Gode råd, links og ressourcer

- når du arbejder med programmering i folkeskolen

Udarbejdet af Eva Petropouleas Christensen,
CFU Absalon

CFU

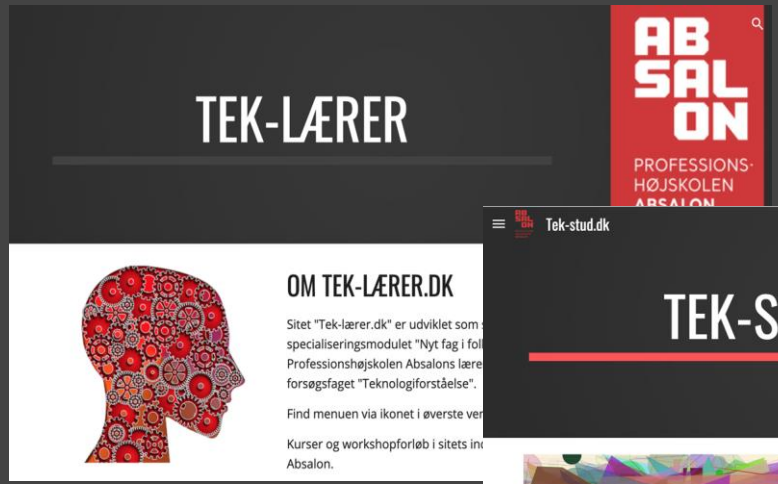
Center for Undervisningsmidler

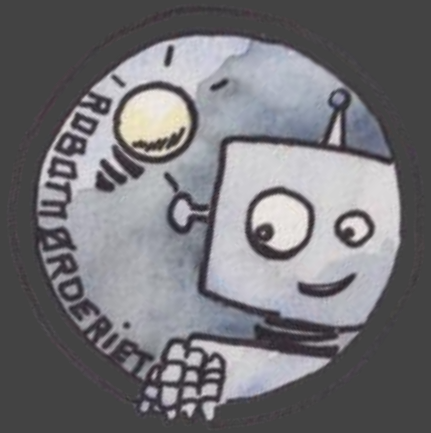
- ❏ sparring og vejledning
- ❏ temadage
- ❏ fagteammøder
- ❏ kurser
- ❏ udlån

Alle CFU-afdelinger har pædagogiske it- og mediekonsulenter, som gerne vejleder i forhold til mulighederne i netop din region.

https://arkiv.emu.dk/modul/cfu-centre-undervisningsmidler-i-danmark#

Websites

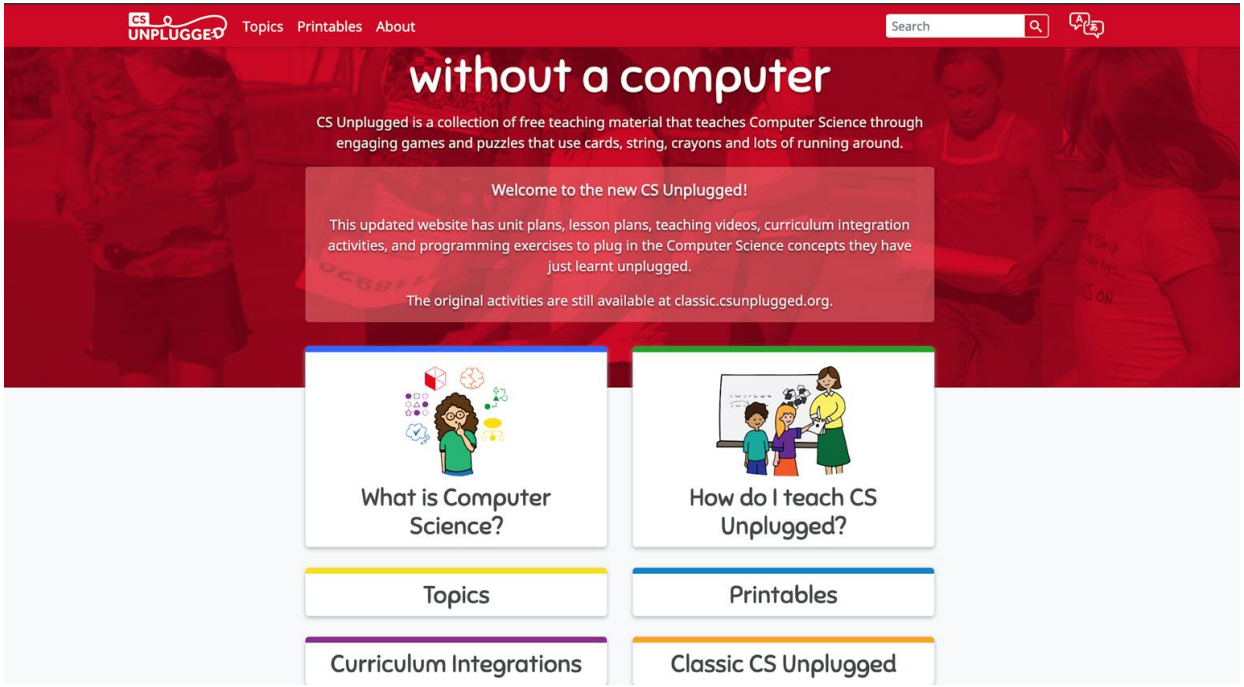




www.tek-lærer.dk www.tek-pæd.dk www.tek-stud.dk www.robotnørderiet.dk

“Tør-programmering” og lign.

Se fx ComputerScienceUnplugged’s ressourcer: https://csunplugged.org/en/



Små videoer fra Khan Academy

Find dem + små øvelser via de to links nedenfor:
https://sites.google.com/pha.dk/tek-laerer/tematiske-forl%C3%B8b-ressourcer/internet-websites-sikkerhed-privatliv/internettet
https://sites.google.com/pha.dk/tek-laerer/tematiske-forl%C3%B8b-ressourcer/internet-websites-sikkerhed-privatliv/sikkerhed

Hour of Code

Hour of Code er en event, som foregår en uge om året, og som arbejder på at få så mange børn og unge som muligt til at bruge 1 time på at kode. Derfor er der udviklet en række små tutorials, som kan bruges – både under eventet og resten af året – de er gratis og af forskellig sværhedsgrad, så alle aldersgrupper kan være med.

https://code.org/learn

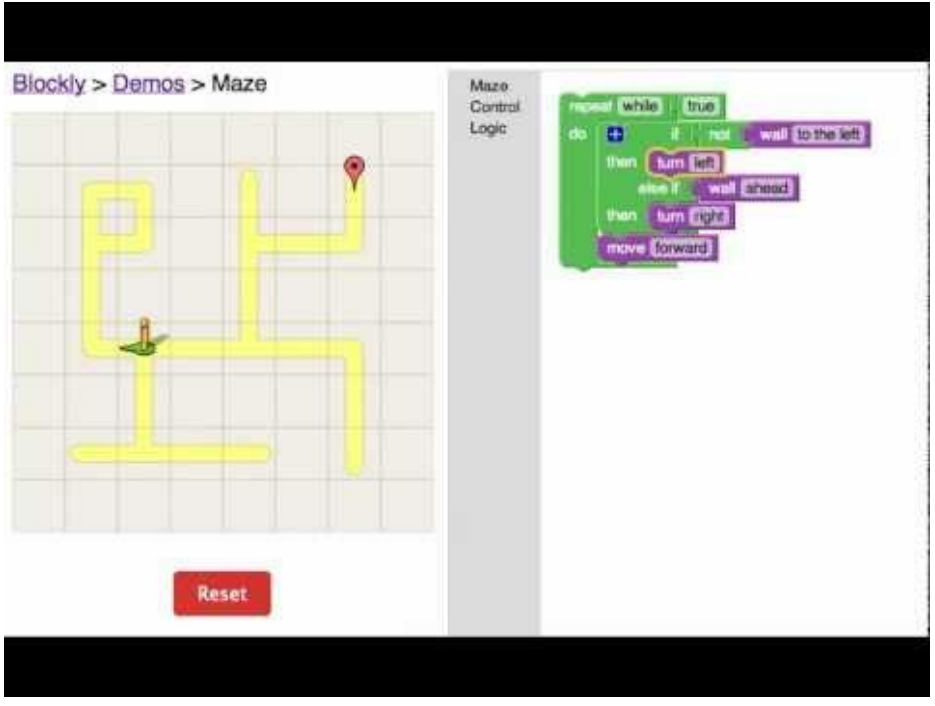
EU Code Week

EU Code Week minder lidt om Hour of Code, men her kan man også deltage på andre måder end ved at lave en times kodning. Derudover kan man – som med Hour of Code – bruge de forskellige ressourcer hele året.

http://codeweek.eu/resources/ Scroll ned på siden – her er der en samling rigtig fine links til diverse platforme (på engelsk).

Siden er nok mest henvendt til jer som undervisere.

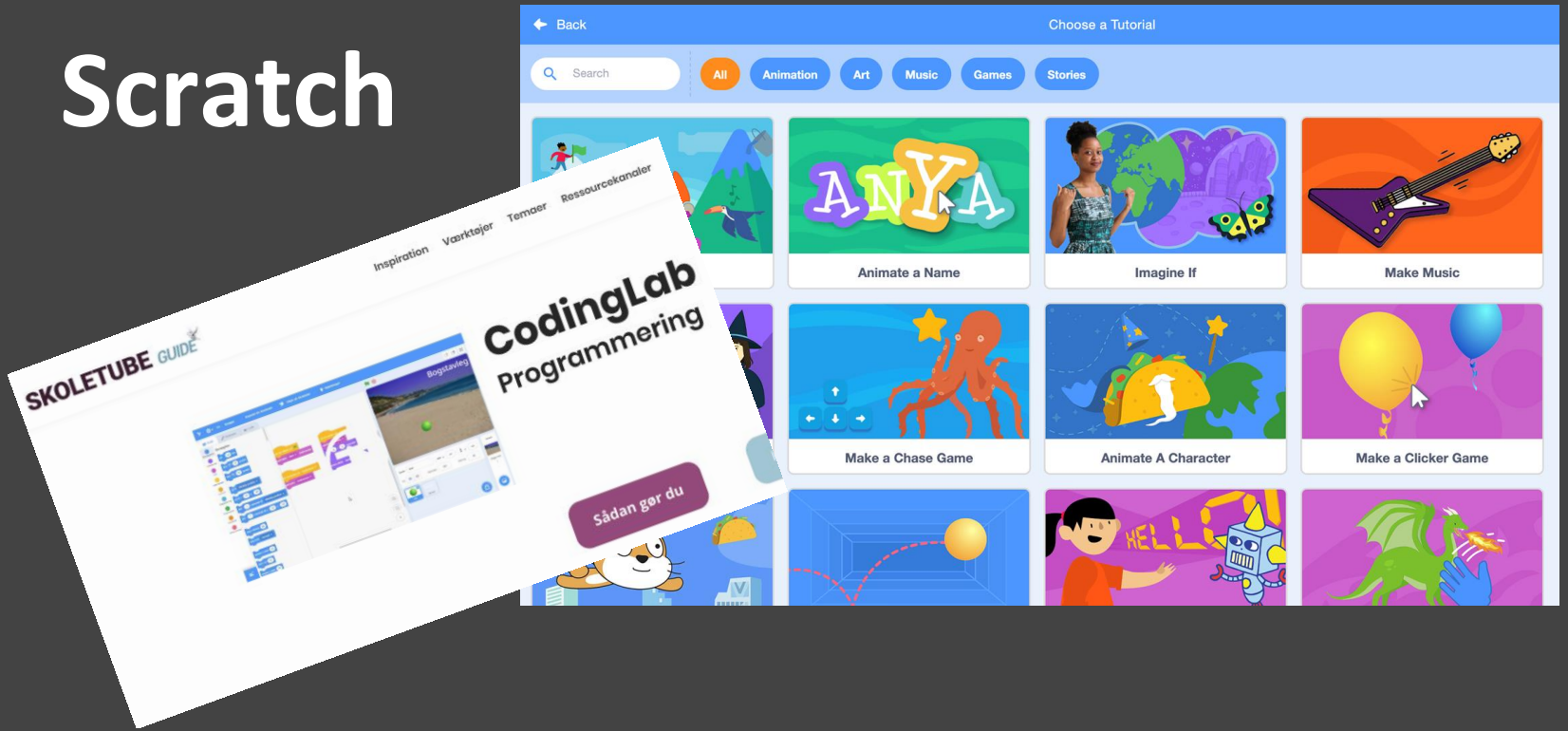
Blockly



Klik på linket her for at komme til en række små opgaver, der introducerer Googles blok-programmeringssprog ”Blockly”

https://blockly-games.appspot.com/

Scratch



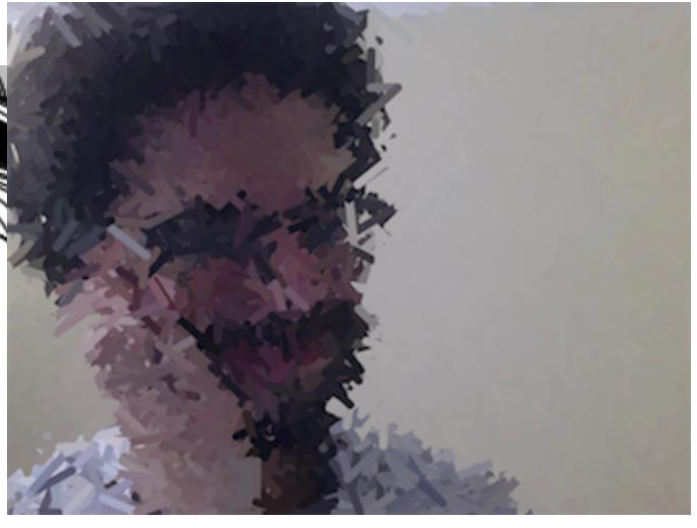
Scratch er udviklet af MIT og rummer rigtig mange muligheder uanset alder. Der findes en række tutorials og hjælperessourcer inde i programmet, som du kan bruge til at komme godt i gang. Klik på linket her: https://scratch.mit.edu/

OBS: Man skal være 13 for at lave sin egen bruger. Alternativt skal der gives forældretilladelse eller I skal lave en lærerkonto med tilhørende elevprofiler.Har I licens til Skoletube, så kan I bruge Scratch her. På skoletube hedder det CodingLab , men det er det samme program.

Udarbejdet af Eva Petropouleas, CFU Absalon

Processing

Udviklet primært til at lave digital kunst, men kan anvendes til rigtig mange grafiske projekter. Nemt at komme i gang med, men rummer også mulighed for ret avancerede projekter. På engelsk. Se de forskellige tutorials her: https://processing.org/tutorials/

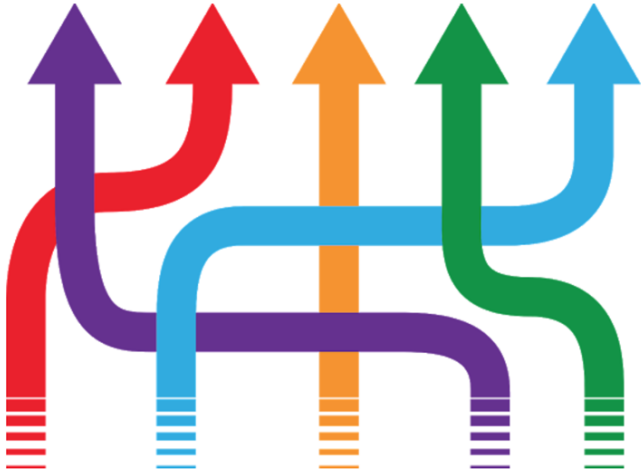


Sprogteknologi

Sprogteknologi er et særligt forskningsfelt, som grundlæggende arbejder med at få computere til at genkende, forstå, tolke, producere og efterligne det menneskelige sprog i dets forskellige former, skrevet såvel som talt. Eksempler på områder, som man kan arbejde med i grundskolen, kan fx være chatbots, interaktive fortællinger og autogenererede digte og sange. Klik på billederne eller links'ene nedenfor for at komme til undersiderne:

CHATBOTS

KLIK PÅ BILLEDE FOR AT KOMME TIL UNDERSIDE

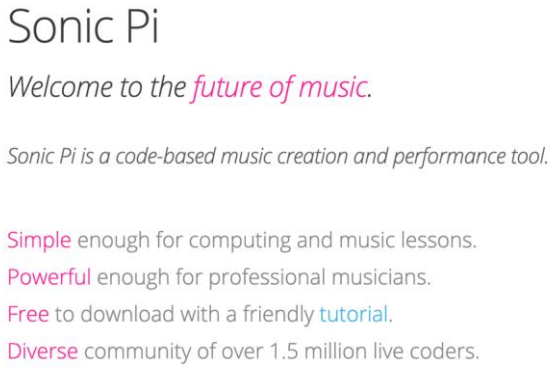


INTERAKTIVE FORTÆLLINGER

KLIK PÅ BILLEDE FOR AT KOMME TIL UNDERSIDE

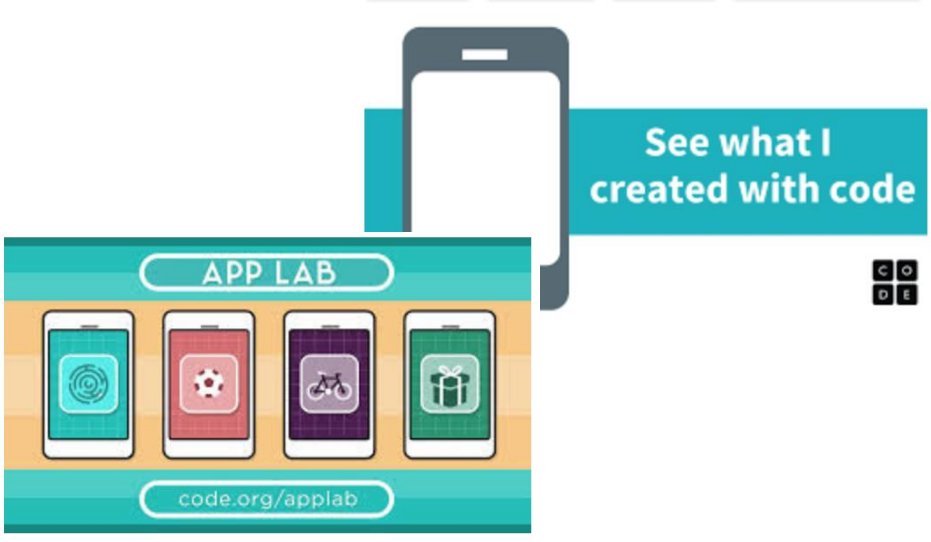
Programmering af musik

Der er indbygget en tutorial i programmet, som er ret nem, men lang og på engelsk. På linket nedenfor kan hentes en bearbejdet og forkortet version på dansk.



https://sites.google.com/pha.dk/tek-laerer/tematiske-forl%C3%B8b-ressourcer/computationel-kreativitet/programmering-af-musik

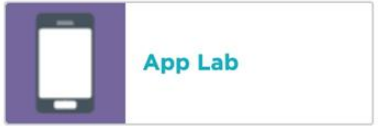
App Lab m.m.



Program til at udvikle apps fra Code.org. Det bygger på blokprogrammering, men der er også mulighed for at arbejde med tekst. Der er en god tutorial til at komme i gang her: https://code.org/educate/applab

App Lab er ikke bundet til et bestemt device, og er rigtig fin, hvis eleverne allerede er fortrolige med blokprogrammering. **Den helt store fordel er, at eleverne kan dele deres apps som en URL. Når modtageren klikker på denne, åbner deres program op og fungerer som en rigtig app.**

Se også de øvrige tutorials fra code.org. Udover nedenstående har de også udviklet "web lab" til websites.

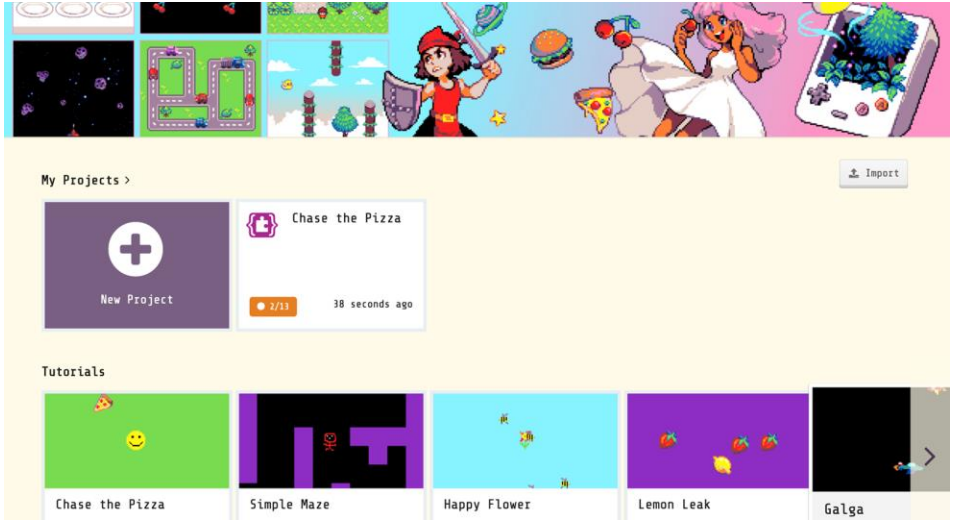


App Inventor

Endnu en måde at arbejde med udvikling af apps. Der er hjælp til at komme i gang og forskellige tutorials, du kan bruge, til at lære programmet at kende: http://appinventor.mit.edu/explore/index-2.html

Programmet er lidt kompliceret og man kan kun arbejde på apps til Android – programmet giver dog mulighed for at afprøve sine apps på en emulator, hvis man ikke har en Android-telefon.

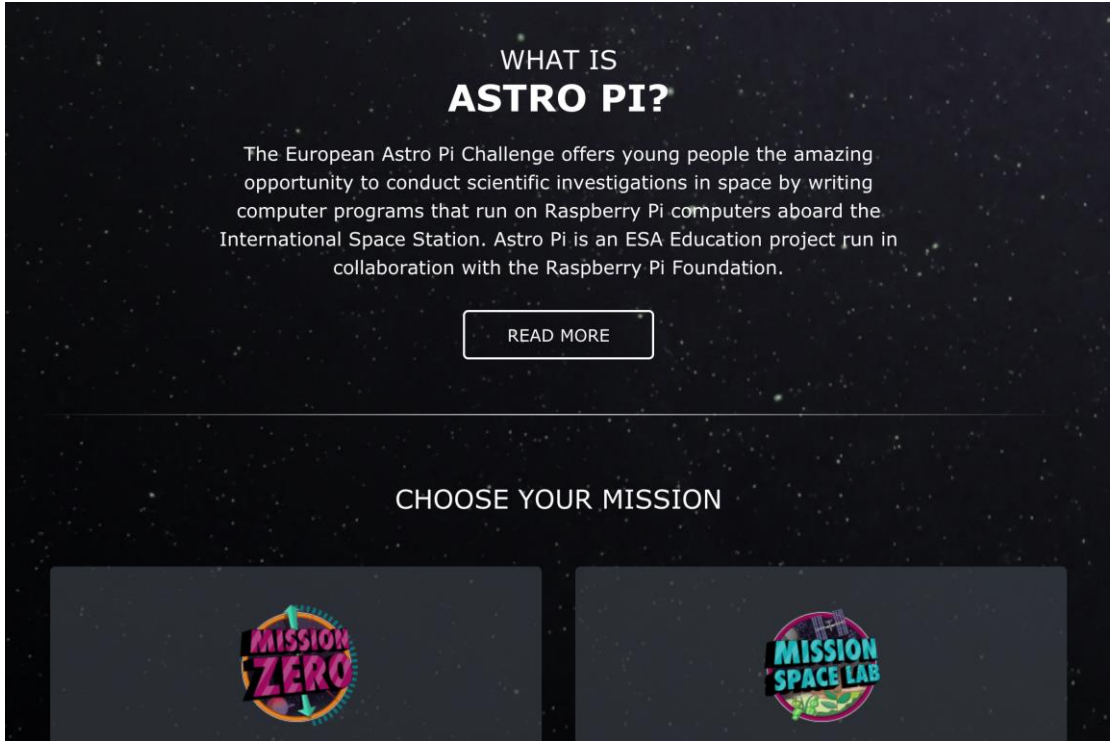
makecode.arcade



Makecode.arcade er Microsofts nye platform til at lave spil og apps. Der er en række tutorials til at komme i gang, som viser dig, hvordan du gør. Du finder det via dette link, hvor du også kan klikke dig videre til hjælperessourcerne:
https://arcade.makecode.com/

Astro pi

Her kan eleverne på to niveauer arbejde med at skrive kode, som bliver afviklet på en virkelig rumstation. Læs mere her: https://astro-pi.org/



Nemprogrammering.dk

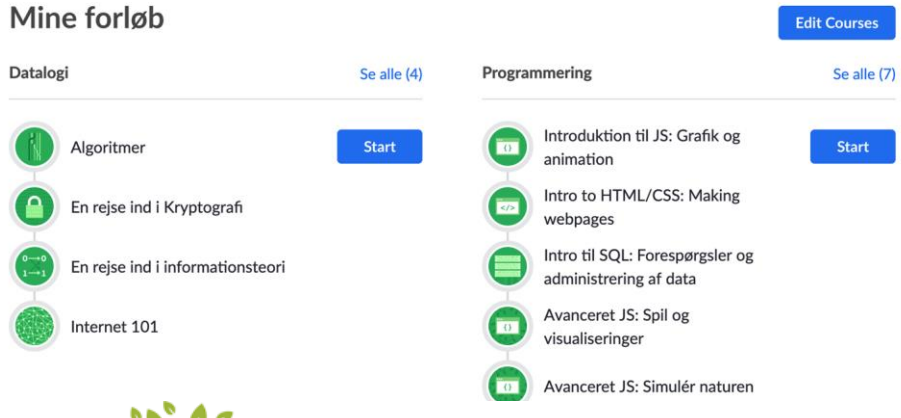
På nemprogrammering.dk's hjemmeside finder du videotutorials på dansk til en lang række programmeringssprog og programmer, bl.a. JavaScript, C# og Unity. Siden er god for de lidt ældre, som gerne vil arbejde med programmering på egen hånd. http://www.nemprogrammering.dk/

Codecademy & Khan Academy

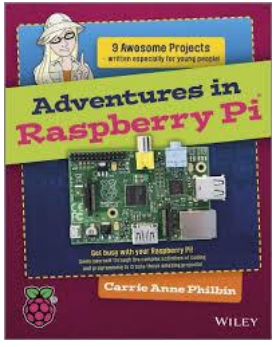
Begge tilbyder masser af interaktive tutorials til forskellige programmeringssprog, hvoraf en hel del er gratis – du skal bare oprette dig som bruger. Begge sites er velegnede til overbygningselever, som gerne vil videre med programmering – de findes her:

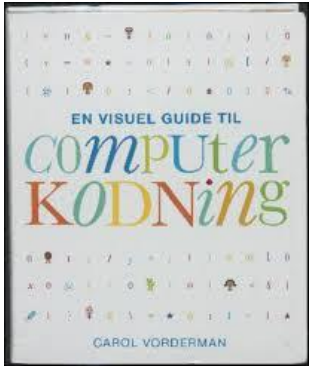
https://www.codecademy.com/

https://www.khanacademy.org/



(Nogle af) Bøgerne…

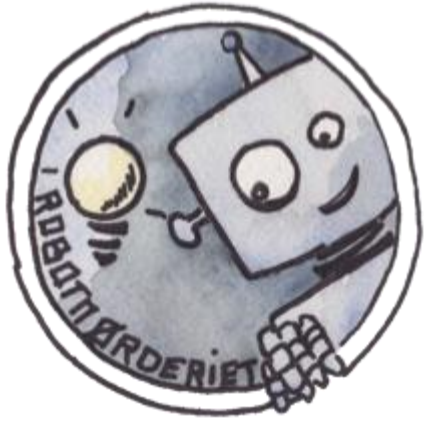




...og robotterne...

Mange af de store robotfirmaer har udviklet rigtig fint materiale til deres læringsrobotter.

Siden www.robotnørderiet.dk bestræber sig på at linke til en del af disse.



...og facebook-grupperne

- ❏ Kodning I Skolen
- ❏ Scratch i undervisningen
- ❏ C.R.A.F.T
- ❏ Micro:bit DK
- ❏ Ultra:bit - for lærere og pædagoger

Mads Remvigs website - http://4code.dk/ - er også fin at følge med på

Didaktik og forståelser

Om implementering af ressourcerne

En (af mange) definition(er)

Karen Brennan og Mitchell Resnicks definition af Computational Thinking fra 2012 er inddelt i tre dimensioner.

- **Computational Concepts:** fundamentale elementer og strukturer, der bruges i programmering, fx sekvenser, løkker, data, betingelser, osv.

- **Computational Practices:** tilgange, som anlægges i en programmeringsproces, fx abstraktion, dekomposition (modularisering), mønstergenkendelse, algoritmisk tænkning, fejlsøgning, m.m.

- **Computational Perspectives:** Det syn på verden, som kommer til udtryk gennem computationel problemløsning

(Brennan & Resnick, 2012)

Kilde: Michael E. Caspersen: “Teaching Programming” s. 112 i “Computer Science Education”

Udfordringer I

- Mange læringsressourcer tager udgangspunkt i et specifikt sprog
- Viden om programmerings**processer** er sjældent eksplicit adresseret

Kilde: Michael E. Caspersen: “Teaching Programming” s. 112 i “Computer Science Education”

Udfordringer II

- Mange læringsressourcer anvender en “bottom-up” tilgang, hvor man først skal mestre en række basale koncepter, inden man gradvist bliver introduceret til mere avancerede koncepter og principper.

Dette kan fungere for særligt teknisk orienterede elever, men kan give alvorlige motivationsproblemer for resten.

- Bottom-up-tilgangen sigter mod udvikling af detaljerede kompetencer inden for et enkelt sprog

Vi ønsker at udvikle interesse, kritisk tænkning, kreativitet og brede kompetencer inden for programmering og Computational Thinking

Kilde: Michael E. Caspersen: "Teaching Programming" s. 112 i "Computer Science Education"

Strategier og didaktiske principper

Se videoer, som folder alle principperne ud, på:
https://sites.google.com/pha.dk/tek-laerer/begreber-metoder-didaktik/didaktik

Progression

- Vær applikationsorienteret
- Lad elever progrediere fra forbruger til producent (use-modify-create)
- Organisér progression ud fra opgavekompleksitet - ikke kompleksitet i programmeringssproget

Eksempler

- Tilbyd eksemplariske eksempler
- Tilbyd "Worked Examples"
- Tilvejebring motivation gennem "passion", "play", "peers" og meningsfulde "projects"

Abstraktion og mønstre

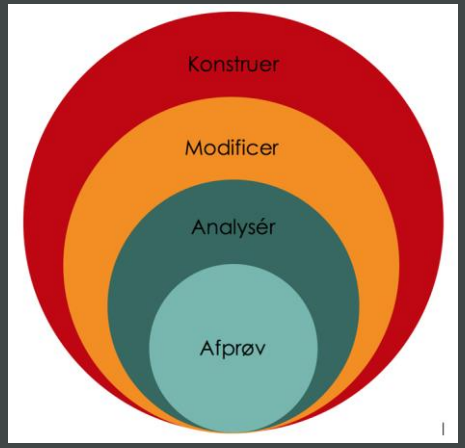
- Forstærk specifikationer
- Forstærk mønstre
- Forstærk modeller og konceptuelle rammeværk (programmering *ind* i et sprog)

Proces

- Afslør proces og pragmatisme
- Tilbyd stilladsering gennem trinvise selv-forklaringer
- Anvend og undervis i inkrementel udvikling gennem "Stepwise improvement" (udvid, konkretisér, omstrukturér)

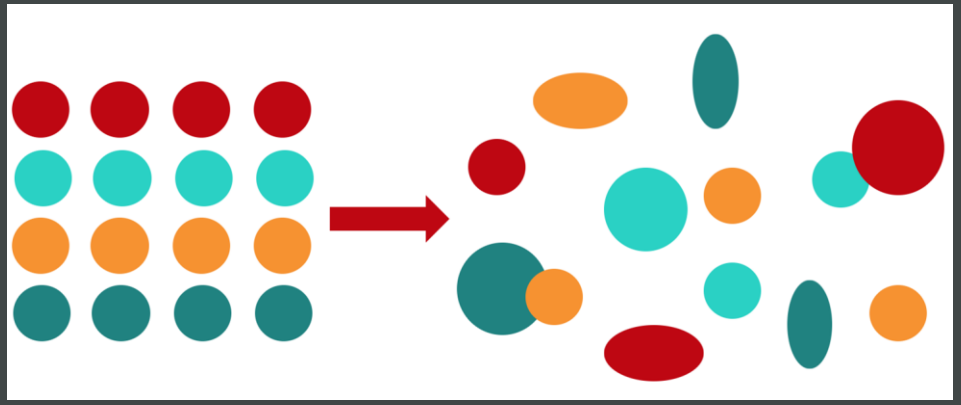
Kilde: Michael E. Caspersen: “Teaching Programming” s. 112 i “Computer Science Education”

Tre didaktiske principper i forhold til progression



Lad elever progrediere fra forbruger til producent (use-modify-create)

Vær applikationsorienteret

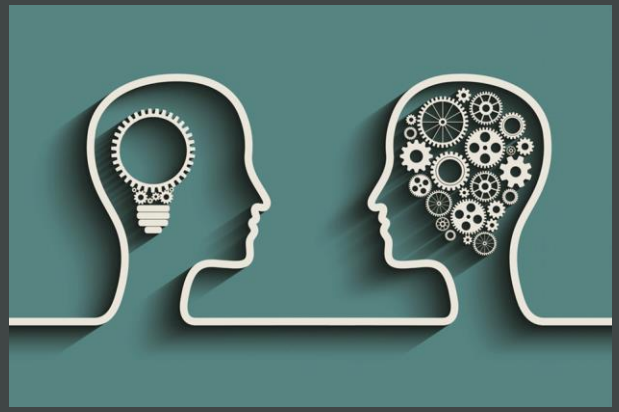


Organisér progression ud fra opgavekompleksitet - ikke kompleksitet i programmeringssproget, jf. low floor, high ceiling, wide walls

Kilde: Michael E. Caspersen: “Teaching Programming” s. 112 i “Computer Science Education”

Tre didaktiske principper i forhold til eksempler

Tilvejebring motivation



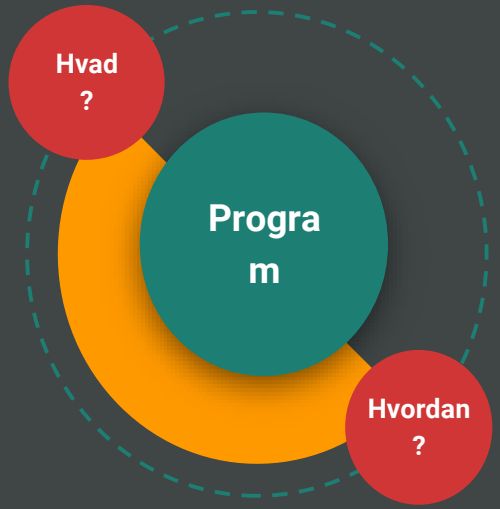
Tilbyd eksemplariske eksempler - lette at forstå, få nye koncepter ad gangen

Tilbyd “Worked Examples”

Kilde: Michael E. Caspersen: “Teaching Programming” s. 112 i “Computer Science Education”

Tre didaktiske principper i forhold til abstraktion og mønstre

Forstærk specifikationer, altså *hvad* et program gør

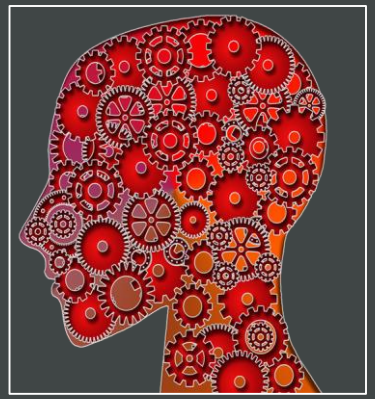


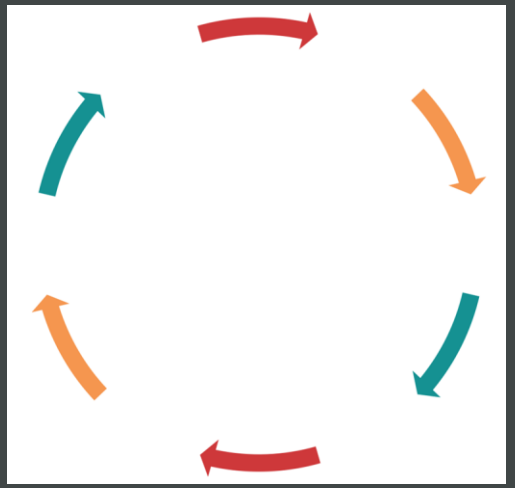
Forstærk mønstre, fx forskellige cover stories

Forstærk modeller og konceptuelle rammeværk (programmering *ind* i et sprog). Først modeller og koncepter, dernæst abstrahering og simulering i den konkrete programmering

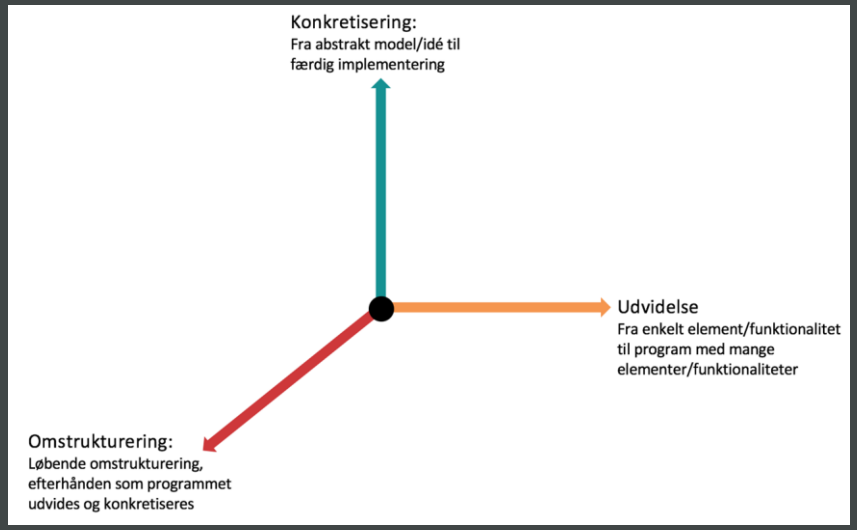
Tre didaktiske principper i forhold til proces

Tilbyd stilladsering gennem trinvise selv-forklaringer (mentale dialoger, når worked examples studeres)





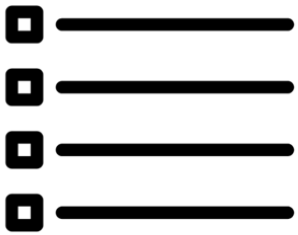
Afslør proces og pragmatisme



Anvend og undervis i inkrementel udvikling gennem “Stepwise improvement” (udvid, konkretisér, omstrukturér)

Vær opmærksom på kontrolstrukturer på tværs af sprog

SEKVENTIEL KONTROL

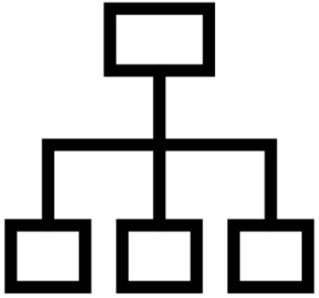


Kommandoer følges i den angivne rækkefølge. Første kommando skal udføres, før programmet går videre til den næste.

SAMMENSÆTNING

Kommandoer sammensættes til en enkelt blok, fx når **variable** defineres ved start af programmet (sæt point til, sæt liv til, osv. eller sæt liv til x, hvis y rører x, sæt x = x-1

UDVÆLGELSE



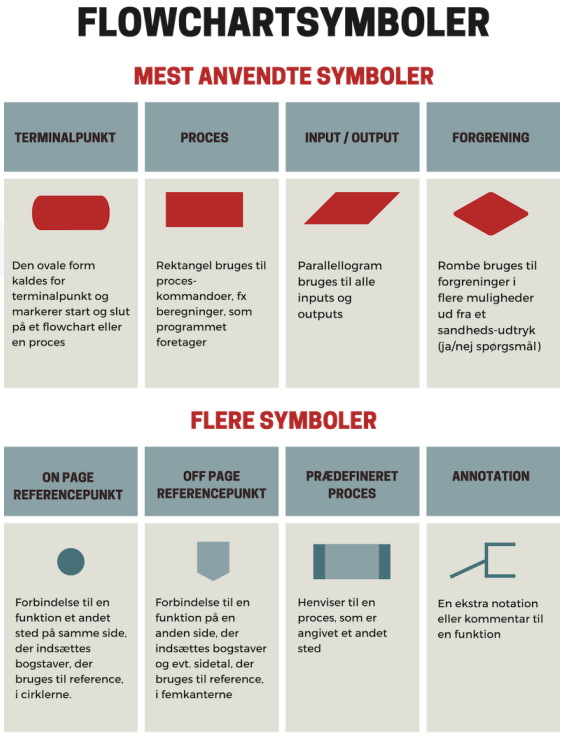
En kommando udvælges blandt flere muligheder: **if** (logisk/ boolsk udtryk); noget er enten **sandt** eller **falskt)...else**

GENTAGELSE

Fx **løkker.** En kommando gentages et antal gange. Antallet af gentagelser kan styres af et angivet fast tal, men styres også ofte af et logisk udtryk

Udtryk logik før programmering

```
                                    // her begynder programmet
     BEGIN
hop1 Mærk efter om du er sulten     // her spørges
     IF sulten
       THEN spis spaghetti          // her hoppes fire linjer op i programmet
       ELSE vent en time            // programmet er færdig
     GOTO hop1
     END
```



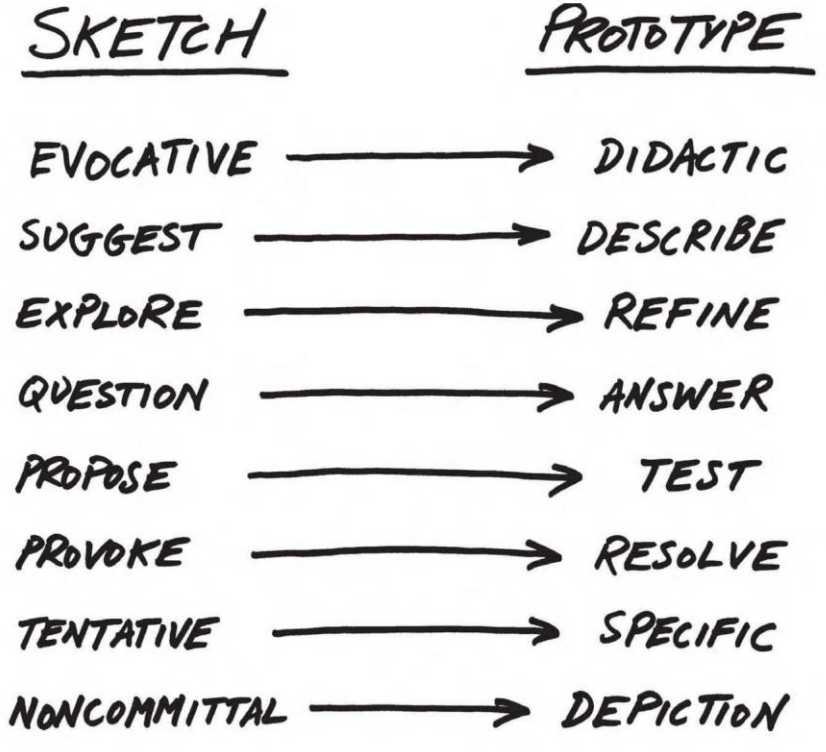
Fx pseudokode og flowcharts - se mere på:
https://sites.google.com/pha.dk/tek-laerer/begreber-metoder-didaktik/s%C3%A6rlige-metoder-og-v%C3%A6rkt%C3%B8jer

Skitser og prototyper

Man kan tale om **“lo-fi”**, fx papirsprototyper, & **“hi-fi”**, fx digitaliserede prototyper.

På siden her linkes til en række forskellige ressourcer, som kan bruges i arbejdet med prototyping:

kortlink.dk/ymcu



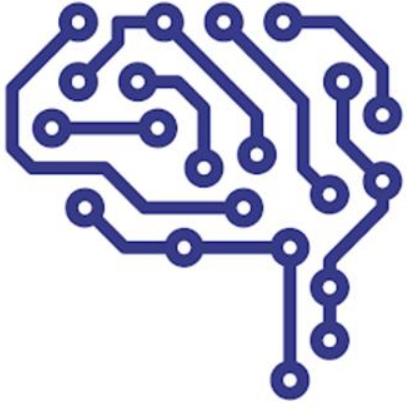
Bill Buxton: Sketching User Experiences, s. 140

Test før, under og efter

- Kortlægning af brugere
- Userflow
- Usability
- UX

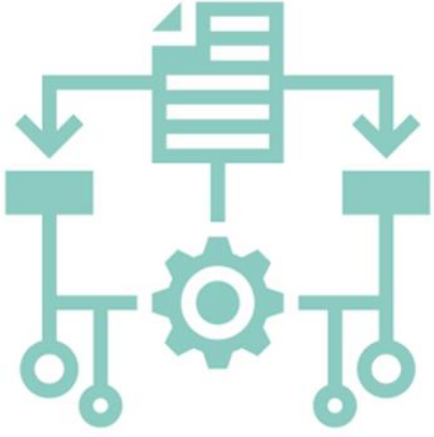
Se mere på: https://sites.google.com/pha.dk/tek-laerer/begreber-metoder-didaktik/usability-ux-og-brugertest

Arbejd med interaktionsdesign



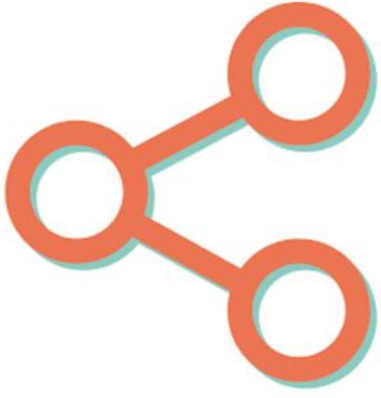
Begrebsmæssigt niveau

Her arbejdes på et overordnet (begrebsligt) niveau med centrale begreber, interaktionsaktiviteter og interaktionsformer



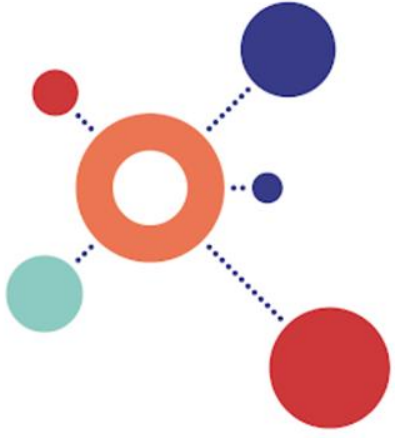
Helhedsniveau

Her arbejdes med brugergrænsefladens elementer, fx alle vinduer, og flowet imellem dem (fx overgange mellem vinduer)



Elementniveau

Her arbejdes med hvert elements dele (fx hver enkelt vindue), og flowet mellem delene inden for elementet



Delniveau

Her arbejdes med delene, fx felter, knapper, m.m, deres placering og deres udformning

Se mere på: https://sites.google.com/pha.dk/tek-laerer/begreber-metoder-didaktik/interaktion-og-interface

Find præsentation her: