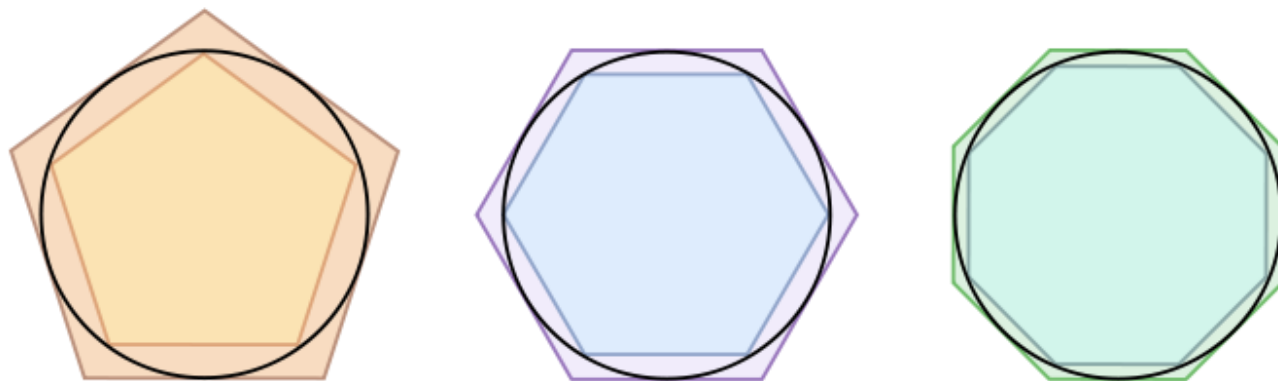


如何近似计算圆周率 π ?



```
3.141592653589793238462643383
279502884197169399375105820974944
59230781640628620899862803482534211
70679821480865132823066470938446095
50582231 725359408 128481117
45028410 270193852 1105559544
622948 954930381 9644288109
75 665933446 128475 6482
3378678716 5271201909
145648566 9284603486
1045432664 8213393607
2602491412 7372458700
66063155881 74881520920 962829
25409171536 43678925903600113305
3054882046652 1384146931941511609
43305727036575 959195309218611738
19326117931051 18548074462379962
7495673518857 527248912279381
8301194912 9833673362
44065 66430
```

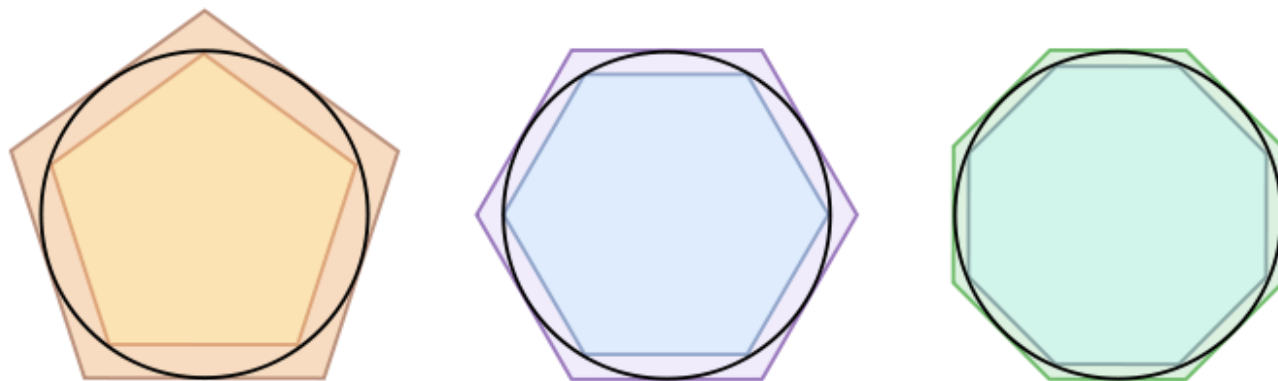
如何近似计算圆周率 π ?



π 可以透过计算圆的外切多边形及内接多边形周长来估算

```
3.141592653589793238462643383
279502884197169399375105820974944
59230781640628620899862803482534211
70679821480865132823066470938446095
50582231 725359408 128481117
45028410 270193852 1105559544
622948 954930381 9644288109
75 665933446 128475 6482
3378678716 5271201909
145648566 9284603486
1045432664 8213393607
2602491412 7372458700
66063155881 74881520920 962829
25409171536 43678925903600113305
3054882046652 1384146931941511609
43305727036575 959195309218611738
19326117931051 18548074462379962
7495673518857 527248912279381
8301194912 9833673362
44065 66430
```

如何近似计算圆周率 π ？



π 可以透过计算圆的外切多边形及内接多边形周长来估算

```
3.141592653589793238462643383
279502884197169399375105820974944
59230781640628620899862803482534211
70679821480865132823066470938446095
50582231 725359408 128481117
45028410 270193852 1105559544
622948 954930381 9644288109
75 665933446 128475 6482
3378678716 5271201909
145648566 9284603486
1045432664 8213393607
2602491412 7372458700
66063155881 74881520920 962829
25409171536 43678925903600113305
3054882046652 1384146931941511609
43305727036575 959195309218611738
19326117931051 18548074462379962
7495673518857 527248912279381
8301194912 9833673362
44065 66430
```

祖冲之在公元480年利用割圆术计算12,288形的边长，得到 $\pi \approx \frac{355}{113}$ （现在称为密率），其数值为3.141592920，小数点后的前六位数都是正确值。在之后的八百年内，这都是准确度最高的 π 估计值。为纪念祖冲之对圆周率发展的贡献，日本数学家三上义夫将这一推算值命名为“祖冲之圆周率”，简称“祖率”。

如何近似计算圆周率 π ?

$$\arctan(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{2k+1} = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \dots$$

反正切泰勒级数

如何近似计算圆周率 π ?

$$\arctan(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{2k+1} = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \dots$$

反正切泰勒级数

当 $x=1$ 时, $\arctan x = \pi/4$

$$\frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

$\arctan 1$

如何近似计算圆周率 π ?

$$\frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

arctan1



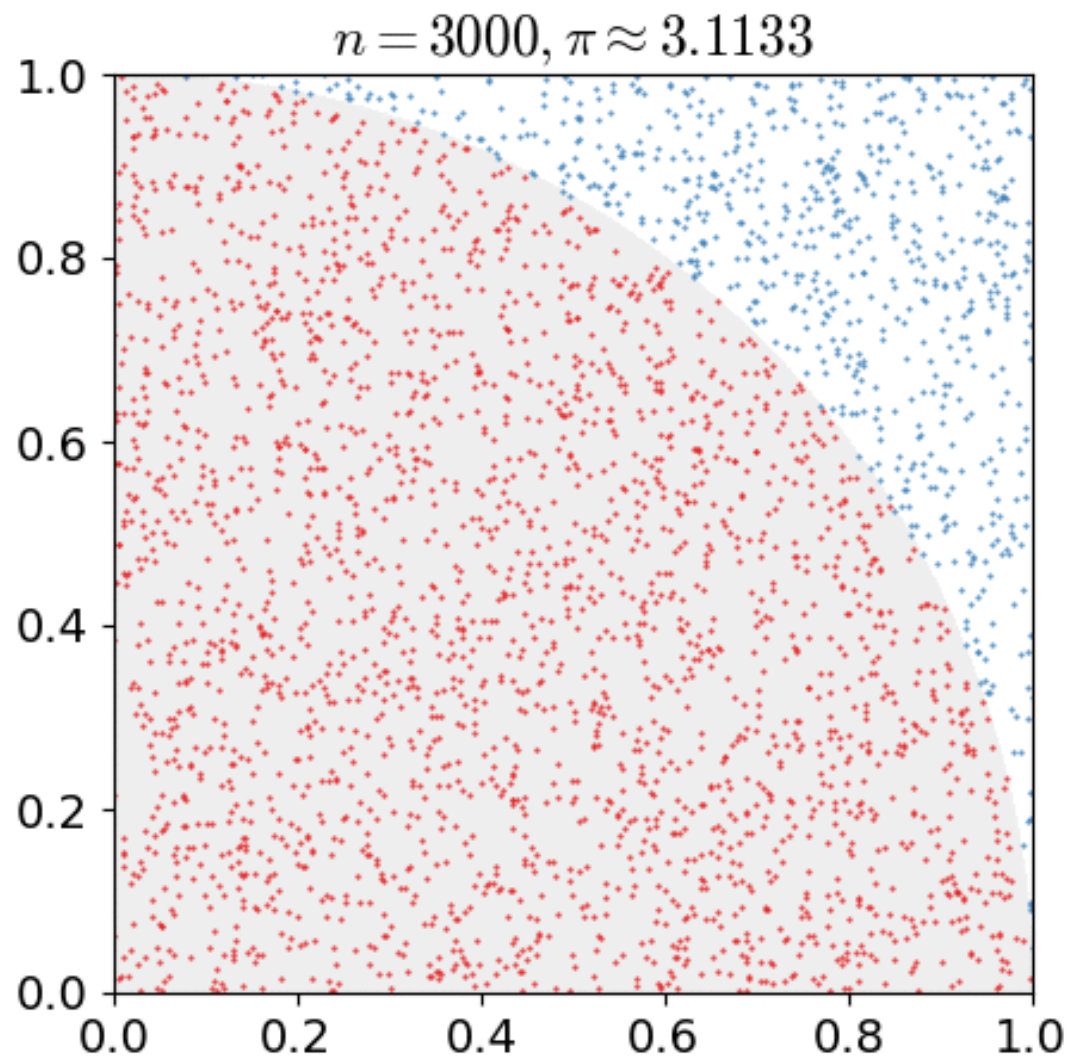
$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

如何近似计算圆周率 π ?

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

```
def TaylorPi(k):  
    sum, odd = 0, True  
    for i in range(1, k):  
        sum += 1/(2*i-1) if odd == True else -1/(2*i-1)  
        odd = not odd  
    print("Taylor PI:", sum*4)
```

如何近似计算圆周率 π ?



$$\frac{S_1}{S_2} = \frac{\pi / 4}{1} = \frac{\pi}{4} \approx \frac{N_1}{N_2}$$

$$\therefore \pi \approx \frac{4N_1}{N_2}$$

当 $N_2=30000$ 时， π 的
估计值与真实值只
相差0.07%

如何近似计算圆周率 π ?

```
1  import random
2
3  ▼ if __name__ == '__main__':
4      N2 = 30000
5      N1 = 0.
6  ▼ for i in range(N2):
7      x = random.random()
8      y = random.random()
9      if x*x+y*y<=1:
10         N1+=1
11     print("PI:", 4*N1/N2)
```

$$\frac{S_1}{S_2} = \frac{\pi/4}{1} = \frac{\pi}{4} \approx \frac{N_1}{N_2}$$

$$\therefore \pi \approx \frac{4N_1}{N_2}$$

当 $N_2=30000$ 时， π 的
估计值与真实值只
相差0.07%

如何近似计算圆周率 π ?

Chudnovsky公式

$$\pi = \frac{426880\sqrt{10005}}{\sum_{k=0}^{\infty} \frac{(6k)!(13591409 + 545140134k)}{(3k)!(k!)^3(-640320)^{3k}}}$$

这个公式可以做到每计算一项得出15位有效数字！1994年，人们利用这个公式，得到了圆周率小数点后40.44亿位。

如何近似计算圆周率 π ?

Chudnovsky公式

```
def Ch_cal(k):  
    ... uper_value = math.factorial(6*k)*(13591409+545140134*k)  
    ... lower_value = math.factorial(3*k)*math.pow(math.factorial(k),3)*math.pow((-640320),3*k)  
    ... return uper_value/lower_value  
  
def Chudnovsky(number):  
    ... uper_value = 426880*math.sqrt(10005)  
    ... lower_sum = 0.  
    ... for k in range(number):  
    ...     lower_sum+=Ch_cal(k)  
    ... print("Chudnovsky PI:", uper_value/lower_sum)
```

如何近似计算圆周率 π ?

迭代算法:

1. 设置初始值:

$$a_0 = 1 \quad b_0 = \frac{1}{\sqrt{2}} \quad t_0 = \frac{1}{4} \quad p_0 = 1.$$

2. 反复执行以下步骤直到 a_n 与 b_n 之间的误差到达所需精度:

$$a_{n+1} = \frac{a_n + b_n}{2},$$

$$b_{n+1} = \sqrt{a_n b_n},$$

$$t_{n+1} = t_n - p_n (a_n - a_{n+1})^2,$$

$$p_{n+1} = 2p_n.$$

3. 则 π 的近似值为:

$$\pi \approx \frac{(a_{n+1} + b_{n+1})^2}{4t_{n+1}}.$$

如何近似计算圆周率 π ?

迭代算法:

```
def Iterative_cal(number):  
    a_now = 1.  
    b_now = 1./math.sqrt(2)  
    t_now = .25  
    p_now = 1.  
    for i in range(number):  
        a = (a_now+b_now)/2  
        b = math.sqrt(a_now*b_now)  
        t = t_now-p_now*math.pow((a_now-a),2)  
        p = 2*p_now  
  
        a_now = a  
        b_now = b  
        t_now = t  
        p_now = p  
    print("Iterative PI:",math.pow(a_now+b_now,2)/(4*t_now))
```