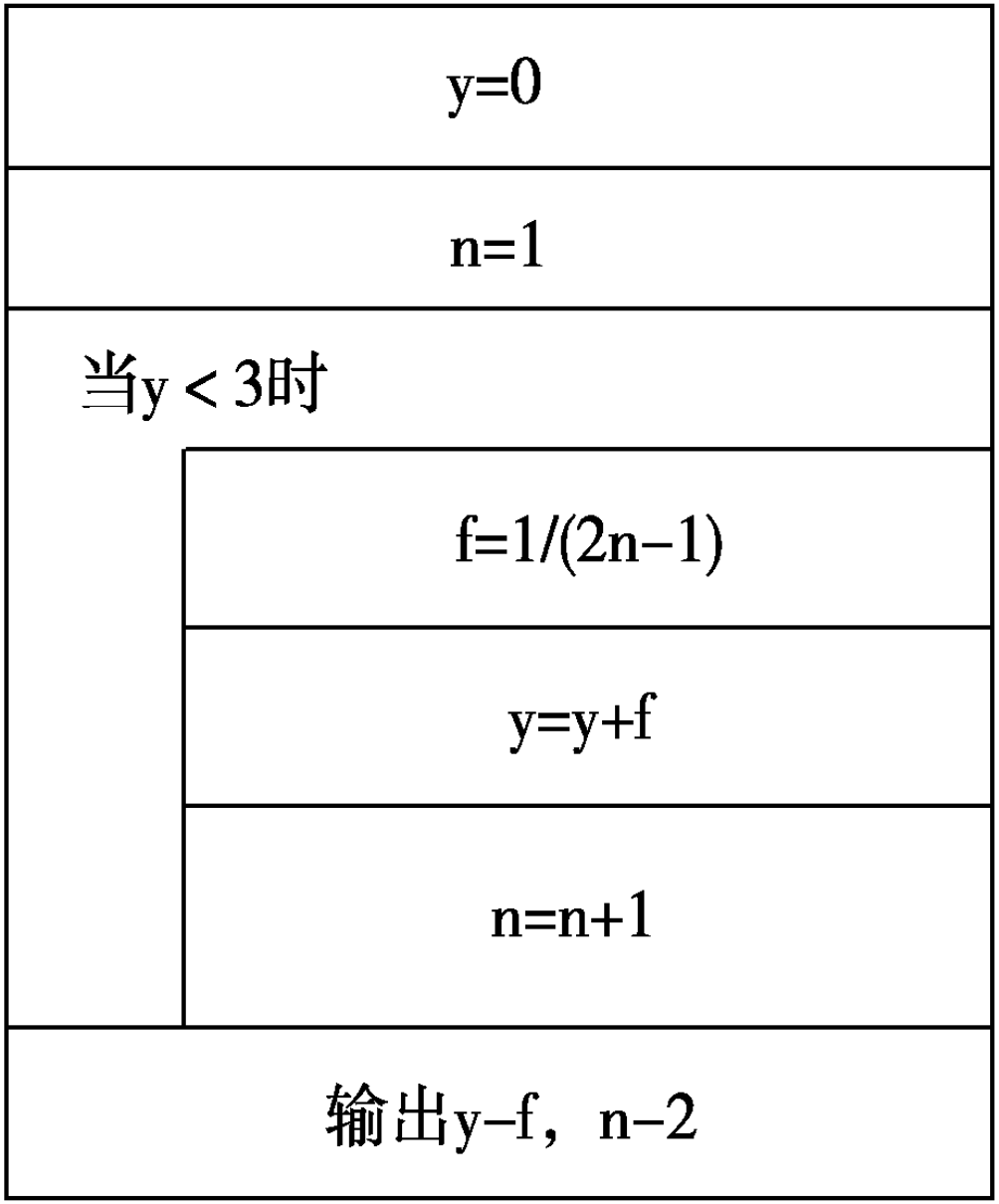
4.5 循环结构程序举例

例4 已知y，求：

（1）y<3时的最大n值。

（2）与（1）的n值对应的y值。

分析：这是一个累加求和问题，循环条件是累加和$y \geqslant 3$，用N-S图表示算法如图所示。当退出循环时，y的值已超过3，因此要减去最后一项，n的值相应也要减去1。又由于最后一项累加到y后，n又增加了1，故n还要减去1，即累加的项数是$n-2$。



求*y*值的算法

思考：

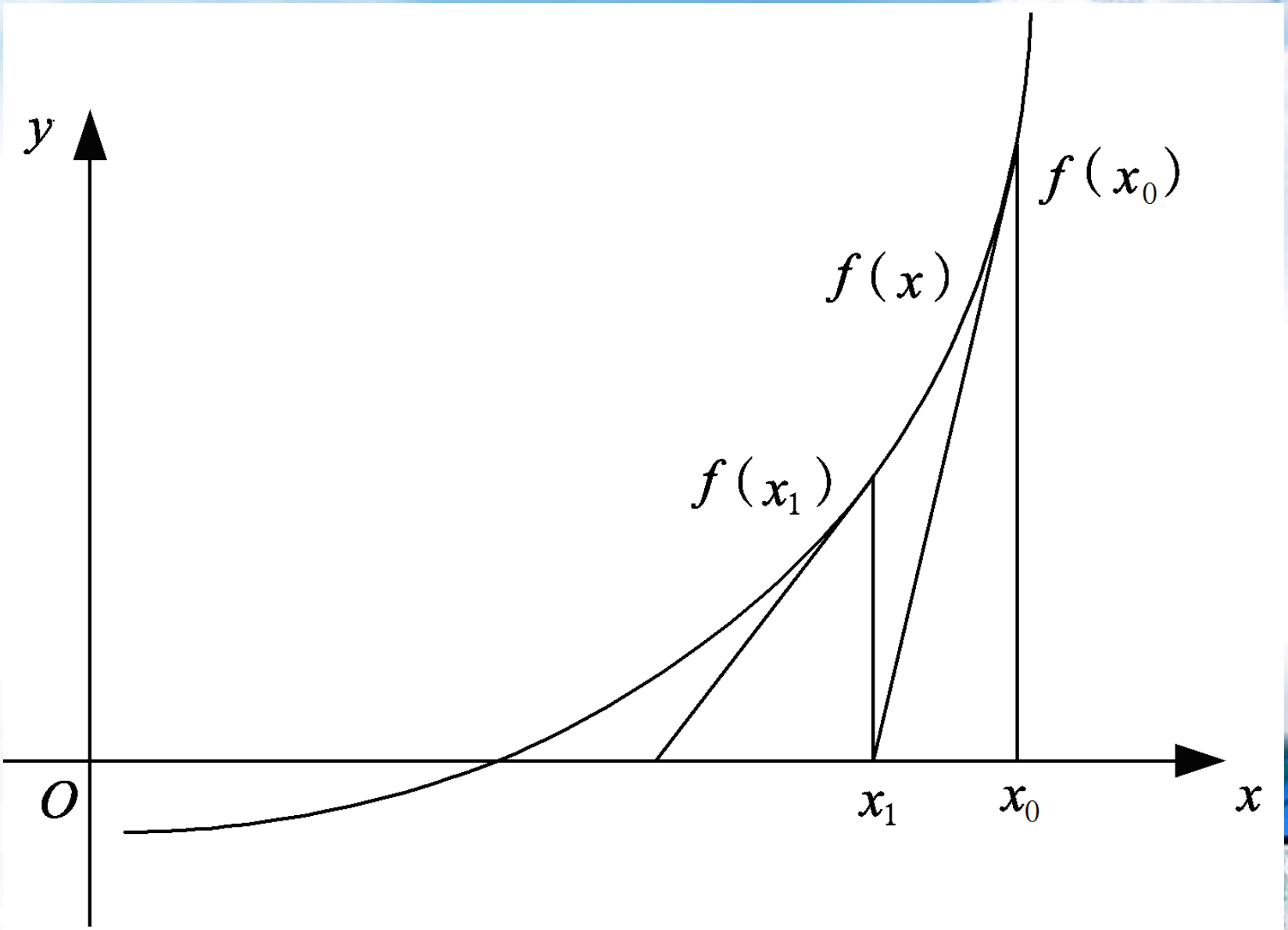
（1）求$y \geqslant 3$时的最小n，如何修改程序?

（2）求y的值，直到累加项小于10^{-6}为止，如何修改程序?

（3）n取100，求y的值，如何修改程序?

例　用牛顿迭代法求方程$f(x)=2x^3-4x^2+3x-7=0$在$x=2.5$附近的实根，直到满足$|x_n-x_{n-1}|\leqslant 10^{-6}$为止。

分析：迭代法的关键是确定迭代公式、迭代的初始值和精度要求。牛顿切线法是一种高效的迭代法，它的实质是以切线与x轴的交点作为曲线与x轴交点的近似值以逐步逼近解，如图所示。



牛顿迭代法

牛顿迭代公式为:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \qquad (n = 1,2,3\cdots)$$

例4-13 求$f(x)$在$[a,b]$上的定积分

$$\int_a^b f(x)dx$$

分析：为了求得图形面积，先将区间$[a,b]$分成n等分，每个区间的宽度为$h=(b-a)/n$，对应地将图形分成n等分，每个小部分近似一个小曲边梯形。近似求出每个小曲边梯形面积，然后将n个小曲边梯形的面积相加，就得到总面积，即定积分的近似值。n越大，近似程度越高。这就是函数的数值积分方法。

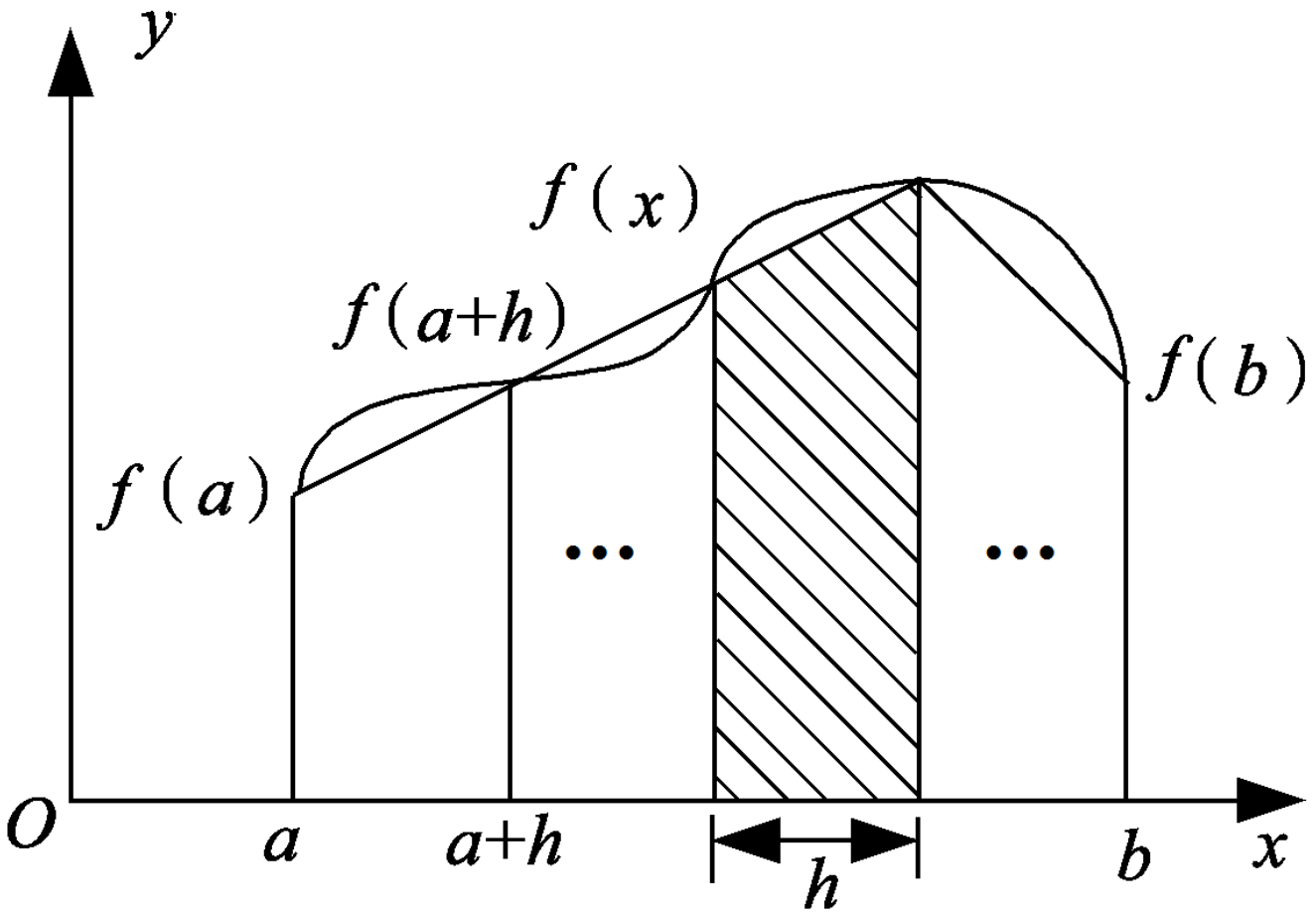
近似求每个小曲边梯形的面积，常用的方法有：

（1）用小矩形代替小曲边梯形，求出各个小矩形面积，然后累加。此种方法称为矩形法。

（2）用小梯形代替小曲边梯形，此种方法称为梯形法。

（3）用抛物线代替该区间的$f(x)$，然后求出抛物线与$x=a+(i-1)h$，$x=a+ih$，$y=0$围成的小曲边梯形面积，此种方法称为辛普生法。

以梯形法为例，求解方法如图所示。



梯形法求定积分

第一个小梯形的面积为：

$$s_1 = \frac{f(a) + f(a+h)}{2} \cdot h$$

第二个小梯形的面积为：

$$s_2 = \frac{f(a+h) + f(a+2h)}{2} \cdot h$$

……

第n个小梯形的面积为：

$$s_n = \frac{f[a+(n-1)\cdot h] + f(a+n\cdot h)}{2} \cdot h$$

例　验证哥德巴赫猜想：任何大于2的偶数，都可表示为两个素数之和。

分析：哥德巴赫猜想是一个古老而著名的数学难题，迄今未得出最后的理论证明。这里只是对有限范围内的数，用计算机加以验证，不算严格的证明。

读入偶数n，将它分成p和q，使n=p+q。p从2开始（每次加1），q=n-p。若p、q均为素数，则输出结果，否则将p+1再试。

例　将1元钱换成1分、2分、5分的硬币有多少种方法。

分析：设*x*为1分硬币数，*y*为2分硬币数，*z*为5分硬币数，则有如下方程：

$$x+2y+5z=100$$

可以看出，这是一个不定方程，没有唯一的解。这类问题无法使用解析法求解，只能将所有可能的*x*，*y*，*z*值一个一个地去试，看是否满足上面的方程，如满足则求得一组解。和前面介绍过的求素数问题一样，程序也是采用穷举法。