# 원주율의 역사와 계산 방법

세종대학교
SEJONG UNIVERSITY

18010367 강다정

19010350 강준모

22010493 김현수

22010494 유 빈

22010495 정용진

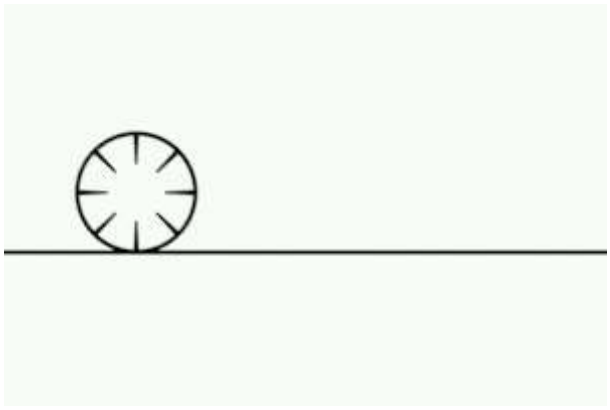
# 목 차

# 원주율의 역사와 증명 방법

## 1. 서론

원의 지름에 대한 둘레의 비를 나타내는 수 '파이'는 흥미로운 숫자다. 파이는 초등학교, 중학교, 고등학교를 다니면서 사용된다. 대학교에서도 정규분포, 등속원운동 등 파이를 활용해 수학, 물리학을 배운다. 이렇게 자주 접하는 파이는 특이하게도 순환하지 않는 무한소수이다. 따라서 우리는 근사값으로만 파이를 알고 있다. 아직도 파이의 끝을 보기 위해 계산기가 작동한다. 또한, 계산 속도를 높이기 위해 원주율을 증명하는 많은 식을 탐색 중이다. 우리는 이 매력적인 수의 다양한 증명 방식을 전개할 것이다.

아주 오래 전 옛 사람들도 파이를 인식했다. 정확한 숫자를 알진 못했지만, 원의 지름과 둘레는 일정한 비를 나타낸다고 어렴풋이 짐작하고 있었다. 기원전 3000년 즈음 발생한 메소포타미아 문명 사람들은 원주율을 3정도임을 짐작하고 있었다. 고대 이집트인은 직접 원을 굴려 둘레를 가늠해 구했다고 한다. 이 뿐 아니라 중국 문명, 성경에서도 원주율에 관한 글을 찾아 볼 수 있다.



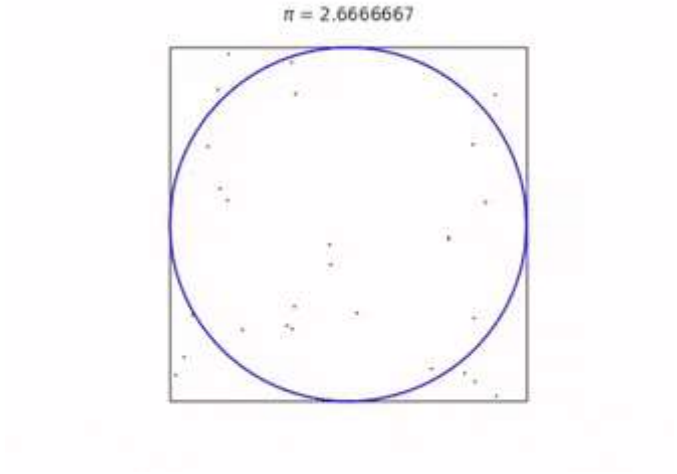
$$\pi \approx 3$$

원주율을 구하는 방법은 크게 3가지로 분류할 수 있다. 확률을 이용한 방식, 도형을 이용한 방식과 전개식을 이용한 방식이다. 본 논문에서는 이들 방법론들을 파이썬으로 구현해 보고자 한다.

## 2. 확률을 이용한 방식

### 2.1. 원의 넓이를 이용한 몬테카를로 실험

무작위 추출된 난수를 이용하여 함수의 값을 계산하는 '몬테카를로 방법'을 이용해 원주율을 구할 수 있다. 공식을 모르는 물리적인 관계성을 밝힐 때, 몬테카를로 방법은 계수를 구하는 등의 관계성 발견에 용이하다. 즉, 적분을 사용하지 않고 파이를 찾아볼 수 있다. 파이썬에서는 random 함수를 통해 몬테카를로 방법을 코딩할 수 있다.




***정사각형의 넓이 4, 원의 넓이 $\pi$***

***→ 전체 점과 원 내부의 점의 개수의 비율은 $4:\pi$로 추정***

In [1]:

```
import random
from time import time
import math

err=1e-7 #오차범위
```

In [2]:

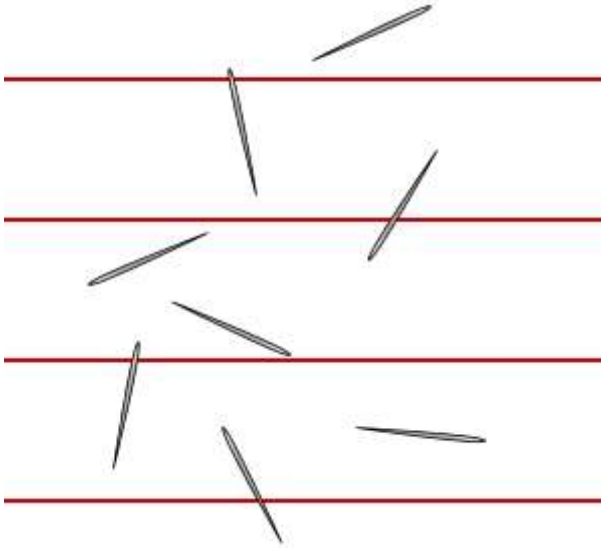
```
count=0
result=0
i=0
start=time()

while abs(math.pi-result)>err:
    x=random.random()
    y=random.random()
    if x*x+y*y<1:
        count+=1
    i+=1
    result=4*count/i
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.141592669830995, 걸린 시간: 528ms, 반복 횟수: 317919

## 2.2. 뷔퐁의 바늘 문제

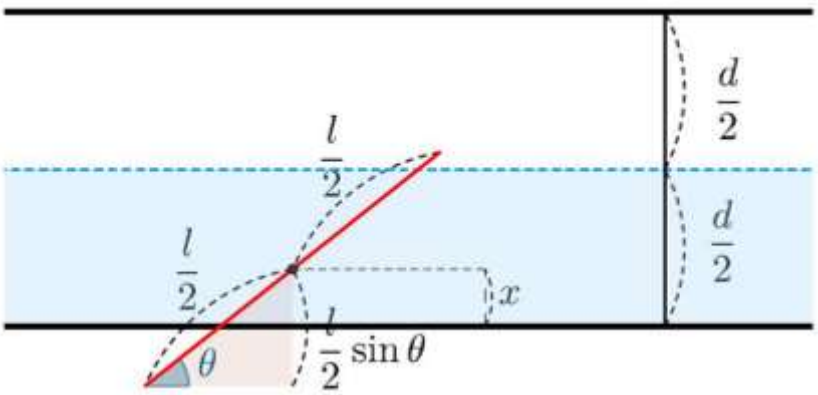
'뷔퐁의 바늘 실험'은 수학자 뷔퐁이 기하학적 확률, 정적분, 원주율이 결합되었다. 바닥에 일정한 간격의 평행선이 있을 때, 평행선과 바늘이 만날 확률을 찾는 실험이다

***바늘이 평행선과 만날 확률?***



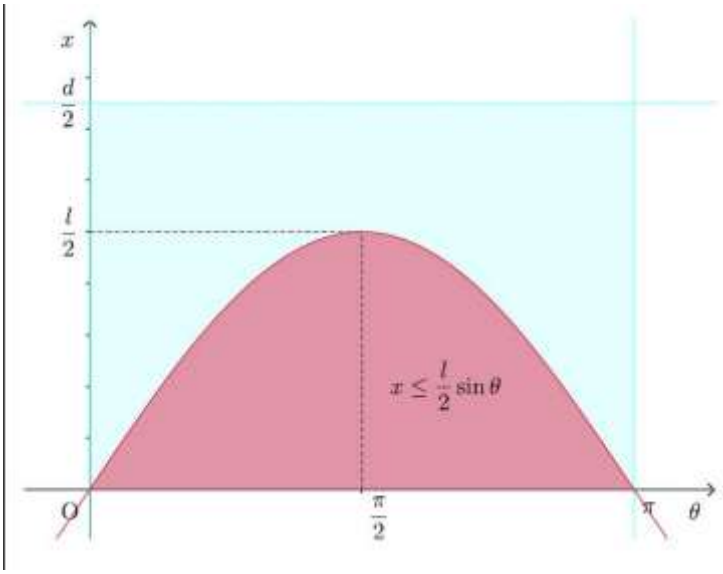
***평행선의 간격: $d$, 바늘의 길이: $l(d \geq l)$***

평행선의 간격을 d, 바늘의 길이를 l, d > l이라고 가정한다. 그림에서 x는 평행선에서 바늘의 중심까지의 거리, $\theta$는 바늘의 평행선이 오른쪽 방향과 이루는 각도다. 바늘이 평행선과 만나는 상황을 생각해보자. 바늘의 중심이 평행선과 아무리 가까이 가더라도 $\theta$가 매우 작으면 바늘은 평행선과 만날 수 없다. 반대로 $\theta$가 $\frac{\pi}{2}$에 가까운 값을 가져 거의 평행선과 수직이라고 하여도, 중심이 평행선과 멀리 떨어져 있으면 바늘과 평행선은 만날 수 없다. 즉 바늘과 평행선이 만나기 위해선 x와 $\theta$가 적절한 값을 가져야 한다. 따라서, x의 범위는 [0, d/2], $\theta$의 범위는 [0, $\pi$] 이어야 한다. 그리고 $x \leq \frac{1}{2}\sin\theta$을 만족해야 하는 것을 밑 그림을 보면 알 수 있다.




$$x \leq \frac{1}{2}\sin\theta$$

$\theta$를 가로축, x를 세로축으로 하는 좌표평면에 위 부등식의 영역을 나타내보자.




$$P = \frac{\frac{l}{2}\int_0^{\pi} \sin\theta \, d\theta}{\frac{d\pi}{2}}$$

$$= \frac{l}{d\pi}\int_0^{\pi} \sin\theta \, d\theta$$

$$= \frac{2l}{d\pi}$$

$$\therefore \pi \approx \frac{2l}{dP} = \frac{2ln}{ds}$$

***(바늘의 갯수: n, 평행선과 만난 바늘의 갯수: s)***

따라서 바늘과 평행선이 만날 확률은 기하학적 정의에 따라 밑의 식으로 나타낼 수 있다. π = 2*l*/*P*d (P=(평행선과 만난 횟수=s/던진 횟수=n)) 즉 π = 2l*n*/*s*d이다.

In [3]:

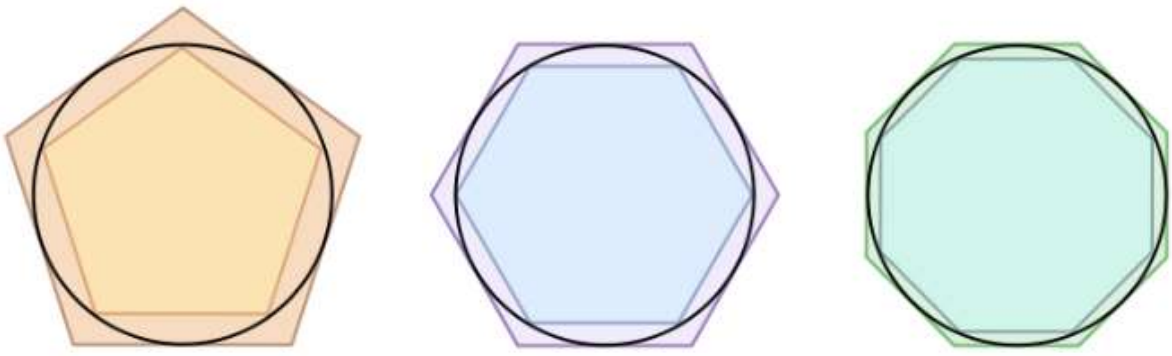
```
import turtle

boardWidth = 40
needleLength = 30
numberOfNeedles = 500
overlappingNeedles = 0

myPen = turtle.Turtle()
myPen.hideturtle()
myPen.speed(0)

y = 180
#Draw floor boards
for i in range(0,10):
    myPen.penup()
    myPen.goto(-200,y)
    myPen.pendown()
    myPen.goto(200,y)
    y-=boardWidth

#Draw Needles
myPen.color("#f442d1")
for needle in range(0,numberOfNeedles):
    x=random.uniform(-200,200)
    y=random.uniform(-180,180)
    angle=random.uniform(0,360)
    myPen.penup()
    myPen.goto(x,y)
    myPen.setheading(angle)
    myPen.pendown()
    a=myPen.ycor()
    myPen.forward(needleLength)
    b=myPen.ycor()

    for i in range(-180,181,boardWidth):
        if a <= i <= b or b<= i <= a:
            overlappingNeedles+=1

print("L = " + str(needleLength))
print("N = " + str(numberOfNeedles))
print("W = " + str(boardWidth))
print("C = "+ str(overlappingNeedles))

pi = (2*needleLength*numberOfNeedles)/(boardWidth*overlappingNeedles)
print("pi ~ ",pi)
```

```
L = 30
N = 500
W = 40
C = 236
pi ~  3.1779661016949152
```

# 3. 도형을 이용한 방식

## 3.1. 아르키메데스 방법

이 장에서는 외접원과 내접원, 구분구적법 등 도형을 활용한 원주율 계산 방법을 볼 것이다.



기원전 3세기, 아르키테미스는 변이 매우 많은 다각형의 외접원과 내접원을 통해 원주율을 구하였다. 다각형의 변이 많아질수록 내접원과 외접원의 둘레의 차이는 작아진다. 변의 개수를 극한으로 취한다면, 원의 둘레에 근사할 것이다. 즉 반지름의 길이가 r인 원에 내접하는 변이 n개인 정다각형의 둘레 $P_n$에 대해 극한을 취하면 원주율을 계산할 수 있다. 아르키테미스는 또한 원의 넓이가 $\pi r^2$이라는 것도 증명하였다.

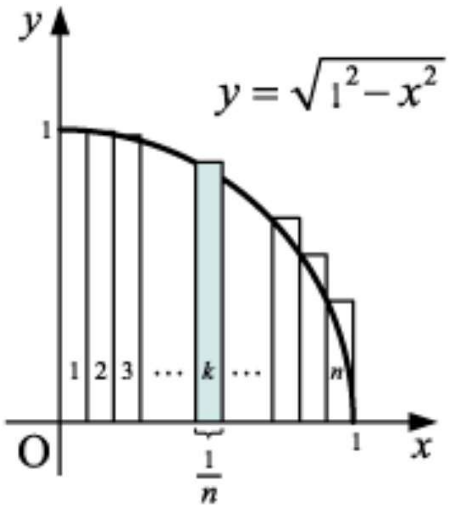
In [4]:

```
n = 5 # 5각형부터 시작

while True:
    degree = 360 / n
    theta = degree / 2
    inner_length = math.sin(math.radians(theta)) * 2 # 내접 정n각형 한 변 길이
    outer_length = math.tan(math.radians(theta)) * 2 # 외접 정n각형 한 변 길이
    difference = outer_length - inner_length
    new_pi = n * ((outer_length + inner_length) / 2) /2 # 아르키메데스 방법으로 구한 파이

    if difference < err:
        break
    else:
        n = n + 1

print(f'pi = {new_pi}')
print(f'정{n}각형')
print("내장 원주율: ", math.pi, "차이: ", math.pi - new_pi)
```

```
pi = 3.141598291251134
정677각형
내장 원주율:  3.141592653589793 차이:  -5.637661340873734e-06
```

## 3.2. 구분구적법

고등학교 때, 배운 구분구적법을 활용해서도 원주율을 증명할 수 있다.
원의 면적을 알면 원주율(π, pi, 파이)을 구할 수 있다. 구분구적법으로 원의 면적을 구해보자. 구분구적법은 해당 영역을 잘게 쪼개서 면적이나 부피를 구하는 방식이다.




반지름이 1인 원을 원점에 놓고 제1사분면의 영역에 대해서 넓이를 구하자. 그림과 같이 밑변을 균일한 간격으로 n개로 쪼개어 각각의 사각형의 면적을 구하자. 각각의 사각형의 높이를 정할 때, 왼쪽 끝을 기준으로 그 높이로 정하자. (높이를 정할 때, 오른쪽을 채택해도 상관없다. 심지어 중간을 높이로 정하는 것도 가능하다. 좌우 중 어느 한쪽을 높이로 정하는데 우리는 왼쪽으로 정하자.)
원을 정리하면, $x^2 + y^2 = 1$이므로, $y = \sqrt{1 - x^2}$이다.
그러므로 k번째 사각형의 높이는 $h = \sqrt{1 - (\frac{k}{n})^2}$이다.

각각의 사각형의 밑변은 $\frac{1}{n}$, 밑변과 높이를 곱한 k번째 사각형의 면적은

$$\frac{1}{n} \cdot h = \frac{1}{n}\sqrt{1 - (\frac{k}{n})^2}$$

이고, 이 n개의 사각형을 모두 합하면 원의 넓이의 근사값이 나온다. 잘게 쪼갤 수록 원의 값에 가까워진다.

In [5]:

```
n = 10000

s=0
for k in range(n):
    s = s + (1/n) * ((1-(k/n)**2)**(1/2))
new_pi = s*4

print(f'결과: {new_pi}')
```

결과: 3.141791477611317

$$(a+b)^n = a^n + \frac{na^{n-1}b^1}{1!} + \frac{n(n-1)a^{n-2}b^2}{2!} + \frac{n(n-1)(n-2)a^{n-3}b^3}{3!} + \cdots$$

$$a = 1, b = -x^2, n = \frac{1}{2} \text{대입}$$

$$\Rightarrow (1-x^2)^{\frac{1}{2}} = 1^{\frac{1}{2}} + \frac{1}{2} \cdot 1^{\frac{1}{2}-1}(-x^2)^1 + \frac{\frac{1}{2}(\frac{1}{2}-1) \cdot 1^{\frac{1}{2}-2}(-x^2)^2}{2!} + \frac{\frac{1}{2}(\frac{1}{2}-1)(\frac{1}{2}-2) \cdot 1^{\frac{1}{2}-3}(-x^2)^3}{3!} + \cdot\cdot$$

$$= 1 - \frac{1}{2}x^2 - \frac{1}{8}x^4 - \frac{1}{16}x^6 + \cdots$$

**이 때, y의 범위는 0≤y≤1이므로**

$$\int_0^1 (1-x^2)^{\frac{1}{2}}dx = [x - \frac{1}{6}x^3 - \frac{1}{40}x^5 - \frac{1}{112}x^7 + \cdots]_0^1$$
$$= 1 - \frac{1}{6} - \frac{1}{40} - \frac{1}{112} + \cdots = \frac{\pi}{4}$$
$$\therefore \pi = 4(1 - \frac{1}{6} - \frac{1}{40} - \frac{1}{112} + \cdots)$$

위의 식은 원의 넓이에 대한 식이므로 $\pi r^2$형태로 바꿔 쓸 수 있다.
여기서 r=1이고 사분원이므로 $\frac{1}{4}$을 붙여주면

$$\frac{\pi}{4} = (1 - \frac{1}{6} - \frac{1}{40} - \frac{1}{112} - \frac{1}{1152} - \cdots)$$
$$\pi = (4 - \frac{4}{6} - \frac{4}{40} - \frac{4}{112} - \frac{41}{1152} - \cdots)$$

이를 계산하면 $\pi$ = 3.14⋯ 가 나온다.

In [6]:

```
temp=1
result=1
i=0
start=time()
while abs(math.pi-result*4)>err:
    temp*=(2*i+1)*abs(2*i-1)/((2*i+3)*(2*i+2))
    result-=temp
    i+=1
result*=4
print(f' 결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

```
 결과: 3.1415927535897232, 걸린 시간: 73ms, 반복 횟수: 24185
```

# 4. 전개식을 이용하는 방식

마지막으로 전개식을 이용해 파이를 구하는 방법을 살펴볼 것이다. 현대에서는 파이값을 컴퓨터를 이용해 계산한다. 더 빠르게 파이를 계산하려면 유용한 식을 짜야 한다. 그래서 수학자들은 많은 전개식을 발견했다. 어느 식이 더 효율적인지 판단하기 위해 수렴속도도 같이 찾아볼 것이다.

## 4.1. 라이프니츠

라이프니츠는 삼각함수를 이용해 원주율을 계산했다. $tan\frac{\pi}{4} = 1$이므로$4 * \tan^{-1}1 = \pi$이다. 라이프니츠는 이 식을 전개해 원주율을 구했다.

$$\arctan x = \frac{x^1}{1} - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$
$$\arctan 1 = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$
$$\therefore \pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots)$$

In [7]:

```
result=0
i=0
start=time()
while abs(math.pi-result*4)>err:
    result+=(-1)**i/(2*i+1)
    i+=1

result*=4
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.1415927535897814, 걸린 시간: 23867ms, 반복 횟수: 10000001

## 4.2. 오일러

오일러를 비롯한 많은 수학자들은 arctan 급수를 이용해 원주율을 계산하는 공식을 발견했다. 또한 오일러는 처음으로 파이라는 기호를 사용했다.

$$\arctan x = \frac{x^1}{1} - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

$$\arctan \frac{1}{2} = \frac{1}{2} - \frac{1}{3 \times 2^3} + \frac{1}{5 \times 2^5} - \frac{1}{7 \times 2^7} + \cdots$$

$$\arctan \frac{1}{3} = \frac{1}{3} - \frac{1}{3 \times 3^3} + \frac{1}{5 \times 3^5} - \frac{1}{7 \times 3^7} + \cdots$$

$$\frac{\pi}{4} = \arctan 1 = \arctan \frac{1}{2} + \arctan \frac{1}{3}$$

$$\pi = 4(\arctan \frac{1}{2} + \arctan \frac{1}{3})$$

In [9]:

```
def taylor_atan(x,i):  # atan(x)의 테일러 급수의 i+1번째 항
    return (-1)**i*x**(2*i+1)/(2*i+1)


start=time()
result=0
i=0
while abs(math.pi-result)>err:
    result+=4*(taylor_atan(1/2,i)+taylor_atan(1/3,i))
    i+=1
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.1415925796063506, 걸린 시간: 1ms, 반복 횟수: 10

## 4.3. 윌리스

파이값을 arctan 급수와 다른방식으로 구한 수학자에는 '윌리스'가 있다. 윌리스는 파이를 무한곱으로 나타낸 '윌리스 곱'을 이용해 파이를 계산했다.

$$I_n = \int_0^{\frac{\pi}{2}} \sin^n x dx$$

$$\int_0^{\frac{\pi}{2}} \sin^{2n} x dx = \frac{2n-1}{2n} \times \frac{2n-3}{2n-2} \times \cdots \times \frac{3}{4} \times \frac{1}{2} \times \frac{\pi}{2} = l_{2n}$$

$$\int_0^{\frac{\pi}{2}} \sin^{2n+1} x dx = \frac{2n}{2n+1} \times \frac{2n-2}{2n-1} \times \cdots \times \frac{4}{5} \times \frac{2}{3} \times 1 = l_{2n+1}$$

$$0 \le x \le \frac{\pi}{2} \rightarrow 0 \le \sin x \le 1$$

$$\therefore \sin^{n+1} \le \sin n \Rightarrow \int_0^{\frac{\pi}{2}} \sin^{n+1} x dx \le \int_0^{\frac{\pi}{2}} \sin^{n} x dx$$

$$\therefore l_{n+1} \le l_n, l_{2n+1} \le l_{2n} \Rightarrow 1 \le \frac{l_{2n}}{l_{2n+1}}, \frac{l_{2n}}{l_{2n+1}} \le \frac{l_{2n-1}}{l_{2n+1}} = \frac{2n+1}{2n}$$

$$1 \le \frac{l_{2n}}{l_{2n+1}} \le \frac{2n+1}{2n} \Rightarrow \lim_{n\to\infty} \frac{l_{2n}}{l_{2n+1}} = 1$$

---

$$\frac{2\times 2}{1\times 3} \times \frac{4\times 4}{3\times 5} \times \frac{6\times 6}{5\times 7} \times \cdots \times \frac{2n\times 2n}{(2n-1)\times(2n+1)} = A_n$$

$$\lim_{n\to\infty} A_n = \frac{2\times 2}{1\times 3} \times \frac{4\times 4}{3\times 5} \times \frac{6\times 6}{5\times 7} \times \cdots \times \frac{2n\times 2n}{(2n-1)\times(2n+1)} \times \cdots$$

$$\frac{I_{2n}}{I_{2n+1}} = \frac{1\times 3}{2\times 2} \times \frac{3\times 5}{4\times 4} \times \cdots \times \frac{(2n-1)\times(2n+1)}{2n\times 2n} \times \frac{\pi}{2}$$

$$\Rightarrow \frac{I_{2n}}{I_{2n+1}} = \frac{1}{A_n} \times \frac{\pi}{2} \Rightarrow 1 = \lim_{n\to\infty} \frac{I_{2n}}{I_{2n+1}} = \frac{\pi}{2} \lim_{n\to\infty} \frac{I}{A_n}$$

$$\therefore \pi = 2 \lim_{n\to\infty} A_n = 2 \times \frac{2\times 2\times 4\times 4\times 6\times 6\times 8\times 8\times \cdots}{1\times 3\times 3\times 5\times 5\times 7\times 7\times 9\times \cdots}$$

In [10]:

```
start=time()
result=1
i=0

while abs(math.pi-result*2)>err:
    result*=((i//2+1)*2)/(((i+1)//2)*2+1)
    i+=1
result*=2

print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.1415927535897836, 걸린 시간: 31394ms, 반복 횟수: 15707653

## 4.4. 뉴턴

뉴턴은 $sin(\frac{\pi}{6}) = \frac{1}{2}$을 이용해 원주율을 구했다. 이식은 수렴속도가 매우 빠르다.

$$\arcsin(x) = x + \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{x^5}{5} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \frac{x^7}{7} + \cdots$$

$$\arcsin\frac{1}{2}=\frac{\pi}{6}=\frac{1}{2}+\frac{1}{2\times3\times2^3}+\frac{1\times3}{2\times4\times5\times2^5}+\frac{1\times3\times5}{2\times4\times6\times7\times2^7}+\cdots$$

$$\therefore\pi=6(\frac{1}{2}+\frac{1}{2\times3\times2^3}+\frac{1\times3}{2\times4\times5\times2^5}+\frac{1\times3\times5}{2\times4\times6\times7\times2^7}+\cdots)$$

In [11]:

```
err=10e-15
factor=1
result=0
i=1
start=time()
while abs(math.pi-result*6)>err:
    result+=factor*0.5**(2*i-1)/(2*i-1)
    factor*=(2*i-1)/(2*i)
    i+=1
result*=6
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i-1}')
```

결과: 3.1415926535897913, 걸린 시간: 1ms, 반복 횟수: 21

## 4.5. 그 외 수학자

이외에도 해튼, 마틴, 메가, 다제, 러더퍼드, 센크스, 샤프 등과 같은 많은 수학자들이 파이를 계산한 방법들을 코딩으로 짜보았습니다. 이 밥법들은 간단히 수렴 속도만 살펴보겠다.

In [12]:

```
# 해튼
start=time()
result=0
i=0
while abs(math.pi-result)>err:
    result+=12*(taylor_atan(1/4,i))+4*taylor_atan(5/99,i)
    i+=1
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.1415926535897998, 걸린 시간: 1ms, 반복 횟수: 11

In [13]:

```
#마틴
start=time()
result=0
i=0
while abs(math.pi-result)>err:
    result+=16*taylor_atan(1/5,i)-4*taylor_atan(1/239,i)
    i+=1
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i+1}')
```

결과: 3.1415926535897927, 걸린 시간: 1ms, 반복 횟수: 11

In [14]:

```
#베가
start=time()
result=0
i=0
while abs(math.pi-result)>err:
    result+=16*taylor_atan(1/5,i)-8*taylor_atan(1/408,i)+4*taylor_atan(1/1393,i)
    i+=1
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.1415926535897922, 걸린 시간: 1ms, 반복 횟수: 10

In [15]:

```
#오일러, 베가
start=time()
result=0
i=0
while abs(math.pi-result)>err:
    result+=20*taylor_atan(1/7,i)+8*taylor_atan(3/79,i)
    i+=1
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.141592653589788, 걸린 시간: 1ms, 반복 횟수: 8

In [16]:

```
# 가우스
start=time()
result=0
i=0

while abs(math.pi-result)>err:
    result+=12*taylor_atan(1/4,i)+4*taylor_atan(1/20,i)+4*taylor_atan(1/1985,i)
    i+=1
print(f'9. 결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

9. 결과: 3.1415926535897998, 걸린 시간: 1ms, 반복 횟수: 11

In [17]:

```
# 다제
start=time()
result=0
i=0
while abs(math.pi-result)>err:
    result+=4*taylor_atan(1/2,i)+4*taylor_atan(1/5,i)+4*taylor_atan(1/8,i)
    i+=1
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.1415926535898016, 걸린 시간: 1ms, 반복 횟수: 21

In [18]:

```
# 러더퍼드
start=time()
result=0
i=0
while abs(math.pi-result)>err:
    result+=16*taylor_atan(1/5,i)-4*taylor_atan(1/70,i)+4*taylor_atan(1/99,i)
    i+=1
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.1415926535897927, 걸린 시간: 1ms, 반복 횟수: 10

In [19]:

```
# 센크스
start=time()
result=0
i=0

while abs(math.pi-result)>err:
    result+=24*taylor_atan(1/8,i)+8*taylor_atan(1/57,i)+4*taylor_atan(1/239,i)
    i+=1
print(f'결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

결과: 3.1415926535897927, 걸린 시간: 0ms, 반복 횟수: 8

In [22]:

```
# 샤프

start=time()
result=0
i=0

while abs(math.pi-result*(2*math.sqrt(3)))>err:
    result+=(-1)**i/((2*i+1)*3**i)
    i+=1

result*=2*math.sqrt(3)
print(f'13. 결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

13. 결과: 3.1415926535897998, 걸린 시간: 1ms, 반복 횟수: 27

In [23]:

```
# 오일러
start=time()
result=2*math.sqrt(3)
result2=2*math.sqrt(3)
factor=0
i=0
while abs(math.pi-result)>err:
    result=result2
    factor+=(-1)**i/(i+1)**2
    result2=result
    result*=math.sqrt(factor)
    i+=1
print(f'15. 결과: {result}, 걸린 시간: {(time()-start)*1000:.0f}ms, 반복 횟수: {i}')
```

15. 결과: 3.141592653589803, 걸린 시간: 13716ms, 반복 횟수: 3441481

## 5. 결론

현재 파이를 구하는 식을 라이프니츠, 오일러와 비교해보면 수렴속도는 훨씬 빨라졌다. 또한, 계산을 위한 코딩은 더 깔끔해졌다. 더불어 컴퓨터가 발전함에 따라, 계산 속도도 더 빨라졌다. 이 과정에서 파이가 초월수, 무리수라는 것 또한 증명되었다. 인간은 끊임없이 발전한다. 미래가 되면 더 빠르고 길게 파이를 구할 수 있는 세상이 올 것이다.