Statped

micro:bit med Python

Komme i gang

Innledning

I denne veilederen forklarer vi hvordan du kommer i gang med å programmere micro:bit med Python i Visual Studio Code for Windows. Visual Studio Code er en kildekode-editor, det vil si et program som brukes til å skrive tekstbasert programkode. Å komme i gang krever litt forarbeid, men gir en løsning som er godt egnet for elever som bruker skjermleser og for elever med moderat og alvorlig synssvekkelse som leser visuelt. For veldig mange av disse elevene er tekstbasert programmering det eneste som fungerer. Men løsningen passer også fint for normalt seende elever som ønsker å bruke tekstbasert programmering.

For å komme i gang må du først gå gjennom kapittelet om installasjon. Tilleggene er nyttige for å få mer ut av programmeringen og micro:bit, men ikke strengt tatt nødvendige for å ta fatt på oppgavene.

micro:bit kan brukes alene eller sammen med forskjellige tilbehør. I forbindelse med prosjektet super:bit er roboten Bit:bot veldig aktuell. Dette dokumentet beskriver derfor også hvordan du kan komme i gang med Bit:bot. Det er viktig at micro:bit programmeres slik at den gir mening for elever som ikke ser. LED-matrisen på micro:bit er derfor ikke interessant å bruke, i hvert fall ikke alene. Derimot kan lyd, både i form av musikk og syntetisk tale, samt bevegelse være spennende.

Husk at denne veilederen ikke gir noen innføring i bruk av Python. Grunnleggende kunnskap om tekstprogrammering eller Python anbefales. Det finnes mange ressurser som kan gi deg en god introduksjon til dette programmeringsspråket. Det finnes også tilbud om videreutdanning flere steder. Et eksempel er Universitetet i Oslos tilbud ProFag. ProFag tilbyr etterutdanning i programmering med kursrekke for henholdsvis ungdomsskolen og videregående skole. Alle ressursene til kursene ligger åpne på universitetets nettsider.

Hvorfor Python og Visual Studio Code?

Python er et av de mest utbredte programmeringsspråkene og brukes mye som introduksjon til tekstbasert programmering. Det regnes som et språk det er enkelt å komme i gang med. Det brukes i veldig mange sammenhenger og er derfor et naturlig valg. Men blokkbasert programmering blir ofte ansett som enklere enn tekstbasert. Man slipper å forholde seg direkte til programkoden. I stedet trenger man bare velge blokker med ferdig kode. Som en konsekvens av det, vil man ikke få syntaksfeil i koden. Men tekstbasert programmering trenger ikke være så vanskelig. Dessuten har blokkbasert programmering sine ulemper. Fordelen ved å slippe syntaksfeil i koden er samtidig en stor ulempe. Det å finne og rette feil er en veldig viktig del av å programmere. Alle skriver feil i programkoden, også profesjonelle. Dessuten er blokkbasert programmering uansett bare en start. Skal man fortsette med programmering, vil man før eller siden møte et tekstbasert programmeringsspråk. Det er dette som brukes profesjonelt og i høyere utdanning.

Tekstbasert programmering er faktisk like morsomt som blokkbasert. For elever som bruker leselist, er blokkbasert programmering umulig. Det gjelder også for mange elever med alvorlig synssvekkelse som leser visuelt. Flere av de mest kjente løsningene for blokkbasert programmering har dårlige tilpasningsmuligheter, det vil si mangelfulle muligheter for forstørring og kontrastforbedringer.

I Norge har vitensentrene en nøkkelrolle i prosjektet super:bit. Det er vitensentrene som gir opplæring i programmering av micro:bit til elever og lærere. Og det du får er opplæring i bruk av blokkbasert programmering med MakeCode. Men MakeCode kan også brukes til å programmere micro:bit med Python. Hvorfor ikke da like gjerne la eleven bruke MakeCode som Visual Studio Code? Jo, det er mulig, men Visual Studio Code vil likevel være den beste løsningen for elever som bruker leselist og for mange elever med alvorlig synssvekkelse som leser visuelt. Grunnen er at programmet har langt bedre tilpasningsmuligheter, flere hurtigtaster og vil fungere bedre med hjelpemiddelteknologien. Både Visual Studio Code og MakeCode fungerer brukbart sammen med skjermleseren JAWS, men aller best med skjermleseren NVDA.

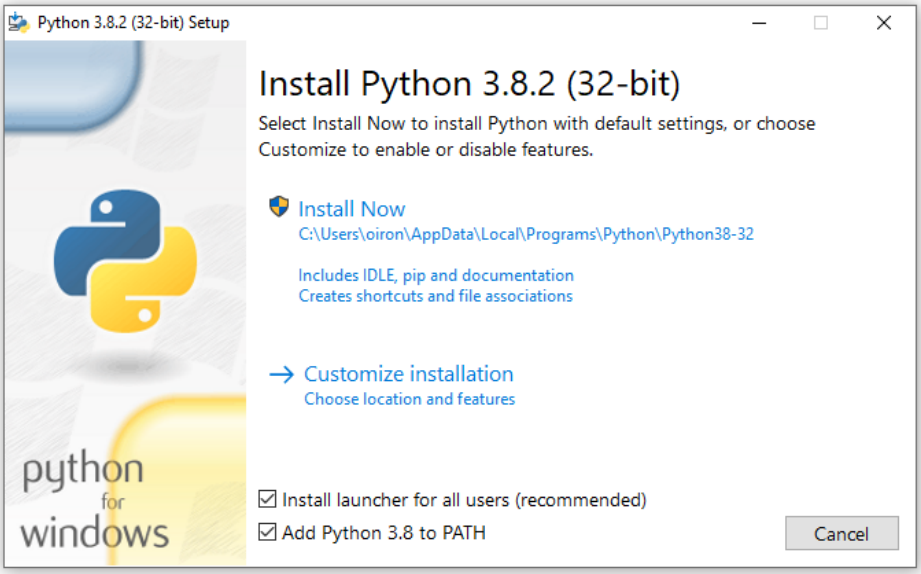
NB! For eleven er det viktig at skolen velger enten Visual Studio Code eller MakeCode. Python-koden for micro:bit og Bit:bot vil se forskjellig ut i de to løsningene. Det er derfor ingen god idé å veksle mellom disse. Vi anbefaler at kildekode-editor velges i samarbeid med rådgiver fra Statped.

Installasjon

Denne installasjonen vil gi deg et utviklingsmiljø som gjør det enkelt for eleven å komme i gang med å programmere micro:bit i Python.

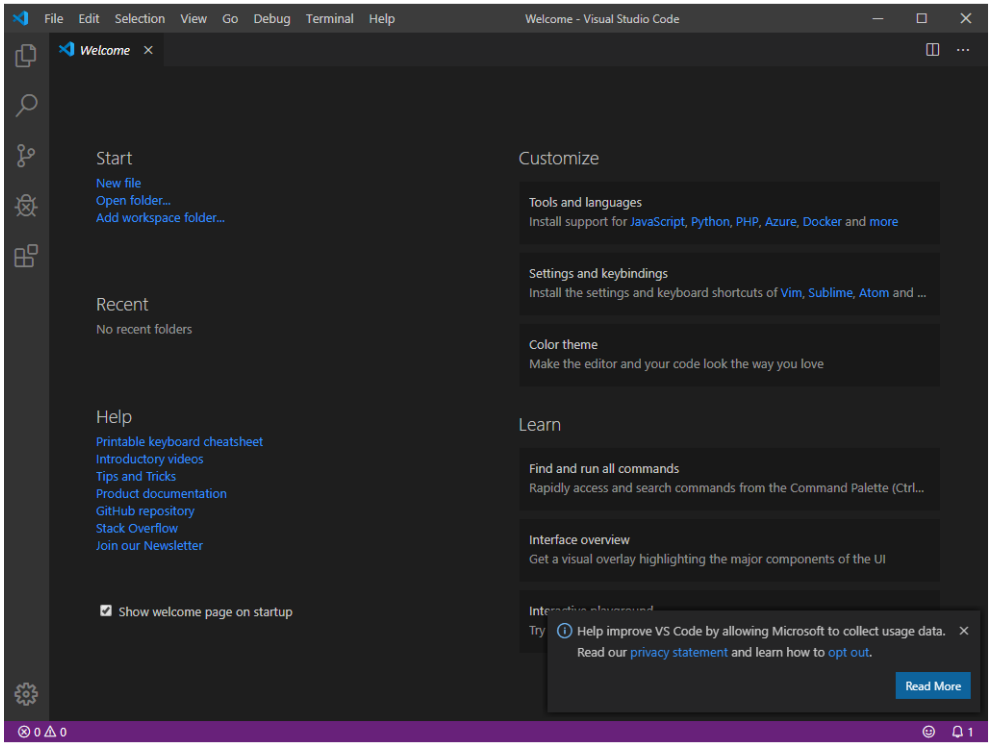
Det første du trenger er selve Python-systemet. Det er en verktøykasse som inneholder forskjellige standard Python-verktøy. Uten den kan ikke datamaskinen forstå Python. Når Python er på plass, kan du installere Visual Studio Code som brukes til å skrive programmene dine. Deretter må du opprette en mappe i Visual Studio Code der programmene dine skal ligge. Og til slutt trenger du to extensions som danner en kobling mellom Visual Studio Code og Python-systemet. Extensions er utvidelser til Visual Studio Code som gir deg funksjonalitet som programmet ikke har som standard.

Python



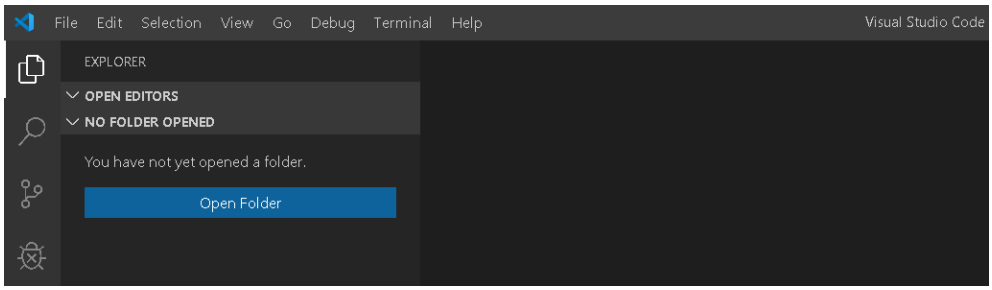
1. Last ned Python 3 fra python.org/downloads.
2. Kryss av for **Add Python to PATH**.
3. Kjør installasjonen ved å velge **Install Now**.
4. Velg **Close** når installasjonen er ferdig.

Visual Studio Code



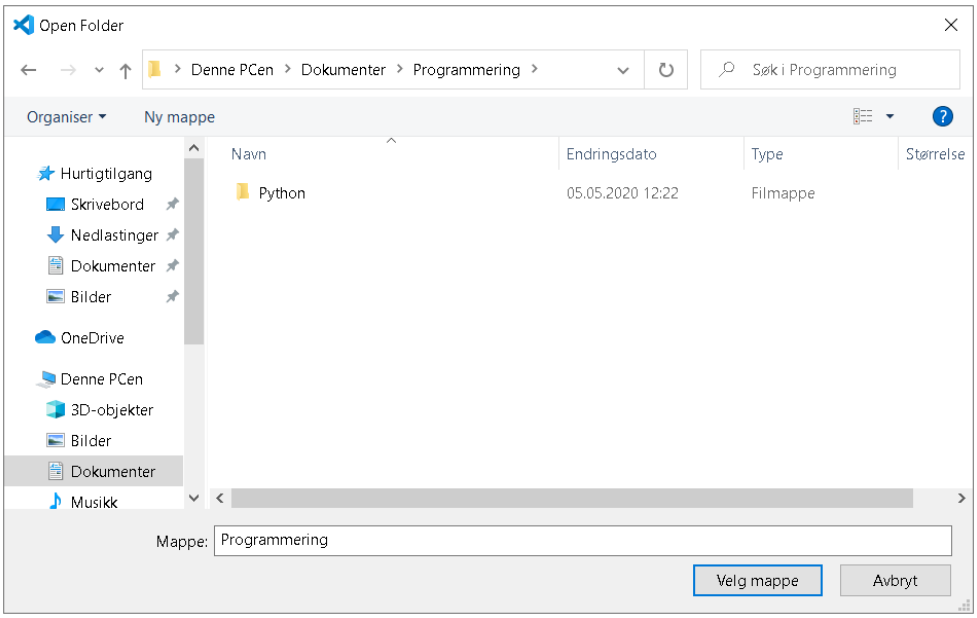
Installasjon

1. Last ned Visual Studio Code fra code.visualstudio.com.
2. Kjør installasjonen.
3. Velg **I accept the agreement** og behold standardvalgene gjennom hele veiviseren.
4. Åpne programmet.

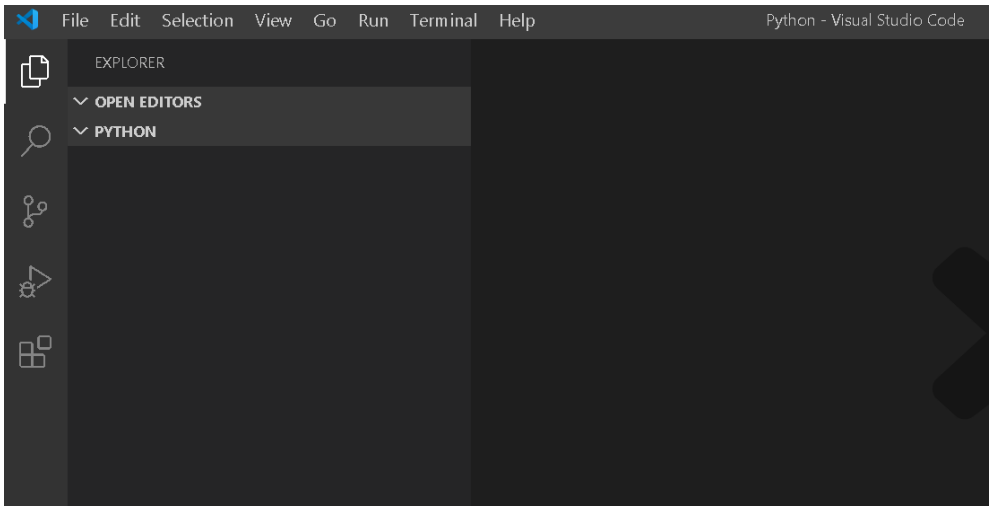


Legg til mappe

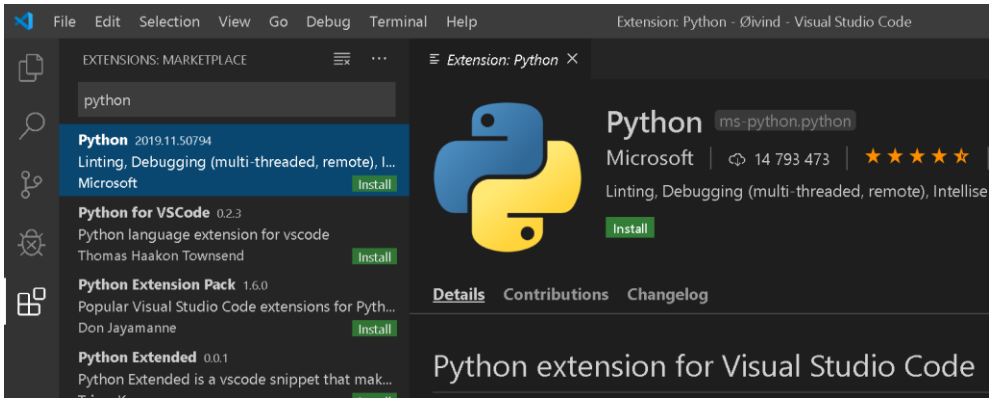
5. Åpne Explorer (**Ctrl+Shift+E**) og legg merke til at du får fram et nytt felt til venstre i programvinduet.
6. I Explorer-feltet vil det stå: «You have not yet opened a folder”.
7. Velg knappen **Open Folder** eller bruk **Ctrl+K** etterfulgt av **Ctrl+O**.



8. Du får fram en dialogboks. Opprett en ny mappe og gi den et beskrivende navn, for eksempel koding, programmering e.l. **Unngå mappenavn som Microbit og Bitbot**. Gå inn i den nye mappen og velg deretter knappen **Velg mappe**.



9. Mappen du valgte åpnes nå i Explorer-feltet i Visual Studio Code. Alle filer du oppretter blir lagret i denne mappen.



Legg til extensions

10. Åpne nå Extensions (**Ctrl+Shift+X**). Dette gir deg et nytt felt til venstre i programvinduet.
11. Søk etter «python» fra søkefeltet øverst i dette feltet. Installer utvidelsen **Python** fra **Microsoft** ved å velge den grønne Install-knappen.
12. Gå tilbake til søkefeltet og søk etter «micro:bit». Installer utvidelsen **micro:bit** fra **Statped**.
13. Start Visual Studio Code på nytt.

Forbered for micro:bit og Bit:bot

Nå har du installert alt som er nødvendig. Kjør nå en kommando i Visual Studio Code for å forberede systemet. Denne kommandoen gjør klar en del filer og innstillinger:

- Åpne Command Palette (**Ctrl+Shift+P**).
- Søk etter og velg kommandoen **micro:bit Prepare**.
- Når du har fått beskjed om at dette var vellykket, trykk **Ctrl+Shift+P** og velg kommandoen **Reload Window**.

Du har nå laget et utviklingsmiljø for programmering av micro:bit i Python. Om du ikke forsto alt, så spiller ikke det noen rolle. Men hvis du ønsker å lære mer, finner du mye informasjon på code.visualstudio.com. Hensikten med denne installasjonsveiviseren, er å gjøre det så enkelt som mulig for eleven.

Komme i gang med programmering

I oppgavene under skal du få prøve deg på å programmere både micro:bit og Bit:bot. Men aller først kan det være lurt å prøve noen rene Python-oppgaver.

Oppgave 1: Skriv navnet ditt (Python)

Utstyr: Visual Studio Code.

Deloppgave A

I denne deloppgaven skal du lage et lite program som skriver en melding på skjermen. Meldingen skal være fornavnet ditt.

1. Opprett en ny fil (**Ctrl+N**).
2. Lagre filen (**Ctrl+S**). Gi filen et navn (for eksempel navn.py).
3. Den første instruksjonen du skal skrive, er en funksjon i Python som heter print. Den brukes for å skrive en melding på skjermen. I dette tilfellet er meldingen navnet ditt (for eksempel Anna). Etter print må selve meldingen stå mellom to parenteser. Og meldingen skal også omsluttes av anførselstegn:

```
print("Anna")
```

4. Lagre filen (**Ctrl+S**).
5. Gå til Terminal i Visual Studio Code (**Ctrl+Ø**).
6. Skriv kommandoen for å kjøre programmet ditt. Du må skrive python etterfulgt av navnet på fila som inneholder programmet ditt:

```
python navn.py
```

7. For å se hva programmet ditt skrev på skjermen, kan du bruke **Ctrl+Pil opp**. Ble det riktig?
8. Gå tilbake til programmet ditt med **Ctrl+1**.
9. Gjør en endring i programmet ditt slik at programmet i stedet skriver «Hei på deg» og så navnet ditt. For eksempel: «Hei på deg, Anna».

```
print(«Hei på deg, Anna»)
```

10. Lagre endringene (**Ctrl+S**) og gå tilbake til Terminal (**Ctrl+Ø**). Kjør programmet på nytt:

```
python navn.py
```

11. Gjorde programmet det det skulle?

Deloppgave B

I denne deloppgaven skal du utvide programmet fra deloppgave A. Vi kan få programmet til å spørre etter navnet på den som skal kjøre programmet. Hun må da taste inn navnet sitt. Til slutt skal programmet skrive navnet på skjermen.

1. Gå først tilbake til programmet ditt (**Ctrl+1**).
2. Opprett en ny fil (**Ctrl+N**).
3. Lagre filen (**Ctrl+S**). Gi filen et navn (for eksempel hvaheterdu.py).
4. Først må vi bruke en funksjon som heter input. Input bruker vi litt på samme måte som print, det vil si at vi trenger parenteser etter input og anførselstegn inne i parentesene. Når programmet inneholder input, vil programmet stoppe opp og vente på svar fra deg. Du må da skrive inn et svar og trykke Enter:

```
input("Hva heter du?")
```

5. For at programmet ditt skal klare å ta vare på svaret du gir, må vi bruke noe som kalles en variabel. En variabel må ha et navn og brukes til å lagre en verdi. En variabel har derfor både et navn og en verdi. Det svaret du gir på funksjonen input blir da lagret i variabelen. Du kan kalle variabelen for navn.

   Hver instruksjon i Python skal skrives på en ny, blank linje. Før du skriver neste instruksjon, må du derfor trykke Enter.

   Skriv variabelnavnet og et likhetstegn foran input-setningen du nettopp skrev:

```
navn = input("Hva heter du?")
```

6. Til slutt i dette programmet skal du bruke print-funksjonen på nytt. Da kan vi få programmet til å skrive ut verdien på variabelen navn. Men du skal bruke print litt annerledes enn i deloppgave A. Når vi bruker anførselstegn, vil meldingen som står mellom anførselstegnene bli skrevet ut nøyaktig slik det står. For å få skrevet ut verdien til variabelen navn, kan vi ikke bruke anførselstegn:

```
print(navn)
```

7. Programmet ditt skal nå se slik ut:

```
navn = input("Hva heter du?")
print(navn)
```

8. Lagre endringene (**Ctrl+S**), gå tilbake til Terminal (**Ctrl+Ø**), og kjør programmet på nytt:

```
python hvaheterdu.py
```

9. Hva skjedde når du kjørte programmet?

10. Du skal nå endre litt på print-funksjonen du brukte. I tillegg til å skrive verdien på variabelen navn, skal programmet også skrive en liten melding. Husk at vi må bruke anførselstegn for å skrive ut en melding nøyaktig slik den står, men at vi må sløyfe anførselstegnene når vi skal skrive ut en variabel. For å skrive ut både en melding og en variabel, må du skrive følgende:

```
print("Hei " + navn)
```

11. Legg merke til at det står et mellomrom etter ordet hei. Det er for at Hei og verdien til variabelen navn ikke skal står helt inntil hverandre. Det ser bedre ut. Programmet ditt ser nå slik ut:

```
navn = input("Hva heter du?")
print("Hei " + navn)
```

12. Lagre endringene (**Ctrl+S**), gå tilbake til Terminal (**Ctrl+Ø**) og kjør programmet på nytt. Hva ble annerledes denne gang?
13. Ta en skjermdump av programvinduet til Visual Studio Code og lim inn bildet her.

Oppgave 2: Spill en melodi (micro:bit)

Nå skal du skrive et lite program som gjør at micro:bit kan spille en melodi.

Utstyr: micro:bit, høyttalere, Visual Studio Code.

1. Opprett en ny fil (**Ctrl+N**).
2. Lagre filen (**Ctrl+S**). Gi filen et navn i feltet Filnavn og velg Python i feltet Filtype. (Eller skriv hele filnavnet direkte i feltet Filnavn, for eksempel melodi.py.) Alle Python-filer skal ha etternavnet .py. Da vet Visual Studio Code at du programmerer i Python.
3. Når du programmerer micro:bit, må programmet alltid starte med at du gjør micro:bit-modulene tilgjengelig for programmet ditt:

```
from microbit import *
```

4. Og for at micro:bit skal kunne spille en melodi, trenger du å importere music-modulen også:

```
import music
```

5. Nå skal du skrive setningen som gjør at micro:bit spiller melodien Happy birthday. Legg merke til at du må skrive navnet på melodien med store bokstaver (BIRTHDAY). La det gjerne være en blank linje mellom import-setningene og resten av programmet.

```
music.play(music.BIRTHDAY)
```

6. Lagre endringene du har gjort i programmet (**Ctrl+S**). Programmet ditt skal nå se slik ut:

```
from microbit import *
import music

music.play(music.BIRTHDAY)
```

7. Sørg for at micro:bit er koblet til PC-en med USB-kabelen som følger med.
8. Overfør programmet ditt til micro:bit med kommandoen **Ctrl+F5**. Når overføringen er ferdig etter noen få sekunder, vil micro:bit kjøre programmet. Du vil da høre melodien i høyttaleren.
9. Hvis du vil høre melodien en gang til, må du trykke på reset-knappen på baksiden av micro:bit. Da kjøres programmet på nytt.
10. Gjør endring i programmet ditt slik at micro:bit spiller en annen melodi.

Tips: Det kan være vanskelig for synshemmede elever å koble høyttalere til micro:bit. Vanligvis bruker man ledninger med krokodilleklemmer eller bananplugger til dette. I stedet går det an å bruke en enhet som består av en batteripakke, høyttaler (buzzer) og et spor der man kobler til micro:bit. Et eksempel på dette er 4tronix Music Box Mk2.

Oppgave 3: Kjør en meter med Bit:bot (Bit:bot)

I denne oppgaven skal du programmere roboten Bit:bot slik at den kjører én meter rett fram.

Utstyr: micro:bit, Bit:bot XL, Visual Studio Code.

1. Lag en ny fil (**Ctrl+N**) i Visual Studio Code.
2. Lagre filen (**Ctrl+S**).
3. I tillegg til den vanlige import-setningen for micro:bit, trenger du en import-setning for Bit:bot:

```
from microbit import *
from bitbot import *
```

4. For å få Bit:bot til å kjøre rett fram, kan du bruke en metode som heter goms (go milliseconds). goms krever at du oppgir 3 såkalte argumenter, nemlig retning (FORWARD eller REVERSE), hastighet (0-100%) og tid (antall millisekunder). Når du skal bruke metodene som hører til Bit:bot, begynner du setningene med bitbot etterfulgt av punktum:

```
bitbot.goms(FORWARD, 50, <millisekunder>)
```

   Hvilken verdi må du sette inn i stedet for <millisekunder> for at roboten skal gå én meter rett fram? Verdien skal altså oppgis i millisekunder. Hvor mye er det?

5. Pass på at Bit:bot er slått av (knapp på baksiden) før du overfører programmet!
6. Lagre programmet ditt, koble til micro:bit og overfør med **Ctrl+F5**.
7. Koble USB-ledningen fra micro:bit og sett det inn i sporet på Bit:bot slik at de to knappene (A og B) vender framover.
8. Slå på Bit:bot med bryteren på baksiden.
9. Hvis ikke roboten starter automatisk, kan du trykke på reset-knappen på baksiden av micro:bit.
10. Hvor langt kjørte roboten? For kort eller for langt? Endre i så fall antall millisekunder i programmet ditt slik at roboten kommer nærmere én meter.

Disse oppgavene er basert på oppgaver fra oppgaver.kidsakoder.no/microbit og www.vitensenter.no/superbit/elev.

Tillegg A: Hurtigtaster

Visual Studio Code har mange hurtigtaster, men strengt tatt trenger man ikke så mange for å bruke programmet.

Generelle

Kommando	Hurtigtast
Ny fil	**Ctrl+N**
Lagre fil	**Ctrl+S**
Åpne fil	**Ctrl+O**
Overføre program til micro:bit	**Ctrl+F5**
Problems (feil og advarsler)	**Ctrl+Shift+M**
Gå til terminal	**Ctrl+Ø**
Gå til editor (fil)	**Ctrl+1**

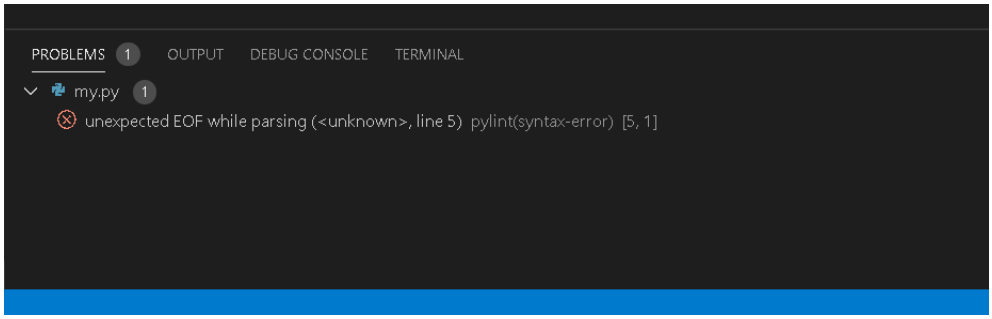
Tillegg for elever som leser visuelt

Kommando	Hurtigtast
Color Theme (fargeoppsett)	**Ctrl+K, Ctrl+T**
File Icon Theme (visning av ikoner)	**Ctrl+Shift+P**, søk etter “file icon theme”
Zoom	**Ctrl++/Ctrl+-**
Zen mode (enkelt oppsett)	**Ctrl+K, Z**
Sidebar vis/skjul	**Ctrl+B**
Panel vis/skjul	**Ctrl+J**

Visual Studio Code hurtigtaster: https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf.

Tillegg B: Rette feil i programmet

Alle som driver med programmering, skriver feil i programkoden. Derfor er det viktig å lære hvordan man kan finne og rette de feilene som oppstår.



Problems

I panelet Problems (**Ctrl+Shift+M**) vises en liste over advarsler og feil i programkoden din. Dette er feil og advarsler som dukker opp mens du programmerer. Bruk pil opp eller ned for å se advarslene i lista. Trykk Enter for å gå direkte til det stedet i programkoden som advarselen viser til.

Vær klar over at lista viser feil og advarsler for alle åpne filer i Visual Studio Code. Det kan derfor lønne seg å bare ha en fil åpen av gangen.

NB! Under Problems vil du ikke få fram feil i koden som er knyttet til modulene music og speech.

Tillegg C: Dokumentasjon

Nedenfor finner du lenker til beskrivelse av funksjonene og metodene som er tilgjengelig for micro:bit og Bit:bot XL. I tillegg er det nyttig å ha litt grunnleggende kunnskap om Python.

- micro:bit: https://microbit-micropython.readthedocs.io/en/latest/index.html
- Bit:bot XL: https://github.com/oivron/bitbotxl eller under

Bit:bot XL dokumentasjon

Alle metodene under tilhører Python-modulen for Bit:bot XL. For å bruke disse metodene, må du først skrive bitbot, etterfulgt av punktum og så navn på metoden.

Husk at Bit:bot XL består av to elektromotorer – en på hvert hjul. Slike motorer er ustabile og gir ikke alltid det resultatet en forventer. Motorene kan for eksempel trekke ulikt slik at roboten drar mot høyre eller venstre. Svært lave hastigheter bør unngås. Det kan bety at roboten ikke beveger seg i det hele tatt. Og du vil også legge merke til at roboten kan kjøre med en annen hastighet bakover enn den gjør forover, selv om du har oppgitt at den skal kjøre med samme fart.

bias

```
# Direction: LEFT eller RIGHT.
# Percent: 0-100%.
bias(direction, percent)
```

Brukes for å korrigere Bit:bot slik at den ikke drar mot høyre eller venstre, men følger en rett linje. Hvis roboten drar mot høyre med for eksempel 10 cm over en distanse på 1 meter, kan dette korrigeres ved å velge direction=left og percent=10:

```
bitbot.bias(LEFT, 10)
```

bias-metoden trenger du bare å bruke én gang i programmet ditt. Sett den øverst, rett etter import-setningene.

go

```
# Direction: FORWARD eller REVERSE.
# Speed: 0-100%.
go(direction, speed)
```

Kjører roboten forover eller bakover med en bestemt hastighet. Retningen bestemmes med direction og hastigheten med speed. Speed oppgis i prosent. Å kjøre roboten forover med 40% hastighet, skrives slik:

```
bitbot.go(FORWARD, 40)
```

goms

```
# Direction: FORWARD eller REVERSE.
# Speed: 0-100%.
# Duration: i millisekunder.
goms(direction, speed, duration)
```

Samme som go, men spesifiserer også hvor lenge roboten skal kjøre. Å kjøre roboten bakover med 60% hastighet i 2 sekunder skrives slik:

```
bitbot.goms(REVERSE, 60, 2000)
```

spin

```
# Direction: LEFT eller RIGHT.
# Speed: 0-100%.
spin(direction, speed)
```

Brukes for å snu eller rotere roboten. Retningen bestemmes av direction og hastigheten med speed. Når roboten roterer, vil det ene hjulet gå forover, mens det andre går bakover. Den vil med andre ord rotere uten å bevege seg forover eller bakover. For å snu roboten mot høyre med hastighet 30%, må vi skrive:

```
bitbot.spin(RIGHT, 30)
```

spinms

```
# Direction: LEFT eller RIGHT.
# Speed: 0-100%.
# Duration: i millisekunder.
spinms(direction, speed, duration)
```

Samme som spin, men spesifiserer også hvor lenge roboten skal rotere. Å rotere roboten mot venstre med 50% hastighet i et halvt sekund skrives slik:

```
bitbot.spinms(LEFT, 50, 500)
```

buzz

```
# Duration: i millisekunder.
buzz(duration)
```

Buzz brukes for å la roboten lage et pipesignal. Hvis vi vil la roboten pipe i et tidels sekund, skriver vi:

```
bitbot.buzz(100)
```

linesensor

```
# Directon: LEFT eller RIGHT.
linesensor(direction)
```

Brukes til å lese av linjesensoren. Bit:bot XL har egentlig to linjesensorer, en foran på hver side. Begge sensorene vil registrere endringer, enten at de havner utenfor en linje, eller ved at de kommer inn på en linje. Vi kan utnytte dette og få roboten til å følge en linje eller å kjøre mellom to linjer. Hvis roboten er i ferd med å forlate linja og dra ut mot venstre, må vi korrigere retningen ved å rotere roboten mot høyre. Slik kan du sjekke om roboten kjører for mye til venstre (dvs. at den forlater linja den har fulgt):

```
if (bitbot.linesensor(RIGHT) == 1):
    # Korriger retningen slik at roboten kjører mer mot høyre.
```

I eksemplet over må du erstatte kommentaren (merket med #) med koden som er nødvendig for å få roboten tilbake til linja.

For å lage linjer som roboten skal følge, bruker du tapen som følger med micro:bit klassesettet. Men du kan like gjerne bruke vanlig svart tape.

sonar

```
# Returnerer avstand i cm.
sonar()
```

Brukes for å finne avstanden til det nærmeste objekt foran roboten. Metoden returnerer avstanden i cm. Følgende setning vil sjekke om avstanden til nærmeste objekt er mindre enn 10 cm.

```
if bitbot.sonar() < 10:
    # Korriger retningen så ikke roboten krasjer med det som er foran.
```

NB! Sonar-metoden forutsetter at sonar-enheten er montert på Bit:bot XL. Denne følger med Bit:bot XL som standard og monteres i sporet helt foran på roboten.

stop

```
stop()
```

Stopper roboten.

Vær oppmerksom på at de metodene som har duration som parameter, automatisk vil bruke metoden stop. Du trenger for eksempel ikke bruke stop etter at du har brukt spinms. spinms stopper roboten automatisk etter angitt tid.

Tillegg D: Ressurser

- Visual Studio Code hurtigtaster: https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf
- micro:bit dokumentasjon: https://microbit-micropython.readthedocs.io/en/latest/index.html
- Bit:bot XL dokumentasjon: https://github.com/oivron/bitbotxl
- Superbit: https://www.vitensenter.no/superbit/

Telefon: 02196

www.statped.no
facebook.com/statped
twitter.com/statped