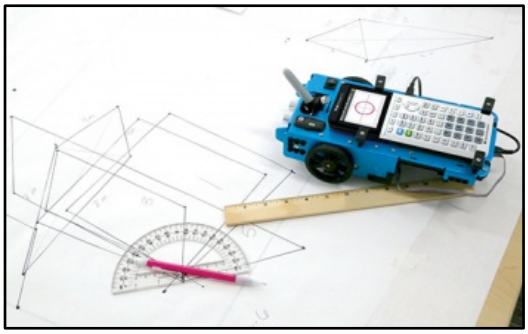
Kapitel 6: Använda Hub & Rover-biblioteket

Tillämpning: Representera en tur med Rover

I kapitel 6 ska du ansluta TI-Innovator™ Rover med **Ti_hub**-biblioteket och skapa ett skript som samlar in punkt-koordinaterna under en färd och sedan också visa dessa koordinater grafiskt.

Syfte:

- Undersöka **ti_rover**-modulen.
- Skriva och använda ett skript för att köra TI-Innovator Rover och dess tillhörande ställdon.
- Använda en sluten slinga.
- Representera data grafiskt

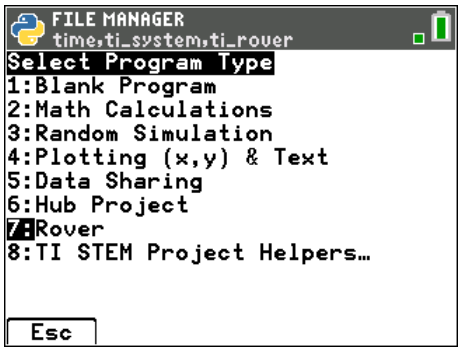


I den här lektionen kommer du att skapa ett skript som ger ti-rover möjlighet att göra en väg som motsvarar ritningen av en polygon.

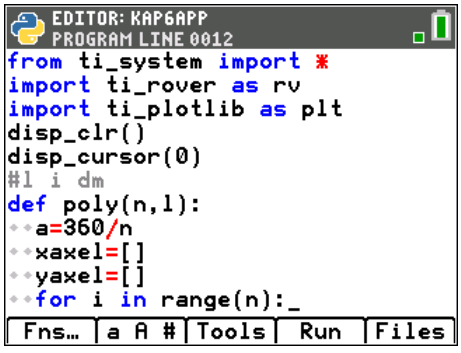
Hörnens koordinater sparas i listor och representeras sedan grafiskt med hjälp av instruktionerna i **ti_plot**-biblioteket.

Genomförande:

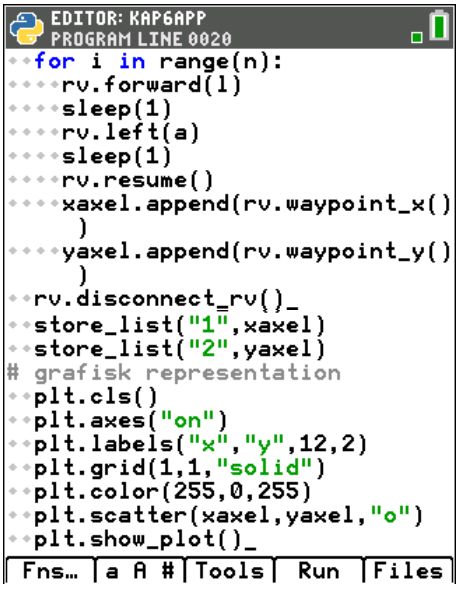
- Starta ett nytt skript och döp det till KAP6APP
- Importera TI-Rover-biblioteket från **modulmenyn**
- Bekräfta genom att trycka på enter.
- Importera även **ti_plotlib**-biblioteket



- Även om det inte är nödvändigt så gör instruktionerna för att rensa skärmen och inte visa markören att skärmen ser trevligare ut. Infoga alltså **disp_clr()** och **disp_cursor(0)** från modulmenyn och **ti_system** sedan.
- Skapa en funktion **poly()** med antalet sidor **n** i polygonen som argument och **l** som standardenheten längd för TI-Rover, dvs dm.
- Vinkeln vid hörnen för varje polygon kommer således att vara lika med a=360/n.
- Skapa två tomma listor **xaxel** och **yaxel**, som är avsedda att ta emot koordinaterna för var och en av dessa hörn.



- Skapa en stor öppen loop **n**.



Följande instruktioner finns i **ti_rover**-biblioteket:

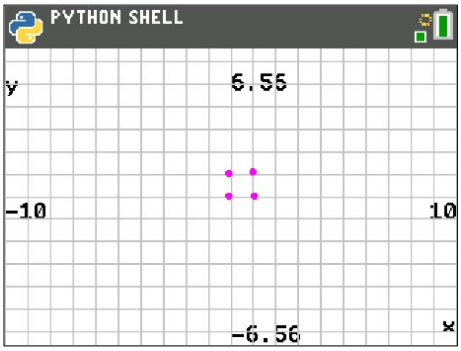
- Flytta fram Rover **l** decimeter *n* gånger
- Sväng vänster med vinkeln **a.**
- Lägg in en fördröjning på 1 s mellan varje steg.
- Lagra koordinaterna för punkterna i x- och y-listorna.
- Koppla bort Rover i slutet av banan.
- Exportera **xaxel-** och **yaxel**-koordinaterna till listorna L_1 respektive L_2 hos räknaren.

Kommentar: Instruktionerna **rv. waypoint_x()** och **rv. waypoint_y()** finns i I/O-menyn. Välj sedan **3: Path...**.

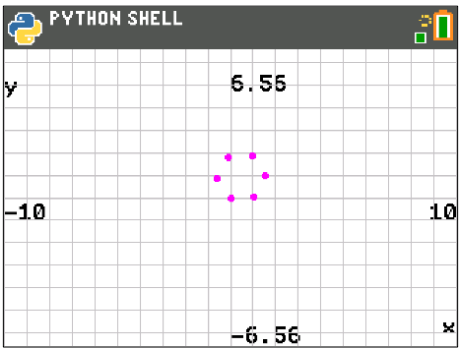
- Byt sedan till den grafiska representationen, som ingår i funktionen här men det är fortfarande möjligt att skapa en graffunktion separat som vi gjorde i andra övningar (kapitel 5).

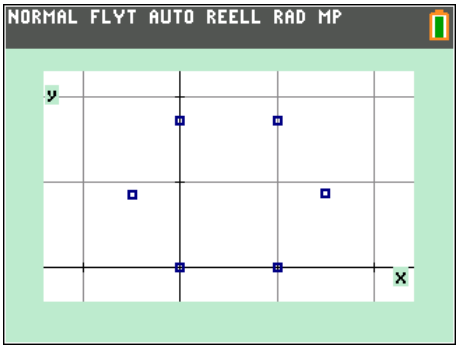
Hur skriptet fungerar:

Kör ditt **skript** och anropa **poly()**-funktionen. Skriv (4, 1) för att rita en kvadrat med sidan1 dm.



Testa också med en hexagon med sidan 1 dm





Om du exporterar listorna till räknarens statistikeditor kan du plotta den grafiska representation som visas här. Om du använder trace-tangenten så kan du visa koordinaterna för varje punkt.

Kommentar: Undvik att använda instruktionerna **rv.pathlist_x()** och **rv.pathlist_y()** i denna typ av övning.

Faktum är att när du ritar ett segment registrerar räknaren koordinaterna för punkterna i ett segment i polygonen och lagrar sedan koordinaterna för den sista punkten som den första punkten i nästa segment. Speciellt eftersom vi har placerat en fördröjning på 1s mellan varje plottning.

Så rv.path instruktionerna är i vårt fall inte lämpliga.

Med satserna **rv.pathlist_x()** och **rv.pathlist_y()**, får vi koordinaterna för ändpunkten två gånger.

Obs: Justera inställningarna på ditt rutnät, beroende på vilka polygoner du vill rita. Detta har avsiktligt ställts in till standardinställningarna här för att observera noggrannheten i ROVER-spåret.

Var också uppmärksam vilken typ av yta som roboten rör sig på. Den får inte erbjuda för mycket motstånd mot rörelse eller tvärtom upp-muntra till glidning.