Lasse Berglund

Bestämma π med Python

I den här artikeln får vi se en rad exempel på hur värdet på π kan bestämmas. Artikeln tar avstamp i Arkimedes geometriskt grundade ansats och avslutar med modern Python-programmering – en kod på bara några rader!

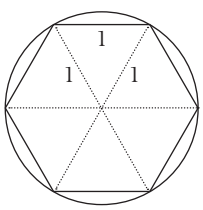
Att bestämma π med stor noggrannhet har varit föremål för intresse i årtusenden. Vi kommer här i några exempel att titta dels på den klassiska metoden med inskrivna polygoner, dels på några modernare formuleringar från 1600-talet. Lite programmering följer naturligt på en uppgift som denna, och vi knyter därmed samman lite av det gamla med lite av det nya. Många gånger kodar man iterativt, det vill säga arbetar med slingor av skilda slag, och allt det som vi kommer att skapa här, kan man också göra med while- och for-loopar. Men denna gång anlägger vi ett rekursivt perspektiv.

Det antika arvet

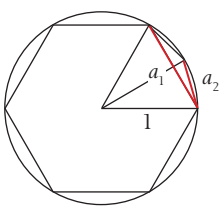
I ett försök att komma åt detta irrationella tal, skriver vi in en regelbunden 6-hörning i en cirkel med radien $r = 1$, på ungefär samma sätt som *Arkimedes* en gång i tiden inledde sin berömda undersökning.

Då medelpunktsvinkeln är $360°/6 = 60°$, blir trianglarna liksidiga och var sida blir därmed 1 längdenhet. Polygonens omkrets blir alltså $6 \cdot 1$, se figur 1. Då diametern är 2, har vi redan här ett första värde på en kvot P som en approximation till π:

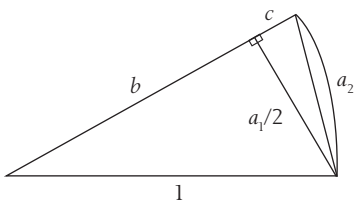
$$P_1 = \frac{Omkretsen}{diametern} = \frac{(antal\ sidor)\cdot(sidlängden)}{2} = \frac{6\cdot 1}{2} = 3$$



Figur 1



Figur 2



Figur 3

För att få igång en process som fördubblar antalet hörn för varje ny polygon, halverar vi 6-hörningens sidor och får därmed en 12-hörning, se figur 2. De röda linjerna markerar de två polygonernas sidlängder, a_1 respektive a_2. Detta zoomar vi nu in, se figur 3. I ett försök att finna ett samband mellan a_1 och a_2 utnyttjar vi *Pythagoras sats*:

$$(a_1/2)^2 + b^2 = 1^2 \qquad (a_1/2)^2 + c^2 = a_2^2 \qquad b + c = 1$$

Ur detta system eliminerar vi b och c, och får efter lite räknande:

$$a_2 = \sqrt{2 - 2\sqrt{1 - a_1^2/4}}$$

Vill man redan här sätta in $a_1 = 1$ så blir $a_2 \approx 0{,}517$ vilket verkar stämma med de röda linjerna i figur 2, där a_2 är drygt halva a_1. Polygonen har i detta skede 12 sidor:

$$P_2 = \frac{12 \cdot 0{,}517}{2} = 3{,}102$$

Vi inser också att a_3 kommer att ges av a_2 enligt samma princip.

Katten på råttan – en rekursiv ansats

En *rekursiv* funktion är en funktion som anropar sig själv. Vår formel, se generaliseringen nedan, är en sådan rekursion, det vill säga en funktion på formen $a_{n+1} = F(a_n)$. Vår rekursiva funktion är:

$$a_{n+1} = \sqrt{2 - 2\sqrt{1 - a_n^2/4}} \qquad (*)$$

Säg att vi vill veta sidlängden i den fjärde polygonen, a_4. Om vi vet a_3 är det bara att sätta in detta värde i (∗) och därmed skulle vi vara klara. Vi vet inte a_3, men vi kan få reda på a_3 om vi vet a_2. Vi vet inte a_2. Men! Vi vet att $a_1 = 1$. Där har vi vårt så kallade *basfall* som är helt avgörande. Utan ett sådant går processen inte igång. Från (∗) får vi, kompakt formulerat:

$$a_4 = F(a_3) = F(F(a_2)) = F(F(F(a_1)))$$

Vi har nått ner till basfallet och nu vänder det uppåt då vi i likhet med ramsan till slut har stött på katten:

$$F(F(F(1))) = F(F(0{,}517)) = F(0{,}261) = 0{,}130 = a_4$$

Arkimedes och Python

Det finns för- och nackdelar med både det iterativa och det rekursiva sättet att programmera. En fördel med det rekursiva sättet är att det många gånger blir oerhört enkelt; koden blir inte mer än ett par tre rader lång.

Även antalet sidor i polygonen, 6, 12, 24, 48 … kan uttryckas rekursivt då varje tal är en fördubbling av talet precis framför, vilket ger:

$b_{n+1} = 2b_n$, där b är antalet sidor med basfallet $b_1 = 6$.

Den intresserade kan så klart utnyttja även denna rekursion i koden, men jag väljer här att istället ta med den här faktorn först på utskriftsraden då antalet sidor också kan anges explicit som $3\cdot 2^n$ med $n=1,2,3\ldots$


```
n = int(input("Ange exponenten: "))
def L(n):
    if n == 1:
        return 1.0
    else:
        return (2 – 2*(1 – (L(n – 1))**2/4.0)**0.5)**0.5
print 3*2**n*L(n)/2.0
```

```
Ange exponenten: 5
3.14103195089
```

Figur 4. Rekursiv Pythonkod.

- Inledningsvis får användaren ange n i uttrycket $3\cdot 2^n$, vilket ger antalet sidor i polygonen.
- Därefter definieras en funktion L(n) som ska utgöra längden av en sida.
- I if-satsens inledning anges det viktiga basfallet.
- Sedan går rekursionen igång där L(n) anropar L(n – 1).
- Utskriften sker enligt (antalet sidor) · (sidlängden)/2.
- I Python är symbolen = = en likhet, symbolen = är en tilldelning och *a**b* betyder *a* upphöjt till *b*.

En körning för L(5) syns i figur 4, där 5:an anger att det är $3\cdot 2^5=96$ sidor i polygonen. Vi ser också att ett π-värde är på väg – programmet är redan ifatt Arkimedes och hans klassiska 96-hörning. Den som sedan väljer $n=6$ är därmed om och förbi.

Pi som produkt och summa

I de sista två exemplen programmerar vi enligt formeln $a_{n+1}=\mathrm{F}(a_n;n)$. I båda fallen kommer det att konvergera ganska långsamt, men vi tar ändå med dem då de har en märklig charm båda två. Istället för en kvot kommer π att anges som en produkt respektive en summa.

Wallis produkt

I och med 1600-talet tog utvecklingen ny fart när den då moderna integralkalkylen gav helt nya möjligheter att komma π inpå livet. *Wallis produk*t från år 1656,

$$\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\cdot\ldots=\frac{\pi}{2}$$

är ett helt enastående resultat då en massa rationella tal utan skymten av någon cirkel ger oss ett numeriskt värde på detta svårflörtade π, som inte bara är irrationellt utan även transcendent. Inför uppgiften att skriva en kod, packar vi ihop faktorerna två och två.

$$\frac{4}{3}\cdot\frac{16}{15}\cdot\frac{36}{35}\cdot\frac{64}{63}\cdot\ldots$$

Som ett exempel betraktar vi nu tredje faktorn a_3 lite närmare:

$$a_3=\frac{36}{35}=\frac{4\cdot 9}{4\cdot 9-1}=\frac{4\cdot 3^2}{4\cdot 3^2-1}\quad\Rightarrow\quad \text{Den allmänna faktorn } a_n=\frac{4\cdot n^2}{4\cdot n^2-1}$$

Produkten fram till och med säg faktor a_7 blir: $P_7=a_1a_2a_3a_4a_5a_6a_7$. Då vi inser att detta kan anges som $P_7=P_6\cdot a_7$ blir det naturligt med en rekursiv kod.

Allmänt får vi:

$$P_n=P_{n-1}\cdot 4n^2/(4n^2-1)\qquad P_1=4/3$$

Exempel:

$$\begin{aligned}P_4 &= P_3\cdot(64/63)\\ &= P_2\cdot(36/35)\cdot(64/63)\\ &= P_1\cdot(16/15)\cdot(36/35)\cdot(64/63) && \text{Vi är nere vid basfallet.}\\ &= (4/3)\cdot(16/15)\cdot(36/35)\cdot(64/63)\end{aligned}$$

Leibniz summa

Lika iögonfallande som produkten ovan, är *Leibniz formel* från år 1676.

$$1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\ldots=\frac{\pi}{4}$$

Om inte tecknen hade skiftat, hade varje term haft formen $1/(2n-1)$. De basala sambanden $(-1)^{udda}=-1$ och $(-1)^{jämnt}=1$, ger oss det teckenskifte vi söker, och den allmänna termen blir:

$$a_n=\frac{(-1)^n}{2n-1}$$

Återigen framstår en rekursion som något självklart. Vi får:

$$S_n=S_{n-1}-(-1)^n/(2n-1)\qquad S_1=1$$

Exempel:

$$\begin{aligned}S_4 &= S_3-1/7\\ &= S_2+1/5-1/7\\ &= S_1-1/3+1/5-1/7 && \text{Vi är nere vid basfallet.}\\ &= 1-1/3+1/5-1/7\end{aligned}$$

Bortom pennans räckvidd

Att någon av dessa märkliga formuleringar ska leda oss till π verkar högst osannolikt. För att se om det som utlovas verkligen stämmer skapar vi två nya Python-filer. Ett par tre rader kod räcker för att vi med egna ögon ska kunna se att de två matematikerna hade rätt båda två. Man kan inte annat än lyfta på hatten. Samtidigt passar vi så klart på att göra en jämförelse. Vem vinner – Wallis eller Leibniz?

Wallis produkt

```
n = int(input("Ange antalet faktorer: "))
def P(n):
    if n == 1:
        return 4.0/3
    else:
        return P(n – 1)*4.0*n**2/(4*n**2 – 1)
print 2*P(n)
```

```
Ange antalet faktorer: 982
3.14079336788
```

Leibniz summa

```
n = int(input("Ange antalet termer: "))
def S(n):
    if n == 1:
        return 1.0
    else:
        return S(n – 1) – (–1.0)**n/(2*n – 1)
print 4*S(n)
```

```
Ange antalet termer: 982
3.14057432391
```

Figur 5. Pythonkod för π som produkt respektive summa.

Är det någon läsare som känner sig hågad att ge sig på någon av dessa beräkningar med papper och penna? Närmare tusen uppställningar. Nej, vi har nog samma känsla allihopa: inte ens med räknaren i hand lockar ett sådant beräkningsarbete.

Arkimedes, 200-talet f Kr, grekisk matematiker

John Wallis, 1616–1703, engelsk matematiker

Gottfried Wilhelm von Leibniz, 1646–1716, tysk matematiker

LITTERATUR

Tambour, T. (2015). *Wallis´ produkt*. PDF.

Firk, F. W. (2009). *A proof of the Gregory-Leibniz series and new series for calculating pi*. Yale University.