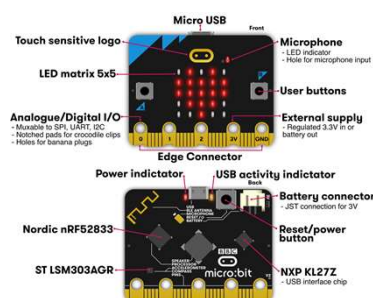


LYD-Kit Spor 1 (LKS1)

Byg med sensorer

Modul a – "Værkstedskørekort"

Micro:Python og indbyggede sensorer



Projekt støttet af:
midt
regionmidtjylland

```
1 # 3c - lys til lyd.py
2 # Konvertering af lys intensitet til lyd (pitch) Version 1.1 10-Nov-2020
3 # Importer de nødvendige biblioteker/moduler.
4 # Den generelle microbit for at nå display.read_light_level() funktionen
5 # og music for at nå music.pitch() funktionen
6 from microbit import *
7 import music
8
9 # Lys niveau kan have værdier mellem 0 og 255
10 # pitch kan håndtere værdier mellem 0 og 3906Hz. Vi kan høre fra ca. 50Hz.
11 # - derfor lys niveau*14+50 som 'fornuftig' frekvens til pitch funktionen
12 while True:
13     x = display.read_light_level()
14     music.pitch(x*14+50)
15     print(x)
16     sleep(100)
```

Til lærer som info/inspiration.

Kan bruges sammen med opslagsværkerne nedenfor som en introduktion til programmering i Python/MicroPython (på Micro:Bit)

[Python Tutorial \(w3schools.com\)](https://www.w3schools.com/python/) og

[BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/)

(Klikbare links på slides sidst i stakken – De er desværre ikke klikbare når de er i noterne! – der er referencer til disse med links i slides undervejs.)

Kan findes ved at søge i google på følgende:

"w3schools python tutorial" og

"micropython microbit"

Og det er en god ide at vise de 2 opslagsværker live ifm undervisningen.

Dækker Spor 1 i LYD-Kit materialet.

Slide stak er tænkt som inspiration til lærer – ikke som et færdigt forløb overfor eleverne.

Version 04-Feb-2022:

Rettet/tilføjet sådan at det kan bruges som 3 forløb af kortere varighed.

Modul a:

Det basale og nødvendige for at kunne bruge materialerne i alle 4 spor i LYD-Kit med en basal forståelse for hvad der sker i koden – ‘Værkstedskørekort’

Modul b:

Et værkstedsmodul for dem som vil vide mere/gå dybere i Microbitten for dem som vil bygge selv med Micro:Bitten, og grundlag for at kunne kikke ind i den færdige kode til Micro:Bit som der udleveres i Spor 2.

Modul c:

En case, hvor man med udgangspunkt i færdig kode kan lave funktions kritik, teknologivurdering og produktdesign.

Version 23-Nov-2020:

Lagt op efter test med 1x og 2x på SSG. Der blev det kørt som et samlet forløb på 10 lektioner.

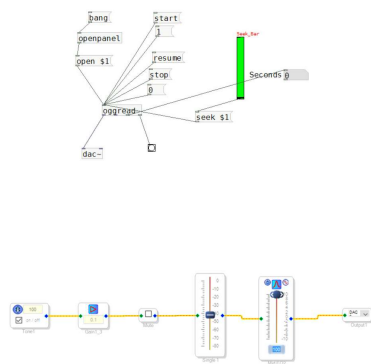
BEMÆRK at LYD-Kit er organiseret sådan at man IKKE behøver at komme dybt i programmeringen, da der er en række færdige programmer og skabeloner, som man kan bygge ovenpå; både uden at ændre i kode og ved at lave simple modifikationer.

Man skal dog have en basis forståelse for Python og et såkaldt udviklingsmiljø (IDE – Integrated Development Environment) for at kunne benytte de færdige programmer. Det er det vi kalder ‘Lukket bog’ tilgangen. Det er omvendt også muligt at bruge det som ‘åben bog’ til at gå dybere i programmeringsdelen, hvis man vil.

Derved understøttes den didaktiske tilgang – Faded Guidance, og UMC (Use Modify Create) kendt fra informatik og programmerings fagene, samt i tilgangen fra Computational Thinking (CT)

Spor 1 Modul a: Programmeringssprog

Projekt støttet af:
midt
regionmidtjylland



```
src file named HelloWorld.java
name
main() method
public class HelloWorld {
    public static void main(String[] args) {
        // Prints "Hello, World" in the terminal window.
        System.out.print("Hello, World");
    }
}
```

Som introduktion til programmeringssprog og udviklingsmiljøer (IDE)

Der er mange slags programmeringssprog. Visuelle, blokbaserede og tekst baserede. Hør hvad eleverne har stødt på. De fleste har nok stødt på Blokbaseret programmering – evt. i form af SCRATCH, men måske også MakeCode til Micro:Bit.

De tekstbaserede bruges til professionel udvikling af programmer i forskellige sammenhænge. Der er også mange forskellige slags tekstbaserede sprog, der har forskellige styrker.

Der findes på nye og de udvikler sig over tid. Så hvis man vil arbejde med programmering skal man være forberedt på at skulle kunne flere sprog og at de ændrer sig undervejs.

Python:

Python er gennemgående for alle spor i LYD-Kit.

Det anses for at være en fordel for begynderen, at man kan bruge det samme grundlag både på en mikrocontroller som Microbitten (men også andre) såvel som på en computer som PC/MAC samt Raspberry PI.

LYD-Kit bruger Python fordi det er et sprog der er let at komme i gang med, men som har rige muligheder for at skalere til mange forskellige anvendelser, fra styring af Micro

kontrollere til Datascience og Machine learning. Det er et sprog der er populært i tiden og har været i vækst over en længere periode.

Man vil kunne få glæde af, og møde Python i forbindelse med mange videregående uddannelser indenfor STEM fagene.

Klikbar link på billedet i midten, som reference til hvor oversigt kommer fra.

Udviklingsmiljø (IDE):

Der er tilsvarende mange forskellige udviklingsmiljøer (IDE – Integrated Development Environment) man kan bruge til at skrive Python programmer i.

Da LYD-KIT retter sig mod begynder og på at kunne bruges uden den store programmeringsindsats er Thonny valgt. Thonny er et enkelt, let tilgængeligt og gratis IDE som fungerer på både PC/MAC og Raspberry PI. Det betyder ikke at man ikke kan bruge andet. Det er bare det som LYD-kit er testet af med. Thonny er også på udviklingstidspunktet default IDE som kommer sammen med Raspberry OS til Raspberry PI.

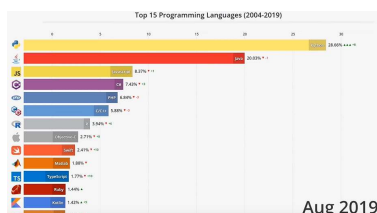
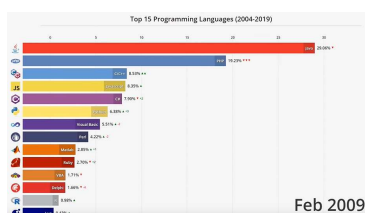
I det følgende er det forudsat at Thonny er installeret på elevernes maskiner og at diverse demoer gennemgås med Thonny. Igen så vil det kunne laves med andre IDE's.

Andre populære IDE's er for eksempel:

- Pycharm
- Visual code

Spor 1 Modul a: Programmeringssprog

Statistikker og lignende optællinger har altid nogle forudsætninger så man kan finde forskellige bud på ex. hvad der er mest udbredt så vær varsom. Men de 2 der er i denne præsentation indikerer at Python er udbredt og er blevet mere og mere anvendt over tid.



Worldwide, Jan 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.74 %	-1.8 %
2		Java	18.01 %	+1.2 %
3		JavaScript	9.07 %	+0.6 %
4	↑	C/C++	7.4 %	+1.1 %
5	↓	C#	7.27 %	+0.7 %
6		PHP	6.06 %	+0.0 %
7		R	4.19 %	+0.3 %
8		Objective-C	2.27 %	-1.4 %
9		Swift	1.91 %	-0.2 %
10		TypeScript	1.74 %	-0.0 %
11		Matlab	1.74 %	+0.0 %

<https://www.youtube.com/watch?v=yL704C1PI4o>

Endnu et eksempel til at underbygge historien om at Python er i vækst og benyttes af mange.

Klikbare links på figurerne – hvis man vil uddybe – vise en udvikling over tid.
Det kan også gives som lektier eller baggrundsmateriale, som eleverne selv kan kikke i.

Spor 1 Modul a: Basis strukturer

```
while <betingelse>:  
    <gør dit>  
    <gør dat>
```

```
if <betingelse 1>:  
    <gør dit når betingelse 1 er opfyldt>  
    <gør dat når betingelse 1 er opfyldt>  
elif <betingelse 2>:  
    <gør noget andet dit når betingelse 2 er opfyldt>  
    <gør noget tredje når betingelse 2 er opfyldt>  
else:  
    <gør noget femte når hverken betingelse 1 eller 2 er opfyldt>
```

The while Loop

With the `while` loop we can execute a set of statements as long as a condition is true.

[Python While Loops \(w3schools.com\)](https://www.w3schools.com/python/python_while_loops.asp)

Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

[Python Conditions \(w3schools.com\)](https://www.w3schools.com/python/python_conditions.asp)

HUSK ":" og indrykning er afgørende for at fortælle fortolkeren hvilken kode stump der hører til i løkker og udføres på baggrund af en betingelse! Det er en del af sprogets 'syntax' [Python Syntax \(w3schools.com\)](https://www.w3schools.com/python/python_syntax.asp)

Andre programmeringssprog benytter andre former for at afgrænse kode stumper. Ex. Do .. End, Begin .. End, og { }.

Kan bruges til en overordnet snak om syntaks inden der laves kodning, og som introduktion til hvor man kan finde mere information.

Slå gerne op på nettet på referencer og vis det som del af gennemgang. ("w3schools Python Tutorial" – Kan bruges som søgeord til google, så kommer man direkte dertil).

Der er klikbare links i slide direkte til relevante steder i reference opslagsværk. De findes også samlet sidst i slide stak, hvis man ønsker at dele dem på anden vis.

Efter at have vist slide(s) med struktur bør man tage en runde hvor man viser programmeringen i Thonny, og kan så bruge de efterfølgende slides (med programeksemples) som opsamling/hand-outs til en opsamlende snak/repetition.

Vis at eksempler kan eksekveres både på egen PC/MAC såvel som på Microbitten. Først gennemgå koden – byg den op og vis den bliver eksekveret på PC

Tilslut Micro:bit på USB og installer Python på den – hvis det ikke allerede er gjort se: "notat om at komme i gang med Thonny.pdf" under installationsvejledninger under folderen generel i Lyd-Kit.

HUSK – Der skal være USB kabler med det rigtige stik til elevernes PC'er – Det kabel der

følger med Micro:Bit har et USB-A, men visse MAC's har kun usb-c – så der skal man have en adapter eller et andet kabel.

Gør en del ud af at få forklaret at der er en væsentlig forskel – på 2 vidt forskellige maskiner! (processoren i hhv. PC/MAC og på Microbitten). ←

Men altså at resultatet er det samme, for de eksempler der gennemgås! De 2 maskiner kan finde ud af at lave det samme grundlæggende!

Spor 1 Modul a: Basis strukturer

```
<listenavn> = (<liste element 1>,<liste element 2>,...)
```

List

Lists are used to store multiple items in a single variable.

[Python Lists \(w3schools.com\)](https://www.w3schools.com/python/python_lists.asp)

```
for <elementer i en sekvens>:
```

```
    <gør dit>
```

```
    <gør dat>
```

Python For Loops

A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

[Python For Loops \(w3schools.com\)](https://www.w3schools.com/python/python_for_loops.asp)

```
def <funktions navn>(<evt. parametre>):
```

```
    <gør dit>
```

```
    <gør dat >
```

```
    <og mere til, som man kan udnytte flere steder/gange>
```

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

[Python Functions \(w3schools.com\)](https://www.w3schools.com/python/python_functions.asp)

KAN udelades, da det kun er nødvendigt med grund forståelsen for while-løkken og if-betingelsen til eksemplerne i modul a.

Men er en god 'overbygning' til den foregående slide, hvis man vil lidt mere.

Igen bør man viser programmeringen i Thonny, og kan så bruge de slides med program som opsamling/handouts efter at man har vist eksempel.

Spor 1 Modul a: Et simpelt program – betingelse og løkke

```
1 # En variabel med navnet 'a' som initielt får tildelt værdi nul (tallet 0)
2 a = 0
3 # En while løkke der gentages sålænge betingelsen (a<=5) "a er mindre eller lig med 5" er opfyldt
4 while a<=5:
5     # udfører en handling/en funktion der udskriver værdien af variabelen a i shell
6     print(a)
7     # en betinget sætning, der tester om a er større eller mindre end 3 og udskriver det i shell
8     if a<3:
9         print("a mindre end 3")
10    elif a>3:
11        print("a større end 3")
12    else:
13        print("a er lig med 3")
14    # der lægges 1 til værdien af variabelen a
15    a = a+1
16 print("while løkken er nu slut - Bemærk at koden både kan køre på PC/MAC og på Micro:Bitten")
```

HUSK – tekstbaseret programmering kræver at man skriver det rigtige. Ellers forstår fortolkeren det ikke og man får en syntaks fejl. Der er også forskel på store og små bogstaver.

Men prøv bare der er ikke noget der går i stykker og man får en fejlmelding i shell som man kan bruge til at få løst fejlen.

Ambitionen er IKKE at eleverne forstår alt her – men at de kort introduceres til begreberne og syntaks for de væsentligste strukturer. Og at de får mulighed for selv at komme videre vha. opslagsværker og tutorials, hvis de ønsker.

De skal bare her få muligheden for at kunne læse færdige programmer, og få en basal fornemmelse for hvad simple programmer gør.

Som lærer kan man vurdere hvor meget denne del skal fylde og også hvor dybt man går. Der er en risiko for at man får givet for meget information og at man taber nogen.

Det kan være bedre at komme hurtigt til eksemplerne med Micro:Bitten og få nogle oplevelser med noget der sker/virker.

Har forsøgt med at kun benytte niveau fra slide 4/6 og så gå derfra til slide 10 eksemplet for at komme hurtigt til et hands-on resultat med microbitten.

Hands-on kommer efterfølgende med fokus på Micro:Bitten.

HUSK igen at modul a primært er tænkt til at eleverne efterfølgende vil kunne bruge færdig kode til forløb i spor 2.

Kode eksempel findes under LKS1>Ma>LKS1Ma-Slide6.py.

De kan udleveres til eleverne efter at man har lavet det i Thonny og vist funktionen. De kan så selv læse kommentarerne og prøve at køre det.

Spør 1 Modul a: Et simpelt program – liste, funktion og for løkke

```
1 # Liste med navne som tekst strenge
2 # Index   0     1     2     3     4   - Der er 5 navne i listen - også kaldet længden af listen
3 navne = ['Jens', 'Peter', 'Ole', 'Anne', 'Sofie']
4
5 # Definition af en funktion der bliver døbt til at hedde "hmd".
6 # Funktion tager et navn som argument og udskriver 4 liniers velkomst i shell
7 def hmd(n):
8     print ('Hej med dig')
9     print (n)
10    print("Godt at se dig")    # bemærk at både ' og " kan bruges omkring en tekst.
11    print()
12
13 # En for løkke der for alle navne i listen navne kalder funktionen hmd
14 for n in (navne):
15     hmd(n)
16
17 # Det samme i modsat rækkefølge vha indexering i listen - man udpeger selv index i listen med navne
18 print("Længden af listen er: "+str(len(navne)))    # Kan sammensætte tekster med + str() funktionen laver tal om til tekst.
19 for i in range(len(navne)):
20     hmd(navne[len(navne)-i-1])    # Der er 5 navne, men index går fra 0 til 4.
```

KAN udelades, da den ikke er nødvendig for at komme i gang med MicroPython og Micro:Bitten.

I resten af forløb a med Micro:Bit er der for eksempel ikke lagt op til noget hvor man selv skal lave funktioner.

Det kan evt. bruges som en option til dem der vil vide mere. Passer sammen med spør 2 eksempler, hvor funktioner og lister er brugt i cases.

Kode eksempel findes under LKS1>Ma>LKS1Ma-Slide7.py.

De kan udleveres til eleverne efter at man har lavet det i Thonny og vist funktionen. De kan så selv læse kommentarerne og prøve at køre det – evt. som hjemmeopgave.

Spor 1 Modul a: Introduktion til biblioteker/moduler

```
1 # import af bibliotek/modul med random funktionaliteter.
2 import random
3 # Liste med navne som tekst strenge
4 # Index 0 1 2 3 4 - Der er 5 navne i listen - også kaldet længden af listen
5 navne = ['Jens', 'Peter', 'Ole', 'Anne', 'Sofie']
6
7 # Definition af en funktion der bliver døbt til at hedde "hmd".
8 # Funktion tager et navn som argument og udskriver 4 liniers velkomst i shell
9 def hmd(n):
10     print ('Hej med dig')
11     print (n)
12     print("Godt at se dig") # bemærk at både ' og " kan bruges omkring en tekst.
13     print()
14
15 # Et tilfældigt udvalgt navn fra listen med navne vha importeret random modul/bibliotek
16 hmd(random.choice(navne)) # virker både i Python og MicroPython
```

[Python Modules \(w3schools.com\)](https://www.w3schools.com/python/python_modules.asp)

[Python Random Module \(w3schools.com\)](https://www.w3schools.com/python/python_random_module.asp)

[Random Number Generation — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://www.bbc.com/microbit/micropython-1.0.1-documentation)

```
1 # import af bibliotek/modul med random funktionaliteter.
2 import random
3
4 # Definition af funktion der sammenligner parameteren i med konstanten 3,
5 # og udskriver om i er mindre end, større end eller lig med 3
6 def lms(i):
7     if i<3:
8         print(str(i)+" er mindre end 3") # Kan sammensætte tekster med + str() funktionen laver tal om til tekst.
9     elif i>3:
10        print(str(i)+" er større end 3")
11    else:
12        print(str(i)+" er lig med 3")
13
14 # metoden/funktionen randrange() fra modulet/biblioteket random finder et tal mellem 0 og 7
15 # returnerer det og gemmer det i variabelen i.
16 # Efterfølgende kaldes funktionen lms med i som argument.
17 i=random.randrange(7) # virker både i Python og MicroPython
18 lms(i)
```

Kan udelades og man kan springe direkte til microbitten og biblioteket Micro:bit for ikke at skabe 'overload' eftersom fokus er på at give eleverne en fornemmelse for sensorerne i Microbitten, og for at kunne lade og bruge færdige kode eksempler til Micro:bitten i Spor 2.

Funktionalitet af basis Python udvides vha. biblioteker (også kaldet moduler) som kan importeres og benyttes i programmerne.

Mange biblioteker virker til flere maskiner og operativsystemer (OC/Windows, MAC/MACos, Raspberry PI/Rapsberry OS/Linux)

Og nogle få er helt generelle og kan også benyttes i MicroPython – Ex. random –MEN ikke nødvendigvis med fuld funktionalitet!

De 2 eksempler i slide kan køre både på PC/MAC og på Micro:bit.

MEN bemærk at der er flere funktioner i random som ikke er med i MicroPython. Vis det evt. ved at bruge dir(random) funktion i shell både med Python og med MicroPython som valgt fortolker.

Kan også vises ved at åbne dokumentationen for Python (W3schools linken) hhv. MicroPython for Micro:bit. (micro:bit MicroPython linken)

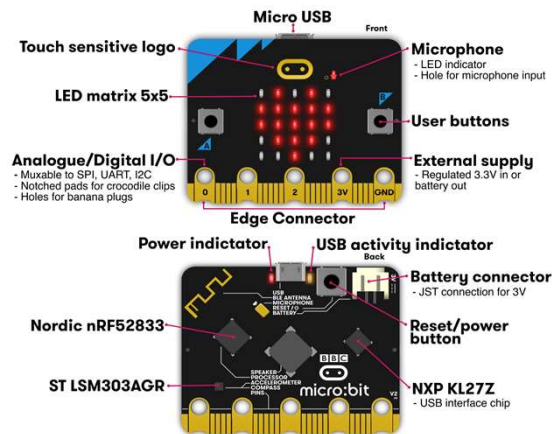
Der er biblioteker der er koblet direkte til den maskine som man kører programmerne på. Det er tilfældet for de specifikke sensorer som findes på Micro:Bit.

De tilgås gennem et bibliotek der hedder Micro:bit, som bruges i det følgende, hvor programmerne bliver eksekveret/udført på Micro:Bitten.

Kode eksemplerne findes under LKS1>Ma>LKS1Ma-Slide8-ex1.py og ..-ex2.py

De kan udleveres til eleverne efter at man har lavet det i Thonny og vist funktionen. De kan så selv læse kommentarerne og prøve at køre det – evt. som hjemmeopgave. De kan så også få en simpel opgave som at udskifte navnene i liste – lave den længere etc.

Spor 1 Modul a: Micro:Bit v2



En kort introduktion til Micro:Bit i version 2, og dens indbyggede sensorer.

- Accelerometer
- Magnetometer (compass)
- Lyssensor
- Mikrofon
- Påmonterede knapper (A,B), samt logo m touch.
- Påmonteret højttaler til at lave lyde
- 5*5 LED's som display.

Version 1 havde ikke monteret mikrofon og højttaler, og logo kunne ikke bruges som knap.

HW reference:

[Hardware \(microbit.org\)](https://hardware.microbit.org)

Spor 1 Modul a: Brug Micro:bit bibliotek

```
1 from microbit import *
2
3 while True:
4     if button_a.is_pressed():
5         display.show(Image.HAPPY)
6     elif button_b.is_pressed():
7         break
8     else:
9         display.show(Image.SAD)
10
11 display.clear()
```

Microbit biblioteket indeholder bl.a. funktioner (også kaldet metoder) der eksempelvis kan fortælle om knap a og/eller knap b er blevet trykket ned/aktiveret, og til at udskrive ting på displayet.

Bemærk at "button_a.is_pressed()" er en funktion der returnerer en værdi som er sand eller falsk (True eller False (None)) som så kan bruges som betingelse i en if ... sætning.

Der er en "break" kommando i python generelt, som kan bruges til at komme ud af en igangværende løkke – Bemærk at betingelsen i while løkken er en konstant True, som jo aldrig vil blive ændret – så hvis ikke break blev brugt ville den aldrig stoppe og komme til den kode som ligger efter while løkken

Prøv med andre billeder/evt. en tekst.

Prøv med display.scroll("En tekst") og se hvad der sker.

Se/Find flere eksempler i MicroBit/MicroPython dokumentationen.

[Buttons — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/en/latest/buttons.html)

Vis dokumentationen for Micropython på Microbit. Vis overordnet hvordan den er struktureret.

- En tutorial del
- En API del med flere detaljer i beskrivelsen
- Et søge felt.

Gå så til siden med buttons – scroll ned til eksemplet som er i slide.

Kopier det ind i Thonny fra dokumentationen ind i editor

Skift fortolker til MicroPython på Micro:Bit (se notat om at komme i gang med Thonny) kød koden.

Lad alle elever gøre det samme!

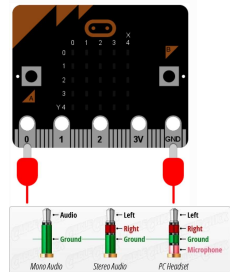
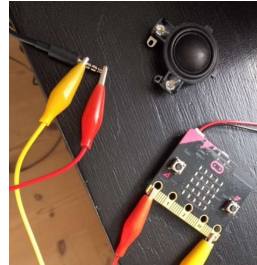
Lad dem lave et par ændringer i koden som beskrevet i Slide.

Spor 1 Modul a: Brug speech og music bibliotek

Projekt støttet af:
midt
regionmidtjylland

```
1 import speech
2
3 speech.say("Hi - Is there anyone out there")
```

```
1 import music
2
3 music.pitch(440,1000)
```



Med version 1 af Microbitten (MB) skal man tilslutte hovedtelefon eller lille højttaler som vist ovenfor. Med version 2 er det ikke nødvendigt da der er monteret højttaler på MB'en.

Prøv speech og music funktionerne til venstre. Find dem i dokumentationen [BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-python.readthedocs.io/)

Ekspirer med andre tekster i speech og andre kald til music. Find inspiration i dokumentationen. Der er for eksempel forud definerede melodier der kan spilles med `music.play(...)`.

Findes i afsnittet "Music" i "Tutorial" sektionen i dokumentationen for MicroPython på MicroBit.

Præsenter eleverne for modulerne/bibliotekerne speech og music
Vis dem søgefunktionen i dokumentationen så de finder referencen.

(klikbare links i slide – og direkte i slide sidst i stakken – MEN bedre at finde dem i dokumentationen på nettet – Bedre at eleverne selv får fornemmelsen for at navigere i dokumentationen og finde det frem som de har behov for.)

Giv dem en opgave med at få sat lyd ind sammen med displayvisningen fra eksemplet med knapper.

Spor 1 Modul a: Udbyg knap eksempel med lyd.

```
1 from microbit import *
2 import music
3
4 while True:
5     if button_a.is_pressed():
6         display.show(Image.HAPPY)
7         music.pitch(440)
8     elif button_b.is_pressed():
9         break
10    else:
11        display.show(Image.SAD)
12        music.stop()
13
14 display.clear()
15 music.stop()
```

Tilføjet at der spilles en tone med frekvens 440 (Kammertonen) når knap a aktiveres – og at tonen stoppes når knap slippes, samt når knap b trykkes ned.

Når der ikke angives flere parametre/argumenter til funktionen "music.pitch(440)" end de 440 så bliver den ved med at spille tonen til den aktivt bliver stoppet med funktionen "music.stop()"

Man kan også kalde music.pitch med flere parametre. Ex. "music.pitch(440,50)" – Det vil resultere i at tonen kun spiller i 50 millisekunder – og så vil "music.stop()" ikke være så vigtig i koden ved siden af – og pitch bliver genstartet så længe knap a er aktiveret – Men det vil lyde anderledes – Prøv det..

PS – det vil fortsat være fornuftigt at bruge "music.stop()" da det sikrer at PIN's på MB er i en veldefineret tilstand når koden afsluttes.

Dette er 'svaret' (et svar) på opgaven med at kombinere det de fandt under buttons med noget lyd.

Skal/kan vises efter opgaven, men naturligvis ikke før.

BEMÆRK – Det er også vigtigt at få udført funktionen music.stop(), for at få ryddet op i PIN tilstand. Hvis ikke kan man risikere at få en fejl hvis/når man bruge pins til noget andet.

Det kan også løses ved at strømmen kommer af Micro:Bitten.

Med eksemplerne i modul a bør man ikke komme i problemer.

Kode eksempel fra slide kan findes i LKS1Ma-Slide12-udbyg-knap-eksempel.py.

Kan evt. udlevere, så elever kan finde tilbage og gense/køre.

Til dem som måtte køre fast og have problemer med syntaks.

Spor 1 Modul a: Find grænsen for hvad pitch kan.

```
1 from microbit import *
2 import music
3
4 for x in range(<start>,<slut>,<step>):
5     music.pitch(x)
6     sleep(100)
7     print(x)
```

Det hørbare område for lyd er 20-20000 Hz

Med music.pitch() funktionen kan I generere toner med frekvenser i området, men når man når under en vis værdi og over en vis værdi kan man ikke længere høre en forskel..

Lav eksperimenter med kode for at finde de værdier

```
1 from microbit import *
2 import music
3
4 for x in range(20,20000,100):
5     music.pitch(x)
6     sleep(100)
7     print(x)
```

Så hvad er den mindste hhv. den største værdi man kan angive som frekvens i pitch funktionen så den er tydeligt anderledes end den/de foregående.

Brug evt. en for-løkke til at eksperimentere med..
Og måske over flere gange for at indsnævre..
Kom med et bud ..

HUSK – Eksperimenter kan man lære af. Brug det til at få en bedre fornemmelse/forståelse af hvad MB'en kan/ikke kan. Eksperimenter er et godt supplement til at læse om tingen. Men begge dele er ikke så tossede. I dokumentationen eller ved at søge på nettet kan I finde inspiration til hvad man kan og så kan jeres egne eksperimenter gøre jer endnu klogere på den specifikke ting.

Kan udelades/Option. Mest som illustration af "for-løkke", så hvis man ikke har fokus på åben bog tilgang (at lære lidt mere om programmering) er det at gå for langt.

Tag en snak om frekvenser og det hørbare område – spørg eleverne om det inden slide.

Højtaler og frekvensgenerator i Micro:Bitten kan ikke lave lyd i hele området, men hvad kan vi lave og høre med den?

Det er opgaven at finde ud af. Med Version 1 af microbitten er der en 'hård' grænse for hvad den kan – Den går simpelt hen ned når man når en bestemt pitch.

Med version 2 er det en anden sag, da den ikke går ned – men man kan så lede efter en værdi, hvor man ikke længere synes at man kan høre forskel mere. Det er så et lidt blødere svar man kan finde.

Men i begge tilfælde kan man bruge eksemplet til at lade eleverne først prøve sig frem, og komme med et svar, og så efterfølgende give dem en måde til at finde det lidt mere struktureret.

Formål overordnet set er at give dem mod på at eksperimentere, for at lære – her for at lære MB'en at kende – efterfølgende for at lære de indbyggede sensorer at kende.

Program findes: LKS1Ma-Slide13-PitchRange.py

Spor 1 Modul a: Lys til lyd.

```
1 from microbit import *
2
3 while True:
4     x = display.read_light_level()
5     print(x)
6     sleep(100)
```

Først eksperimenteres lidt med sensoren, der læser lys niveau.

Lav et simpelt program der læser lys niveau og putter det over i én variabel, der udskrives. Her bruger vi navnet 'x' til variabelen. Det vil være en god ide at bruge et mere sigende navn i længere programmer. Det vil hjælpe med læsevenligheden mht. at forstå koden.

I dokumentationen står at man kan få værdier mellem 0 og 255. Stemmer det overens med det I oplever. Det I får udskrevet i shell med print() funktionen? Prøv med lommelygten.

Det hørbare område er 20-20000 Hz. Ved at eksperimentere med music.pitch() funktionen så vil man opleve at man kan skelne forskel på lyd på MB fra 50 til 4000 Hz.

Så hvad skal der til for at få lys sensor input til at passe med en fornuftig pitch værdi?

Lad eleverne finde "read light level" med søgefunktionen i dokumentationen.

Lad dem køre program fra slide

Lad dem forholde sig til sensorværdierne som læses ud mens de hhv. skygger for display og lyser på det med lommelygte (evt. fra deres mobil telefoner)

-De kan selv taste det ind, men så skal man forvente at der er nogle der får problemer mht. syntaks, så der skal være tid til at komme rundt.

Man kan også udlevere koden færdig så eleverne bare skal loade det og opleve sensor værdien der ændres.

Og så snakke om opgaven med konvertering til lyd og lave den sammen.

Brug lejlighed til at tage en snak om forskellen på at importere biblioteker/moduler vha.

import microbit

Versus

from microbit import *

Der er nemlig lidt inkonsistens i dokumentationen. I tutorial benyttes den sidste, hvor den første bruges i API delen.

Når man bruger den sidste så behøver man ikke lave en fuld reference til modul og metode/funktion. Men det skal man hvis man bruger den første. Den første anses for at være den mest korrekte, da der så ikke kan være tvivl, hvis man har flere moduler med metoder/funktioner med samme navn. Den kræver til gengæld ikke så lange navne – og hvis man opererer med få biblioteker, som her, så kan det være en fordel.

Altså:

```
microbit.display.read_light_level()
```

Versus

```
display.read_light_level()
```

Program findes: LKS1Ma-Slide14-LysSensor.py

Spor 1 Modul a: Lys til lyd.

```
1 # Konvertering af lys intensitet fra MB's indbyggede lyssensor til til lyd (pitch/frekvens)
2 # afspillet på MB's indbyggede højttaler (V2 af MB) eller på ekstern højttaler/hovedtelefon
3 # koblet til PIN 0 og GND.
4 #
5 # Benytter funktionen/metoden read_light_level() fra display i modulet/biblioteket microbit
6 # til at aflæse lys niveau, og pitch funktionen/metoden til at spille en frekvens/lyd fra biblioteket
7 # music, hvorfor de 2 (microbit og music) skal importeres.
8 #
9 # Værdierne fra lysmåler er fra 0 til 255. For at få det til at blive en 'fornuftig' (hørbar) frekvens
10 # ganges lys intensitet med 15 (giver værdier mellem 0 og 15*255 = 3825) og adderer 50 for at komme til
11 # en frekvens der ligger mellem 50 og 3825+50 = 3875. Altså i det hørbare område og i et område som den indbyggede
12 # højttaler på MB'en kan reproducere.
13 #
14 # For at kunne følge lysintensiteten udskrives den til shell med den indbyggede print() funktion
15 # Og for at der skal være lidt tid til at høre lyden og se intensiteten udskrevet ventes der 100 msek mellem
16 # hver omgang i while løkken.
17 #
18 from microbit import *
19 import music
20
21 while True:
22     x = display.read_light_level()
23     music.pitch(x*15+50)
24     print(x)
25     sleep(100)
```

Et svar. Ganger med 15 og adderer 50 til lys niveau for at konvertere til en pitch. HUSK – Der er ikke kun eet rigtigt svar mht. kode!
Det er en god ide at tilføje kommentarer til sin kode. Det kan hjælpe med til at man forstår den bedre når man på et senere tidspunkt kommer tilbage til den.
Også værdifuldt hvis andre skal se og forstå koden. Naturligvis alt med måde ;-) MEN tænk over læseren/modtageren!

Et svar til gennemgang på klassen efter at eleverne selv har prøvet.

Program findes: LKS1Ma-Slide15-LysTilLyd.py

BEMÆRK at program aldrig afsluttes – andet end når det stoppes med stop eller strømmen tages.

Det betyder:

- 1) At man kan putte det ud lokalt på MB og få den til at have funktionen fra der kommer strøm på indtil strømmen fjernes. Vent med at vise det indtil man har været rundt om flere eksempler. Bedst at have programmer gemt lokalt på PC/MAC indtil man er lidt mere tryk med kodedelen.
- 2) At man som tidligere nævnt får efterladt PIN0 i en mode så den ikke kan bruges til andet, og det er så heller ikke noget problem i modul a, men der burde være en music.stop() inden man bruger pin 0 til andet. Det kan man gøre i shell, eller ved at tage strømmen af MB (tage den ud af USB), hvis altså ikke programmet kører lokalt på MB'en (mere om det i Slide 21). Det er IKKE noget man behøver at snakke om her, men god viden at have hvis der pludselig opstår 'sjove' fejl....

Spor 1 Modul a: Magnet felt til lyd.

```
1 from microbit import *
2
3 while True:
4     x = compass.get_field_strength()
5     print(x)
6     sleep(100)
```

Først eksperimenteres lidt med sensoren, der læser magnet feltet.

Lav et simpelt program der læser styrken af magnet feltet og putter det over i én variabel, der udskrives. Her bruger igen navnet 'x' til variablen. Her kan I se at det måske ville være godt med et andet navn – for det er jo IKKE længere det samme som i opgaven med lysintensiteten. Så var det et program med begge ville andre navne være værdifuldt.

I dokumentationen står der IKKE noget om hvilke værdier der kan komme tilbage fra funktionskaldet. Kun at det er i nano tesla. Så der er kun en vej til at finde ud af det. Prøv/Eksperimenter med en magnet omkring Micro:Bitten!

Hvad skal der til for at få magnet felt værdi omsat til en pitch der kan håndteres af funktionen music.pitch()?

HUSK – Eksperimenter kan man lære af. Brug det til at få en bedre fornemmelse/forståelse af hvad MB'en kan/ikke kan. Eksperimenter er et godt supplement til at læse om tingen. Men begge dele er ikke så tossede. I dokumentationen eller ved at søge på nettet kan I finde inspiration til hvad man kan og så kan jeres egne eksperimenter gøre jer endnu klogere på den specifikke ting.

Kan udelades, hvis man kun vil give grundlaget for at kunne komme i gang med Spor 2.

Man kan gå fra at have lavet koden med lys til lyd til at vise hvordan man får koden lagt på MB'en (slide 21) , så den kan køre for sig selv, og derfra gå over til at fortælle om øvrige sensor funktioner og derfra til Spor 2, hvor man så fortæller om de forskellige typer af kode som er lavet og som kan loades på MB'en.

Tilsvarende som med lys, men nu med magnetfelt.
Nyt programmeringsteknisk – operators and casting.

Udlever en magnet til eleverne.

Lad eleverne finde get_field_strength() med søgefunktionen i dokumentationen. (Der er også en klikbar link på kodelistump i slide – og til sidst i slide stak)

Lad dem køre program fra slide

Få dem til at forholde sig til de værdier de kan få fra sensoren, så de kan få en fornemmelse for hvad det er for en størrelse.

For dem der hurtigt er færdige kan man lade dem udskifte kaldet compass.get_field_strength() med funktionerne:
compass.get_x(), compass.get_y() og compass.get_z()

Og få en fornemmelse for hvad forskellen er.

Og så som med lys få dem til at lave en konvertering fra magnet felts værdier til lyd.

Pas på med magnet på den indbyggede højttaler – Det kan få den slået ud af kurs!!

Så magnet rundt på og omkring MB's forside. Slås den ud af kurs kommer den sandsynligvis tilbage efter at strømmen har været af.

Program findes: LKS1Ma-Slide16-Magnetometer.py

Spor 1 Modul a: Magnet felt til lyd.

```
1 # Konvertering af magnetfelts værdier fra MB's indbyggede magnetometer til til lyd (pitch/frekvens)
2 # afspillet på MB's indbyggede højttaler (V2 af MB) eller på ekstern højttaler/hovedtelefon
3 # koblet til PIN 0 og GND.
4 #
5 # Benytter funktionen/metoden get_field_strength() fra compass i modulet/biblioteket microbit
6 # til at aflæse magnet feltets styrke, og pitch funktionen/metoden til at spille en frekvens/lyd fra biblioteket
7 # music, hvorfor de 2 (microbit og music) importeres.
8 #
9 # Værdierne fra magnetometer er altid positive og kan gå op til ihvertfælde 2.5 mill.
10 # For at få det til at blive en 'fornuftig' (hørbar) frekvens divideres med 1000, og adderer 50.
11 # MEN skal desuden få det til at være et heltal' og ved normal division (/) får vi decimaler.
12 # - derfor bruges funktionen int() til at konvertere til heltal!
13 #
14 from microbit import *
15 import music
16
17 # Da felt styrken ser ud til at kunne nå op til ca. 2.5 mill og altid er positiv
18
19 while True:
20     x = compass.get_field_strength()
21     print(x)
22     music.pitch(int(x/1000)+50)      # Kan også bruge floor division // - Se python aritmetiske operatorer
23     sleep(100)
```

Et svar. Deler med 1000 og adderer 50 til magnet felt værdi for at konvertere til en pitch MEN skal også lave det til heltal (ikke decimaltal), da funktionen music.pitch() kun forstår positive heltal!
Igen - HUSK – Der er ikke kun eet rigtigt svar mht. kode!

[Python Operators \(w3schools.com\)](https://www.w3schools.com/python/python_operators.asp) (afsnit om aritmetiske operatorer) eller [Python Casting \(w3schools.com\)](https://www.w3schools.com/python/python_casting.asp) (int()) funktionen til konvertering til heltal)

Kan udelades, hvis man kun vil give grundlaget for at kunne komme i gang med Spor 2.

Program findes: LKS1Ma-Slide17-MagnetTilLyd.py

Spor 1 Modul a: Accelerometer til lyd.

```
1 from microbit import *
2
3 while True:
4     acc_x, acc_y, acc_z = accelerometer.get_values()
5     # acc_all = accelerometer.get_values()      # alternativ - tuple/sæt af værdier
6     print(acc_x)
7     # print(acc_all[0])                        # alternativ - udpeg i tuple
8     sleep(100)
```

Først eksperimenteres lidt med sensoren, der læser g-værdier fra accelerometeret .

Hvad skal der til for at få magnet felt værdi omsat til en pitch der kan håndteres af funktionen music.pitch()?

Læs evt. om tuples:
[Python Tuples \(w3schools.com\)](https://www.w3schools.com/python/python_tuples.asp)

Kan udelades, hvis man kun vil give grundlaget for at kunne komme i gang med Spor 2.

En gang mere – Denne gang med accelerometer.

Find i dokumentation – søg på "accelerometer".

Eksperimenter med sensor, så man får en fornemmelse. Vip med den.

Konverter til lyd.

Bemærk at dokumentationen siger at man får værdier i området +/- 2000, men i test så kan man få værdier der er større end 2000 – Det sker ved et 'kraftigt' stød.

Men ved et normalt vip får man værdier der ligger i området +/- 1100. Den viden skal man så bruge i konverteringen efterfølgende.

Ny ting programteknisk – Tuples og/eller flere variable retur i eet kald, samt abs() funktionen i math til at lave en negativ værdi om til en positiv.

Program findes: LKS1Ma-Slide18-Accelerometer.py

Spor 1 Modul a: Accelerometer til lyd.

```
1 # Konvertering af g-kraftspåvirknings værdier fra MB's indbyggede accelerometer til til lyd (pitch/frekvens)
2 # afspillet på MB's indbyggede højttaler (V2 af MB) eller på ekstern højttaler/hovedtelefon
3 # koblet til PIN 0 og GND.
4 #
5 # Benytter funktionen/metoden get_values() fra accelerometer i modulet/biblioteket microbit
6 # til at aflæse kraftpåvirkningen, og pitch funktionen/metoden til at spille en frekvens/lyd fra biblioteket
7 # music, hvorfor de 2 (microbit og music) importeres.
8 #
9 # Værdierne fra accelerometer i x-aksen er ifølge specifikationen +/- 2000, men kan måle værdier over 2000 ved
10 # kraftig påvirkning. Ved 'normal' vip højre/venstre får man ca. værdier i området +/- 1100.
11 # For at få det til at blive en 'fornuftig' (hørbar) frekvens skal det først og fremmest være positivt. Det kan
12 # man få det til at blive med abs() funktionen, men så kan man ikke skelne højre versus venstre vip. Derfor
13 # adderes istedet 1500 og så for at være sikre på at det kommer indenfor en fornuftig range testes på værdi inden
14 # kaldet til music.pitch() - MEN det kan laves på andre måder!! De 3907 kommer fra at version 1 af MB går ned hvis
15 # man bruger en frekvens der er 3907 eller større. Det er ikke tilfældet for version 2!!
16 #
17 from microbit import *
18 import music
19
20 while True:
21     acc_x, acc_y, acc_z = accelerometer.get_values()
22     print(acc_x)
23     # music.pitch(abs(acc_x)) # den hurtige løsning - abs() laver neg til pos
24
25     freq = acc_x+1500 # adderer tilstrækkelig stor værdi
26     if freq>50 and freq<3907: # tester at det er i området for at være sikker
27         music.pitch(freq)
28     sleep(50) # Venter kun 50msek (hvor der i andre eksempler har stået 100
29             # Det er for at give en lidt mere 'flydende' oplevelse - Prøv forskellen!
```

Læs evt.: [Python Math \(w3schools.com\)](https://www.python-math.com/) om abs() funktionen.

Kan udelades, hvis man kun vil give grundlaget for at kunne komme i gang med Spor 2.

Ny ting programteknisk – Tuples og/eller flere variable retur i eet kald, samt abs() funktionen i math til at lave en negativ værdi om til en positiv.

MEN – med abs() får vi ikke muligheden for at lave en lav frekvens ved vip til venstre og høj ved vip til højre – så derfor den anden løsning.

Kan bruge eksemplet til at snakke om normalisering af data, men her er der bare en enkelt løsning.

Program findes: LKS1Ma-Slide19-AccTilLyd.py

Spor 1 Modul a: Mikrofon måling.

```
1 # Udlæsning af lyd niveau fra indbygget mikrofon.  
2 from microbit import *  
3  
4 while True:  
5     s = microphone.sound_level()  
6     if s>0: print(s)  
7     sleep(10)
```

[Microphone V2 — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](#)

Eksperimenter med sensoren, der udlæser lyd tryk.

Hvad skal der til for at give udslag af en vis størrelse?

Hvad kan man bruge en måling af lyd tryk til?

Kan udelades, hvis man kun vil give grundlaget for at kunne komme i gang med Spor 2.

For at vise den sidste indbyggede sensor - mikrofon

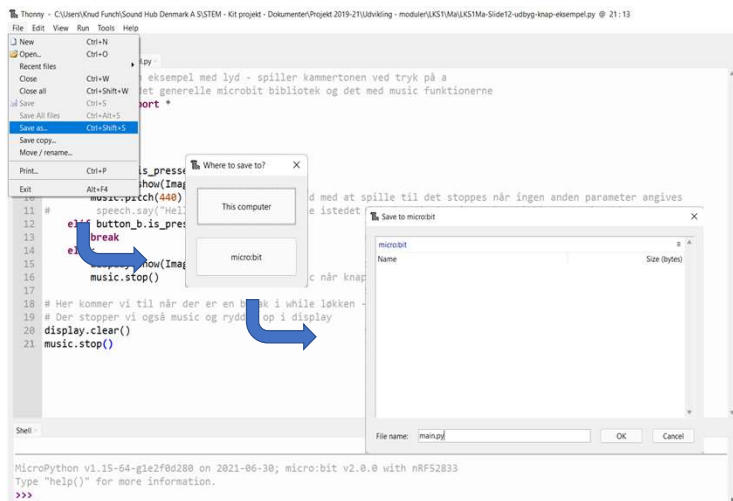
Find i dokumentation – søg på "sound_level".

Eksperimenter med sensor, så man får en fornemmelse.

Her giver det ikke så meget mening at konvertere til lyd, så det springes over – men man kan jo tage en snak om hvad en støj måling kan bruges til.

Program findes: LKS1Ma-Slide20-Microphone.py

Spor 1 Modul a: Gemme program lokalt på MB



Man kan gemme et program lokalt på Micro:Bitten og ved at tilslutte en batteripakke så kan det program køre lokalt uden at MB er tilsluttet PC/MAC.

Det gør man ved at stoppe evt. program som er i gang! Med stop! Sikre at Shell står klar som vist til venstre.

Så gå til file>save as og så vælge Micro:Bitten. Man skal så gemme program med navnet "main.py" (alle bogstaver **SKAL** være små).

Det program startes så snart MB får strøm.

Når I så efterfølgende skal gennem noget/eller have anden kode til at køre på den MB skal I først stoppe programmet med STOP og sikre jer at I har prompt i shell som vist. Ellers kan I få div fejl meldinger/eller ikke kunne gemme etc.

Udlever batteriholder til elever som kan kobles til MB.

For at gemme et program lokalt på Micro:Bitten (så det kan køre når der kommer strøm på uden at være koblet til PC/MAC) skal man:

- 1) Sikre sig at der IKKE er noget program som kører på MB – Brug stop – Skal se ">>>" og "MicroPython v1..." i Shell!
- 2) Gå ind via menu'en Filer og vælge save as, og så vælge microbit – Hvis ikke der står microbit er der noget galt – se 1)
- 3) Gemme under navnet "main.py" SKAL være med små bogstaver.

Det er den grundlæggende ting man skal vide for at kunne load færdig kode ned på MB's ifm Spor 2.

Og så at kunne skifte tilbage til at køre fortolkeren på PC/MAC/PI fremfor på MB.

BEMÆRK – Ved at gemme et af de programmer der laver sensor input om til lyd på MB'en. Ex LysTilLyd så kommer der lyd ligeså snart der er strøm på MB'en
Også når man kobler den tilbage på PC/MAC via USB kablet! Det kan godt give nogle stressede situationer! ;-) Lyden bliver også ved selvom man får stoppet program med stop og har prompt i shell!

- Man skal så skynde sig at køre et program fra Thonny og stoppe det
- Man kan bruge `music.stop()` kald direkte fra fra shell, men skal så huske at lave en `import music` først (også bare i shell)
- Man kan reload micropython fortolker fra run menu, men det er et lidt voldsomt skridt.

Og så kan man jo tage en snak om at man evt. skal lave programmet så det kan stoppe på 'ordentlig vis' ved et knap tryk etc.. ..

Hvis man vil undgå den slags så kan man bruge det første program fra slide 12 som det man gemmer!

Den korteste vej, hvis man ønsker at køre lukket bog og videre i spor 2 vil være:

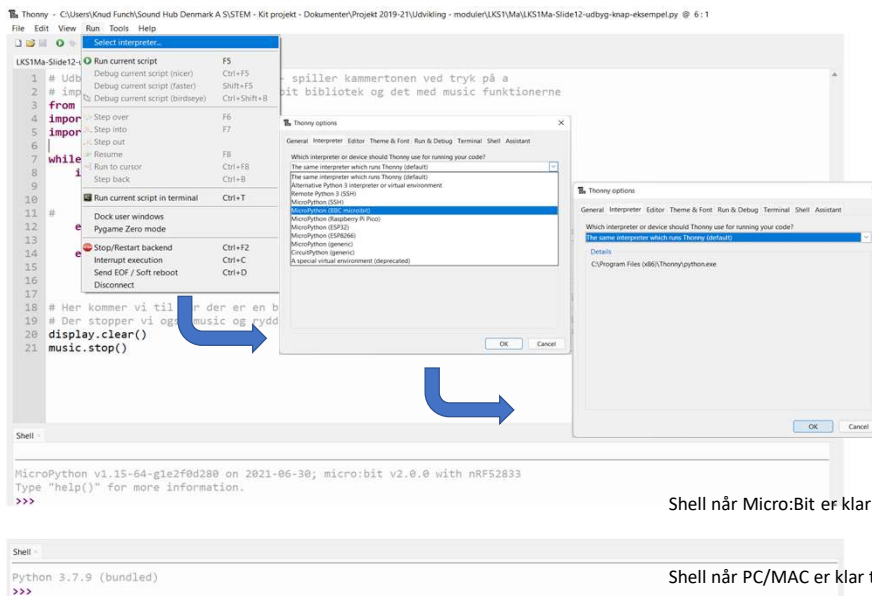
- Introduktion til (IDE) Thonny – se notat om Thonny
- Introduktion til programmering md fortælling og demo -Slide 4 og 6
- Introduktion til MB – Slide 9,10 og 12
- Gemme program – Slide 21 med kode fra Slide 12.

Vil man klæde eleverne på til at kunne gå åben bog går man dybere, samt tilbyder muligheden for Modul B

Supplerende Materiale

Skift mellem fortolkere. På PC/MAC skift til på Micro:Bit
Installation af MicroPython fortolker på Micro:Bit
Sammenligning MakeCode Blokprogrammering og MicroPython
Sammenligning MicroPython og Arduino C
Samling af links til materiale, som der henvises til undervejs.

Spor 1 Modul a: skift mellem fortolkere



Sekvens når man skifter mellem Fortolkere (maskiner) at udføre Python/Micropython programmer på.

Shell når Micro:Bit er klar til at eksekvere et MicroPython program

Shell når PC/MAC er klar til at eksekvere et Python program

Bedst at demonstrere med Thonny og Micro:bit tilkoblet, men som oversigt til at forklare er der her en slide.

Det er en væsentlig forudsætning for at bruge LYD-Kit (alle spor) at man kan skifte mellem at køre/loadere programmer på Micro:Bitten og på PC/MAC/PI – At skifte mellem MicroPython fortolkeren på Micro:Bitten og Python fortolkeren på PC/MAC/PI. Derfor er det vigtigt også i en 'lukket bog' tilgang at få gjort denne forskel tydelig.

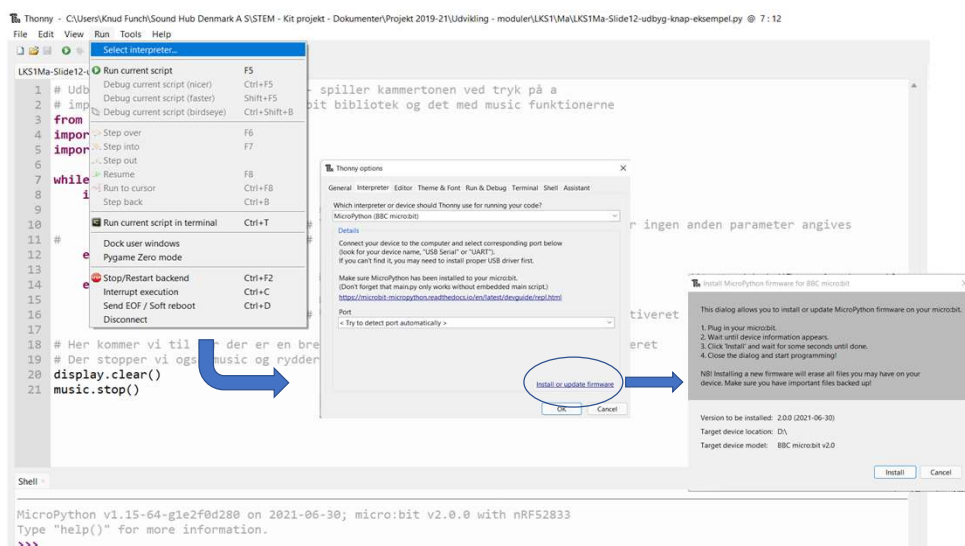
Den væsentligste feedback med Thonny som IDE er gennem prompt. og fortolker udmelding i Shell.

Når der er >>> er fortolkeren klar – og der er ikke et program som er aktivt

Når der står: "Python" er det fortolkeren på PC/MAC/PI der er klar.

Når der står: "MicroPython" er der fortolkeren på Micro:bitten som er klar.

Spor 1 Modul a: Installation af MicroPython på Micro:Bit



For at gøre Micro:bitten klar til at køre MicroPython programmer skal der installeres en ny firmware – (MicroPython fortolker) i stedet for den kode som ligger i Micro:Bitten fra fabrikken, som kan håndtere blokprogrammer fra microsoft Makecode. Det bliver gjort med sekvensen vist i Slide, men igen så er det bedre at demonstrere det med Thonny og Microbit koblet til USB stikket.

Husk USB kabel – og at der er adapter der passer på PC/MAC siden.

BEMÆRK:

- En MicroBit af fabrik kommer med en lille startsekvens som fortæller om dens egenskaber når der kommer strøm på. Det kan man godt udnytte til at fortælle om den inden man skifter firmware.
- Makecode editor viser at man kan lave python programmer – Det er ikke rigtig MicroPython, men kun pseudo kode så på ingen måde sammenlignelig!!

Spor 1 Modul a: Makecode python vs MicroPython

The Python in the MakeCode editor is different to the MicroPython in the [micro:bit Python Editor](#). Although both editors appear like Python 3 they actually are completely different in the way their programs are interpreted and run.

This explains why you may find programs that run fine in the MicroPython editor cause bugs in the MakeCode editor, or vice versa. For example, our simple "Hello, World" program in MakeCode Python would appear as:

```
basic.show_string("Hello!")  
basic.show_icon(IconNames.HEART)
```

Where as in MicroPython you would have to import modules first, and then call different classes as follows:

```
from microbit import *  
  
while True:  
    display.scroll('Hello, World!')  
    display.show(Image.HEART)  
    sleep(2000)
```

MakeCode Python

MakeCode Python runs the micro:bit DAL (Data Abstraction Layer) just as the other languages in MakeCode do. It is in fact just static TypeScript surfaced with Python syntax. This is the same structure as the blocks in the MakeCode editor, which is why it is possible to switch between the different languages. However, this does mean that there are certain limitations to MakeCode Python, as it shares the same limitations as the other languages in MakeCode. One example of this would be the limitation to access of only 3 PWM pins on the micro:bit. You can access the MakeCode Python API in the documentation <https://makecode.microbit.org/reference>. This shows how the different functions map on to each other across languages.

Micropython

MicroPython is different to this, as this has its own runtime on the micro:bit. MicroPython is in fact a tiny python interpreter that runs on the board (at a low level) and is a popular programming language because it is optimised to work on microcontrollers — just like the micro:bit. It is almost a full re-implementation of python 3 but is designed to be able to run in a low memory and low power environment. The python editor provided by micro:bit is the perfect environment for building programs to run on your micro:bit, as you can easily access the REPL (Read, Evaluate, Print, Loop) and also connect directly to your micro:bit to flash programs via serial connection. Furthermore, there is an easy to use file system where you can drag in modules you have created. You can access the MicroPython API documentation here <https://microbit-micropython.readthedocs.io/en/latest/>.

[MakeCode Python and MicroPython : Help & Support \(microbit.org\)](#)

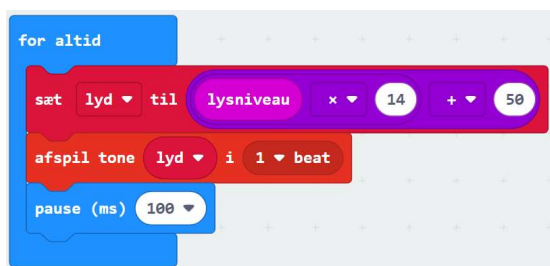
Dette er med MicroPython fortolker fra

[BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](#)

Og brugt via Thonny IDE'en som pt (2020/2021) er den som understøttes af Raspberry PI Foundation.

Bemærk forskel til den Python som er med i Makecode fra MicroSoft!!

Spor 1 Modul a: Makecode (Blok) versus MicroPython (Tekst)



```
from microbit import *
import music

while True:
    lyd = display.read_light_level()*14+50
    music.pitch(lyd)
    sleep(100)
```

MicroPython udgaven her er med MicroPython fortolker fra [BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-python.readthedocs.io)

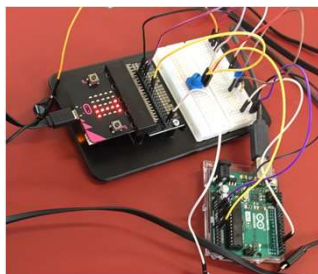
Og brugt via Thonny IDE'en som pt (2021/2021) er den som understøttes af Raspberry PI Foundation. [Thonny, Python IDE for beginners](https://thonny.org)
Bemærk forskel til den Python som er med i Makecode fra Microsoft!

Lyd ændres mere glidende i python udgaven, da den kan sættes til at afspille i baggrund indtil ny.

Pausen i blok kan udelades og så bliver det lidt mere glidende i dette eksempel, men ikke som Python udgaven.

Så kode er næsten ens men ikke helt samme resultat. (blokprogrammeringen har ikke samme grad af muligheder). Men med Blokprogrammering undgår man syntaksfejl, så det er lettere at gå til. Omvendt er Python et sprog, der er forholdsvis let at gå til og som skalerer til 'professionelt' brug i mange domæner.

Spor 1 Modul a: Arduino versus MicroPython



```
Thonny - C:\Users\KIT\MicroPython_for_microbit\Software\readPot.py @ 10:102
File Edit View Run Tools Help

readPot.py
1 # Reading a Potentiometer and set LED intensity accordingly
2
3 from microbit import *          # Import the basic microbit library
4
5
6 while True:
7     a = pin2.read_analog()      # read the analog value from potentiometer connected to pin 2 - value: 0-1023
8     print("Analog read pin 2 ",a) # print the read value for debug purpose in shell
9     pin0.write_analog(a)        # write the analog value to pin where LED is connected - here pin 0 - value: 0-1023
10    sleep(50)                   # Pause program for 50 msec - That's a fair latency in this use case
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
26
```

stor sandsynlighed for at eleverne vil kunne få gælde af Python indenfor mange STEM relateret fag i deres videre uddannelse.

- 4) Det virker til at der kommer mere og mere 'Python' (MicroPython) på mikrokontrollere eftersom HW (CPU'en) bliver kraftigere og kraftigere på diverse kommercielt tilgængelige HW til DIY folket.

MEN når eleverne når til at ville mere avancerede ting med Micro Kontrollere så er det ikke et valg mellem det ene eller det andet. Så er det både-og. Valget vil blive betinget af hvad man ønsker at lave. Og det er valg af både HW og SW.

Spor 1 Modul a: Reference links

[Thonny, Python IDE for beginners](#)

[Python Tutorial \(w3schools.com\)](#)

[BBC micro:bit MicroPython documentation — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](#)

[Stack Overflow Developer Survey 2021](#)

[Top 15 Programming Languages by Popularity \(2004-2019\) – YouTube](#)

[PYPL Popularity of Programming Language index](#)

[Python While Loops \(w3schools.com\)](#)

[Python Conditions \(w3schools.com\)](#)

[Python Syntax \(w3schools.com\)](#)

[Python Lists \(w3schools.com\)](#)

[Python For Loops \(w3schools.com\)](#)

[Python Functions \(w3schools.com\)](#)

[Python Modules \(w3schools.com\)](#)

Spor 1 Modul a: Reference links

[Python Random Module \(w3schools.com\)](https://www.w3schools.com/python/random.asp)

[Random Number Generation — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/en/latest/tutorial/03-random-number-generation.html)

[Hardware \(microbit.org\)](https://microbit.org/micro/python/hardware/)

[Buttons — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/en/latest/tutorial/04-buttons.html)

[Music — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/en/latest/tutorial/05-music.html)

[Speech — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/en/latest/tutorial/06-speech.html)

[Display — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/en/latest/tutorial/07-display.html)

[Compass — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/en/latest/tutorial/08-compass.html)

[Python Operators \(w3schools.com\)](https://www.w3schools.com/python/operators/)

[Python Casting \(w3schools.com\)](https://www.w3schools.com/python/python_casting.asp)

[Accelerometer — BBC micro:bit MicroPython 1.0.1 documentation \(microbit-micropython.readthedocs.io\)](https://microbit-micropython.readthedocs.io/en/latest/tutorial/09-accelerometer.html)

[Python Tuples \(w3schools.com\)](https://www.w3schools.com/python/python_tuples.asp)

[Python Math \(w3schools.com\)](https://www.w3schools.com/python/python_math.asp)

[MakeCode Python and MicroPython : Help & Support \(microbit.org\)](https://microbit.org/micro/python/help/)