## Kapitel 7 : micro:bit med Python

## Övning 2 : Knappar och rörelse

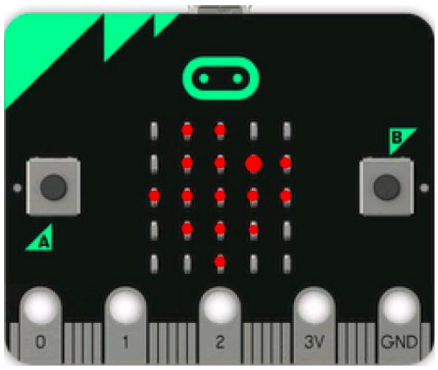
I den här lektionen lär du dig att använda micro:bits **knappar** och **rörelsesensor**. Vi ska skapa ett program som simulerar tärningskast och sedan spara resultatet i en lista som kan exporteras till, och visas med TI-Nspires dataplot.
Denna lektion består av två delar:
**Del 1**: Utforska knapparnas funktion och effekten av rörelser
**Del 2**: Använda rörelse och knappar för att generera data

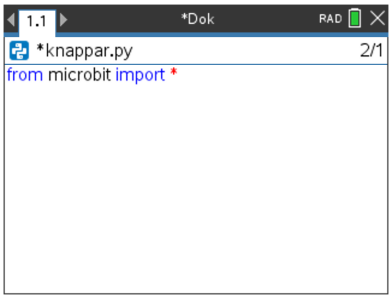
**MÅL :**

- Att läsa in om knapp A eller B trycks in.
- Veta skillnaden mellan .was och .is
- Att utbyta data mellan Python och TI-Nspire CX
- Att hantera data från micro:bit
- Att använda micro:bit-gester

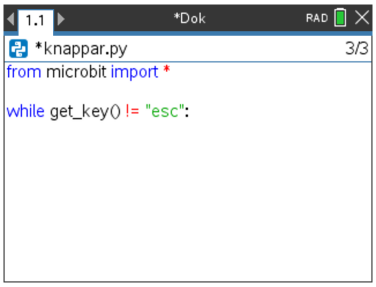
1. Micro:bit-enheten har två knappar, märkta A och B, på vardera sidan av displayen.
   Python micro:bit-modulen har metoder för att läsa knapparnas läge och sedan utföra uppgifter beroende på resultatet.
   Först ska vi testa dessa metoder och sedan så ska vi skriva ett Python-program som samlar in och analyserar data med TI-Nspire CXII-T.

   Det finns också en accelerometer/kompass-chip på baksidan av micro:bit-enheten och det finns metoder (kommandon) för att mäta enhetens rörelse och orientering med hjälp av dessa.



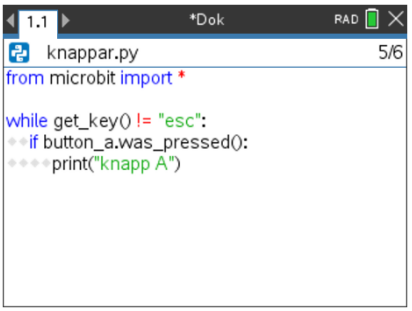

> **Lärartips**: På version 2 av micro:bit: finns en touch-knapp. Denna befinner sig ovanpå displayen och är guldpläterad.
> I micro:bit-modulen kan du nå knappen med **is_touched()**-kommandot under **[menu]>BBC micro:bit>Buttons and Logo Touch>pin_logo>' 'is_touched()**. I detta kapitel kommer den här knappen inte att användas men man kan naturligtvis alltid använda denna på samma sätt som knapparna A och B.




2. **Del 1: Använda knappar och rörelser**
   Starta ett nytt Python-program i ett nytt dokument.
   **(**Tryck på **[Home]-knappen**, välj **Nytt** och sedan **Lägg till Python> Nytt...).**
   Ge den ett namn till exempel "knappar".
   Börja igen med **från microbit import *** som du hittar under **[menu]> Fler moduler> BBC micro:bit.**

3. Vi börjar med att lägga till samma **while**-loop som tidigare med **[menu]> Fler moduler> BBC micro:bit> Commands**.

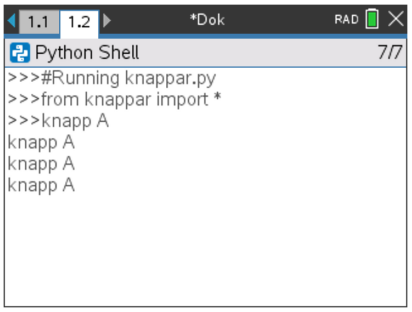
   Nästan alla exempel med micro:bit har denna struktur.




4. För att testa om knapp A har tryckts in använder vi **if** -strukturen:

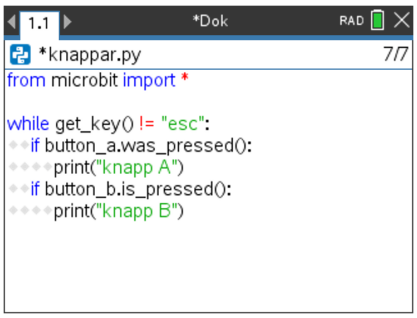
   **if button_a.was_pressed():**
   **◆ ◆ print("knapp A ")**

   If -satsen ska vara inne i while-loopen och vi måste därför lägga till rätt indrag. Utskriftskommandot är extra indraget eftersom det tillhör if -satsen.
   if-satsen finns under **[menu]> Inbyggda> Kontroller**
   och **button_a_was_pressed ()** finns under:
   **[menu]> Fler moduler> BBC micro: bit> Buttons and Logo Touch**.

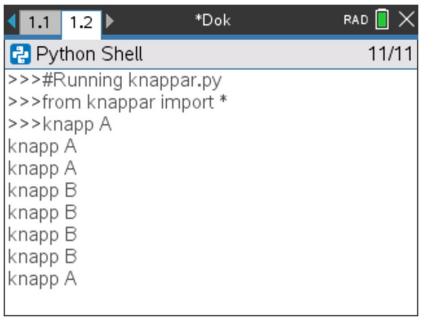



5. Vi ska nu testa om detta fungerar. Tryck på **[ctrl] [R]** för att köra programmet.
   Först ser det ut som inget händer, men varje gång du trycker på knapp A visas texten “knapp A” på TI-Nspires skärm.

   Tryck på **[esc]** för att stoppa programmet.




6. Lägg nu till en andra if-sats för knapp B men använd nu kommandot: if button_b.is_pressed ()
   Observera att det nu står ".is" istället för ".was".
   Återigen, var uppmärksam på att ha korrekta indragningarna.




7. Kör programmet igen och prova att trycka på knapparna A och B.

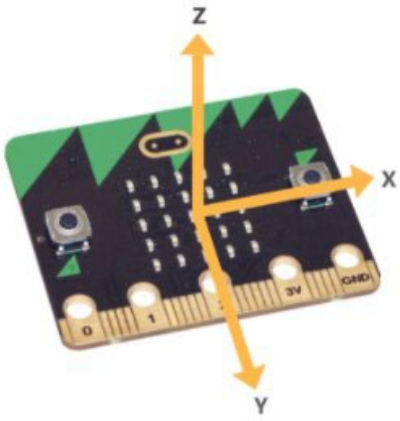
Skillnaden är att texten ”knapp A” inte skrivs ut förrän knappen släpps (was=var), medan ”knapp B” skrivs ut om och om igen så länge den hålls intryckt (is=är).

*Observera att om du trycker in och snabbt släpper knappen B, kan det hända att det inte sker något eftersom knappen inte trycktes in just då motsvarande if -sats kördes.*

**Lärartips**: De två funktionerna beter sig olika. Det beror på vad du vill göra, vilken av de två som passar bäst.
Metoden **.was_pressed()** kontrollerar om knappen har tryckts in tidigare i loopen.
Metoden **.is_pressed()** kontrollerar om knappen är intryckt just nu.

8. **Rörelsedetektering (gester).**
   Micro:bit-enheten har en inbyggd accelerometer som kan mäta accelerationer i tre olika riktningar 90 grader mot varandra (det är en 3D accelerometer).
   Förutom numeriska värden för accelerationen i varje riktning (se figuren) så har micro:bit också tydliga så kallade “gester” som "uppåt" och "med framsidan nedåt" baserat på dessa numeriska värden.

   Vi kommer att undersöka detta med ett program.




```
1.1 1.2  *Dok  RAD
knappar.py  9/9
from microbit import *

while get_key() != "esc":
    if button_a.was_pressed():
        print("knapp A")
    if button_b.is_pressed():
        print("knapp B")
    g=accelerometer.current_gesture()
    print(g)
```

9. Kommandot **g = accelerometer.current_gesture()** lagrar en sträng som visar vilken gest vi har gjort i variabeln g.
   Du hittar det här kommandot under:
   **[menu]>More Modules>BBC micro:bit>Sensors>Gestures**

   Skriv sedan ut värdet av g till konsolen med **print(g)**.

   *Se till att du använder rätt indrag av texten. De två sista raderna ska höra till while-satsen, inte if-satsen.*

```
1.1 1.2  *Dok  RAD
Python Shell  2884/2884
down
up
up
up
up
up
face up
>>>
down
down
```

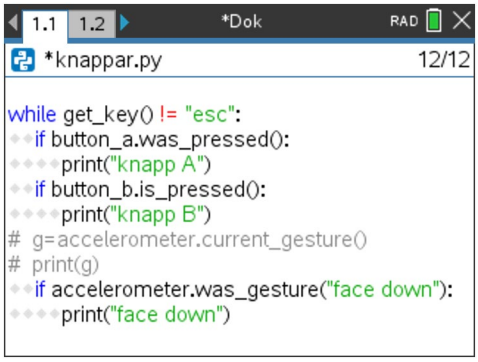
10. Kör programmet och titta på de olika meddelandena på handenheten när du skakar micro:bit-enheten, vänder på den, ställer den upp, etc.

    Några tillgängliga gester är: **face up**, **face down**, **up**, **down**, **left**, **right**, **shake**.

    Prova gärna med att släppa enheten (fånga den säkert).

Kommentera bor de två sista raderna i ditt program med kommentarsymbolen (#) (vilket innebär att programmet ignorerar dessa rader när det körs) och lägg till följande två rader:

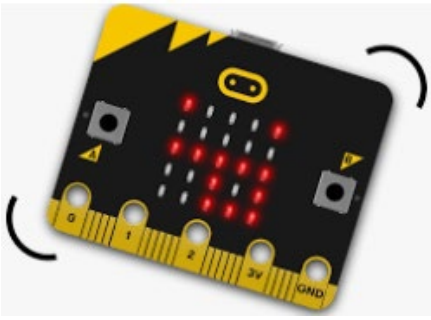
**if accelerometer.was_gesture(“face down”):**
⋄ ⋄ **print(“face down”)**




Du hittar alla gestalternativ och funktioner under:
**[menu]>More Modules>BBC micro:bit>Sensors>Gestures**

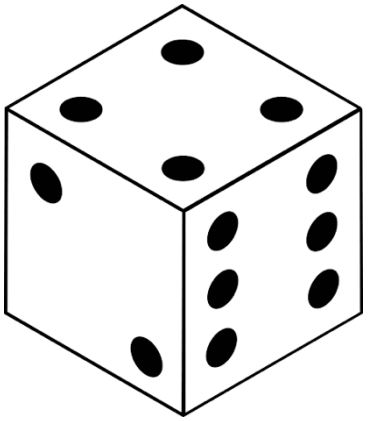
*Tips: för att markera en programrad som en kommentarsrad kan du använda [ctrl] [T].*

Om du kör det senare programmet kommer du att märka en skillnad med den tidigare versionen;
.current_gesture() skriver ut meddelandet kontinuerligt.
.was_gesture() skrivs bara ut om något ändras.
Dessutom vill was_gesture() ha en parameter, ett värde, motsvarande ett möjlig gest som den kan testa om vi har gjort.



11. Knappar och gester låter dig båda hämta inmatning från micro:bit.
I nästa avsnitt ska vi samla in data med hjälp av micro:bit och sedan bearbeta det med TI-Nspire CX.

12. **Del 2: Kasta en tärning**
När knappen A trycks in ska en variabel ges ett slumpmässigt heltal från 1 till 6. Detta simulerar tärningskastet.
(I stället för knapp A kan du också använda knapp B eller en av gesterna.)
Skriv ut talet på micro:bit -skärmen.
Innan du fortsätter kan du först försöka skriva programmet själv.



13. Starta ett nytt Python -program och lägg till micro:bit -modulen igen.
    För att kunna välja ett slumpmässigt (slumpmässigt) nummer behöver vi en funktion från slumpmodulen. Vi lägger också till denna (se andra raden i programmet).
    Vi ska lägga till samma while-sats som tidigare. Så behöver vi en if-sats för att test om knapp A är intryckt. Sedan lägger vi till raden **utfall = randint(1,6)** som nu ger variabeln 'utfall' ett slumpmässigt värde (1 till 6) varje gång knappen A trycks in. (Tänk på indrag.)

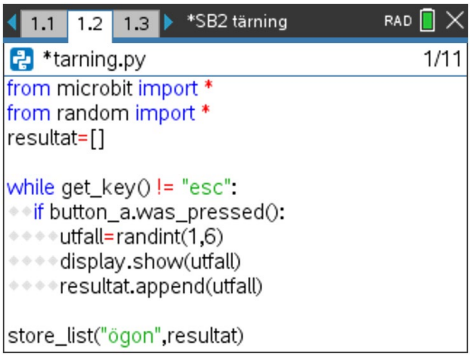

14. Lägg nu till en rad  för att visa detta värde på micro:bit-enhetens display.
    Detta görs med **display.show (utfall)**.

    Kör programmet. Varje gång du trycker på knapp A visas resultatet i micro:bit-displayen.

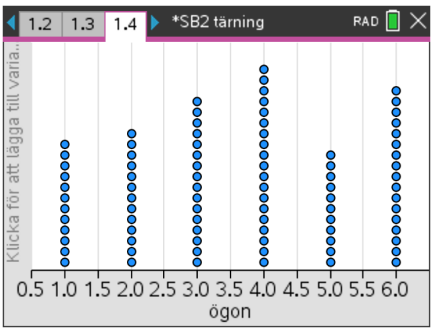

15. **Datainsamling:** Att kasta tärning genom att trycka på en knapp må vara kul, men vi vill spara resultaten så att vi kan dra slutsatser av dem.

16. Börja med att lägga till en tom lista där resultat ska lagras, i detta program kallas listan “resultat”.
    Varje gång knappen A trycks in ska utfallet läggas till i resultat-listan. Du kan göra det med resultat.append().
    Du hittar detta under **[meny]> Inbyggda > Listor**.

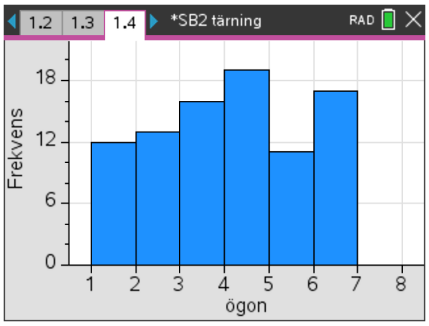
    I slutet av programmet ska vi spara elementen i listan till en TI-Nspire-lista med kommandot:
    store_list ("ögon", resultat).
    Du hittar den här funktionen under:
    **[meny]> BBC micro:bit>  Commands**



17. Kör programmet och tryck på knappen A ganska många gånger. Tryck sedan på **[esc]** för att stoppa programmet. Python-programmet innehåller nu en lista över "utfall" med alla utfall och TI-Nspire har nu också en lista med samma resultat, den här kallas "ögon".
Tryck på **[ctrl]-[doc]** eller **[ctrl]-[i]** för att lägga till en ny sida i dokumentet och välj sedan Data och statistik.
Du kommer att se en skärm med ett stort antal punkter. Genom att klicka längst ner på skärmen och välja variabeln "ögon" visas en figur som den till vänster.



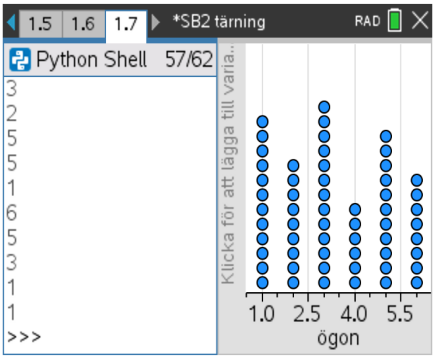

I menyn i det här programmet kan du ändra diagramtypen och välja till exempel ett histogram.

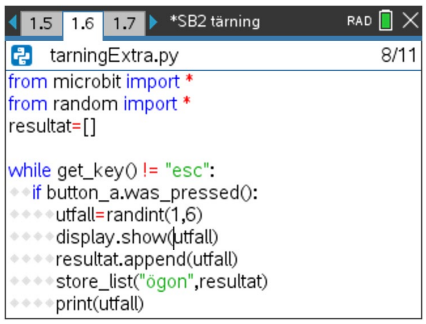



18. **Extrauppgift:** Ändra programmet så att varje gång knappen A trycks ned sparas resultatlistan och resultatet skrivs ut.

Kombinera applikationsskärmarna genom att trycka på **[ctrl]-[4]** i konsolen.
Om du nu kör programmet igen kommer du att se resultaten i den vänstra delen och histogrammet i den högra delen av skärmen ständigt justeras.

*Tips: Ändra diagramtypen till punkdiagram så kommer den automatiskt ändra höjdskalan när antalet punkter blir stort.*



**Tips för läraren**: TI-Nspire -skärmen kan delas upp både horisontellt och vertikalt. Detta kan göras på handenheten med:

**[doc]>Sidlayout>Välj layout>Layout 3**