matematik

origo

Daniel Dufåker
Attila Szabo
Niclas Larson

Programmering

SANOMA UTBILDNING

Innehåll

Aktivitet	Kurs	Beskrivning
Gissa ett tal	1c	I den här aktiviteten får eleverna via gissningar försöka finna vilket heltal mellan 1 och 100 som deras kamrat har angett. Syftet med uppgiften är att upptäcka den metod med intervallhalvering som garanterar att man inte behöver mer än 7 gissningar för att hitta talet. Samtidigt får eleverna bekanta sig med programmerings-kommandon som `if` och `while`.
Slumpförsök med tärningar	1c	Vilken tärningssumma är vanligast vid kast med 3, 4 respektive 5 tärningar? I den här aktiviteten får eleverna undersöka den frågeställningen med hjälp av dator-simuleringar.
Fibonacci	1c, 5	Eleverna får ta reda på slutsiffran i det 159:e Fibonaccitalet och undersöka kvoten mellan två på varandra följande Fibonaccital.
Perfekta tal	1c, 5	I den här aktiviteten får eleverna använda programmering för att undersöka om ett heltal är fattigt, perfekt eller rikt.
Kvadratrötter	1c, 3b	Eleverna får använda programmering för att numeriskt bestämma kvadratrötter. Eftersom uppgiften inte kräver så mycket kod, passar den bra för att introducera programmering.
Approximera π	1c, 3c	I den här aktiviteten får eleverna använda programmering för att uppskatta värdet av konstanten π.
Eratosthenes såll	1c, 3b	Syftet med den här aktiviteten är att fördjupa elevernas kunskaper om primtal genom att skriva program som utför algoritmen *Eratosthenes såll*. Samtidigt får eleverna se hur man kan utnyttja programmering för att underlätta utförandet av tidsödande beräkningar.
Viètes samband	2c, 3b, 3c	I den här aktiviteten får eleverna undersöka sambanden mellan andragradsekvationens konstanter och dess rötter.
Avståndsformeln	2c	Den här aktiviteten syftar till att introducera eller fördjupa elevernas kunskaper om avståndsformeln. Eleverna får slumpa fram tal i ett intervall, respektive punkter i ett koordinatsystem, och undersöka sannolikheten att avståndet mellan talen/punkterna är större än ett bestämt tal.
Antal punkter på cirkeln	3c	I den här aktiviteten får eleverna undersöka egenskaper hos cirklar med hjälp av programmering. Samtidigt får de se hur man med programmering kan utföra ett stort antal beräkningar och på så sätt undersöka något man är nyfiken på. Aktiviteten är lämplig att använda i Matematik Origo 3c i samband med avsnittet om cirkelns ekvation.
Sinussatsen	3c, 4	Den här aktiviteten syftar till att fördjupa elevernas förståelse av sinussatsen. Eleverna får skriva program som kan användas för att beräkna okända sidor och vinklar i trianglar. På så sätt skaffar sig eleverna ett verktyg för att undersöka när det finns ingen, en eller två trianglar som uppfyller givna villkor.
Buffons nålproblem	3c	I den här aktiviteten får eleverna använda programmering för att undersöka *Buffons nålproblem*. Det klassiska problemet går ut på att uppskatta sannolikheten att en nål skär någon linje, om man släpper den på ett papper med ekvidistanta parallella linjer.

Aktivitet	Kurs	Beskrivning
Geometrisk talföljd och geometrisk summa	3b, 5	Den här aktiviteten syftar till att fördjupa elevernas kunskaper om geometriska talföljder och summor. För att aktiviteten ska vara lämplig att använda i Matematik Origo 3b, där eleverna inte tidigare har arbetat med programmering som problemlösningsverktyg i gymnasieskolan, har vi valt att presentera ett färdigt program som eleverna får modifiera och bygga vidare på.
Olika typer av lån	3b, 5	I den här aktiviteten får eleverna använda programmering för att jämföra annuitetslån med rak amortering.
Summor och gränsvärden	3b, 4	Den här aktiviteten syftar till att fördjupa elevernas kunskaper om bevisföring genom att de själva får undersöka om en serie konvergerar eller divergerar.
Komplexa tal och transformationer	4	I den här aktiviteten får eleverna använda programmering för att undersöka transformationer i det komplexa talplanet.

A

Aktivitet

Gissa ett tal

Här nedanför ser du ett program skrivet i programspråket Python 3.

```
from getpass import *
tal = int(getpass(prompt = "Skriv ett heltal mellan 1 och 100:"))
gissning = int(input("Gissa talet som din kamrat har angett:"))
antal_gissningar = 1
while gissning != tal:
  if gissning < tal:
    gissning = int(input("För litet! Gissa igen."))
  elif gissning > tal:
    gissning = int(input("För stort! Gissa igen."))
  antal_gissningar = antal_gissningar + 1
print("Rätt! Din kompis tal var", tal)
print("Du behövde", antal_gissningar, "gissningar.")
input("Tryck Enter för att avsluta programmet")
```

1 Spara programmet på din dator och dubbelklicka på filnamnet. Då körs programmet i terminalläge. Arbeta tillsammans med en kompis och kör programmet ett par gånger. Vad gör det?

2 Förklara tillsammans vad varje del av programkoden gör.

3 Kör programmet 10 gånger. Växla mellan att vara den som anger talet och att vara den som gissar. Notera hur många gissningar ni behöver varje gång.

4 Hur många gissningar behövde ni i genomsnitt?

5 Oskar säger att han aldrig behöver mer än 7 gissningar för att gissa rätt tal. Kan ni komma på hur Oskar gör?

6 Hur många gissningar behöver man som mest med Oskars strategi, om talet ligger mellan

a) 1 och 1 000

b) 1 och 10 000

7 Vad finns det för samband mellan det högsta antalet gissningar som behövs och potenser med basen 2?

Syfte och centralt innehåll

I den här aktiviteten får eleverna gissa ett heltal mellan 1 och 100 som deras kamrat har angett. Syftet med uppgiften är att upptäcka den metod med intervallhalvering som garanterar att man inte behöver mer än 7 gissningar för att hitta talet. Samtidigt får eleverna bekanta sig med programmeringskommandon som `if` och `while`.

Förkunskaper i programmering

Programmet som vi presenterar i aktiviteten innehåller kommandon som `print`, `input`, `while` och `if`. Det kan vara en fördel att eleverna har sett dessa kommandon tidigare, men aktiviteten kan också vara ett sätt att låta eleverna bekanta sig med dem. I inledningen av aktiviteten importerar vi modulen `getpass` som gör att kompisens inmatning inte syns på skärmen när programmet körs i terminalläge.

Genomförande

Om eleverna har tillräckliga förkunskaper i programmering kan du låta dem arbeta med elevstencilen enskilt eller i par. I annat fall kan du tillsammans med eleverna analysera programmet innan du låter dem använda det.

Du kan också inleda lektionen genom att diskutera med eleverna hur man på ett effektivt sätt kan hitta ett heltal (som man inte känner till) inom ett bestämt intervall. På så sätt kan man sätta i gång elevernas tankar om vilka strategier de kan använda i aktiviteten.

Lösning

1 Programmet räknar antalet gissningar som behövs för att en person ska gissa ett heltal mellan 1 och 100.

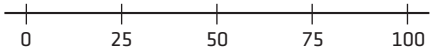
2

```
from getpass import *      #Vi importerar modulen getpass för att den följande inmatningen ska vara osynlig på skärmen.
tal = int(getpass(prompt = "Skriv ett heltal mellan 1 och 100:"))
#Programmet låter en användare mata in ett heltal mellan 1 och 100.
gissning = int(input("Gissa talet som din kamrat har angett:"))#En annan användare får mata in en gissning.
antal_gissningar = 1       #Variabeln antal_gissningar räknar antalet gissningar
while gissning != tal:     #Så länge gissningen inte är lika med det inmatade talet, gör programmet följande:
  if gissning < tal:
    gissning = int(input("För litet! Gissa igen."))   #Om gissningen är mindre än det inmatade talet skriver programmet "För litet! Gissa igen."
  elif gissning > tal:
    gissning = int(input("För stort! Gissa igen."))   #Om gissningen är större än det inmatade talet skriver programmet "För stort! Gissa igen."
  antal_gissningar = antal_gissningar + 1             #För varje ny gissning ökar antal_gissningar med 1
print("Rätt! Din kompis tal var", tal)
print("Du behövde", antal_gissningar, "gissningar.")  #När användaren gissar rätt skrivet programmet ut det rätta svaret och antalet gissningar.
```

3 –

4 –

5 Oskar använder sig av intervallhalvering. Han börjar med att gissa på talet 50, mitt emellan 0 och 100. Om programmet säger att gissningen är för stor, gissar han på talet mitt emellan 0 och 50, dvs. 25. På så sätt halverar han i varje steg det intervall där talet kan ligga.



6 a) 10

b) 14

7 I intervallet 1–100 krävs det 7 gissningar eftersom 2^7 är den första potensen med basen 2 som är större än 100. Det beror på att det krävs 7 halveringar av intervallet innan intervallet är mindre än eller lika med 1.

$\frac{100}{2} = 50$

$\frac{50}{2} = 25$

$\frac{25}{2} = 12{,}5 \approx 13$ Vi avrundar till heltal eftersom eleverna gissar heltal i uppgiften.

$\frac{13}{2} = 7{,}5 \approx 8$

$\frac{8}{2} = 4$

$\frac{4}{2} = 2$

$\frac{2}{2} = 1$

Det största antalet gissningar som krävs är lika med exponenten i den första tvåpotens som är större än eller lika med antalet möjliga tal som man kan välja på (100, 1 000 respektive 10 000).

Att lyfta fram

Uppgift 6 kan eleverna undersöka empiriskt genom att ändra i programmet eller fundera på utan hjälp av ett program. Diskutera gärna hur man kan veta att det endast krävs 7 gissningar, t.ex. med hjälp av en tallinje. Diskutera även hur man ska hantera att intervallhalveringen leder till tal med decimaler. Då gissar man helt enkelt på ett av de två talen som är i mitten av intervallet.

Utvidgning och variation

En möjlig utvidgning är att låta eleverna spela spelet mot datorn, när datorn använder intervallhalvering. Hur kan ett sådant program se ut? Klarar datorn det på i genomsnitt färre gissningar än klassen?

Här är ett förslag till sådan kod.

```
print("Tänk på ett tal mellan 0 och 100.")
input("Tryck Enter när du är klar.")
a = 0
b = 100
gissning = (a + b)//2
print("Min gissning är:", gissning)
antal_gissningar = 1
feedback = input("Om gissningen är rätt,
skriv Rätt. Om ditt tal är högre eller
lägre, skriv Högre eller Lägre.")
while feedback != "Rätt":
  if feedback == "Högre":
    a = gissning
    gissning = (gissning + b)//2
    print("Min gissning är",gissning)
    feedback = input("Om gissningen är
    rätt, skriv Rätt. Om ditt tal är hö-
    gre eller lägre, skriv Högre eller
    Lägre.")
  elif feedback == "Lägre":
    b = gissning
    gissning = (gissning + a)//2
    print("Min gissning är",gissning)
    feedback = input("Om gissningen är
    rätt, skriv Rätt. Om ditt tal är hö-
    gre eller lägre, skriv Högre eller
    Lägre.")
  antal_gissningar = antal_gissningar + 1
print("Okej! Jag behövde", antal_giss-
ningar, "gissningar.")
```

A Aktivitet
Slumpförsök med tärningar

I den här aktiviteten får du skriva program som kan simulera slumpförsök med tärningar.

1 a) Skriv ett program som simulerar 100 000 kast med en vanlig tärning och beräknar den relativa frekvensen för de sex möjliga utfallen.

b) Undersök hur väl resultatet stämmer överens med vad du borde få teoretiskt.

2 Vilken tärningssumma är mest sannolik när man kastar två tärningar? Undersök detta genom att skriva ett program som simulerar 100 000 kast med två tärningar och beräknar deras ögonsumma. Programmet ska beräkna den relativa frekvensen för de olika utfallen (tärningssummorna).

Tärningssumman är summan av ögonen på tärningarna.

3 Vilken tärningssumma är mest sannolik när man kastar tre tärningar? Undersök detta genom att skriva ett program som simulerar 100 000 kast med tre tärningar och beräknar deras ögonsumma. Programmet ska beräkna den relativa frekvensen för de olika utfallen.

4 Vilken tärningssumma är mest sannolikt vid kast med 4, 5 respektive 6 stycken tärningar? Undersök med hjälp av ett program som simulerar detta.

5 Hur stor är sannolikheten att få Yatzy, dvs. att alla tärningar visar lika vid ett kast med fem tärningar? Undersök med hjälp av ett program som simulerar detta.

Syfte och centralt innehåll

Den här aktiviteten syftar till att fördjupa kunskaperna i sannolikhetslära genom att eleverna får skapa simuleringar som beräknar den relativa frekvensen för olika utfall. En central idé i aktiviteten är att man kan utnyttja relativ frekvens för att uppskatta sannolikheten för en händelse.

Aktiviteten passar bra i kapitlet om sannolikhetslära i Matematik Origo 1c. Man kan till exempel använda den som en utvidgning av uppgift **6110,** där eleverna får undersöka sannolikheten för olika tärningssummor vid kast med två tärningar. Vilken tärningssumma är vanligast vid kast med 4, 5 respektive 6 tärningar? Den frågeställningen får eleverna undersöka i aktiviteten.

Förkunskaper i programmering

Eleverna bör känna till hur man arbetar med variabler, skapar listor och skriver slingor med hjälp av kommandot `for`. För att arbeta med slumpförsök måste eleverna även importera modulen `random`. Det föreslår vi att man går igenom gemensamt med eleverna (se *Genomförande*).

Genomförande

Vi presenterar ett förslag på hur aktiviteten kan genomföras för elever som inte har någon större erfarenhet av programmering.

- Dela upp eleverna så att de arbetar två och två med en dator. Uppmana dem att låta den som har minst erfarenhet av datorer och programmering sköta datorn. Är båda eleverna lika erfarna så kan de turas om.
- Låt eleverna granska koden här nedanför och försöka förstå vad varje del i koden gör. Detta sätt att arbeta kallas inom programmering för *tinkering*. Vad har man simulerat i programmet? Om det här inte är elevernas första programmeringsaktivitet kan man ta bort de förklarande kommentarerna.

```
from random import *
resultat = []                    # Skapar en tom lista med namnet resultat

for n in range(1, 1001):
  a = randint(1, 2)              # Slumpar ett heltal i intervallet [1, 2]

  resultat.append(a)             #Lägger till talet a i listan resultat

for i in range(1, 3):
  print("Relativa frekvensen för", i,
  "är", resultat.count(i)/1000)
  #Skriver ut den relativa frekvensen för vart och ett av utfallen 1 och 2.
```

Programmet skriver ut följande:

```
Relativa frekvensen för 1 är 0.49866
Relativa frekvensen för 2 är 0.50133
```

Programmet simulerar 1 000 slantsinglingar och beräknar den relativa frekvensen för de två möjliga utfallen *krona* och *klave*. I programmet har vi valt att kalla krona för 1 och klave för 2. Vi kan på detta sätt nöja oss med att slumpa fram ettor (1) och tvåor (2) och beräkna den relativa frekvensen för antalet ettor och antalet tvåor.

- Diskutera i helklass vad varje del i koden gör, med särskilt fokus på `from random import *` och på kommandot `randint()`. Notera att vi har valt att skriva `from random import *` i stället för `import random` på den första raden i programmet. Så har vi gjort eftersom man då kan skriva `randint()` i stället för `random.randint()`.
- Uppmuntra eleverna att skriva av koden och testköra den. De kommer att kunna lösa flera av uppgifterna på aktivitetsstencilen genom att modifiera koden på lämpligt sätt.
- Dela ut aktivitetsstencilen till eleverna och låt dem arbeta två och två.

Utvidgning och variation

Om man vill utvidga uppgiften kan man låta eleverna beräkna sannolikheten för att få tretal, kåk eller stege på ett kast i Yatzy.

Att lyfta fram

Jämför gärna den relativa frekvensen i simuleringarna med den teoretiska sannolikheten i uppgift 1 och 2. Det ger eleverna möjlighet att se att den relativa frekvensen för ett utfall närmar sig den teoretiska sannolikheten när antalet slumpförsök ökar. Det är svårare att beräkna den vanligaste tärningssumman vid kast med 3, 4 eller 5 stycken tärningar. På så sätt illustrerar aktiviteten programmeringens relevans.

Jämför i uppgift 5 den relativa frekvensen i simuleringen med den teoretiska sannolikheten: $P(\text{Yatzy}) = \frac{1}{6^4}$.

Lösning

1
```
from random import *
resultat = []                    #Skapar en tom lista med namnet resultat
for n in range(1, 100001):       #Programmet upprepar följande bit kod 100 000 gånger
  a = randint(1, 6)              #Slumpar ett heltal mellan 1 och 6 (upp till 7 – 1 = 6)
  resultat.append(a)             #Lägger till talet a i listan resultat
for i in range(1, 7):
  print("Relativa frekvensen för", i,
  "är", resultat.count(i)/100000)
  #Skriver ut den relativa frekvensen för vart och ett av utfallen 1, 2, 3, 4, 5 och 6.
```

2
```
from random import *
resultat = []                    #Skapar en tom lista med namnet resultat
for n in range(1, 100001):       #Programmet upprepar följande bit kod 100 000 gånger
  a = randint(1, 6)              # Slumpar ett heltal mellan 1 och 6
  b = randint(1, 6)              # Slumpar ett heltal mellan 1 och 6
  g = a + b                      # Adderar slumptalen och kallar summan för g
  resultat.append(g)             # Lägger till talet g i listan resultat
for i in range(2, 13):
  print("Relativa frekvensen för", i,
  "är", resultat.count(i)/100000)
  #Skriver ut den relativa frekvensen för vart och ett av utfallen 2–12.
```

3
```
from random import *
resultat = []
for n in range(1, 100001):
  a = randint(1, 6)              # Slumpar ett heltal mellan 1 och 6
  b = randint(1, 6)
  c = randint(1, 6)
  g = a + b + c                  # Adderar slumptalen och kallar summan för g
  resultat.append(g)             # Lägger till talet g till listan resultat
for i in range(3, 19):
  print("Relativa frekvensen för", i,
  "är", resultat.count(i)/100000)
  #Skriver ut den relativa frekvensen för vart och ett av de möjliga utfallen 3–18.
```

4
```
from random import *
resultat = []
for n in range(1, 100001):
  a = randint(1, 6)
  b = randint(1, 6)
  c = randint(1, 6)
  d = randint(1, 6)
  g = a + b + c + d              #Adderar de fyra slumptalen och kallar summan för g
  resultat.append(g)             #Lägger till talet g till listan resultat
for i in range(4, 25):
  print("Relativa frekvensen för", i,
  "är", resultat.count(i)/100000)
  #Skriver ut den relativa frekvensen för vart och ett av de möjliga utfallen 4–24.
```

Koden här ovanför kan modifieras för att undersöka fallen med fem respektive sex tärningar.

5 T.ex.
```
from random import *
antal_yatzy = 0
for n in range(1, 100001):
  a = randint(1, 6)
  b = randint(1, 6)
  c = randint(1, 6)
  d = randint(1, 6)
  e = randint(1, 6)
  if a == b == c == d == e:
    print('yatzy')
    antal_yatzy = antal_yatzy + 1
print("Antal yatzy blev: ", antal_yatzy)
print("Sannolikheten blir enligt simuleringen:", antal_yatzy/100000)
print("Den teoretiska sannolikheten är:", 1/6**4)
```

A Aktivitet
Fibonacci

Leonardo Fibonacci levde omkring år 1200. Under sina många resor runt Medelhavet kom han i kontakt med både arabiska och grekiska matematiker. I boken *Liber abbaci*, som utkom år 1202, sammanfattade han vad han hade lärt sig om aritmetik (räknekonst) och algebra. I den boken presenterade han också de siffror vi använder i dag. Men mest känd är Fibonacci för att han har gett namn åt en talföljd. Talföljden börjar med 0 och följs sedan av två ettor. Varje tal i talföljden är sedan summan av de två föregående talen.

0, 1, 1, 2, 3, 5, 8, 13, 21, …

1 Vilka är de nästa två talen i följden?

2 Skriv ett program som skriver ut de 50 första Fibonaccitalen.

3 Vilken slutsiffra har det 159:e Fibonaccitalet?

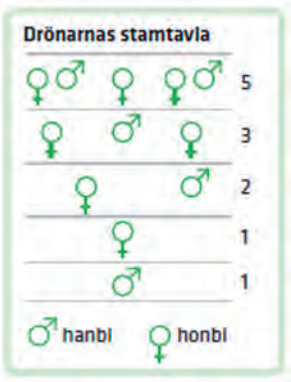
4 Ändra programmet så användaren får ange hur många Fibonaccital som ska skrivas ut.

Tips! I Python3 kan du använda dig av kommandot `int(input())` när du vill låta användaren mata in ett heltal.

Om vi dividerar varje Fibonaccital med det föregående talet i talföljden, får vi en ny talföljd:

$\frac{1}{1}, \frac{2}{1}, \frac{3}{2}, \frac{5}{3}, \frac{8}{5}, \frac{13}{8}, \ldots$

5 Skriv ett program som skriver ut de 20 första talen i denna talföljd. Ser du något mönster?



Hanbiets stamtavla växer precis som Fibonaccis talföljd.

Syfte

I den här aktiviteten får eleverna ta reda på slutsiffran i det 159:e Fibonaccitalet och undersöka kvoten mellan två på varandra följande Fibonaccital. Aktiviteten passar bra i algebrakapitlet i Matematik Origo 1c, t.ex. i samband med avsnittet *Mönster och formler* eller i avsnittet *Historia* på sidan 97 i elevboken. Eleverna får naturligtvis använda valfritt programspråk, men vi kommer att visa lösningar i programspråket Python 3.

Förkunskaper i programmering

Det finns många olika sätt att skriva program som skriver ut Fibonaccitalen, men eleverna behöver veta hur man använder kommandona `print` och `input`, hur man sparar tal i variabler och hur man skriver slingor med `for` eller `while`.

Genomförande

I aktivitetens inledning presenterar vi Fibonaccitalen. Presentera dem gärna gemensamt i helklass, kanske utifrån hanbiets stamtavla. Om eleverna har nödvändiga förkunskaper i programmering kan du låta dem arbeta med resten av elevstencilen enskilt eller i par. I annat fall kan du tillsammans med eleverna skriva ett program som skriver ut Fibonaccitalen, t.ex.

```
a = 0              #Vi sparar de första två Fibonaccitalen,
                   0 och 1, i variablerna a och b
b = 1
print("De 10 första Fibonaccitalen är:")
for n in range (1, 11): #Programmet ska upprepa
                        följande kod 10 gånger:
  print(a)         #Skriv ut värdet av variabeln a
  c = b            #Spara värdet av variabeln b i c.
  b = a + b        #Det nya värdet på b ska vara summan
                   av a och b
  a = c            #Det nya värdet på a ska vara värdet på c,
                   dvs. det gamla värdet på b
```

Diskutera med eleverna hur koden är uppbyggd och låt dem sedan arbeta vidare med uppgifterna på aktivitetsstencilen. Om eleverna arbetar i par kan det vara bra att låta den elev som har minst erfarenhet av programmering sköta datorn. Är båda eleverna lika erfarna kan de turas om.

I aktiviteten bygger vi stegvis upp mot frågeställningen vilken slutsiffra det 159:e Fibonaccitalet har, men det går också bra att låta lektionen helt utgå från den frågeställningen. Det ger aktiviteten en mer problemlösande karaktär.

Lösning

1 34 och 55

2 T.ex.

```
a = 0              #Vi sparar de första två Fibonaccitalen,
                   0 och 1, i variablerna a och b
b = 1
print("De 50 första Fibonaccitalen är:")
for n in range (1, 51): #Programmet ska upprepa följande kod
                        50 gånger:
  print(a)         #Skriv ut värdet av variabeln a
  c = b            #Spara värdet av variabeln b i c.
  b = a + b        #Det nya värdet på b ska vara summan
                   av a och b
  a = c            #Det nya värdet på a ska vara värdet
                   på c, dvs. det gamla värdet på b
```

3 Det 159:e Fibonaccitalet är 468340976726457153752543329995929. Slutsiffran är alltså 9.

4 T.ex.

```
a = 0
b = 1
n = int(input("Det här programmet skrivet ut de n första Fibonaccitalen. Ange n:"))
print("De", n, "första Fibonaccitalen är:")
for n in range (1, n + 1):
  print(a)
  c = b
  b = a + b
  a = c
```

5 Kvoten stabiliserar sig kring ca 1,618. Talet brukar kallas för det gyllene snittet och kan exakt skrivas $\frac{1+\sqrt{5}}{2}$. Uppgiften kan t.ex. lösas med programmet:

```
a = 0
b = 1
for n in range (1, 21):
  c = b
  b = a + b
  a = c
  print(b/a)
```

Att lyfta fram

I den sista deluppgiften får eleverna undersöka kvoten mellan två konsekutiva Fibonaccital. Eleverna kommer att märka att kvoten stabiliserar sig kring talet 1,618. Lyft gärna fram att talföljden konvergerar mot det så kallade gyllene snittet: $\frac{1+\sqrt{5}}{2}$.

Det finns många olika sätt att lösa deluppgift 2. Här nedanför visar vi fyra olika program, som utnyttjar olika programmeringskommandon. Det första programmet använder en stödvariabel. Det andra programmet är elegant, eftersom det uppdaterar värdet på variablerna a och b på en enda rad. Det tredje programmet lägger Fibonaccitalen i en lista och skriver ut listan. Det fjärde programmet utnyttjar rekursion. Det kan vara intressant att jämföra programmens körningstid och diskutera varför det fjärde programmet tar så lång tid att köra.

(1) Med stödvariabel

```
a = 0                    #Vi sparar de första två Fibonaccitalen, 0 och 1, i variablerna a och b
b = 1
print("De 50 första Fibonaccitalen är:")
for n in range (1, 51):  #Programmet ska upprepa följande kod 50 gånger:
  print(a)               #Skriv ut värdet av variabeln a
  c = b                  #Spara värdet av variabeln b i c.
  b = a + b              #Det nya värdet på b ska vara summan av a och b
  a = c                  #Det nya värdet på a ska vara värdet på c, dvs. det gamla värdet på b
```

(2) Med två variabler

```
a = 0
b = 1
for n in range (1, 51):
  print(a)
  a, b = (b, a + b)      #Här sätter vi värdet av variabeln a till b och värdet av variabeln b till a + b
```

(3) Med listor:

```
print("Det här programmet skriver ut de 50 första Fibonaccitalen")
Fibonacci = [0, 1, 1]
for n in range (2, 49):
  Fibonacci.append(Fibonacci[n - 1] + Fibonacci[n])
print(Fibonacci)
```

(4) Med rekursion:

```
def fib(a):              #Vi definierar en funktion som tar fram Fibonaccital nummer a
  if a == 1:
    return (0)
  if a == 2:
    return 1
  else:
    return (fib(a - 1) + fib(a - 2))
for n in range (1, 51):
  print(fib(n))          #Vi anropar funktionen och skriver ut alla Fibonaccital mellan 1 och 50
```

Utvidgning och variation

Fibonaccitalen har många intressanta egenskaper som man kan undersöka med hjälp av programmering, t.ex.

> Om man kvadrerar Fibonaccitalen, får man en ny följd av tal. Om man adderar två på varandra följande tal i den följden, så får man alltid ett Fibonaccital.

Ett annat sätt att utvidga aktiviteten är att introducera Binets formel. Det är en sluten formel för det n:te Fibonaccitalet.

$$\frac{1}{\sqrt{5}} \cdot \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right)$$

Man kan också låta eleverna undersöka de så kallade Lucastalen. Den talföljden är uppbyggd på samma sätt som Fibonaccitalen, men med 2 och 1 som de första elementen i stället för 0 och 1.

2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, …

Det finns intressanta samband mellan Lucastalen och Fibonaccitalen, som eleverna kan undersöka med hjälp av programmering, t.ex.
$L_n = F_{n-1} + F_{n+1}$, $n > 1$.

A Aktivitet
Perfekta tal

Ett tal där summan av talets alla delare (utom talet självt) är lika med talet självt, kallas för ett *perfekt* tal. Talet 6 är delbart med 1, 2, 3 och 6. Summan av delarna (utom 6) är 1 + 2 + 3 = 6. Talet 6 är alltså ett perfekt tal.

1 Skriv ett program som avgör om ett inmatat tal är perfekt. Så här ska programmet se ut när det körs.

```
Det här programmet undersöker om ett tal är perfekt.
Skriv in ett positivt heltal: 6
Talet är perfekt!
```

2 Testa ditt program med talen: 6, 10, 28 och 99. Vilka av talen är perfekta?

Ett tal där summan av delarna (förutom talet självt) understiger talet självt, kallas för ett *fattigt* tal. Om summan överstiger talet, så kallas det för ett *rikt* tal.

3 Skriv ett program som avgör om ett inmatat tal är fattigt, perfekt eller rikt.

4 Skriv ett program som tar reda på vilka tal mellan 1 och 10 000 som är fattiga, perfekta respektive rika.

5 Skriv ett program som undersöker hur stor andel av talen mellan 1 och 10 000 som är fattiga, perfekta respektive rika.

Syfte och centralt innehåll

I den här aktiviteten får eleverna använda programmering för att undersöka om ett heltal är fattigt, perfekt eller rikt. På så sätt får de utnyttja programmering för att lösa problem som det är svårt och arbetsamt att lösa för hand. I den avslutande uppgiften får de undersöka hur stor andel av de positiva heltalen som är fattiga, perfekta respektive rika.

Aktiviteten kan användas i Matematik Origo 1c i samband med avsnittet *Primtal och delbarhet* eller i Matematik Origo 5 i kapitlet *Talteori*.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras nedan är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör känna till hur man skriver slingor med hjälp av kommandot `for`, villkorssatser med hjälp av kommandona `if`, `else` och `elif` samt hur man använder operatorn `%` för att beräkna resten vid heltalsdivision. Det är en fördel om eleverna även känner till hur man arbetar med listor.

Genomförande

- Förklara för eleverna vad ett perfekt tal är och låt dem för hand undersöka om något av talen 5, 6 eller 7 är perfekt. Berätta eventuellt lite historik om perfekta tal (se rubriken *Att lyfta fram*).
- Dela in eleverna i par och dela ut aktivitetsstencilen. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Om eleverna inte har så goda förkunskaper i programmering kan man välja att lösa den första uppgiften gemensamt i klassen. Det ger möjlighet att lyfta fram hur operatorn `%`, och kommandona `if`, `for` och `.append`, fungerar. Förslag på kod till den första uppgiften presenteras här nedanför.

Lösning

1
```
print("Det här programmet undersöker om
ett tal är perfekt.")
n = int(input("Skriv in ett positivt
heltal:"))
delare = []
for x in range(1, n):
  if n % x == 0:
    delare.append(x)
b = sum(delare)
if b == n:
  print("Talet är perfekt!")
else:
  print("Talet är inte perfekt.")
```

Kommentar: Det är också möjligt att skriva programmet utan att använda listor:

```
print("Det här programmet avgör om ett
tal är perfekt.")
n = int(input("Skriv ett heltal:"))
summa = 0
for x in range(1, n):
  if n % x == 0:
    summa = summa + x
if summa == n:
  print("Talet är perfekt.")
else:
  print("Talet är inte perfekt.")
```

2 6 och 28

3
```
print("Det här programmet undersöker om
ett tal är perfekt, rikt eller fat-
tigt.")
n = int(input("Skriv in ett positivt
heltal."))
delare = []
for x in range(1, n):
  if n % x == 0:
    delare.append(x)
b = sum(delare)
if b == n:
  print("Talet är perfekt!")
elif b > n:
  print("Talet är rikt!")
elif b < n:
  print("Talet är fattigt!")
```

4
```
perfekta_tal = []
rika_tal = []
fattiga_tal = []
for x in range(1, 10000):
  delare = []
  for a in range(1, x):
    if x % a == 0:
      delare.append(a)
    b = sum(delare)
    if b == x:
     perfekta_tal.append(x)
    elif b > x:
     rika_tal.append(x)
    else:
     fattiga_tal.append(x)
print("De perfekta talen mellan 1 och
10000 är:",perfekta_tal)
print("De rika talen mellan 1 och
10000 är:", rika_tal)
print("De fattiga talen mellan 1 och
10000 är:",fattiga_tal)
```

5
```
print("Det här programmet undersöker
andelen tal mellan 1 och 10000 som är
perfekta, rika och fattiga")
n = 10000
perfekta_tal = []
rika_tal = []
fattiga_tal = []
for x in range(1, n + 1):
  delare = []
  for a in range(1, x):
    if x % a == 0:
     delare.append(a)
  b = sum(delare)
  if b == x:
    perfekta_tal.append(x)
  elif b > x:
    rika_tal.append(x)
  else:
    fattiga_tal.append(x)
print("Andelen perfekta tal mellan 1
och 10000 är:",len(perfekta_tal)/n)
print("Andelen rika tal mellan 1 och
10000 är:", len(rika_tal)/n)
print("Andelen fattiga tal mellan 1
och 10000 är:",len(fattiga_tal)/n)
```

Att lyfta fram

Perfekta tal förekommer redan i Euklides Elementa. Där bevisade han att talet $\frac{q(q+1)}{2}$ är ett jämnt perfekt tal om q är ett Mersenneprimtal, dvs. ett primtal i formen $2^p - 1$ för något primtal p. Euler bevisade senare att alla jämna perfekta tal kan skrivas på denna form. Däremot är det många egenskaper hos perfekta tal, som man inte har kunnat bevisa. Det är till exempel okänt om det finns oändligt många perfekta tal eller om några av dem är udda.

Utvidgning och variation

Programmering är ett bra sätt att undersöka mönster hos tal. Eleverna kan till exempel få använda programmering för att undersöka någon av följande frågeställningar:

- Är några perfekta tal udda?

  Alla perfekta tal man känner till är jämna, men ingen har kunnat bevisa att det inte finns några udda perfekta tal.

- Hur många perfekta tal finns det?

  Det är inte känt huruvida det finns oändligt många perfekta tal.

- Undersök slutsiffran i de perfekta talen. Ser du något mönster?

  Alla jämna perfekta tal slutar på 6 eller 8.

Eleverna kan också få ställa egna frågor om fattiga, perfekta och rika tal och undersöka dem med hjälp av programmering.

A Aktivitet
Kvadratrötter

I den här aktiviteten får du skriva program som beräknar ett närmevärde till kvadratroten av ett positivt tal.

För att beräkna ett närmevärde till kvadratroten ur 2 kan man först göra en gissning och därefter använda följande formel.

$$\textit{Nästa värde} = \frac{\textit{Gissning} + \dfrac{2}{\textit{Gissning}}}{2}$$

Nästa värde kommer att vara ett bättre närmevärde till $\sqrt{2}$ än gissningen. Genom att låta *Nästa värde* vara *Gissning* i formeln, får man ett ännu bättre närmevärde. Proceduren kan man sedan upprepa tills man får ett så bra värde på $\sqrt{2}$ som man önskar.

1 Skriv ett program som beräknar ett närmevärde till $\sqrt{2}$ med hjälp av en gissning och formeln här ovanför.

2 Utgå från formeln här ovanför och skriv ett program som beräknar ett närmevärde till $\sqrt{3}$.

3 Skriv ett program som beräknar ett närmevärde till $\sqrt{x}$, där x är ett tal som användaren matar in i programmet.

Syfte och centralt innehåll

I den här aktiviteten får eleverna använda programmering för att numeriskt bestämma kvadraträtter. Eftersom uppgiften inte kräver så mycket kod, passar den bra för att introducera programmering. Programmen som eleverna skriver är rätt så enkla och det är lätt att jämföra resultaten med kända värden.

Aktiviteten är lämplig att använda som en första programmeringsaktivitet i Matematik Origo 1c eller Matematik Origo 3b.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör känna till hur man skriver slingor med hjälp av for-satser.

Genomförande

Hur aktiviteten ska genomföras beror på elevernas förkunskaper i programmering. Här presenteras ett förslag på hur aktiviteten kan genomföras för elever som inte har så stor erfarenhet av programmering.

- Dela upp eleverna så att de arbetar två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Eftersom uppgifterna kräver relativt lite kod kan det räcka med att påminna eleverna om hur man skriver for-satser, kanske genom att tillsammans skriva ett program som skriver ut de 100 första positiva heltalen:

```
for n in range (1, 101):
  print(n)
```

- För att lösa uppgift 2, kan eleverna modifiera koden i uppgift 1 på lämpligt sätt. Det är lätt att tro att man kan ersätta talet 2 i koden med talet 3, men det är bara den övre 2:an i formeln som ska bytas ut mot en 3:a. För att kunna lösa uppgiften, måste eleverna alltså först förstå hur programmet fungerar. Det kan vara klokt att gå igenom detta gemensamt i klassen efter att eleverna har arbetat med uppgift 1:

  Nästa värde är medelvärdet av två tal, där det ena talet är större än $\sqrt{2}$ och det andra talet är mindre än $\sqrt{2}$. Om *Gissning* är större än $\sqrt{2}$, kommer nämligen $\frac{2}{Gissning} = \frac{\sqrt{2} \cdot \sqrt{2}}{Gissning}$ att vara mindre än $\sqrt{2}$. Omvänt gäller att om *Gissning* är mindre än $\sqrt{2}$, så är $\frac{2}{Gissning}$ större än $\sqrt{2}$. Det är rimligt att medelvärdet av ett tal större än $\sqrt{2}$ och ett tal mindre än $\sqrt{2}$, är en bättre approximation av $\sqrt{2}$ än den första gissningen. På så sätt ger iteration en bättre och bättre uppskattning av $\sqrt{2}$.

Lösningar

1
```
gissning = float(input("Gissa ett värde på roten ur 2:"))   #Vi låter användaren mata in ett decimaltal och sparar det som gissning
for n in range(1, 15):
  nästavärde = (gissning + 2/gissning)/2
  gissning = nästavärde   #Vi döper om så att nästavärde blir gissning vid nästa iteration
  print(gissning)
print(gissning, "är ett bra närmevärde till roten ur 2.")
```

Kommentar: Roten ur 2 är med 15 decimalers noggrannhet: 1,414213562373095.

2
```
gissning = float(input("Gissa ett värde på roten ur 3:"))
for n in range(1, 15):
  nästavärde = (gissning + 3/gissning)/2
  gissning = nästavärde
  print(gissning)
print(gissning, "är ett bra närmevärde till roten ur 3.")
```

Kommentar: Roten ur 3 är med 15 decimalers noggrannhet: 1,732050807568877

3
```
x = float(input("Ange det tal x som du vill dra roten ur:"))
gissning = float(input("Gissa ett värde på roten ur x:"))
for n in range(1, 15):
  nästavärde = (gissning + x/gissning)/2
  gissning = nästavärde
  print(gissning)
print(gissning, "är ett bra närmevärde till roten ur", x)
```

Att lyfta fram

Låt gärna eleverna undersöka hur många iterationer som behöver göras för att få önskat antal korrekta värdesiffror i svaret. Diskutera hur svaret beror på precisionen i den första gissningen.

Lyft gärna fram att liknande algoritmer används i miniräknare och datorer för att beräkna närmevärden till t.ex. kvadratroten ur två och kvadratroten ur tre.

Utvidgning och variation

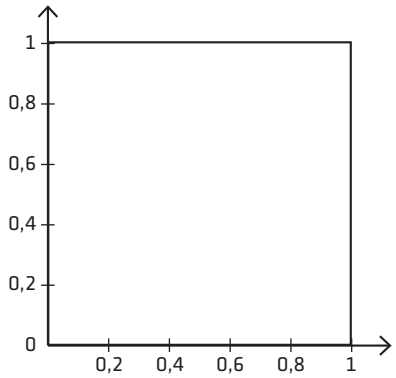
En utvidgning av aktiviteten är att ge eleverna följande kod och låta dem försöka lista ut vad programmet gör. Programmet utnyttjar en liknande strategi som den föregående algoritmen. I det här programmet får dock användaren göra två gissningar: en gissning som är större än $\sqrt{2}$ och en gissning som är mindre än $\sqrt{2}$, och programmet använder därefter intervallhalvering för att hitta ett närmevärde till $\sqrt{2}$.

```
print("Skriv ett tal som är större än roten ur 2.")
gh = float(input())   #gh står för gissning_hög
print("Skriv ett tal som är mindre än roten ur 2.")
gl = float(input())   #gl står för gissning_låg
m = (gh + gl)/2   #Den nya uppskattningen m är medelvärdet av de två gissningarna
for x in range (0, 100):
  if m * m > 2:
    gh = m
    m = (m + gl)/2
    print (m)
  else:
    gl = m
    m = (m + gh)/2
    print (m)
```

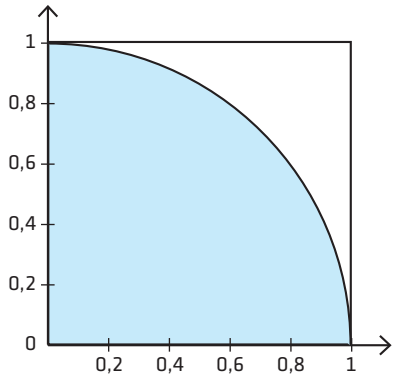
A Aktivitet
Approximera π – version 1

I den här aktiviteten får du skriva program som ger ett närmevärde till π.

1 Skriv ett program som slumpar fram 100 punkter i en kvadrat med sidan 1 l.e.



2 Utveckla programmet så att det beräknar hur många av de slumpade punkterna i uppgift 1 som hamnar innanför den inskrivna kvartscirkeln med radien 1 l.e.



3 Hur stor andel av kvadratens yta täcks av det skuggade området?

4 Använd slutsatsen i uppgift 3 och modifiera ditt program i uppgift 2 så att det beräknar ett närmevärde till π.

5 Ungefär hur många punkter måste du simulera för att få

a) två korrekta decimaler på π

b) tre korrekta decimaler på π

Ge svaret i form av en tiopotens.

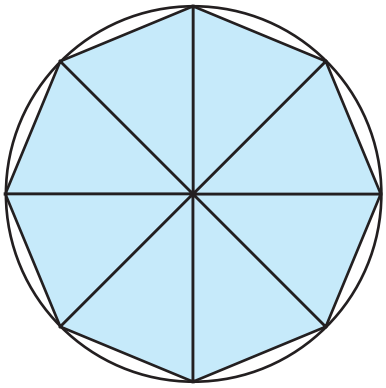
A

Aktivitet

Approximera π – version 2

Genom att följa stegen här nedanför kan du skriva ett program som ger ett närmevärde till π.

I cirkeln här nedanför har vi inskrivit en regelbunden åttahörning.



1 Beräkna den inskrivna månghörningens area.

Tänk dig att vi inskriver en regelbunden månghörning med n hörn i cirkeln.

2 Skriv en formel för hur arean av månghörningen beror av cirkelns radie r och antalet hörn n.

3 a) Skriv ett ungefärligt uttryck för π genom att jämföra månghörningens area med cirkelns.

b) Skriv ett program som beräknar uttryckets värde för $n = 8, \ldots, 100$.

4 Undersök med hjälp av programmet hur många hörn månghörningen ska ha för att du ska få

a) tre korrekta decimaler på π

b) sex korrekt decimaler på π

Ge svaret som en tiopotens.

Syfte och centralt innehåll

Syftet med den här aktiviteten är att eleverna ska få använda programmering för att uppskatta värdet på konstanten π. I den första aktiviteten får eleverna uppskatta π genom att slumpa fram punkter i enhetskvadraten och undersöka hur många av dem som hamnar innanför enhetscirkeln i första kvadranten. Det kräver att de har kunskaper om Pythagoras sats eller cirkelns ekvation. I den andra aktiviteten får eleverna uppskatta π genom att jämföra cirkelns area med arean av en inskriven månghörning. Det är svårt att göra utan kunskaper i trigonometri. Aktiviteterna kan därför vara lämpliga att använda i samband med sannolikhetslära och trigonometri i Matematik Origo 1c eller i samband med cirkelns ekvation och trigonometri i Matematik Origo 3c.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör känna till hur man upprepar kod med hjälp av kommandot `for`.

Genomförande

Eftersom aktiviteten inte kräver så mycket kod, kan den passa bra för elever som inte har så stor erfarenhet av programmering.

- Dela upp eleverna så att de arbetar två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Påminn vid behov eleverna om hur man skriver for-satser, t.ex. genom att tillsammans skriva ett program som skriver ut de 100 första positiva heltalen:

```
for n in range (1, 101):
  print(n)
```

Lösning

Approximera π – version 1

1
```
from random import *
for n in range(1, 101):
  x = uniform(0,1)
  y = uniform(0,1)  #Slumpar fram en punkt i intervallet [0, 1] med koordinaterna (x, y)
  print(x,",", y)
```

2
```
from random import *
a = 0
for n in range(1, 101):
  x = uniform(0,1)
  y = uniform(0,1)
  if x**2 + y**2 <= 1:
    a = a + 1
print(a)
```

3 Arean av cirkelsektorn är $\frac{\pi \cdot 1^2}{4}$ och arean av kvadraten är 1. Cirkelsektorns andel av kvadraten är alltså $\frac{\pi}{4}$.

4 Sannolikheten P att en slumpvist vald punkt i enhetskvadraten ligger innanför kvartscirkeln är $\frac{\pi}{4}$, dvs.

$P = \frac{\pi}{4}$

$\pi = 4 \cdot P$

Det ger programmet:

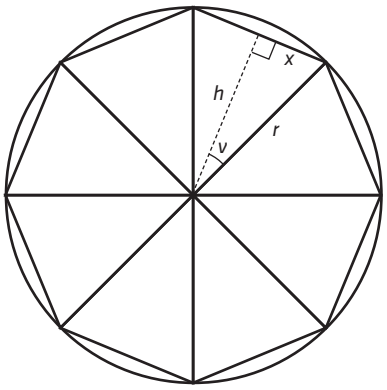
```
from math import *
from random import *
a = 0
for n in range(1, 1000001):
  x = uniform(0,1)
  y = uniform(0,1)
  if x**2 + y**2 <= 1:
    a = a + 1
print(4 * a/1000000)   #Skriver ut approximationen av π
print(pi)              #Skriver ut π med 15 korrekta decimaler
```

5 a) Ca 10^5

b) Ca 10^7

Approximera π – version 2

1



Eftersom åttahörningen är regelbunden är vinkeln v

$\frac{360°}{16} = 22{,}5°$

$\frac{x}{r} = \sin 22{,}5°$ som ger $x = r \sin 22{,}5°$

$\frac{h}{r} = \cos 22{,}5°$ som ger $h = r \cos 22{,}5°$

Arean för månghörningen blir

$A_m = n \cdot 2 \cdot \frac{x \cdot h}{2} = n \cdot r^2 \sin 22{,}5° \cos 22{,}5° =$

$= 8 \cdot 1^2 \cdot \sin 22{,}5° \cdot \cos 22{,}5° \approx 2{,}83$ a.e.

2 Med ett liknande resonemang som i uppgift 1 får vi:

$$A_m = n \cdot r^2 \sin\left(\frac{180°}{n}\right) \cos\left(\frac{180°}{n}\right)$$

Kommentar: Med areasatsen får man

$$A_m = n \cdot \frac{r^2 \cdot \sin\left(\frac{360°}{n}\right)}{2}$$

3 a) Eftersom månghörningens area närmar sig cirkelns area, $A_c = \pi r^2$, då antalet hörn blir fler, sätter vi $A_m \approx A_c$, vilket ger följande approximation av talet π:

$$\pi \approx n \cdot \cos\left(\frac{180°}{n}\right) \cdot \sin\left(\frac{180°}{n}\right)$$

Kommentar: Med areasatsen får man

$$\pi \approx n \cdot \frac{\sin\left(\frac{360°}{n}\right)}{2}$$

b)
```
from math import *
for n in range (8, 101):
   print(n * cos(pi/n)*sin(pi/n))
```
#I koden måste vi använda talet pi, eftersom kommandona cos och sin kräver argument i radianer.

4 a) $10^2\ (2 \cdot 10^2)$

b) Ca 10^4

Utvidgning och variation

Om man vill utvidga uppgiften, kan man låta eleverna skriva program som approximerar talet π genom att använda någon av följande serier

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

$$\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \ldots = \frac{\pi^2}{6}$$

Att lyfta fram

I aktiviteten presenterar vi två metoder för att approximera talet π. Båda metoderna bygger på att man redan vet att arean av en cirkel är πr^2. Första metoden utnyttjar att förhållandet mellan arean av kvartscirkeln och arean av enhetskvadraten bör vara detsamma som förhållandet mellan antalet slumpade punkter som hamnar i kvartscirkeln och det totala antalet slumpade punkter. Att utnyttja areor för att uppskatta en sannolikhet på det här sättet ingår normalt inte i någon av matematikkurserna och kan därför vara värt att förklara. I den andra metoden blir approximationen bättre ju fler hörn månghörningen har. Det kan ge tillfälle att lyfta fram begreppet gränsvärde.

A Aktivitet
Eratosthenes såll

I den här aktiviteten får du skriva ett program som skriver ut alla primtal från och med 2 till och med ett tal n.

Den grekiske matematikern Eratosthenes formulerade ca 200 f.Kr. en algoritm för att hitta primtal. Algoritmen brukar kallas *Eratosthenes* såll. Här presenterar vi en variant av algoritmen:

- Gör en lista med alla heltal från och med talet 2 till och med ett tal n.
- Talet 2 är ett primtal. Ta bort alla tal *större* än 2 som är delbara med 2 från listan. Det första tal efter talet 2 som inte är bortplockat är ett primtal, p.
- Ta bort alla tal som är större än p och delbara med p från listan. Det första tal efter talet p som inte är bortplockat är ett primtal.
- Upprepa föregående punkt så länge det senaste primtalet du fann är mindre än eller lika med $\sqrt{n}$, där n är det största talet i din lista. De tal som sedan finns kvar i listan är primtalen från och med 2 till och med n.

1 Utför algoritmen för hand när $n = 15$.

För små värden på n går algoritmen lätt att utföra för hand, men för en lång lista av tal blir den snabbt tidsödande. Din uppgift är att skriva ett program som utför Eratostenes såll. Om du vill kan du lösa problemet genom att lösa följande deluppgifter.

2 Skriv ett program som skapar en lista med alla heltal från och med 2 till och med 100.

3 a) Skriv ett program som tar bort alla jämna tal större än 2 från listan som du skapade i uppgift 1. Kom ihåg att talet 2 ska vara kvar i listan eftersom det är ett primtal.

b) Vilket är nästa primtal efter talet 2?

4 Använd Eratosthenes såll och skriv ett program som sållar fram alla primtal i din lista.

5 Utveckla programmet så att det sållar fram primtalen i en lista med heltal från och med 2 till och med n, där n är ett heltal som användaren matar in.

Syfte och centralt innehåll

Syftet med den här aktiviteten är att fördjupa elevernas kunskaper om primtal genom att skriva program som utför algoritmen *Eratosthenes såll*. Samtidigt får eleverna se hur man kan utnyttja programmering för att underlätta utförandet av tidsödande beräkningar.

Aktiviteten är lämplig att använda i samband med att man arbetar med primtal i Matematik Origo 1c eller i kapitlet *Talteori* i Matematik Origo 5.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör känna till hur man arbetar med listor, t.ex. hur man lägger till och tar bort element. De behöver också känna till kommandot %, som undersöker delbarhet, och hur man skriver slingor med hjälp av kommandona `for` och `while`.

Genomförande

Hur aktiviteten ska genomföras beror på elevernas förkunskaper i programmering. Här presenteras ett förslag på hur aktiviteten kan genomföras för elever som inte har så stor erfarenhet av programmering.

- Dela upp eleverna så att de arbetar två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Gå vid behov igenom hur Eratosthenes såll fungerar, t.ex. genom att lösa den första deluppgiften tillsammans i klassen. Om eleverna är osäkra på hur man skapar listor, hur man lägger till eller tar bort element ur listorna och hur man skriver for-satser, kan man påminna dem genom att även lösa uppgift 2 gemensamt.
- När eleverna arbetar med uppgift 4 har de nytta av programmen i uppgift 2 och 3. Ett sätt att lösa uppgift 4 är att skriva ett program som testar delbarhet med i tur och ordning 2, 3, 5 och 7. I uppgift 5 däremot behöver eleverna skriva programmet så att det testar delbarhet upp till och med primtal mindre än eller lika med $\sqrt{n}$. Ett tips kan vara att använda en while-sats.

Lösning

1 –

2
```
lista = []                  #Skapar en tom lista med namnet lista
for i in range(2, 101):
  lista.append(i)           #Lägger till alla heltal från 2 till och med 100 i listan
print(lista)
```

3 a)
```
lista = []
  for i in range(2, 101):
  lista.append(i)
  for x in lista:           #Om talet x i listan är större än 2 och delbart med 2, tas det bort .
    if x > 2 and x % 2 == 0:
      lista.remove(x)
  print(lista)
```

b) Primtalet efter 2 är 3.

4
```
lista = []
for i in range(2, 101):
  lista.append(i)
for x in lista:
  if x > 2 and x % 2 == 0:
    lista.remove(x)
for x in lista:
  if x > 3 and x % 3 == 0:
    lista.remove(x)
for x in lista:
  if x > 5 and x % 5 == 0:
    lista.remove(x)
for x in lista:
  if x > 7 and x % 7 == 0:
    lista.remove(x)
print(lista)
```

5
```
from math import *
n = int(input("Det här programmet gör en lista med alla primtal från 2 till och med n. Ange n:"))
lista = []
for i in range(2, n + 1):
  lista.append(i)
m = 0
while lista[m] <= sqrt(n):   #Programmet upprepar följande bit kod, så länge talet med index m i listan är mindre än √n
  for x in lista:            #Om talet x är större än primtalet med index m och delbart med primtalet med index m, tas det bort från listan.
    if x > lista[m] and x % lista[m] == 0:
      lista.remove(x)
  m = m + 1                  #Vi ökar index m med 1, dvs. vi upprepar koden med nästa (prim)tal i listan
print(lista)
```

Kommentar: För att förstå vad programmet gör, kan det vara en bra strategi att själv utföra varje steg i programmet för $n = 10$.

Att lyfta fram

När eleverna har arbetat med uppgift 1, kan man samla klassen och diskutera:

- Varför behöver man inte ta bort de tal som är delbara med 4?
- Varför räcker det att genomföra algoritmen upp till och med $\sqrt{n}$?

Då kan eleverna inse att man inte behöver kontrollera delbarhet med 4 eftersom alla tal som är delbara med 4 också är delbara med 2 och därmed tas bort när man eliminerar alla tal delbara med 2.

Anledningen till att det räcker att genomföra algoritmen upp till och med $\sqrt{n}$, är att varje sammansatt tal har minst en faktor mindre än eller lika med $\sqrt{n}$. (Ty om n är ett sammansatt tal, $n = a \cdot b$, kan inte *både* a och b vara större än $\sqrt{n}$, eftersom då $a \cdot b > \sqrt{n} \cdot \sqrt{n} = n$.) Om vi undersöker delbarhet med alla heltalsfaktorer från och med 2 till och med $\sqrt{n}$ utan att hitta någon, kan vi alltså vara säkra på att n är ett primtal.

Utvidgning och variation

Lyft gärna fram att det i dag finns andra och effektivare algoritmer för att sortera ut primtal från en lista med heltal.

A Aktivitet
Viètes samband

Francois Viète (1540–1603) var en fransk matematiker och jurist, som bland annat studerade ekvationer. Ett av hans viktigaste bidrag till algebran är de så kallade Viètesambanden.

Viètesambanden

Om x_1 och x_2 är rötter till ekvationen $x^2 + px + q = 0$, så gäller följande samband:

$x_1 + x_2 = -p$

$x_1 \cdot x_2 = q$

Om man vill ställa upp en andragradsekvation $x^2 + px + q = 0$ som har rötterna, x_1 och x_2, kan man använda Viètesambanden för att bestämma konstanterna p och q.

1 Skriv ett program där användaren matar in två rötter till en andragradsekvation och programmet anger motsvarande andragradsekvation i formen $x^2 + px + q = 0$.

2 Ändra programmet så att ekvationen skrivs i formen $x^2 + px = 0$, om $q = 0$ och i formen $x^2 + q = 0$, om $p = 0$.

3 Testa programmet genom att mata in följande rötter och anteckna de andragradsekvationer som programmet ger. Kontrollera att ekvationerna verkligen har de givna rötterna.

a) $x_1 = 1$ och $x_2 = 2$

b) $x_1 = 1$ och $x_2 = -1$

c) $x_1 = -4$ och $x_2 = 0$

4 Ge exempel på tre *andra* ekvationer, i formen $ax^2 + bx + c = 0$, som har rötterna

a) $x_1 = 1$ och $x_2 = 2$

b) $x_1 = 1$ och $x_2 = -1$

c) $x_1 = -4$ och $x_2 = 0$

5 Skriv ett program som, när man matar in två rötter, ger exempel på 10 olika andragradsekvationer, i formen $ax^2 + bx + c = 0$, som har de givna rötterna.

Viètes samband

Syfte och centralt innehåll

I den här aktiviteten får eleverna fördjupa sin förståelse av andragradsekvationer genom att arbeta med sambanden mellan andragradsekvationens konstanter och dess rötter. Aktiviteten passar bra när man arbetar med kapitlet om andragradsekvationer i Matematik Origo 2c eller i kapitlet om polynomekvationer i Matematik Origo 3b och Matematik Origo 3c.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör ha grundläggande kunskaper i programmering och känna till hur man sparar tal i variabler, hur man upprepar kod med hjälp av kommandot `for` och hur man skriver villkorssatser med hjälp av kommandona `if`, `elif` och `else`.

Genomförande

- Det kan vara en god idé att låta eleverna arbeta två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Uppmana elever, som har svårt att komma i gång, att använda Viètesambanden för att för hand skapa en ekvation med rötterna $x_1 = 3$ och $x_2 = 4$. När de väl förstått hur Viètesambanden kan användas, kan själva programmeringen ta vid. Vill man stötta ytterligare, kan man lösa den första uppgiften tillsammans i helklass.
- Uppgift 4 utmanar elevernas begreppsförmåga. Här behöver eleverna inse att man får en ekvivalent ekvation genom att multiplicera ekvationens båda led med en konstant.
- Avsluta lektionen med en gemensam diskussion. Se mer under rubriken *Att lyfta fram*.

Lösning

1
```
print("Detta program konstruerar en
andragradsekvation utifrån dess rötter.")
print("Ange ekvationens rötter (x1 och
x2).")
x1 = float(input("x1:"))
x2 = float(input("x2:"))
p = -(x1 + x2)
q = x1 * x2
print("Ett exempel på en ekvation med
rötterna x =",x1,"och x =",x2,"är
x^2",p, "x +",q, "= 0")
```

2
```
print("Detta program konstruerar en
andragradsekvation utifrån dess rötter.")
print("Ange ekvationens rötter (x1 och
x2).")
x1 = float(input("x1:"))
x2 = float(input("x2:"))
p = -(x1 + x2)
q = x1 * x2
if p == 0:
  print("Ett exempel på en ekvation
  med rötterna x =",x1,"och x
  =",x2,"är x^2 +" ,q, "= 0")
elif q == 0:
  print("Ett exempel på en ekvation
  med rötterna x =",x1,"och x
  =",x2,"är x^2 +" ,p, "x = 0")
else:
  print("Ett exempel på en ekvation
  med rötterna x =",x1,"och x
  =",x2,"är x^2 +" ,p, "x +" ,q, "=
  0")
```

3 a) T.ex. $x^2 - 3x + 2 = 0$

b) T.ex. $x^2 - 1 = 0$

c) T.ex. $x^2 + 4x = 0$

4 a) T.ex. $-x^2 + 3x - 2 = 0$, $2x^2 - 6x + 4 = 0$ och $3x^2 - 9x + 6 = 0$

b) T.ex. $-x^2 + 1 = 0$, $2x^2 - 2 = 0$ och $\frac{x^2}{3} - \frac{1}{3} = 0$

c) T.ex. $-x^2 - 4x = 0$ och $\frac{x^2}{2} + 2x = 0$

5
```
print("Detta program konstruerar tio andragradsekvationer som har samma rötter.")
print("Ange ekvationens rötter (x1 och x2).")
x1 = float(input("x1: "))
x2 = float(input("x2: "))
p = -(x1 + x2)
q = x1 * x2
print("Tio exempel på ekvationer med rötterna x = ", x1, "och x =", x2, "är:")
for n in range (1, 11):
  if p == 0:
    print(n, "x^2 +" ,n * q, "= 0")
  elif q == 0:
    print(n, "x^2 +" ,n * p, "x = 0")
  else:
    print(n, "x^2 +" ,n * p, "x +" ,n * q, "= 0")
```

Att lyfta fram

En av de viktiga insikterna i aktiviteten är att det finns oändligt många andragradsekvationer som har samma rötter.

En annan insikt värd att lyfta fram är att man av ekvationens form, $x^2 + px + q = 0$, $x^2 + px = 0$ eller $x^2 + q = 0$, kan avgöra rötternas natur, t.ex. om rötterna är motsatta tal eller om någon rot är noll.

Utvidgning och variation

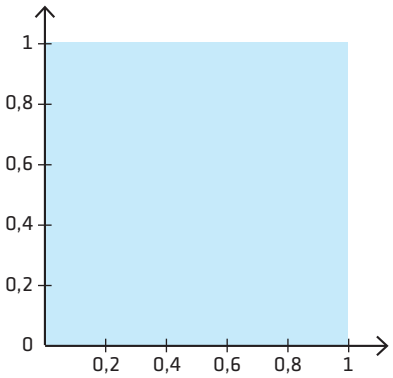
- I uppgift 2 får eleverna använda villkorssatser för att ta hand om fallen när $p = 0$ och $q = 0$. I matematik skriver man ju normalt inte ut termer vars koefficient är noll: $7x^2 + 0x - 49 = 0$ skrivs $7x^2 - 49 = 0$. Utvidga eventuellt med att diskutera hur man kan ändra i programmet så att det inte skriver ut $x^2 + -3x + 4 = 0$, utan $x^2 - 3x + 4 = 0$.
- I den här aktiviteten har vi valt att låta eleverna konstruera andragradsekvationer med givna rötter utifrån Viètesambanden. Men eleverna kan även få skriva ett program som utnyttjar faktorsatsen för att ställa upp en andragradsekvation i faktorform. Ett exempel på en ekvation med rötterna $x_1 = 2$ och $x_2 = 1$ är t.ex. $(x - 2)(x - 1) = 0$.
- Insikten att det finns oändligt många andragradsekvationer som har samma rötter, kan leda fram till upptäckten att det finns oändligt många andragradsfunktioner som har samma nollställen. Diskutera på vilket sätt graferna till funktioner med gemensamma nollställen skiljer sig åt.
- Viètesambanden kan generaliseras till polynomekvationer av högre grad. En utvidgning av aktiviteten är därför att låta eleverna undersöka sambanden för exempelvis tredjegradsekvationer. En tredjegradsekvation i formen $x^3 + ax^2 + bx + c = 0$ med rötterna x_1, x_2 och x_3 kan enligt faktorsatsen skrivas $(x - x_1)(x - x_2)(x - x_3) = 0$. Eleverna kan förenkla det faktoriserade uttrycket för att finna uttryck för konstanterna a, b och c.

A Aktivitet
Avstånd

1 Tänk dig att du slumpar fram två tal på tallinjen mellan 0 och 1.

Hur stor är sannolikheten att avståndet mellan talen är större än 0,5? Uppskatta sannolikheten och skriv sedan ett program som kan besvara frågeställningen.

2 Tänk dig att du slumpvis väljer två punkter inom kvadraten i figuren här nedanför.



a) Hur stor är sannolikheten att avståndet mellan dessa två punkter är större än 1 l.e.? Uppskatta sannolikheten och skriv sedan ett program som kan besvara frågeställningen.

b) Tänk dig att du slumpar fram två punkter i en annan rektangel, med samma area, men med sidlängderna 0,5 l.e. respektive 2 l.e. Hur stor är då sannolikheten att avståndet mellan punkterna är större än 1 l.e.? Uppskatta sannolikheten och skriv sedan ett program som kan besvara frågeställningen.

3 Tänk dig nu att du slumpar fram två punkter i en kub med sidan 1 l.e. Hur stor är sannolikheten att avståndet mellan punkterna är större än 1 l.e.?

Syfte och centralt innehåll

Den här aktiviteten syftar till att fördjupa elevernas kunskaper om avståndsberäkningar. Eleverna får slumpa fram tal i ett intervall, respektive punkter i ett koordinatsystem, och undersöka sannolikheten att avståndet mellan talen/punkterna är större än ett bestämt tal. I aktiviteten utnyttjar vi att man kan använda relativ frekvens för att uppskatta sannolikheten att en händelse ska inträffa.

Aktiviteten kan användas för att introducera eller behandla avståndsformeln, om man tror att eleverna är väl bekanta med Pythagoras sats.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör känna till hur man skriver slingor (särskilt for-satser) och hur man kan använda modulen `random` för att simulera slumpförsök.

Genomförande

Hur aktiviteten ska genomföras beror på elevernas förkunskaper i programmering. Här presenteras ett förslag på hur aktiviteten kan genomföras för elever som inte har någon större erfarenhet av programmering.

- Dela upp eleverna så att de arbetar två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Det kan vara lämpligt att lösa den första uppgiften tillsammans med eleverna och diskutera vad varje steg i koden gör. Ett förslag på kod till den första uppgiften presenteras här nedanför.

Lösning

1 Sannolikheten är ca 25 %.

```
from random import *
antal = 0
n = 100000
for n in range(1, n + 1):
  a = uniform(0, 1)    #Slumpar fram ett tal mellan 0 och 1 och sparar det i variabeln a
  b = uniform(0, 1)    #Slumpar fram ett tal mellan 0 och 1 och sparar det i variabeln b
  if (a - b) > 0.5 or (a - b) < -0.5:
    antal = antal + 1
print("Antal sträckor längre än 0.5:",antal)
print("Antal sträckor:",n )
print(antal/n)
```

- Notera att vi har valt att skriva `from random import *` i stället för `import random` på den första raden i programmet. Det har vi gjort eftersom man då kan skriva `uniform(0, 1)` i stället för `random.uniform(0, 1)`.
- Kommandot `uniform(0, 1)` genererar ett decimaltal i intervallet [0, 1]. Programmet slumpar alltså två tal på tallinjen i intervallet och kontrollerar om avståndet mellan dem är större än 0,5 l.e.
- Om man vill kan man lyfta fram att villkoret `if (a - b) > 0.5 or (a - b) < -0.5:` enklare kan skrivas med absolutbeloppsfunktionen: `if abs(a - b) > 0.5:`. Då måste man dock först importera modulen `math` med raden:

  ```
  from math import *
  ```

- Eleverna kan lösa övriga uppgifter i aktiviteten genom att modifiera ovanstående kod på lämpligt sätt.

2 a) Sannolikheten är ca 2,5 %.

```
from random import *
antal = 0
n = 100000
for n in range(1, n + 1):
  x_1 = uniform(0, 1)  #Vi slumpar fram en punkt med koordinaterna (x_1, y_1)
  y_1 = uniform(0, 1)
  x_2 = uniform(0, 1)  #Vi slumpar fram en punkt med koordinaterna (x_2, y_2)
  y_2 = uniform(0, 1)
  if ((x_1 - x_2)**2 + (y_1 -
  y_2)**2)**0.5 > 1:
    antal = antal + 1
print("Antal sträckor större än
1:",antal)
print("Antal sträckor:",n )
print(antal/n)
```

b) Sannolikheten är ca 26 %.

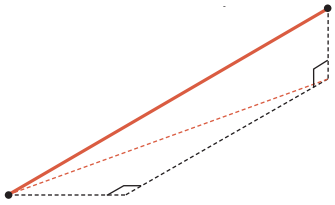
```
from random import *
antal = 0
n = 100000
for n in range(1, n + 1):
  x_1 = uniform(0, 0.5)
  y_1 = uniform(0, 2)
  x_2 = uniform(0, 0.5)
  y_2 = uniform(0, 2)
  if ((x_1 - x_2)**2 + (y_1 -
  y_2)**2)**0.5 > 1:
    antal = antal + 1
print("Antal sträckor större än
1:",antal)
print("Antal sträckor:",n )
print(antal/n)
```

3 Sannolikheten är ca 9,0 %.

```
from random import *
antal = 0
n = 100000
for n in range(1, n + 1):
  x_1 = uniform(0, 1)  #Vi slumpar fram en punkt med koordinaterna (x_1, y_1, z_1)
  y_1 = uniform(0, 1)
  z_1 = uniform(0, 1)
  x_2 = uniform(0, 1)  #Vi slumpar fram en punkt med koordinaterna (x_2, y_2, z_2)
  y_2 = uniform(0, 1)
  z_2 = uniform(0, 1)
  if ((x_1 - x_2)**2 + (y_1 - y_2)**2
  + (z_1 - z_2)**2)**0.5 > 1:
    antal = antal + 1
print("Antal sträckor större än
1:",antal)
print("Antal sträckor:",n )
print(antal/n)
```

Att lyfta fram

I aktiviteten får eleverna möjlighet att på egen hand upptäcka att avståndsformeln är en tillämpning av Pythagoras sats, där man använder koordinater för att beräkna kateternas sidlängder. Det kan kräva lite algebraisk färdighet att härleda ett korrekt uttryck för avståndet mellan två punkter i rummet. Stötta gärna med följande figur.



Utvidgning och variation

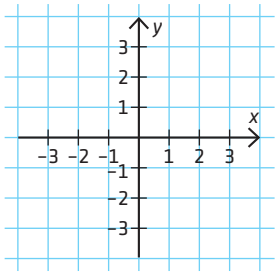
Om man vill kan man utvidga aktiviteten med följande problem:

Om du har ett sugrör med en viss längd och slumpvis väljer två punkter längs sugröret, vad är då sannolikheten att sugröret kan vikas till en triangel där hörnen ligger i de slumpade punkterna?

A Aktivitet
Antal punkter på cirkeln

I den här aktiviteten får du undersöka egenskaper hos cirklar genom att utföra beräkningar med hjälp av ett datorprogram.

1 Skriv ett program som skriver ut koordinaterna för alla punkter med heltalskoordinater som finns i koordinatsystemet här nedanför.



2 a) Skriv ett program som skriver ut koordinaterna för de punkter i uppgift 1 som ligger på cirkeln med medelpunkt i origo och radien 3 l.e.

Kontrollera att programmet fungerar genom att för hand ta reda på vilka av punkterna i uppgift 1 som ligger på cirkeln.

b) Ändra programmet så att det skriver ut *antalet* punkter med heltalskoordinater som ligger på cirkeln.

3 Tänk dig att vi ritar fler cirklar i koordinatsystemet. Var och en av cirklarna har medelpunkt i origo och en heltalsradie mellan 1 l.e. och 30 l.e. Vilken av cirklarna har flest punkter med heltalskoordinater? Hur många punkter med heltalskoordinater har den cirkeln?

Antal punkter på cirkeln

Syfte och centralt innehåll

I den här aktiviteten får eleverna undersöka egenskaper hos cirklar med hjälp av programmering. Samtidigt får de se hur man med programmering kan utföra ett stort antal beräkningar och på så sätt undersöka något man är nyfiken på. Aktiviteten är lämplig att använda i Matematik Origo 3c i samband med avsnittet om cirkelns ekvation.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör känna till hur man skriver slingor och nästlade slingor (särskilt for-satser).

Genomförande

Hur aktiviteten kan genomföras beror på elevernas förkunskaper i programmering. Här presenteras ett förslag på hur aktiviteten kan genomföras för elever som inte har så stor erfarenhet av programmering.

- Dela upp eleverna så att de arbetar två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Det kan vara bra att påminna eleverna om hur man skriver for-satser och nästlade for-satser, t.ex. genom att tillsammans gå igenom lösningen till den första deluppgiften.

Lösning

1
```
for a in range(-3, 4):
    for b in range (-3, 4):
      print ("(", a , ",", b, ")")
```

Kommentar: Vi har valt att bara skriva ut punkterna med heltalskoordinater i intervallet [–3, 3].

2 a)
```
r = 3
for a in range(-r**2, r**2 + 1):
  for b in range (-r**2, r**2 + 1):
    if a**2 + b**2 == r**2:
      print ("(", a , ",", b, ") är
      en punkt på cirkeln.")
```

b)
```
r = 3
antal = 0
for a in range(-r**2, r**2 + 1):
  for b in range (-r**2, r**2 + 1):
    if a**2 + b**2 == r**2:
      antal = antal + 1
print("Det finns", antal, "punkter
med heltalskoordinater på cirkeln
med radien", r, "l.e.")
```

Svar: Det finns 4 punkter med heltalskoordinater på cirkeln med radien 3 och medelpunkt i origo.

3
```
for r in range(1, 31):
  antal = 0
  for a in range(-r**2, r**2 + 1):
    for b in range (-r**2, r**2 + 1):
      if a**2 + b**2 == r**2:
        antal= antal + 1
  print("Det finns", antal, "punkter
  med heltalskoordinater på cirkeln
  med radien", r)
```

Svar: Cirkeln med radien 25 l.e. har flest punkter med heltalskoordinater (20 st).

Att lyfta fram

Lyft gärna fram att undersökningar, där man strukturerat utför en stor mängd beräkningar, i stort sett är omöjliga att genomföra för hand, men relativt snabbt kan genomföras med ett välskrivet och genomtänkt program. Resultatet som programmet ger kan sedan leda till nya matematiska insikter och frågeställningar. Diskutera gärna med eleverna vilka frågor resultatet väcker hos dem. Man kan t.ex. fråga sig varför antalet punkter med heltalskoordinater på cirklarna alltid är delbart med 4 och om det finns någon formel som kan förutsäga antalet punkter med heltalskoordinater på en cirkel med radien r l.e.

I den här aktiviteten får du skriva program som beräknar okända sidor och vinklar i trianglar.

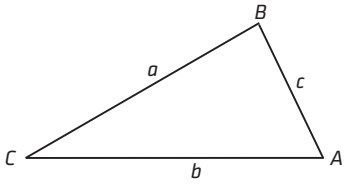
Sinussatsen

I en triangel med vinklarna A, B, C och de motstående sidorna a, b, c gäller sambandet

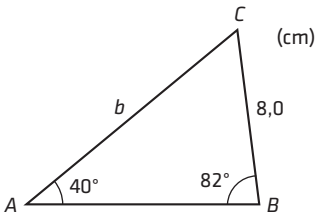
$$\frac{\sin A}{a} = \frac{\sin B}{b} = \frac{\sin C}{c}$$

eller

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$$



1 Använd sinussatsen för att beräkna längden av sidan b i figuren.



2 Det finns olika enheter för att mäta vinklar. Programspråket Python använder vinkelmåttet *radianer*, där $1° = \frac{\pi}{180}$ radianer. Det här programmet skriver ut värdet av $\sin v$ för en vinkel v angiven i radianer.

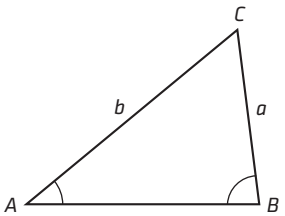
```
from math import *
v = float(input("Ange en vinkel i radianer:"))
print(sin(v))
```

Ändra i programmet så att det skriver ut sinus för en vinkel som är given i grader.

Tips! Talet π skrivs `pi` i Python 3.

3 Skriv ett program som beräknar längden av sidan b i figuren här ovanför.

4 a) Utveckla programmet så att det beräknar sidan b, om användaren matar in vinklarna A och B samt längden av sidan a för en triangel som den i figuren här nedanför.

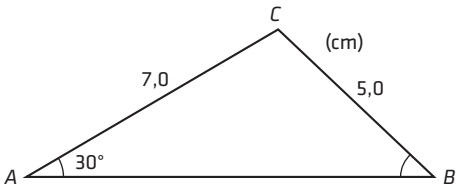


b) Mata in $B = 90°$, $A = 45°$, $a = 1$ i programmet och kontrollera att det ger samma svar som när du bestämmer längden av sidan b för hand.

I den här delen av aktiviteten får du använda sinussatsen för att beräkna storleken av okända *vinklar* i en triangel.

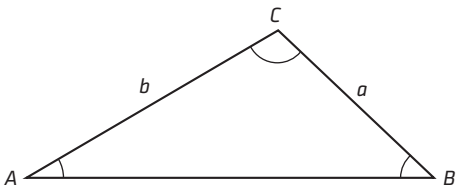
5 Använd sinussatsen för att beräkna vinkeln B i figuren här nedanför. Observera att figuren är en principskiss och att det finns två möjliga värden på vinkeln B utifrån informationen i figuren.

Den inversa sinusfunktionen $\sin^{-1}$ skrivs `asin()` i Python 3.



6 Skriv ett program som löser uppgift 5.

7 Utveckla programmet så det beräknar triangelns samtliga vinklar om man matar in längden av sidorna a och b samt vinkeln A för en triangel som den i figuren här nedanför.



8 Beroende på storleken av vinkel A och längden av sidorna a och b, finns det ingen, en eller två trianglar som uppfyller villkoren i figuren här ovanför. Du ska få använda ditt program för att undersöka när det *inte* finns någon triangel som uppfyller villkoren.

- Välj en spetsig vinkel A och variera längden av sidorna a och b. När finns det ingen triangel som uppfyller villkoren?
- Välj en trubbig vinkel A och variera längden av sidorna a och b. När finns det ingen triangel som uppfyller villkoren?

Syfte och centralt innehåll

Den här aktiviteten syftar till att fördjupa elevernas förståelse av sinussatsen. I aktiviteten får eleverna först för hand bestämma okända sidor och vinklar i trianglar och därefter skriva program som löser samma uppgift. Eleverna får sedan utveckla sitt program så att det kan användas för att beräkna okända sidor och vinklar i godtyckliga trianglar. På så sätt skaffar sig eleverna ett verktyg för att undersöka när det finns ingen, en eller två trianglar som uppfyller givna villkor.

Aktiviteten passar bra i avsnittet om triangelsatserna i Matematik Origo 3c. Uppgift 2, där eleverna får skriva ett program för att beräkna sin v för en vinkel v som är angiven i grader, kan även användas vid arbetet med radianer i Matematik Origo 4.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör ha grundläggande kunskaper i programmering, t.ex. känna till hur man avrundar tal till lämpligt antal decimaler och hur man sparar tal i variabler. I aktiviteten visar vi hur trigonometriska funktioner som sin x och $\sin^{-1} y$ fungerar i Python 3.

Genomförande

- Det kan vara en god idé att låta eleverna arbeta två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Aktiviteten är uppdelad på två sidor. Det gör det möjligt att arbeta med de två sidorna vid olika tillfällen eller välja att bara jobba med aktivitetens ena del. Om man väljer att arbeta med aktiviteten i sin helhet kan det vara klokt att ha en gemensam uppsamling efter att klassen har jobbat med den första delen, dvs. efter uppgift 1–4.
- Om eleverna inte är så vana vid programmering kan man stötta dem genom att skriva programmet i uppgift 3 tillsammans i helklass.
- En svårighet i uppgift 8 är att välja längden av sidorna a och b på ett strukturerat sätt. Här är det ju inte de absoluta längderna av a och b som är intressanta, utan snarare hur längderna a och b förhåller sig till varandra. Ett tips till eleverna kan vara att undersöka fallen $a < b$, $a = b$ och $a < b$.
- Avsluta lektionen med en gemensam diskussion. Se mer under rubriken *Att lyfta fram*.

Lösning

1 $b \approx 12$ cm

2
```
from math import *
v = float(input("Ange en vinkel i grader:"))
print(sin(v * pi/180))
```

3
```
from math import *
a = 8
A = 40
B = 82
sin_A = sin(A * pi/180)
sin_B = sin(B * pi/180)
b = a * sin_B/sin_A
print("Den sökta sidan är ca",
round(b, 0), "cm.")
```
#Kommandot round() avrundar här svaret till heltal

4 a)
```
from math import *
A = float(input("Ange storleken av vinkel A:"))
B = float(input("Ange storleken av vinkel B:"))
a = float(input("Ange längden av sidan a:"))
sin_A = sin(A * pi/180)
sin_B = sin(B * pi/180)
b = a * sin_B/sin_A
print("Längden av sidan b är ca",
round(b, 2), "l.e.")
```
#Kommandot round() avrundar här svaret till två decimaler

b) Om programmet fungerar ger det svaret $b = \sqrt{2} \approx 1{,}41$ l.e.

5 $\angle B \approx 44°$ och $\angle B \approx 136°$

6
```
from math import *
a = 5
b = 7
A = 30
sin_A = sin(A * pi/180)
sin_B = b * sin_A/a
B = asin(sin_B)     #Kommandot asin() är den inversa sinusfunktionen
B = B * 180/pi      #Vi omvandlar vinkeln B till grader
if B - A > 0:
  print("Den sökta vinkeln är ca"
  ,round (B, 0), "grader eller",
  round(180 - B, 0), "grader.")
          #Om C = 180 – A – (180 – B) = B – A > 0 finns det två möjliga värden på vinkel B
else:
  print("Den sökta vinkeln är ca"
  ,round (B, 0), "grader.")
```

7
```
from math import *
a = float(input("Ange längden av sida a:"))
b = float(input("Ange längden av sida b:"))
A = float(input("Ange storleken av vinkel A:"))
sin_A = sin(A * pi/180)
sin_B = b * sin_A/a
B = asin(sin_B)
B = B * 180/pi
C = 180 - A - B
if B - A > 0:
  print("Triangelns vinklar är ca" ,A,
  "grader," ,round(B, 0), "grader och"
  ,round(C, 0), "grader eller", A,
  "grader," ,round(180 - B, 0), "grader och" ,round(B - A, 0), "grader.")
else:
  print("Triangelns vinklar är ca,"
  ,A, "grader," ,round(B, 0), "grader
  och" ,round(C, 0), "grader.")
```

8
- Om vinkel A är spetsig och $a < b \cdot \sin A$ finns det ingen triangel som uppfyller villkoren.
- Om vinkel A är trubbig och $a \leq b$ finns det ingen triangel som uppfyller villkoren.

Att lyfta fram

Eftersom syftet med aktiviteten är att eleverna ska utveckla en fördjupad förståelse av sinussatsen och få insikt om vilka mått på sidor och vinklar som ger faktiska trianglar, bör aktiviteten avslutas med en lärarledd diskussion i helklass.

- Det kan vara intressant att diskutera varför det bara finns en möjlig längd av sidan b i uppgift 1, men däremot två möjliga vinklar B i uppgift 5. En del av förklaringen ligger i kongruensfallen för trianglar. Eftersom vinkel C i uppgift 1 kan bestämmas med triangelns vinkelsumma, är triangeln faktiskt entydigt bestämd enligt kongruensfallet VSV. Triangeln i uppgift 5 uppfyller dock inte något av kongruensfallen SSS, VSV eller SVS. Därför kan det finnas flera trianglar som uppfyller villkoren.
- Diskutera i vilka fall den inmatade informationen i uppgift 6 och 7 ger två möjliga trianglar. Problemet kan omformuleras till i vilka fall som den andra möjliga vinkeln $B_2 = 180 - B_1$ ger rimliga värden på triangelns tredje vinkel, C_2. Vi får

  $$C_2 = 180 - A - B_2 = 180 - A - (180 - B_1) = B_1 - A$$

  Så länge $B_1 - A > 0$, så finns det alltså två möjliga trianglar med var sin uppsättning vinklar:

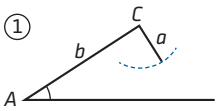
  A, B_1, C_1 och A, B_2, C_2

  Diskutera med eleverna hur man kan formulera programmet, så att det bara skriver ut två lösningar i de fall då två lösningar existerar.
- Sammanfatta elevernas slutsatser i uppgift 8. Undersök om de har kommit fram till att:
  - Om vinkel A är spetsig och $a < b \cdot \sin A$ finns det ingen triangel som uppfyller villkoren.
  - Om vinkel A är trubbig och $a \leq b$ finns det ingen triangel som uppfyller villkoren.

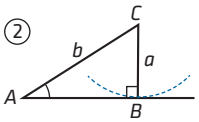
Utvidgning och variation

Aktiviteten kan utvidgas till att reda ut i vilka fall man får ingen, en eller två möjliga trianglar i uppgift 8. Följande bilder kan då vara ett stöd:

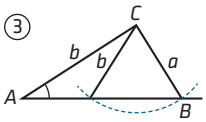
Vinkel *A* är spetsig

① 

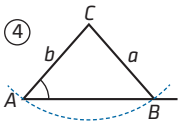
Fall 1: $a < b$. Ingen lösning. Sidan a är för kort för att nå triangelns bas.

② 

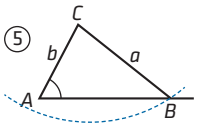
Fall 2: $a < b$. En lösning. Sidan a är precis så lång att den (vinkelrätt) når triangelns bas.

③ 

Fall 3: $a < b$. Två lösningar. Sidan a kan möta triangelns bas på två sätt.

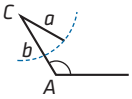
④ 

Fall 4: $a = b$. En lösning. Triangeln blir likbent.

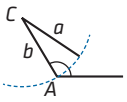
⑤ 

Fall 5: $a > b$. En lösning. Sidan a möter basen i *en* punkt.

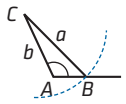
Vinkel *A* är trubbig eller rät

⑥ 

Fall 6: $a < b$. Ingen lösning. Sidan a är för kort för att nå triangelns bas.

⑦ 

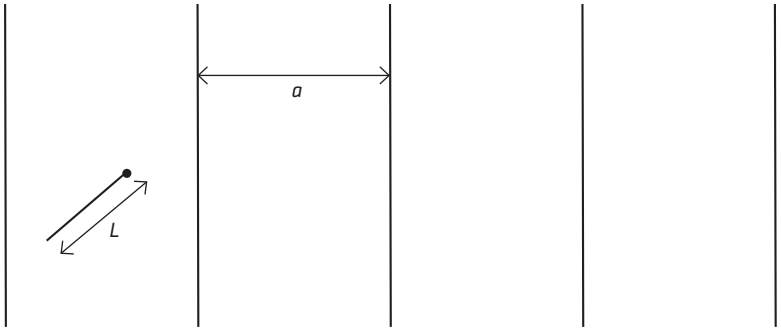
Fall 7: $a = b$. Ingen lösning. Sidan a är för kort för att nå triangelns bas.

⑧ 

Fall 8: $a > b$. En lösning. Sidan a möter basen i *en* punkt.

A Aktivitet
Buffons nålproblem

Tänk dig att du har ett oändligt stort vitt papper med parallella linjer framför dig. Det är lika stort avstånd mellan varje par av linjer.



Du släpper en nål på pappret. Hur stor är sannolikheten att nålen landar så att den skär (träffar) någon av de parallella linjerna? I den här uppgiften får du undersöka hur sannolikheten beror av nålens längd L l.e. och avståndet mellan linjerna a l.e.

Du kan börja din undersökning så här:

1 Skriv ett program som undersöker sannolikheten att nålen träffar en linje när nålen är lika lång som avståndet mellan linjerna, dvs. om $a = L$. Simulera 1 000 000 kast med nålen.

2 Fundera: Hur beror sannolikheten att nålen träffar en linje på nålens längd?

3 Undersök sannolikheten att nålen träffar en linje när nålen är *kortare* än avståndet mellan linjerna, genom att skriva ett program som simulerar experimentet när nålens längd är

a) hälften av avståndet mellan linjerna, $L = \frac{a}{2}$

b) en fjärdedel av avståndet mellan linjerna, $L = \frac{a}{4}$

Simulera 1 000 000 kast med nålen.

4 Jämför resultaten i de simuleringar som du har gjort hittills. Vilka slutsatser kan du dra av dina undersökningar?

5 Skriv ett program som undersöker sannolikheten för att nålen träffar en linje om nålen är *längre* än avståndet mellan linjerna. Börja med fallet då $L = 2a$. Simulera 1 000 000 kast med nålen. Vilka slutsatser kan du dra av din undersökning?

Syfte och centralt innehåll

I den här aktiviteten får eleverna använda programmering för att undersöka *Buffons nålproblem*. Det klassiska problemet går ut på att uppskatta sannolikheten att en nål skär någon linje, om man släpper den på ett vitt papper med ekvidistanta parallella linjer. Problemet har fått sitt namn efter fransmannen Georges-Louis Leclerc de Buffon (1707–1788).

Det räcker att slumpa två parametrar, nämligen var nålens mittpunkt kommer att hamna och vinkeln som nålen kommer att vrida sig i förhållande till de parallella linjerna. Uppgiften kräver att eleverna kan utföra grundläggande trigonometriska beräkningar (trigonometri i rätvinkliga trianglar). Därför kan aktiviteten vara lämplig att använda i samband med sannolikhetslära och trigonometri i Matematik Origo 1c, i trigonometrikapitlet i Matematik Origo 3c eller i trigonometrikapitlet i Matematik Origo 4.

Eftersom Python 3 använder radianer som vinkelmått är det en fördel om eleverna är bekanta med radianer. Detta är dock ingen nödvändighet om man visar dem hur man omvandlar mellan grader och radianer.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör ha grundläggande kunskaper i programmering, känna till hur man importerar och använder modulen `random` samt hur man skriver slingor (särskilt for-satser).

Genomförande

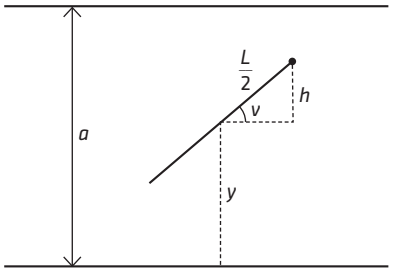
Hur aktiviteten ska genomföras beror på elevernas förkunskaper i programmering. Här presenteras ett förslag på hur aktiviteten kan genomföras för elever som inte har så stor erfarenhet av programmering.

- Introducera gärna aktiviteten genom att demonstrera problemet för eleverna. Utför ett antal försök tillsammans i klassen, där du släpper en riktig nål på ett papper med parallella linjer.
- Det kan krävas lite ledning för att eleverna ska komma fram till att det är nålens mittpunkt och vridningsvinkeln som är lämpliga att använda som parametrar för att bestämma nålens läge. Gå eventuellt igenom detta tillsammans, efter att eleverna själva fått fundera en stund. I samband med det kan det också vara lämpligt att visa hur man omvandlar från grader till radianer, eftersom de trigonometriska funktionerna i Python 3 använder radianer som vinkelmått.
- Dela upp eleverna så att de arbetar två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Det kan vara lämpligt att gå igenom den första uppgiften tillsammans med eleverna i helklass och diskutera vad varje steg i koden gör. Ett förslag på kod till den första uppgiften presenteras här nedanför.

Lösning

1 Vi slumpar fram nålens position genom att slumpa fram två parametrar:

1. Var nålens mittpunkt befinner sig i höjdled i förhållande till de parallella linjerna $(0 \le y \le a)$
2. Nålens vridning kring mittpunkten $(0 \le v \le 90°)$



Med hjälp av dessa parametrar kan man bestämma höjden h i figuren med hjälp av trigonometri

$$\sin v = \frac{h}{\frac{L}{2}}$$

$$h = \frac{L}{2} \cdot \sin v$$

Om $y + h \ge a$ eller om $y - h \le 0$, kommer nålen att skära någon av de parallella linjerna. Vi skriver nu ett program som simulerar slumpförsöket med nålen.

```
from math import *
from random import *
antal = 0
n = 1000000
a = 1
L = a
for n in range(1, n + 1):
  y = uniform(0, a)        #Vi slumpar fram y-koordinaten för nålens mittpunkt som ett tal mellan 0 och a
  v = uniform(0, pi/2)     #Vi slumpar fram en vinkel i radianer mellan 0 och π/2.
  h = sin(v) * L/2
  if (y + h) >= a or (y - h) <= 0:
    antal = antal + 1
print("Antal gånger nålen träffar någon linje:",antal)
print("Antal kast:",n )
print("Relativ frekvens:",antal/n)
```

Man kan teoretiskt visa att den sökta sannolikheten är

$$P(\text{skära någon linje}) = \frac{2L}{a\pi},\ L \le a$$

I det undersökta fallet $L = a$ bör vi alltså i simuleringen få

$$P(\text{skära någon linje}) = \frac{2}{\pi} \approx 0{,}636\ldots$$

Kommentar: De trigonometriska funktionerna i Python 3 kräver att vinkelns anges i radianer. Därför slumpar vi i programmet här ovanför fram en vinkel mellan 0 och $\frac{\pi}{2}$. Man kan också välja att slumpa fram en vinkel v mellan 0° och 90° och sedan omvandla denna vinkel till radianer: $v \cdot \frac{\pi}{180}$.

2 Sannolikheten att nålen träffar linjen minskar när nålens längd minskar.

3 a) Ersätt `L = a` i koden ovan med `L = a/2`. Teoretiskt bör vi i simuleringen få resultatet

$$P(\text{skära någon linje}) = \frac{1}{\pi} \approx 0{,}318\ldots$$

b) Ersätt `L = a` i koden ovan med `L = a/4`. Teoretiskt bör vi i simuleringen få resultatet

$$P(\text{skära någon linje}) = \frac{1}{2\pi} \approx 0{,}159\ldots$$

4 Man kan t.ex. se att sannolikheten att nålen träffar linjen halveras när nålens längd halveras. Det kan leda till slutsatsen att den sökta sannolikheten är proportionell mot $\frac{L}{a}$ med proportionalitetskonstanten $\frac{2}{\pi} \approx 0{,}636\ldots$

5 Ersätt `L = a` i koden ovan med `L = 2 * a`. Med programmet får vi att den sökta sannolikheten är ca 83,7 %. Sannolikheten att nålen träffar linjen är inte längre proportionell mot förhållandet $\frac{L}{a}$, när nålen är längre än avståndet mellan linjerna.

Kommentar: Man kan teoretiskt visa att sannolikheten i fallet då $L > a$ bestäms av uttrycket

$$\frac{2}{\pi} \cdot \cos^{-1}\left(\frac{a}{L}\right) + \frac{2}{\pi} \cdot \frac{L}{a}\left(1 - \sqrt{\left(1 - \left(\frac{a}{L}\right)^2\right)}\right)$$

Med $L = 2a$ får vi:

$$\frac{2}{\pi} \cdot \cos^{-1}\left(\frac{1}{2}\right) + \frac{4}{\pi} \cdot \left(1 - \sqrt{\frac{3}{4}}\right) \approx 0{,}837$$

Att lyfta fram

Förmodligen är det inte enkelt för eleverna att förstå vilken matematisk modell de bör välja för att lösa problemet. Här kan man behöva stötta, t.ex. genom att tipsa om att nålens läge kan modelleras med läget för dess mittpunkt och dess vinkel i förhållande till de parallella linjerna. (Se lösningen till Uppgift 1). Det kan värt att diskutera med eleverna att det räcker att betrakta endast två parallella linjer och att nålens läge i x-led inte är intressant.

Det kan vara givande att nämna att de sökta sannolikheterna kan beräknas teoretiskt och att de alla kommer att innehålla talet π. Med hjälp av simuleringar av Buffons nålproblem kan man alltså uppskatta ett värde på π.

Utvidgning och variation

Om man vill utvidga uppgiften, kan man låta eleverna använda problemet för att uppskatta ett värde på talet π. Då utnyttjar man att den teoretiska sannolikheten ges av $\frac{2L}{a\pi}$ när $L \leq a$. Genom att köra simuleringen, får man ett närmevärde till denna sannolikhet, och kan på så sätt bestämma ett närmevärde till π.

Om problemet används i kurs 4 eller kurs 5, kan man resonera sig fram till de sökta sannolikheterna och beräkna dem med hjälp av integraler. Men det kräver att man kan integrera de inversa trigonometriska funktionerna. En härledning går att finna på nätet om man söker på Buffons nålproblem.

A

Aktivitet

Geometrisk talföljd och geometrisk summa

Programmet här nedanför beräknar det *n*:te elementet i en geometrisk talföljd.

```
n = int(input("Ange elementets index n:"))
print("Elementet med index", n, "i talföljden är", 5 * 2**(n - 1))
```

1 Ange talföljdens

a) första element

b) kvot

2 Ändra i programmet så att det skriver ut det *n*:te elementet i den geometriska talföljden

$1, \frac{1}{3}, \frac{1}{9}, \frac{1}{27}, \ldots$

3 Ändra programmet som du skrev i uppgift 2, så att det skriver ut de

a) 100 första elementen i talföljden

b) *m* första elementen i talföljden, där användaren matar in talet *m*

Om man adderar elementen i en geometrisk talföljd, får man en geometrisk summa.

4 Skriv ett program som adderar de *m* första termerna i den geometriska summan

$1 + \frac{1}{3} + \frac{1}{9} + \frac{1}{27} + \ldots$

5 Undersök värdet av den geometriska summan i uppgift 4 för $m = 10, 20, 30, 40, \ldots$

a) Vilken slutsats drar du?

b) Bevisa din slutsats.

Geometrisk talföljd och geometrisk summa

Syfte och centralt innehåll

Den här aktiviteten syftar till att fördjupa elevernas kunskaper om programmering, geometriska talföljder och summor. Aktiviteten är lämplig att genomföra i samband med kapitlet *Geometrisk summa* i Matematik Origo 3b eller i kapitlet *Talteori* i Matematik Origo 5.

För att aktiviteten ska vara lämplig att använda i Matematik Origo 3b, där eleverna inte tidigare har arbetat med programmering som problemlösningsverktyg i gymnasieskolan, har vi valt att presentera ett färdigt program som eleverna får modifiera och bygga vidare på.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Det är en fördel om eleverna är bekanta med kommandona `print` och `input()` och att de känner till hur man skriver for-satser.

Genomförande

Hur aktiviteten ska genomföras beror på elevernas förkunskaper i programmering. Här presenteras ett förslag på hur aktiviteten kan genomföras för elever som inte har så stor erfarenhet av programmering.

- Gå igenom koden som presenteras i aktivitetens inledning och diskutera vad varje rad i koden gör. Förklara kommandona `print()`, `input()` och `int()` och betydelsen av tecken som citattecken och kommatecken. Kör koden och diskutera den första uppgiften.
- Låt eleverna arbeta med aktiviteten två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- När eleverna löser uppgift 3 behöver de använda sig av en for-sats. Gå vid behov gemensamt igenom hur man använder for-satser för att upprepa kod.

Lösning

1 a) 5

b) 2

2 Ändra uttrycket `5 * 2**(n - 1)` på sista raden i programmet till `1 * (1/3)**(n - 1)`.

3 a)
```
for n in range (1, 101):
    print("Element", n, "är", 1 *
    (1/3)**(n - 1))
```

b)
```
m = int(input("Ange m:"))
for n in range (1, m + 1):
    print("Element", n, "är", 1 *
    (1/3)**(n - 1))
```

4 T.ex.
```
m = int(input("Ange m:"))
summa = 0
for n in range (1, m + 1):
  summa = summa + 1 * (1/3)**(n - 1)
print(summa)
```

Eller med listor:
```
m = int(input("Ange m:"))
lista = []
for n in range (1, m + 1):
  lista.append(1 * (1/3)**(n - 1))
print(sum(lista))
```

5 a) Summan verkar konvergera mot 1,5.

b) Med formeln för geometrisk summa får vi:

$$\frac{1\left(1-\left(\frac{1}{3}\right)^m\right)}{1-\frac{1}{3}} = \frac{3}{2}\cdot\left(1-\left(\frac{1}{3}\right)^m\right) \to \frac{3}{2}$$

när $m \to \infty$

Att lyfta fram

Aktiviteten ger möjlighet att introducera och befästa programmeringskommandon som `print()`, `input()`, `int()` och for-satser.

Den geometriska summan i uppgift 5 närmar sig 3/2 när antalet termer ökar. Att serien konvergerar mot 3/2 kan eleverna bevisa med formeln för geometrisk summa och ett gränsvärdesresonemang. Det är värt att lyfta fram att programmet ger resultatet 1,5 redan för $m = 40$, men att serien i verkligheten aldrig når 1,5. Att programmet ändå ger svaret 1,5, beror på att det utför beräkningarna med begränsad noggrannhet.

Aktiviteten är ett exempel på hur programmering kan användas för att göra hypoteser som sedan bevisas matematiskt.

Utvidgning och variation

Elever som behöver extra utmaning kan få använda programmering för att undersöka konvergensen av andra serier, t.ex.

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots$$

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots$$

$$1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \ldots$$

Ett sätt att göra en sådan undersökning är att låta eleverna skriva program där användaren matar in den geometriska summans första element, kvot och antal termer, och programmet beräknar den geometriska summans värde.

Eleverna kan också använda programmering för att beräkna element i aritmetiska talföljder, eller i talföljder som varken är geometriska eller aritmetiska, t.ex. Fibonaccis talföljd.

Aktiviteten kan även utvidgas till att lösa problem relaterade till annuitetslån. (Se aktiviteten *Olika typer av lån*)

A Aktivitet
Olika typer av lån

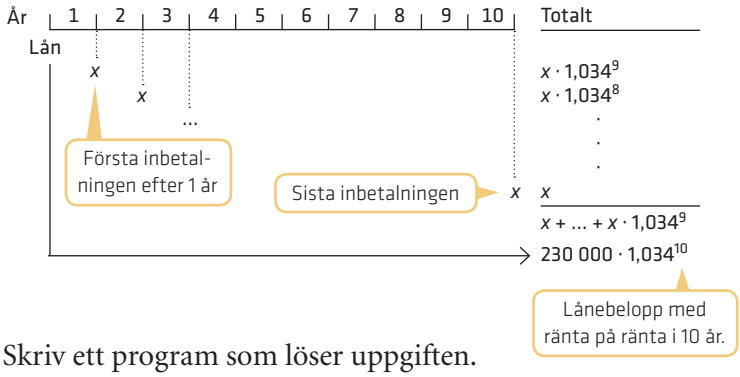
Celine har lånat 230 000 kronor för att köpa en bil till räntesatsen 3,4 %. Hon vill betala tillbaka lånet med tio lika stora annuiteter under 10 år, med start efter ett år.

Foto: Shutterstock

Annuitetslån

Ett annuitetslån betalas tillbaka med ett lika stort belopp vid varje återbetalningstillfälle tills lånet är återbetalt. Beloppet kallas *annuitet* och består av både ränta och amortering.

1 Hur stor ska varje annuitet vara? Lös uppgiften med en ekvation. Ta hjälp av bilden här nedanför.



2 Skriv ett program som löser uppgiften.

3 Ändra programmet så att Celine kan mata in

- belopp
- räntesats
- antal år (annuiteter)

och få reda på hur stor varje annuitet blir.

4 Celine funderar på om hon i stället ska välja rak amortering. Då betalar hon ett fast belopp i amortering varje år och 3,4 % i ränta per år på den kvarvarande skulden. Amorteringen är alltså konstant, medan räntekostnaderna sjunker varje år i takt med att skulden minskar. Undersök med programmering hur mycket det skiljer i *total* kostnad om Celine väljer rak amortering i stället för annuitetslån och lånet avbetalas på 10 år.

Syfte och centralt innehåll

I den här aktiviteten får eleverna använda programmering för att beräkna storleken av annuiteten vid annuitetslån. De får också utnyttja programmering för att jämföra annuitetslån med rak amortering. Aktiviteten är lämplig att genomföra i samband med avsnittet *Olika typer av lån* i Matematik Origo 1c, kapitlet *Geometrisk summa* i Matematik Origo 3b eller i kapitlet *Talteori* i Matematik Origo 5.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras här nedanför är gjord i Python 3.

Lämpliga förkunskaper i programmering

Det är en fördel om eleverna är bekanta med kommandona `print` och `input()` samt känner till hur man skriver while-satser.

Genomförande

Hur aktiviteten ska genomföras beror på elevernas förkunskaper i programmering. Här presenteras ett förslag på hur aktiviteten kan genomföras för elever som inte har så stor erfarenhet av programmering.

- Om eleverna inte är bekanta med annuitetslån kan man behöva gå igenom detta gemensamt. Till den första uppgiften finns en bild som kan vara ett stöd. I uppgift 4 får eleverna jämföra rak amortering med annuitetslån. Förklara för eleverna vad rak amortering innebär.
- Låt eleverna arbeta med aktiviteten två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- I uppgift 4 kan man vid behov gemensamt gå igenom hur man använder while-satser för att upprepa kod tills ett visst villkor är uppfyllt.

Lösning

1 27 516, 30 kr

2
```
print(230000 * 1.034**10 * (1.034 - 1)/
(1.034**10 - 1))
```

3 T.ex.
```
belopp = float(input("Ange beloppet du
vill låna (i kr):"))
ränta = float(input("Ange räntesatsen
i %:"))
år = int(input("Ange på hur många år
lånet ska avbetalas:"))
förändringsfaktor = ränta/100 + 1
print(belopp * förändringsfaktor**år *
(förändringsfaktor - 1)/(förändrings-
faktor**år - 1))
```

4 Vi skriver ett program som räknar ut den totala kostnaden vid rak amortering:
```
belopp = 230000
amortering = 23000
årskostnader = []
while belopp > 0:
  ränta = belopp * 0.034
  årskostnader.append(amortering +
  ränta)      #Vi sparar årskostnaden i en lista som
              vi kallar årskostnader

  print(amortering + ränta)  #Vi skriver ut
                             varje årsbelopp

  belopp = belopp - amortering
  #Beloppet minskar i takt med amorteringen

print(sum(årskostnader))  #Vi skriver ut
                          summan av talen i
                          listan årskostnader
```

Vi kan också skriva programmet utan att använda listor:
```
belopp = 230000
amortering = 23000
total_kostnad = 0
while belopp > 0:
  ränta = belopp * 0.034
  belopp = belopp - amortering
  total_kostnad = total_kostnad +
  amortering + ränta
print(total_kostnad)
```

Programmet ger den totala kostnaden 273 010 kr. Den totala kostnaden vid annuitetslån är $10 \cdot 27\,516{,}30 = 275\,163$ kr. Skillnaden är alltså 2 153 kr.

Svar: Det är 2 153 kr dyrare att välja annuitetslån. Fördelen med annuitetslån är dock att man betalar samma belopp varje år. Rak amortering innebär att man inledningsvis får betala en större årlig summa.

Att lyfta fram

Avsluta gärna aktiviteten med en gemensam diskussion om fördelar och nackdelar med rak amortering respektive annuitetslån. Rak amortering ger en lägre total kostnad för lånet, men innebär också inledningsvis en högre årskostnad. Annuitetslån är totalt sett dyrare, men ger en jämn kostnadsfördelning varje år under hela återbetalningstiden.

A Aktivitet
Summor och gränsvärden

I den här aktiviteten får du skriva program som beräknar summor. En summa sägs *konvergera* om den går mot ett bestämt tal när antalet termer i summan går mot oändligheten. Summan sägs *divergera* om den inte går mot något bestämt värde när antalet termer går mot oändligheten. En summa med ett oändligt antal termer kallas för en serie.

1 a) Undersök om serien här nedanför divergerar eller konvergerar.

$$\sum_{n=1}^{\infty} \frac{1}{2^n}$$

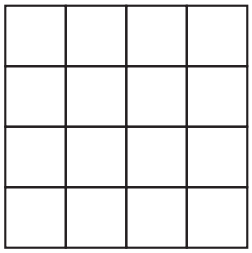
Du kan göra undersökningen så här:

- Skriv ut de fem första termerna i summan.
- Skriv ett program som kan beräkna summan för ett bestämt antal termer.
- Beräkna med hjälp av programmet:

$$\sum_{n=1}^{5} \frac{1}{2^n} \qquad \sum_{n=1}^{10} \frac{1}{2^n} \qquad \sum_{n=1}^{50} \frac{1}{2^n} \qquad \sum_{n=1}^{100} \frac{1}{2^n}$$

b) Konvergerar eller divergerar serien? Om den konvergerar, ange det värde som den konvergerar mot.

c) Försök att komma på ett sätt att matematiskt argumentera för att slutsatsen du dragit är sann. Figuren här nedanför kan vara till hjälp.



2 a) Undersök om serien här nedanför divergerar eller konvergerar.

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

Du kan göra undersökningen så här:

- Skriv ut de fem första termerna i summan.
- Skriv ett program som kan beräkna summan för ett bestämt antal termer.
- Beräkna med hjälp av programmet:

$$\sum_{n=1}^{50} \frac{1}{n} \qquad \sum_{n=1}^{100} \frac{1}{n} \qquad \sum_{n=1}^{10^4} \frac{1}{n} \qquad \sum_{n=1}^{10^6} \frac{1}{n}$$

b) Konvergerar eller divergerar serien? Undersök summan med fler termer om du behöver. Om den konvergerar, ange det värde som den konvergerar mot.

c) Försök att komma på ett sätt att matematiskt argumentera för att slutsatsen du dragit är sann. Kan du jämföra din summa med en annan summa som du känner bättre till?

Syfte och centralt innehåll

Den här aktiviteten syftar till att fördjupa elevernas kunskaper om bevisföring genom att de själva får undersöka om en serie konvergerar eller divergerar. Ett angreppssätt är att beräkna delsummor med succesivt fler termer och se vad som händer med delsummornas värde. Även om man kan bli övertygad av en empirisk undersökning är det inte ett stringent matematiskt bevis, och den förklarar heller inte *varför* serien divergerar eller konvergerar. Behovet av att besvara dessa frågor med ett logiskt resonemang uppstår förhoppningsvis spontant.

Aktiviteten kan användas som en introduktion till kapitlet *Matematisk bevisföring* i Matematik Origo 4. Aktiviteten kan också användas i kapitlet *Geometrisk summa* i Matematik Origo 3b.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Förslag på kod som presenteras nedan är gjord i Python 3.

Lämpliga förkunskaper i programmering

Eleverna bör känna till hur man kan upprepa kod med hjälp av slingor, särskilt for-satser.

Genomförande

Hur aktiviteten ska genomföras beror på elevernas förkunskaper i programmering. Här presenteras ett förslag på hur aktiviteten kan genomföras för elever som inte har någon större erfarenhet av programmering.

- Dela upp eleverna så att de arbetar två och två med en dator. Uppmana dem att låta den som känner sig minst van vid datorer och programmering sköta datorn. Är båda lika vana kan de turas om.
- Repetera vid behov hur man tolkar summor skrivna med summatecken, Σ, och hur man skriver for-satser. Det senare kan göras genom att tillsammans skapa ett program som skriver ut de 100 första positiva heltalen.

```
for n in range (1, 101):
 print(n)
```

- Begreppen divergens och konvergens förklaras kortfattat i den inledande uppgiftstexten. En fördjupad diskussion om begreppen kan med fördel sparas till en avslutande gemensam diskussion.
- Efter att eleverna har fått fundera en stund på deluppgift 2c kan de behöva en ledtråd. Tipsa dem om att jämföra den befintliga serien med serien:

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{4}+\frac{1}{4}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{16}+\ldots$$

Lösningar

1 a) • $\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\frac{1}{32}$

•
```
n = 100
b = 0
for n in range(1, n + 1):
  a = 1/2**n
  b = b + a
  print(b)
```

• $\sum_{n=1}^{5}\frac{1}{2^n}\approx 0{,}96875$

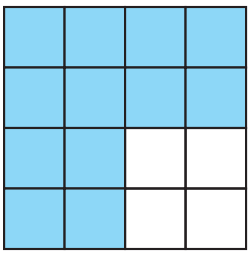
$\sum_{n=1}^{10}\frac{1}{2^n}\approx 0{,}9990234375$

$\sum_{n=1}^{50}\frac{1}{2^n}\approx 0{,}9999999999999991$

$\sum_{n=1}^{100}\frac{1}{2^n}\approx 1{,}0$

b) Från undersökningen här ovanför kan vi dra slutsatsen att summan verkar konvergera mot 1.

c) Genom att börja med att färglägga hälften $\left(\text{representerar talet } \frac{1}{2}\right)$ av figuren här nedanför och sedan successivt fortsätta med att i varje steg färglägga hälften av det som återstår $\left(\frac{1}{4}, \frac{1}{8}, \text{osv.}\right)$, inser vi att vi kan komma godtyckligt nära att hela figuren är färglagd. Det motsvarar att summan kan komma godtyckligt nära, men aldrig överstiga, 1.



2 a) • $\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}$

•
```
n = 100
b = 0
for n in range(1, n + 1):
  a = 1/n
  b = b + a
  print(b)
```

• $\sum_{n=1}^{50}\frac{1}{n} \approx 4{,}499205338329423$

$\sum_{n=1}^{100}\frac{1}{n} \approx 5{,}187377517639621$

$\sum_{n=1}^{10^4}\frac{1}{n} \approx 9{,}787606036044348$

$\sum_{n=1}^{10^6}\frac{1}{n} \approx 14{,}392726722864989$

b) Från undersökningen i punkt 3 kan vi dra slutsatsen att summan verkar divergera.

c) Genom att skriva ut termerna i summan och jämföra dem med något som vi vet är mindre och ändå divergerar, så kan vi vara säkra på att summan divergerar.

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+\frac{1}{8}+\frac{1}{9}+\frac{1}{10}+\ldots$$
$$>\frac{1}{1}+\frac{1}{2}+\frac{1}{4}+\frac{1}{4}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{16}+$$
$$+\frac{1}{16}+\ldots=1+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}+\ldots$$

Att lyfta fram

När man kör programmet i uppgift 1, med tillräckligt många termer, skriver programmet till slut ut resultatet 1. Det kan vara värt att särskilt lyfta fram att summan, för ett ändligt antal termer, alltid är *mindre* än 1. Att summan blir 1 vid beräkningarna beror på programmet bara kan spara och skriva ut tal med begränsad noggrannhet.

Aktiviteten ger eleverna möjlighet att på egen hand upptäcka om en serie konvergerar eller divergerar. Även om programmering kan vara ett kraftfullt hjälpmedel för att empiriskt undersöka konvergens och divergens, så är det inte tillräckligt att genomföra ett antal beräkningar för att vara säker på att en serie konvergerar eller divergerar. Det kan därför vara värt att poängtera behovet av ett övertygande logiskt resonemang.

I uppgift 1c föreslår vi ett geometriskt resonemang för att övertyga sig om att serien verkligen konvergerar. Men om eleverna är bekanta med formeln för geometrisk summa kan de genomföra ett annat bevis.

Summan $\sum_{1}^{n}\frac{1}{2^n}$ kan skrivas

$$\frac{1}{2}\cdot\frac{1-\left(\frac{1}{2}\right)^n}{1-\frac{1}{2}}=1-\left(\frac{1}{2}\right)^n$$

som går mot 1 när $n \to \infty$.

Utvidgning och variation

Om man vill utvidga uppgiften kan man undersöka fler konvergerande och divergerande serier, t.ex.

$$1-1+1-1+1-1+1-1+\ldots$$

eller

$$1+\frac{1}{3}+\frac{1}{9}+\frac{1}{27}+\ldots$$

Den första serien divergerar eftersom delsummorna växlar mellan 0 och 1. Den andra serien konvergerar. Det kan eleverna bevisa genom att jämföra den med serien i uppgift 1 eller genom att använda formeln för geometrisk summa.

Komplexa tal och transformationer

I den här aktiviteten får du undersöka vad som händer med tal i det komplexa talplanet när man utför räkneoperationer på dem.

Precis som man kan arbeta med funktioner där definitionsmängden är reella tal, kan funktioner vara definierade över de komplexa talen.

1 Låt $f(z) = z + (1 + i)$. Beräkna $f(z)$ utan digitala hjälpmedel om

a) $z = 2 + 3i$

b) $z = 1 - i$

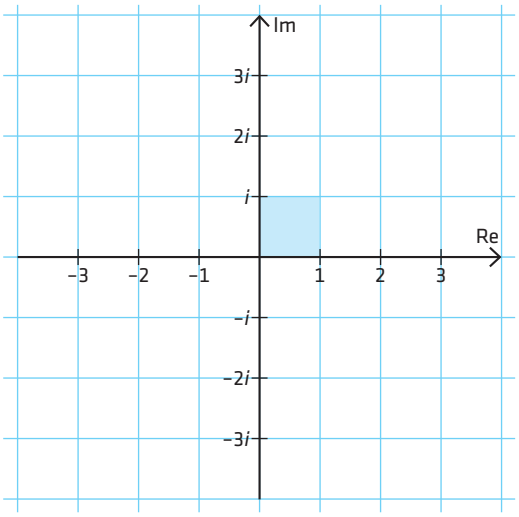
c) $z = 4$

Om man markerar både talet z och talet $f(z)$ i det komplexa talplanet, ser man att funktionen f förflyttar talet z till talet $f(z)$. Därför kallas funktioner som f ibland för *transformationer*.

2 a) Beskriv med ord hur transformationen $f(z) = z + (1 + i)$ förflyttar talet z i det komplexa talplanet.

Tips! Ta hjälp av dina beräkningar i uppgift 1.

b) Tänk dig att vi för varje tal z innanför enhetskvadraten i det komplexa talplanet här nedanför, markerar talet $f(z) = z + (1 + i)$ i det komplexa talplanet. Vad blir resultatet?



Programmet här nedanför slumpar fram 1 000 punkter i enhetskvadraten i det komplexa talplanet på föregående sida och markerar dem i rött. För var och en av de framslumpade punkterna beräknar programmet också $f(z) = z + (1 + i)$ och markerar den punkten i blått.

```
import matplotlib as mpl
mpl.use("Agg")
import matplotlib.pyplot as plt
from math import *
from random import *
X = []
Y = []
X_f = []
Y_f = []
w = complex(1,1)          #w är det komplexa talet med realdel 1 och imaginärdel 1, dvs. w = 1 + i
for n in range (1, 1001):
  z = complex(uniform(0,1), uniform(0,1))  #Programmet slumpar fram 1 000 punkter
                                           vars realdel och imaginärdel ligger i intervallet
                                           0 ≤ Re, Im ≤ 1.
  X.append(z.real)
  Y.append(z.imag)          #Vi sparar realdelen och imaginärdelen av z som koordinater (x, y)
  X_f.append((z + w).real)
  Y_f.append((z + w).imag)  #Vi sparar realdelen och imaginärdelen av f(z) = z + w
                            som koordinater (x, y)
plt.plot(X,Y,"or")
plt.plot(X_f, Y_f, "ob")    #Vi markerar talen z och f(z) = z + w i det komplexa talplanet.
plt.savefig("graph.png")
```

3 Kör programmet och kontrollera att resultatet stämmer överens med ditt svar i uppgift 2.

För att köra koden måste modulen `matplotlib` vara installerad.

Undersök följande transformationer på samma sätt som i uppgift 1–3 genom att ändra i koden på lämpligt sätt. Fundera på hur resultatet kommer att se ut innan du kör programmet.

4 a) $f(z) = z + 1$

b) $f(z) = -z$

c) $f(z) = 2z$

5 a) $f(z) = zi$

b) $f(z) = zi^2$

c) $f(z) = zi^3$

6 a) $f(z) = wz$ där $w = 1 + i\sqrt{3}$

b) $f(z) = \frac{z}{w}$ där $w = 1 + i\sqrt{3}$

7 a) $f(z) = z^2$

b) $f(z) = z^3$

c) $f(z) = \frac{1}{z}$

Syfte och centralt innehåll

I kurs 4 får eleverna lära sig vad som händer geometriskt när man adderar, subtraherar, multiplicerar och dividerar två komplexa tal. I den här aktiviteten får eleverna undersöka sådana transformationer i det komplexa talplanet med hjälp av programmering. Aktiviteten är lämplig att använda i samband med kapitlet om *Komplexa tal* i Matematik Origo 4.

Materiel

Dator med lämplig kompilator (t.ex. IDLE). Aktiviteten kräver att modulen `matplotlib` är installerad. Förslag på kod som presenteras är gjord i Python 3.

Lämpliga förkunskaper i programmering

I aktiviteten får eleverna modifiera ett givet program. För att kunna förstå och analysera programmet är det bra om eleverna känner till hur man importerar modulen `random`, hur man sparar tal i listor och hur man skriver slingor med hjälp av kommandot `for`.

Genomförande

Ett sätt att arbeta med aktiviteten är att lösa de tre första uppgifterna gemensamt i helklass. Det ger möjlighet att presentera begreppet *transformation* och att förklara varje steg i programmet som vi presenterar. Här följer en möjlig arbetsgång:

- Låt eleverna beräkna funktionsvärdena i uppgift 1 och därefter komma fram till tavlan och markera talen z samt motsvarande funktionsvärden $f(z)$ i det komplexa talplanet (uppgift 1).
- Diskutera hur funktionen f kan sägas förflytta talet z till $f(z)$ i det komplexa talplanet. Förklara att funktionen f därför ibland kallas för en *transformation* och låt eleverna beskriva transformationen med ord (uppgift 2a).
- Diskutera vad som händer om vi för varje punkt innanför enhetskvadraten i det komplexa talplanet, markerar talet $f(z)$ i det komplexa talplanet. Vad blir resultatet? (uppgift 2b)
- Presentera koden på elevstencilen och gå igenom vad varje del av programmet gör. Kör koden och jämför resultatet med slutsatsen från diskussionerna i uppgift 2 (uppgift 3). Här ger vi några ytterligare kommentarer till programmet på elevstencilen:

```
import matplotlib as mpl #Vi importerar modulen matplotlib. För att kunna köra programmet måste modulen stödjas av kompilatorn man använder.
mpl.use("Agg")
import matplotlib.pyplot as plt
from math import *       #Vi importerar modulen math för att t.ex. kunna skriva komplexa tal
from random import *     #Vi importerar modulen random för att kunna slumpa fram tal
X = []
Y = []   #Vi skapar tomma listor där vi kommer att spara koordinaterna för talen z och f(z)
X_f = []
Y_f = []
w = complex(1,1) #w är det komplexa talet med realdel 1 och imaginärdel 1, dvs. w = 1 + i
for n in range (1, 1001):
  z = complex(uniform(0,1), uniform(0,1))
  #Programmet slumpar fram 1 000 komplexa tal vars realdel och imaginärdel ligger i intervallet 0 ≤ Re, Im ≤ 1.
  X.append(z.real)
  Y.append(z.imag) #Vi sparar realdelen och imaginärdelen av z som koordinater (x, y)
  X_f.append((z + w).real)
  Y_f.append((z + w).imag) #Vi sparar realdelen och imaginärdelen av f(z) = z + w som koordinater (x, y)
plt.plot(X,Y,"or")
plt.plot(X_f, Y_f, "ob")
#Vi markerar talen z och f(z) = z + w i det komplexa talplanet. Kommandot "or" anger att punkten är rund och röd (red) medan kommandot "ob" anger att punkten är rund och blå (blue).
plt.savefig("graph.png")
```

- Låt sedan eleverna jobba med resten av aktiviteten i par. Samla slutligen ihop klassen till en gemensam diskussion (se mer under rubriken *Att lyfta fram*).

Lösning

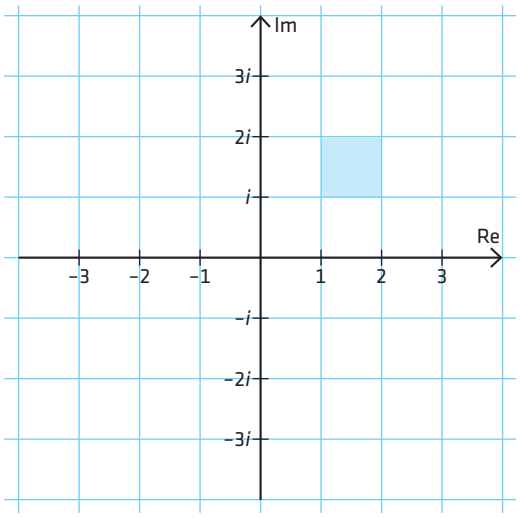
1 a) $z = 3 + 4i$

b) $z = 2$

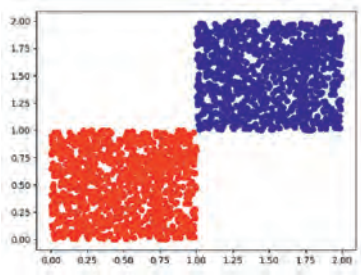
c) $z = 5 + i$

2 a) Transformationen f förflyttar talet z ett steg till höger och ett steg uppåt i det komplexa talplanet.

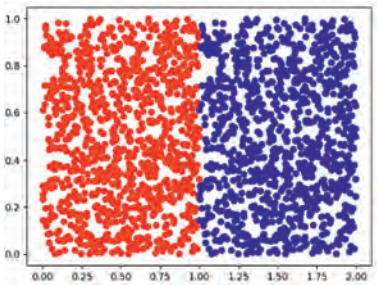
b)



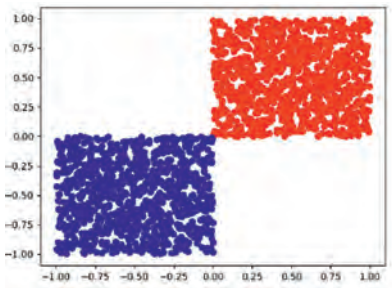
3



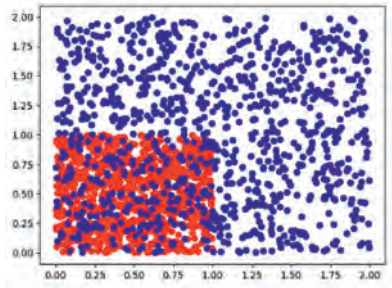
4 a) T.ex. ändra `w = complex (1, 1)` i koden på elevstencilen till `w = complex(1, 0)`. Kvadraten förflyttas ett steg till höger.



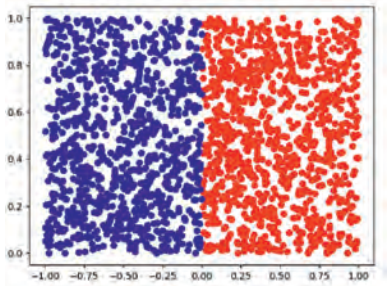
b) T.ex. ändra `z + w` i koden på elevstencilen till `-z`. Kvadraten roteras 180° kring origo.



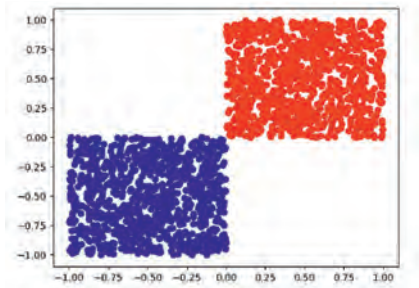
c) T.ex. ändra `z + w` i koden på elevstencilen till `2 * z`. Kvadratens sidlängd blir 2.



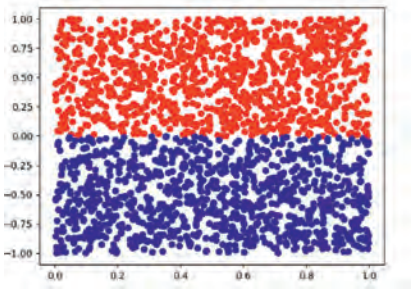
5 a) T.ex. ändra `w = complex (1, 1)` i koden på elevstencilen till `w = complex(0, 1)` och `z + w` i koden på elevstencilen till `z * w`. Kvadraten roteras 90° i positiv riktning kring origo.



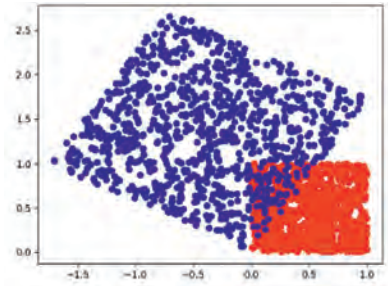
b) T.ex. ändra `w = complex (1, 1)` i koden på elevstencilen till `w = complex(0, 1)` och `z + w` i koden på elevstencilen till `z * w**2`. Kvadraten roteras 180° kring origo.



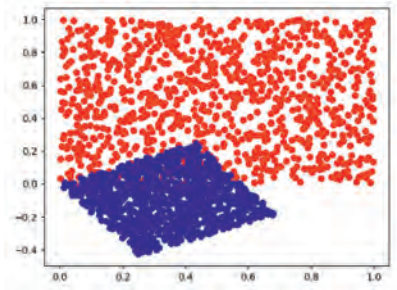
c) T.ex. ändra `w = complex (1, 1)` i koden på elevstencilen till `w = complex(0, 1)` och `z + w` i koden på elevstencilen till `z * w**3`. Kvadraten roteras 270° i positiv riktning kring origo.



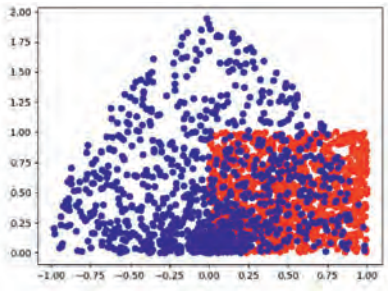
6 a) T.ex. ändra `w = complex (1, 1)` i koden på elevstencilen till `w = complex(1, sqrt(3))` och `z + w` i koden på elevstencilen till `z * w`. Om vi utnyttjar reglerna för multiplikation med komplexa tal i polär form, förstår vi varför den blå figuren är roterad 60° i positiv riktning kring origo och har sidlängden 2.



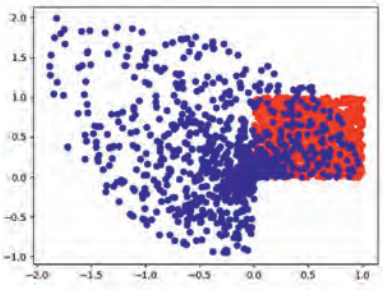
b) T.ex. ändra `w = complex (1, 1)` i koden på elevstencilen till `w = complex(1, sqrt(3))` och `z + w` i koden på elevstencilen till `z/w`. Om vi utnyttjar reglerna för division av komplexa tal i polär form, förstår vi varför den blå figuren är roterad 60° i negativ riktning kring origo och har sidlängden 0,5.



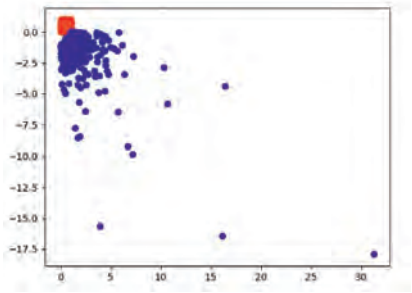
7 a) T.ex. `z + w` i koden på elevstencilen till `z**2`. Enligt de Moivres formel kommer argumentet för varje komplext tal i enhetskvadraten att fördubblas, vilket gör att funktionsvärdena får argument mellan 0° och 180°. Eftersom absolutbeloppet kvadreras, kommer tal med ett absolutbelopp mindre än 1 att komma närmare origo, medan tal med ett absolutbelopp större än 1 kommer att komma längre bort från origo.



b) T.ex. ändra `z + w` i koden på elevstencilen till `z**3`. Enligt de Moivres formel kommer argumentet för varje tal i enhetskvadraten att tredubblas, vilket gör att funktionvärdena får argument mellan 0° och 270°. Eftersom absolutbeloppet i sin tur upphöjs till tre, kommer tal med ett absolutbelopp mindre än 1 att komma närmare origo, medan tal med ett absolutbelopp större än 1 kommer att komma längre bort från origo.



c) T.ex. ändra `z + w` i koden på elevstencilen till `1/z`. Utnyttjar vi reglerna för division av komplexa tal i polär form, förstår vi att argumentet för varje tal i enhetskvadraten byter tecken, vilket gör att funktionvärdena får argument mellan 0° och –90°. Eftersom absolutbeloppet i sin tur inverteras, kommer tal med ett absolutbelopp större än 1 att komma närmare origo, medan tal med ett absolutbelopp mindre än 1 kommer att komma längre bort från origo.



Att lyfta fram

Det kan vara bra att uppmärksamma eleverna på att koordinataxlarna inte är synliga i den figur som programmet genererar och att skalan i koordinatsystemet kan ändras när man undersöker en ny transformation. För att tolka figuren rätt är det därför viktigt att läsa av skalan på axlarna.

Uppgift 5–7 ger möjlighet att lyfta fram vad som händer geometriskt när man dividerar eller multiplicerar två komplexa tal. Uppgift 5 fokuserar särskilt på multiplikation med talet i, som motsvarar en rotation med 90°, medan uppgift 6 och 7 ger eleverna möjlighet att undersöka mer allmänna transformationer. Genom att analysera vad som händer med argument och absolutbelopp kan eleverna göra förutsägelser om hur mängden av funktionsvärden $f(z)$ kommer att placeras i det komplexa talplanet.

Någon elev uppmärksammar kanske att transformationerna i uppgift 4b och 5b ger samma resultat. Det ger möjlighet att poängtera att multiplikation med i^2 är detsamma som multiplikation med –1 och att dessa operationer motsvarar en rotation med 180° kring origo.

Utvidgning och variation

Om man vill utvidga aktiviteten, kan man låta eleverna komma med förslag på egna transformationer att undersöka och förklara. Vad händer t.ex. i uppgift 6 om man varierar talet w?

Uppgift 7 kan utvidgas med frågeställningar om avstånd till origo. Hur långt bort är den punkt $f(z)$ som är närmast/längst bort från origo? I deluppgift 7a och 7b kan dessa avstånd bestämmas, medan det i uppgift 7c kan finnas punkter vars avstånd till origo kan vara hur stort som helst.