Dag 1

Bli kjent med opplegget og Python

Symbolforklaring

Her vil symbolene vi bruker i presentasjonen bli definert

\+ *legg til dette*
\- *fjern dette*
$ *Kjør dette i kommandolinjen*

Eksempel:

main.py

```
+ print “Python er kalt opp etter Monty Python, ikke slangene”
- print “Hallo verden!”
```

Her vil man da fjerne “Hallo Verden” og legge til “Python er kalt opp etter Monty Python, ikke slangene”

```
$ cd dag1
$ kivy main.py
```

Her vil man bytte til mappen dag1, og så kjøre main.py filen

Oppsett

For Kivy og python vil vi i dette kurset bruke den portable Kivy distribusjonen. Kivy er det *rammeverket* vi vil bruke senere i kurset for å lage spillet, og Python er *Programmeringsspråket* vi skriver i. Tenk på det som å skrive et *dikt* på *norsk*.

For å installere Kivy distribusjonen på egen PC eller Laptop kan man gå hit for Windows:

`http://kivy.org/docs/installation/installation-windows.html#installing-the-portable-version`

Instruksjonene er veldig lette å følge, og vil ikke permanent installere noe på datamaskinen din.

Oppsett forts.

For å skrive programmene våre vil vi bruke en åpen kildekode Text editor som heter Atom. Atom er mye brukt, og er laget av den mest populære kode-delings siden, Github.

For å installere Atom går dere her:

`https://atom.io/`

For å lagre arbeidet, har vi opprettet owncloud kontoer her:

`http://95.34.24.118/owncloud`

Loginen vil være *fornavn* og *etternavn23*

Python - print

Python fungerer veldig simpelt. Man har en tekstfil som slutter med *.py*, og i den ligger instruksjonene til programmet.

Eksempel:

print1.py

```
+   print ‘Hallo Verden!’
+   print ‘Dette er print instruksjonen.’
+   print ‘Den sender ting til kommando-linjen’
```

Man kan da kjøre dette og få følgende resultat:

```
$ kivy main.py
Hallo Verden!
Dette er print instruksjonen.
Den sender ting til kommando-injen
```

https://github.com/cruor99/kivyteach/tree/master/day1

Python - typer

I python som i annen programmering har vi forskjellige typer.

En av dem har du sett før.

String	Tekst, f.eks ‘Hei på deg!’, identifiseres med ‘ eller “
Int	Hele tall, f.eks 1 og 42 men ikke 2.0
Float	Tall med desimal, f.eks 2.0 eller 2.6
Boolean	True eller False, alltid stor forbokstav
None	Mangel av verdi, er ikke det samme som 0

https://github.com/cruor99/kivyteach/tree/master/day1

Python - kommentarer

Kommentarer er en måte for programmereren å gi beskjeder til folk som leser koden, slik at man kan bedre forstå hva som skjer her og der. Kommentarer bruker # tegnet før det du skriver, slik at Python ignorerer den linjen.

print2.py

```
+    # -*- coding: utf-8 -*-

+    # Kommentarer er linjer som ikke vil vises i sluttprogrammer
+    print ‘Hallo verden!’
+    # Som du vil se, kommer de ikke frem når man kjører programmet
+    # Det finnes derimot noen få unntak, som toplinjen her. Dette er noe python bruker for
+    # at vi kan bruke æøå og andre slike tegn i python koden, uten at det blir
+    # problemer. Ikke-engelske tegn er ofte et gjennomgående problem med
+    # programmering, så prøv å unngå dem i starten.
```

Python - Matte1

I python gjør vi matte med operatører:

- + pluss
- - minus
- / delt på
- * gange
- % modulus

matte1.py

```
+    print 2 + 2
+    print 12 - 4
+    print 6 / 2
+    print 3 * 6
+    print 5 % 2
+    print “Jeg har skrevet”, 2 * 3, “printer”
```

Resultatet:

```
$ kivy matte1.py
4
8
3
18
1
Jeg har skrevet 6 printer
```

Python - Matte2

I python gjør vi matte med operatører:

- < mindre enn
- > større enn
- == helt lik
- <= mindre eller lik
- >= større eller lik

matte2.py

```
+   print ‘Er Python gøy?’, 1 < 2
+   print ‘Er lekser bra?’, 3 > 1 * 4
+   print ‘Er Minecraft kult?’ , 2 * 3 <= 6
+   print ‘Skal vi ha en pause?’,
    3 % 2 == 2 * 7 / 14
```

Resultatet:

```
$ kivy matte2.py
Er Python gøy? True
Er lekser bra? False
Er Minecraft kult? True
Skal vi ha en pause? True
```

Python - Variabler1

Vi kan lagre verdier i en variabel der verdiene kan endres og gjennbrukes

Variabler kan legges til hverandre og lage nye variabler av en eller flere andre variabler

variabler1.py

```
+    python = ‘Er Python gøy?’, 1 < 2
+    print python
+    a = 2
+    b = 3
+    print a + b
+    c = a + b
+    print ‘Dette kurset går over’, c, ‘uker’
```

Resultatet:

```
$ kivy variabler1.py
Er Python gøy? True
5
Dette kurset går over 5 uker
```

Python - Variabler2

Vi kan legge variabler direkte inn i strenger

variabler2.py

```
+   navn = ‘Yngve’
+   alder = 23
+   hoyde = 174
+   vekt = ‘det vil jeg ikke si’
+   print ‘En av lærerne heter’, navn, ‘han er’,
    alder, ‘år og er’, hoyde, ‘høy. Men om
    dere spør hvor høy han er vil han si:’,
    vekt
```

Resultatet:

```
$ kivy variabler2.py
en av lærerne heter Yngve han er 23 år og er
174 høy. men om dere spør hvor høy han er vil
han si: det vil jeg ikke si
```

Python - Strenger og variabler

Det å blande variabler og strenger er veldig enkelt i python.

Her bruker man {} inne i strenger, etterfulgt av .format() funksjonen.

Eksempler:

```
+    a = “Kjetil”
+    b = “Yngve”
+    print “Vi heter {} og {}”.format(a, b)
```

```
-    print “Vi heter {} og {}”.format(a, b)
+    print “Vi heter {first} og {second}.format(first=b, second=a)
```

Python - Input

Av og til er det greit å få litt input man kan jobbe med!

I python gjør man det slik:

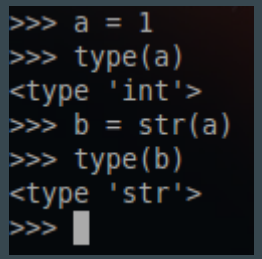
```
+    # -*- coding: utf-8 -*-
+
+    print “Hva heter du?
+    svar = raw_input()
+
+    print “Hyggelig å bli kjent, {}”.format(svar)
```

Python - Konvertering av verdier

I python er det nødvendig å la ting være av samme “type” for at de skal kunne jobbe sammen. Dette betyr at man ikke kan gjøre regne-operasjoner på strenger, og kan ikke sette sammen tall (integers) som strenger.

Måten dette håndteres i python er veldig enkelt:

```
+    a = 1
+    print type(a) # dette vil gi int som svar
+    b = str(a)
+    print type(b) # dette vil gi string
```



Oppgave!

La oss repetere litt!

- Lag et script som spør om tre tall, og regner ut gjennomsnittet.
- Lag et script som holder en samtale med deg selv!

Python - Metoder

Tenk på metoder som lagrede handlinger, som venter på å bli brukt!

Inne i metoder legger man mange hendelser, og kan repetere dem så mange ganger man vil.

```
+    # -*- coding: utf-8 -*-
+
+    def heiNavn(navn):
+        print “Hallo, {}”.format(navn)
+
+    a = raw_input()
+    heiNavn(a)
```

Python - Metoder

Et viktig konsept i python er “block levels” - blokk nivåer.
Dette er snakk om hvor koden er “dyttet inn” i forhold til resten av koden.
Standard python er at for hver nye “blokk” med kode, skal den være dyttet inn 4 mellomrom.

```
+    def example():
+    1234newblock = “Dette er en ny blokk”
+    1234sameblock = “Dette er den samme blokken”
+    1234print newblock + sameblock
+
+    newblock = “Dette er en helt ny blokk”
+    print newblock
+    example()
```

Python - Betingelser

Betingelser er mye brukt i programmering.

I python, i forhold til andre språk kan det være lurt å bruke engelske ord som nøkkelord, istedenfor noen vanlige former for betingelser:

```
+    verdi = 1
+    if verdi is 0:
+        print “verdien er null!”
+    else:
+        print “Verdien er: {}”.format(verdi)
```

Python - Løkker (loops)

Løkker blir brukt når man skal repetere ting.
Det finnes både “while” løkker - “Gjør dette, mens dette er sant” og “for” løkker - “Gjør dette for så og så”

```
+    for i in range(10):
+        print i
```

```
+    i = 0
+    while i < 20:
+        print i
+        i += 1
```

Oppgave:

- Lag et program som skriver ut ti linjer, hvor annenhver linje skal være fornavn og etternavnet ditt.
- Lag et program som hvor man skal gjette seg frem til riktig tall mellom 1 og 20. Her skal man ha 5 forsøk, og du skal si ifra om det er lavere eller høyere enn tallet.

```
+    import random
+    #bruk randrange metoden for å generere tallet
+    print random.randrange(1,21)
```