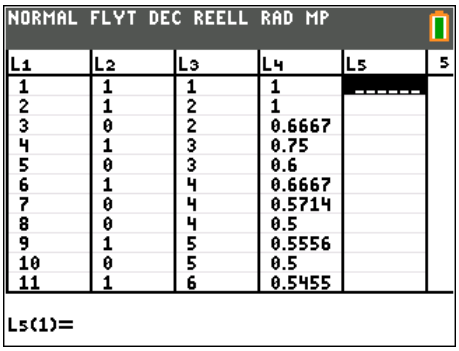
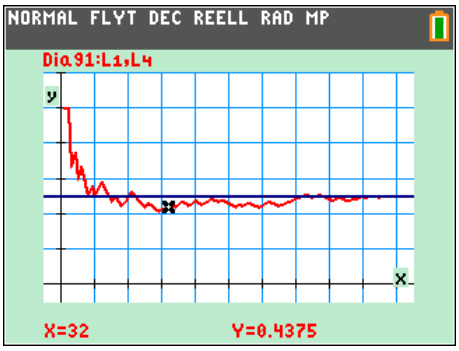
Slantsingling och relativ frekvens

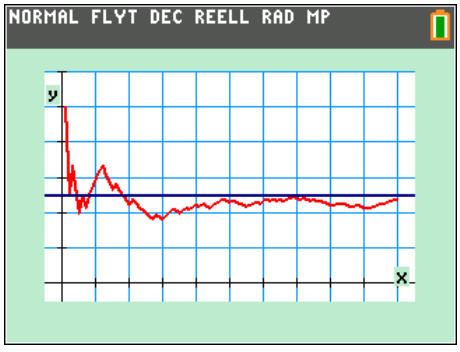
I denna aktivitet ska vi bl.a. utnyttja grafräknarens och Pythons outtröttliga förmåga att singla slant, dvs. vi ska använda slumptalsgeneratorn i räknaren. Vi människor och programmerade datorer har minne och vet att det t.ex. ska bli ungefär lika många klave som krona om vi göra ett stort antal kast. Det vet inte myntet när det hänger i luften. I ett äkta slumptals-försök finns det ingenting som heter minne. Allt börjar om igen när vi kastar myntet nästa gång.

Vi ska först använda grafräknarens statistikeditor för att alstra slumptal och sedan göra beräkningar på dessa slumptal. Här gör vi nu inte noggranna instruktioner hur det går till utan hänvisar till aktiviteten **”Stora talens lag eller det jämnar ut sig”.**

- Först alstrar vi i lista L1 en serie, talen 1 till 100 och lägger dessa i lista L1. Att skriva in denna tal-serie för hand är jobbigt men det finns verktyg på räknaren för detta
- Sedan alstrar vi i L2 en slumptalsserie med tal 0 eller 1. Grafräknaren har en speciell funktion för att alstra heltaliga slumptal. Arbetar du med svenska på räknaren heter den ”SlumpHel” och kör du på engelska heter den ”randInt”.
- I det tredje steget i L3 beräknar vi nu de *kumulerade* fre-kvenserna för slumptalen. Räknaren har en funktion ”kumSum” för detta.
- Till sist alstrar vi en lista i L4 som rad för rad är kvoten av data i L3 och L1. Vi får då den relativa frekvensen beräknad rad för rad. Om vi går till editorn kan det se ut som på skärmbilden till höger:
- Nu kan vi plotta data från lista L1 och L4. Med en bra inställning av skalan på x- och y-axeln kan det se ut som på skärmen till höger. Vi ser hur den hackiga kurvan närmar sig relativa frekvensen 0,5, vilket visas av den vågräta blå linjen.
- Gör vi en ny serie kan det se ut som det nedre diagrammet.







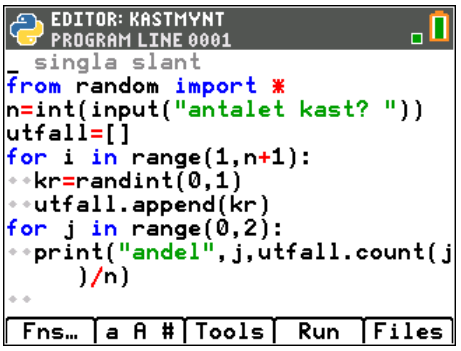
Vad kan vi dra för slutsatser av dessa försök. Jo, det verkar som den relativa frekvensen stabiliseras omkring det förväntade värdet när vi tittar på resultatet efter många försök. Dock kan vi få svängningar i resultatet så att den relativa frekvensen efter många kast plötsligt förändras från ”nära det förväntade värdet” till ”lite längre bort från det förväntade värdet”. Man kan t.ex. få 6 krona eller klave i rad.

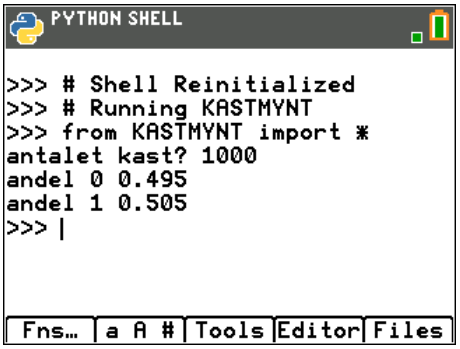
En intressant artikel om de stora talens lag.

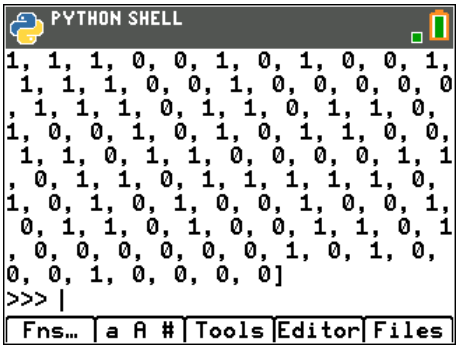
Här har vi ett kort Pythonprogram som slumpar fram ett antal nollor eller ettor (står för krona/klave vid slantsingling).

- En input-sats i början gör att man vid körning får en fråga om antalet gånger som det ska slumpas fram ett tal 0 eller 1.
- Satsen ”for i in .. ” gör att programmet upprepar koden som kommer efter *n* antal gånger.
- Från början har vi en tom lista som vi döper till utfall. När vi sedan slumpar fram tal så fylls de sedan på i listan Utfall med satsen utfall.append(kr)
- Den sista loopen gör att vi får med vart och ett av de möjliga utfallen. I printsatsen skrivs sedan andelen nollor och ettor ut (eller krona klave om man vill). Instruktionen .count returnerar *antal förekomster* av ett objekt, i detta fall antal nollor och ettor, i listan utfall.

I bilden till höger ser vi resultatet av en körning med 1000 slumptal. Kör några gånger och se hur andelarna varierar. Om man trycker på tangenten [vars] och sedan på variabeln utfall så kan du se listan med nollor och ettor bläddra förbi på skärmen.

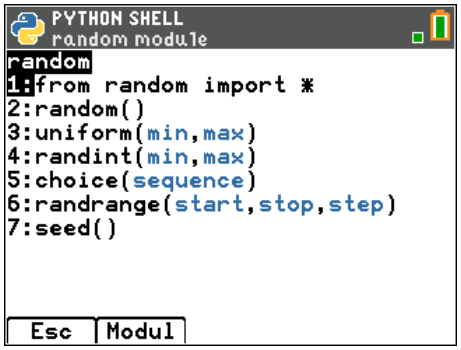






Gå igenom med eleverna de olika funktionerna som finns under modulen Random. *choice* returnerar slumpmässigt ett tal från en lista. *randrange* returnerar ett slumpmässigt tal från start till stopp med ett visst steg. randrange(10,100,10) kan t ex ge resultatet 70.

Seed är när man initialiserar slumptalsgeneratorn. Man kan läsa mer om detta i aktiviteten *Koda med TI, kapitel 2 övning 3*. Oftast behöver man inte tänka på detta eftersom det vid slumptalsalstring hela tiden alstras nya slumptalskärnor.



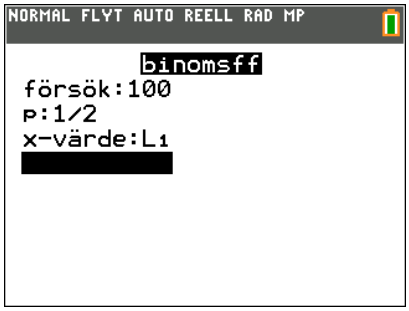
Pröva nu att ändra i programmet ovan så att du istället alstrar *summan* av två slantsinglingar. Fundera vilka utfall som är möjliga innan du börjar med kompletteringarna i programmet. Du kan till exempel först köra två serier av slantsinglingar i statistikeditorn. Du kan också öka antalet simuleringar till 200.

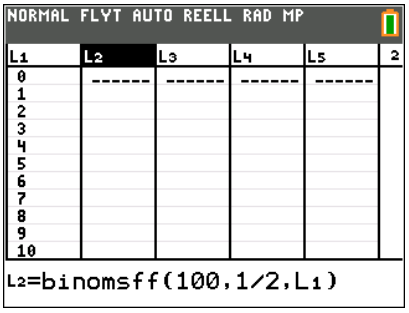
Man kan också tänka sig att du kastar tärning och beräknar den relativa frekvenserna för utfallen 1 till 6.

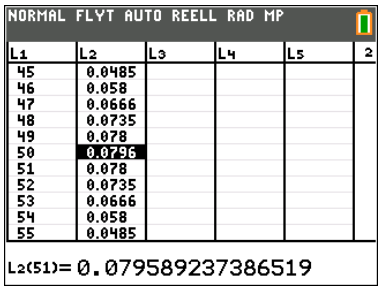
Fördjupning: Hur stor är den teoretiska sannolikheten att vi får exakt 50 krona när vi kastar ett juste mynt 100 gånger.

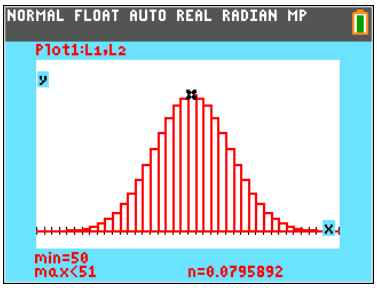
Det är ju ett exempel på en *binomialfördelning*: vi upprepar ett försök, som har en viss sannolikhet *p* att lyckas, *n* gånger. Hur många gånger lyckas vi? Det här ligger ju lite utanför kursen men väldigt många händelser inom sannolikhetsläran kan beskrivas med en binomial-fördelning.

Hos räknaren finns nu inbyggd funktion för att beräkna dessa sannolikheter. Den heter binomsff( ). Vi kastar alltså ett mynt 100 gånger. Hur stor är då den teoretiska sannolikheten att vi får 0, 1, 2 … 99, 100 krona. Vi har placerat talen 0 till 100 i lista L1.









Det visar sig att sannolikheten att få exakt 50 krona är ca 8 %.
Se diagrammet. X-skalan i figuren är xmin=30 och xmax=70.