质数

质数又称素数。指整数在一个大于1的自然数中，**除了1和此整数自身外，没法被其他自然数整除的数。**换句话说，只有两个正因数（1和自己）的自然数即为素数。比1大但不是素数的数称为合数。1和0既非素数也非合数。素数在数论中有着很重要的作用。
质数的分布规律是以36N（N+1）为单位，随着N的增大，素数的个数以波浪形式渐渐增多。
孪生质数也有相同的分布规律。

> 素数，普遍认为的分布规律是没有规律。时而连续出现，时而又相隔很远很远。有远亲、有近邻，地球对面也还有几个好朋友。
> 素数,真的就没有规律吗?
> 合数可以用公式来表示，而素数且不能用公式来表示。这就是素数。
> 不过这里其实就蕴含着秘密。
> 既然合数能用公式表示，间接的也就说明了，素数必须服从这些公式的限制。而研究合数，其实也是研究素数。
> 有2个根深蒂固的观念：
> 1、素数的个数总是按照自然数增加10倍来统计展现的。因为这里一直沿用π(x)与x/lnx的统计方法。
> 2、100以内有25个素数，1000以内有168个素数。就产生了一种根深蒂固的观念：素数越

> 来越稀疏。
> 当然这些都没有错误，否则也不会一直陪伴着素数研究到现在，但它禁锢了人们的思想。
> 有一些数据似乎与之相悖。
> 列举一些四胞胎素数的例子，
> 四胞胎素数是很少的，在自然数1000亿以内仅仅有1209317组。平均间距为82691。两组之间相距是很远的。但总有一些间距仅仅为30的两对四胞胎素数稀稀拉拉的出现。在1000亿以内共有这样四胞胎素数267对，他们是如何分布的呢？
> 200亿以内有90个；200-400亿之间有55个；
> 剩下的如何分布的呢，你不会相信的：
> 400-600亿之间有41个；
> 600-800亿之间有41个；
> 800-1000亿之间有40个；
> 这样的分布说明了什么？均匀分布？大家肯定不会相信的，我也不信，那似乎就只能是巧合了。大家一定也会认为是这纯属巧合。素数嘛，飘忽不定，怎么分布都有可能，但就是没有规律。至少大家还没有发现其分布规律。

打印质数

打印100以内的质数

```
count = 0
for i in range(2,101):
    for x in range(2,i):
        if i%x == 0:
            break
    else:
        print(i)
        count += 1
print("\n","Total: ",count,"number")

----
2
3
5
7
11
13
17
19
23
29
31
37
41
43
```

```
47
53
59
61
67
71
73
79
83
89
97

 Total:  25 number
```

> 这种方法的思路，时间复杂度是O(n²)，2层循环，虽然会break，但效率还是很低的

优化算法，寻找中点

其实我们发现我们求解质数的时候，根本不需要从2除到N-1，当除数大于商的时候我们就不用计算了。
用数学的话来说我们只需除到平方根就好了

```
count = 0
for i in range(2,100000):
    for x in range(2,int(i**0.5)+1):
        if i%x == 0:
            break
    else:
        print(i)
        count += 1
print("\n","Total: ",count,"number")

----
2
3
5
7
11
13
.......
99907
99923
99929
99961
99971
99989
99991
```

```
 Total:  9592 number
```

继续优化，大于2的偶数都不是质数

所以对于偶数都不用判断是不是素数，修改步长

```
count = 1
print(2)
for i in range(3,100000,2):
    for x in range(2,int(i**0.5)+1):
        if i%x == 0:
            break
    else:
        print(i)
        count += 1
print("\n","Total: ",count,"number")

----
2
3
5
7
11
13
.......
99907
99923
99929
99961
99971
99989
99991

 Total:  9592 number
```

进行优化，第二层的for循环

第二层for循环判断的是奇数/range(2，奇数的开方)
但是`2,4,6,8···`这种数肯定不能被奇数整除，不用考虑，可以不加判断

```
count = 1
print(2)
for i in range(3,100000,2):
    for x in range(3,int(i**0.5)+1,2):
```

```
            if i%x == 0:
                break
        else:
            print(i)
            count += 1
print("\n","Total: ",count,"number")
```

```
----
2
3
5
7
11
13
.......
99907
99923
99929
99961
99971
99989
99991

 Total:  9592 number
```

计算以上方法的效率

计算以上的程序运行时间，取1000000以内的质数

方法一

```
from datetime import datetime
t1 = datetime.now()
count = 0
for i in range(2,1000000):
    for x in range(2,i):
        if i%x == 0:
            break
    else:
        #print(i)
        count += 1
print("\n","Total: ",count,"number")
t2 = datetime.now()
print("Total_Cost:",(t2-t1).total_seconds(),"s")
```

我喝了一杯咖啡，还没计算完…已经超过五分钟了，不等了
然后减少10倍，测试100000以内的数据…用了`40s`

方法二

```
from datetime import datetime
t1 = datetime.now()
count = 0
for i in range(2,1000000):
    for x in range(2,int(i**0.5)+1):
        if i%x == 0:
            break
    else:
        #print(i)
        count += 1
print("\n","Total: ",count,"number")
t2 = datetime.now()
print("Total_Cost:",(t2-t1).total_seconds(),"s")

----
 Total:  78498 number
Total_Cost: 6.26467 s
```

用了`6.26467s`,效率大大提升

方法三

```
from datetime import datetime
t1 = datetime.now()
count = 1
#print(2)
for i in range(3,1000000,2):
    for x in range(2,int(i**0.5)+1):
        if i%x == 0:
            break
    else:
       # print(i)
        count += 1
print("\n","Total: ",count,"number")

t2 = datetime.now()
print("Total_Cost:",(t2-t1).total_seconds(),"s")

----
Total:  78498 number
Total_Cost: 5.80345 s
```

用了`5.80345s`,效率稍微进步

方法四

```
from datetime import datetime
t1 = datetime.now()
count = 1
#print(2)
for i in range(3,1000000,2):
    for x in range(3,int(i**0.5)+1,2):
        if i%x == 0:
            break
    else:
        #print(i)
        count += 1
print("\n","Total: ",count,"number")

t2 = datetime.now()
print("Total_Cost:",(t2-t1).total_seconds(),"s")

----
 Total:  78498 number
Total_Cost: 3.375002 s
```

用了`3.375002s`,效率约提高50%

进阶方法

在上述方法四的基础上，引入列表

先把第二层循环用列表替代

```
from datetime import datetime
t1 = datetime.now()
count = 1
lst = [2]
for i in range(3,1000000,2):
    #for x in range(3,int(i**0.5)+1,2):
    for x in lst:
        if i%x == 0 and x <= i**0.5:
            break
    else:
        lst.append(i)
```

```
            count += 1
    t2 = datetime.now()
    print("Total Number:", count, "Total_Cost:", (t2-t1).total_seconds(),"s")

    ----
    Total Number: 78498   Total_Cost: 234.643142 s
```

因为每次都要if判断两次结构（a and b）,效率会低，修改下方案

修改if and 结构

```
from datetime import datetime


t1 = datetime.now()
count = 1
lst = [2]
for i in range(3,1000000,2):
    flag = False
    middle = int(i**0.5)
    for x in lst:
        if i%x == 0:
            break
        if x > middle:
            flag = True
            break
    if flag:
        lst.append(i)
        count += 1
t2 = datetime.now()
print("Total Number:", count, "Total_Cost:", (t2-t1).total_seconds(),"s")

----
Total Number: 78498 Total_Cost: 1.560107 s
```

可以可以，上面的方法四计算1000000内素数用时是3.37s，而现在，只需要1.56s,效率又提高50%以上

还能优化吗?

有一个数在做无用功，它就是2，任何素数都不能整除2

```
from datetime import datetime


t1 = datetime.now()
count = 2 #[2]
lst = [3]
```

```
    for i in range(5,1000000,2):
        flag = False
        middle = int(i**0.5)
        for x in lst:
            if i%x == 0:
                break
            if x > middle:
                flag = True
                break
        if flag:
            lst.append(i)
            count += 1
    t2 = datetime.now()
    print("Total Number:", count, "Total_Cost:", (t2-t1).total_seconds(),"s")

    ----
    Total Number: 78498 Total_Cost: 1.513069 s
```

略微提升，也有效果

还有吗?
>
在求无限质数的时候，我们不能预测有多少结果
但是对于求1000000内质数，我们现在知道了有多少结果
这样就可以提前开辟内存空间，替代append()

孪生素数

还有别的方法吗？当然有！孪生素数了解下

> 孪生素数就是指相差2的素数对，例如3和5，5和7，11和13…。孪生素数猜想正式由希尔伯特在1900年国际数学家大会的报告上第8个问题中提出，可以这样描述：存在无穷多个素数p，使得p + 2是素数。素数对（p, p + 2）称为孪生素数。

总结下来就是一句话：当素数大于3时，素数都在 `6N-1 和 6N+1 左右分布`

素数	2	3	5	7	11	13	17	19	23	25
步长			2	4	2	4	2	4	2	4

由此，在循环中用一个可变步长就可以，C语言有可变步长;
然而Python没有可变步长这一说- -

下面上代码实现

```
    from datetime import datetime
```

```
t1 = datetime.now()

count = 3 #2,3,5
lst = [3,5]
step = 4
i = 7
while i < 1000000:
    if i%5 != 0:
        middle = int(i**0.5)
        flag = False
        for x in lst:
            if i%x == 0:
                break
            if x > middle:
                flag = True
                break
        if flag:
            lst.append(i)
            count += 1

    i += step
    step = 4 if step == 2 else 2

t2 = datetime.now()
print("Total Number:", count, "Total_Cost:", (t2-t1).total_seconds(),"s")

----
Total Number: 78498 Total_Cost: 1.35155 s
```

`1.513069s` —> `1.35155s` 还可以哈哈

质数相关的理论

- 哥德巴赫猜想：任何大于5的奇数都是三个素数之和
- 衍生：任何一个大于2的偶数都是两个素数之和
- 伯特兰猜想：对于任意正整数n>1，存在一个素数p，使得n<p<2n
- 孪生素数猜想：存在无穷多的形如p和p+2的素数对
- n2+1 猜想：存在无穷多个形如n2+1的素数，其中n是正整数

> 量子计算机，了解一下……密码学、区块链都将被重新定义~