Dag 1

•••

Getting to know the course and Python

Symbol explanation

Here are the symbols we are going to use in this presentation

+	*add this*
-	*remove this*
$	*run this*

Example:

main.py

```
+    print “Python is named after Monty Python, not the snake”
-    print “Hello World!”
```

Here we remove “Hello World!” and add “Python is named after Monty Python, not the snake!

```
$ cd dag1
$ python main.py
```

Here, you change to the folder day1, and run the file main.
py

Setup

For Kivy and Python in this course, we will use the portable Kivy distribution.
Kivy is the *framework* we will be using later in the course, and Python is the *language* we are writing in. Think about it like *a poem* written in *Norwegian*.

In order to install the Kivy distribution on your windows computer, go to this URL and follow the instructions:

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

The instructions are quite simple, and will not make permanent changes to your computer.

The file to start the python command line is *kivy2.7.bat*, not the *kivy-bash.bat* file

Setup continued

In order to write our programs, we are going to use an open source text editor called *Atom*. Of course, you can use any text editor of your preference, such as Sublime, Notepad++, Brackets.io or even vim and emacs if you feel brave.

In order to install Atom, go here:

https://atom.io/

In order to save your work, we created owncloud accounts here:

Removed for public view

The login will be *firstname* and password *lastname23*

Python - print

Python works quite simplistically. You have a text file ending with *.py*, and within it contains the instructions for the program..

Example:

print1.py

```
+    print ‘Hello world!’
+    print ‘This is the print function.’
+    print ‘It sends things to the command line’
```

You can then run it and get the following result:

```
$ python main.py
Hello world!
This is the print function.
It sends things to the command line
```

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

Python - types

In python, like other programming languages, we have different types

One of them you have noticed earlier:

String	Text, like: ‘How do you do’, is identified with ‘ or “
Int	Whole numbers, like: 1 and 42 but not 2.0
Float	Numbers with decimals, like: 2.0 or 2.6
Boolean	True or False, always with a capital letter
None	The absence of a value, is not the same as 0

Python - comments

Comments is a way for the programming leave messages for people reading the code, so you can better understand what is happening at those parts. Comments is the # sign in front of what you write, so that Python ignores the line.

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 - Math1

In python, we do math with operators:

- + pluss
- - minus
- / division
- * times
- % 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:

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

Python - Math2

In python we do math with operators:

- < less than
- > greater than
- == equals

- <= less than or equals
- >= greater than or equals

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:

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

Python - Variables1

We can store values in a variable where the values can be changed and used again.

Variables can be added to each other and create new variables from one or more different variables.

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 - Variables2

We can add variables directly to strings

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 - Strings and variables

Mixing variables and strings is very simple in python.

Here we use {} within the strings, followed by the .format() function.

Examples:

```
+    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

Some times, it’s nice to get some input to work with!

In python, we do it like this:

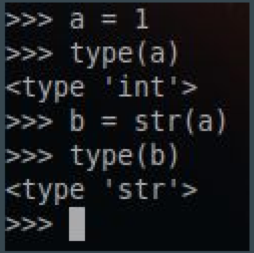
```
+    # -*- coding: utf-8 -*-
+
+    print “Hva heter du?
+    svar = raw_input()
+
+    print “Hyggelig å bli kjent, {}”.format(svar)
```

Python - Converting values

In python, it is necessary for things to be of the same type if they are to work together. That means you can’t do calculations with strengs, and can’t put integers together like strings.

The way this is done in python is as follows:

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



```
>>> a = 1
>>> type(a)
<type 'int'>
>>> b = str(a)
>>> type(b)
<type 'str'>
>>> 
```

Tasks!

Some repitition

- Create a script that asks for 3 numbers, and returns the average.
- Create a script to hold a conversation with yourself.

Python - Methods

Think of methods like prepared actions, waiting to be used!

Within methods, you can add many actions and repeat them as often as you want.

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

Python - Methods

An important concept in python is “block levels”.
This relates to where the code is indented in relation to the rest of the code
The standard in python is that for every new “block” of code, it is to be indented by 4 spaces.

```
+    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

Conditions are often used in programming.

In python, compared to some other languages, we can use some english keywords instead of the normal conditions:

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

Python - Loops

Loops are used for repeating things.

There are both “while” loops - “Do this while the condition is True” and “for” loops, “Do this for this condition”

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

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

Task:

- Create a program that prints out 10 lines, and the lines should alternate between your first and last name
- Create a program to guess a number between 1 and 20. You are to have only 5 attempts, and give feedback on whether or not you are above or below the answer.

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