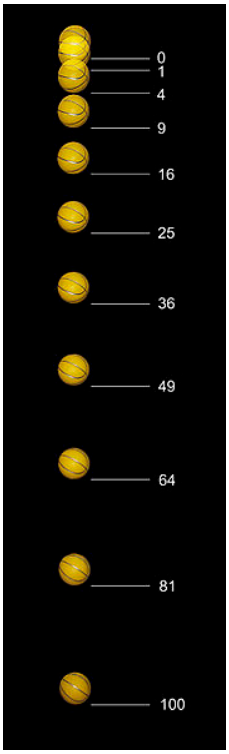
Kapitel 5: Använda biblioteksmodulen ti-plotlib — Tillämpning: Visning och timeout

I denna avslutande övning i kapitel 5 ska du fördjupa begreppen som tagits upp i kapitel 4 och 5 för att skapa en simulator som beskriver fysikalisk rörelse.

Syfte:
- Göra en simulering utifrån stroboskopbilder
- Exportera resultat från simuleringen till listor hos räknaren

Här ska Pythonspråket användas för att skapa en simulator av ett fysiskt fenomen. Det handlar om fritt fall.

- Skriptet bör tillåta beskrivning av en rörelse;
- Den grafiska representationen av data ska i grafik visa själva rörelsen
- Data ska exportera data till kalkylatorlistorna.



Scenario: En boll tappas utan begynnelsehastighet från höjden *h*. Bilderna för rörelsen tas med 60 millisekunders intervall. Figuren visar positionen för en boll som är tagen med ett s.k. stroboskop. Ett **stroboskop** är ett instrument som sänder ut ljuspulser för att lysa upp ett föremål så man kan studera dess rörelse eller hastighet. Ljuspulserna har sänts ut med frekvensen 20 pulser per sekund. Siffrorna visar hur långt bollen har fallit. (*Källa*: Wikipedia)

Du ska nu skriva ett skript som beräknar positionen för bollen vid olika tidpunkter och sedan representera detta grafiskt.

a) Beräkna bollens läge vid olika tidpunkter

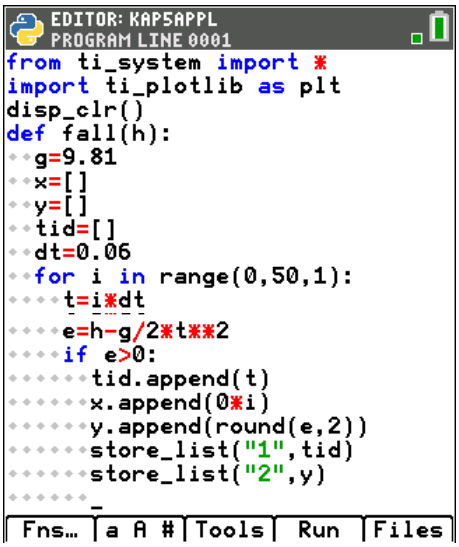
Bollens position *y* m beräknas med hjälp av uttrycket där h_0 är ursprunglig höjd i m och t är tiden efter nedsläpp:

$$y = h_0 - \frac{1}{2} \cdot g \cdot t^2 \qquad g = 9{,}81 \text{ m/s}^2.$$

Skapa ett nytt skript och döp det till KAP5APPL

- Importera **ti_system-** och **ti_plotlib-**modulerna**.**
- Rensa skärmen.
- Skapa en funktion **fall** där argumentet är den ursprungliga höjden varifrån bollen släpps och som visar positionen för bollen vid olika tider.
- Skapa tre tomma listor, **x, y** och **tid.**
- Höjden (y-värden) ska beräknas var 60:e millisekund (0,06 s).
- Värdena sparas i listorna om *y*>0 (bollen kan ju inte sjunka ner i marken).

Skapa en avgränsad loop för att beräkna höjden hos en boll som funktion av tiden. Resultaten ska uttryckas med en noggrannhet på 1/100 och lagras i sina respektive listor. Listorna tid och y exporteras sedan till räknarlistorna L_1 och L_2.



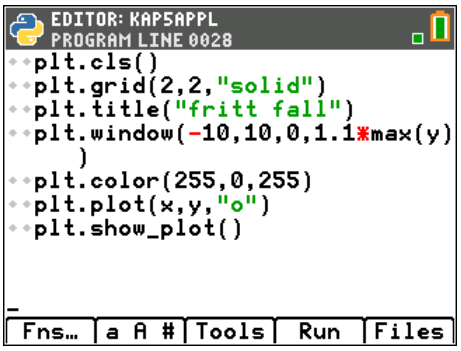
b) Visning av grafiken

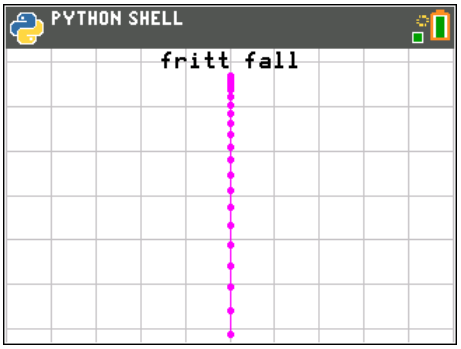
Inställningar för att plotta diagrammet

- Rensa skärmen med instruktionen **plt.cls().**
- Visa rutnät **plt.grid(xscl, yscl, type,(r,g,b))**.
- Gör fönsterinställningar **plt.window(xmin, xmax, ymin, ymax).**
- Ställ in färg för punkter till magenta **plt.color(255,0,255**).
- Visa spridningsdiagram (punkterna) **plt.plot(x-list, y-list, mark)**.
- Visa plottningen **plt.show_plot().**

Kör ditt script och anropa din funktion fall genom att trycka på [vars]. Ange sedan som argument ursprungshöjden, t ex. 8 m.

Diagrammet för rörelsen visas nu.

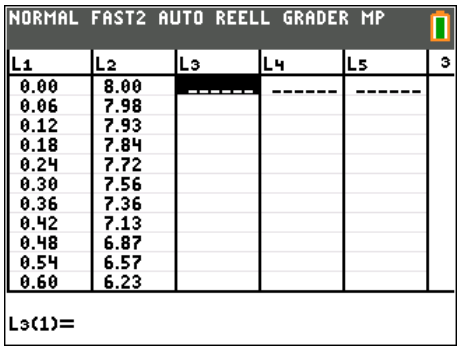


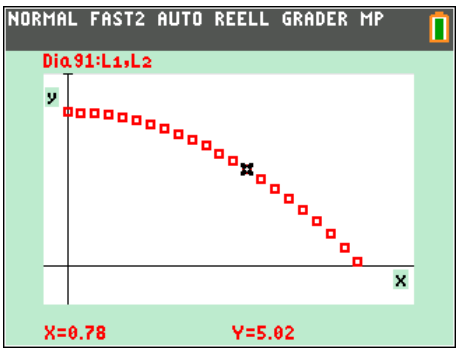


Kommentar: Från listan över bollens position kan vi beräkna hastigheten på bollen, sedan visa hastighetsvektorer också.

b) Visualisering av de exporterade listorna.

Avsluta nu Pythonappen och visa listorna i statistikeditorn hos räknaren. Plotta sedan ett diagram som visar hur höjden beror av tiden. Med [trace] kan man spåra i diagrammet.





Kommentar: Det finns ett smart sätt att inte bara visa hur fallsträckan beror på tiden utan man kan faktiskt med räknaren visa själva rörelsen utan att programmera. Vi gör en speciell inställning på räknaren. Tryck på [mode] och välj parametrisk på femte raden. Med denna inställning kan man skriva in formler som anger hur *x*- och *y*-koordinaten beror av variabeln T, tiden.

X1T sätts nu till 2. Det har bara att göra med läget på skärmen. Vi ska ju visa en fallrörelse och då sker den vertikalt längs *y*-axeln.

Y1T skriver du in enligt inmatningsrutan. Det är ju samma formel som förut egentligen men här får man ett T när man trycker på □-knappen.

En sak som du kanske inte uppmärksammat är att man kan ställa in hur linjer och kurvor ska ritas på skärmen. I inmatningsfältet på första raden, där det nu står 2, kommer du till olika ritningsalternativ om du flyttar dig längst ut till vänster med |. Genom att upprepade gånger trycka på ⊆ bläddrar du fram olika alternativ. Välj nu det alternativ som är en nyckel (🔑).

Sedan ställer man in ett bra fönster för att visa rörelsen.

Vi har på bilden här fryst rörelsen genom att trycka på [enter]. Om man trycker på [enter] igen fortsätter linjen med den lilla bollen sin framfart nedåt. Du märker kanske att det går fortare och fortare. Det är ju en accelererad rörelse vi visar.

Ändra nu visningen av rörelsen så man har en fet prickad linje istället

för nyckeln. Ändra också Tsteg till 0.06 som vi hade i Pythonappen.

Tryck nu på [trace] och bläddrar dig framåt 0,06 sekunder i taget. Se bilden.

Man kan även plotta lite mer komplicerade rörelser.

Du kastar en boll rakt upp luften med begynnelsehastigheten 12 m/s och fångar den på samma höjd när den kommer ner igen. Man ger samtidigt bollen en vindpust på 2 m/s rakt horisontellt.

Visa själva rörelsen!