Dag 2

...

Mer i dybden!

Men først!

Litt repetisjon!

De gamle filene ligger her:

http://95.34.24.118/owncloud/

Denne gangen har vi forberedt USB-pinner til dere som bruker bibliotekets maskiner!

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:

```
$ python 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:

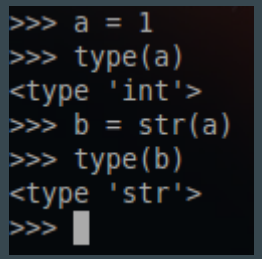
```
+    # -*- coding: utf-8 -*-
+
+    print “Hva heter du?
+    svar = raw_input()
+
+    print “Hyggelig å bli kjent, {}”.format(svar)
```

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)
```

Lister

```
$ python
Python 3.5.0 (default, Sep 20 2015, 11:28:25)
[GCC 5.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> liste = [1, 2, 3, 4]
>>> liste.append(5)
>>> print(liste)
[1, 2, 3, 4, 5]
>>> liste.pop()
5
>>> print(liste)
[1, 2, 3, 4]
>>> liste[0]
1
>>> liste[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> liste[3]
4
>>>
```

Løkker og lister

Man kan lett gå over en liste, slik vi har sett på med løkker tidligere!

```
+    listen = [“Kari”, “Kåre”, “Olekårejohnnyper”]
+    for navn i listen:
+        print navn
```

Husk encoding i toppen av filen!

```
+    # -*- coding: utf-8 -*-
     listen = [“Kari”, “Kåre”, “Olekårejohnnyper”]
     ...
```

Dicts! “Ordbøker”

```
$ python
Python 2.7.9 (default, Dec 19 2014, 06:05:48)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.56)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> ordbok = {}
>>> ordbok["Kjetil"] = "92882979"
>>> print ordbok["Kjetil"]
92882979
>>> print ordbok
{'Kjetil': '92882979'}
>>>
```

Oppgave!

Lag en liste med fem navn.

Gi alle personene et terningkast fra 1 til 6.

Skriv ut personen med det høyeste tallet.

Lagre resultatene i en dict med navn som nøkkel og terningkast som verdi.