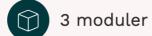
En rolig start

Introduksjon til programmering





- Passer for alle
- Dybdekurs



Om veiviseren

I denne veiviseren vil du lære om hva kunstig intelligens er, hvordan det fungerer – og hvordan kunstig intelligente systemer kan brukes på en etisk og ansvarlig måte på arbeidsplassen.

PROGRAMMERING I GRUNNSKOLEN - NY LÆREPLAN

	Matematikk	Naturfag	Musikk	Kunst og håndverk
2	lage og følgje reglar og trinnvise			
	instruksjonar i leik og spel			
3	lage og følgje reglar og trinnvise		Eksperimentere med rytmer, melodier	
	instruksjonar i leik og spel knytte til		og andre grunnelementer, sette	
	koordinatsystemet		sammen mønstre til komposisjoner,	
4	lage algoritmar og uttrykkje dei ved		også ved bruk av digitale verktøy,	
	bruk av variablar, vilkår og lykkjer		og beskrive arbeidsprosesser og	
			resultater	
5	lage og programmere algoritmar med	Utforske, lage og	Bruke teknologi og digitale verktøy til å	Bruke programmering til å
	bruk av variablar, vilkår og lykkjer	programmere teknologiske	skape, øve inn og bearbeide musikk	skape interaktivitet og visuelle
6	bruke variablar, lykkjer, vilkår og	systemer som består av deler		uttrykk
	funksjonar i programmering til	som virker sammen		
	å utforske geometriske figurar og			
	mønster			
7	bruke programmering til å			
	utforske data i tabellar og datasett			
8	utforske korleis algoritmar kan	Utforske, forstå og lage	Skape og programmere musikalske	Utforske hvordan digitale
	skapast, testast og forbetrast ved	teknologiske systemer som	forløp ved å eksperimentere med lyd	verktøy og ny teknologi kan gi
	hjelp av programmering	består av en sender og en	fra ulike kilder	muligheter for
9	simulere utfall i tilfeldige forsøk og	mottaker		kommunikasjonsformer og
	berekne sannsynet for at noko skal			opplevelser i skapende
	inntreffe ved å bruke programmering	Bruke programmering til		prosesser og produkter
10	utforske matematiske eigenskapar og	å utforske naturfaglige		
	samanhengar ved	fenomener		
	å bruke programmering			

Dagens agenda

Hva er python	
Noen praktiske datatyper	
Variabler	
- oppgave	
Løkker	
Python Turtle	
- oppgave	
Onneummering og avelutning	

Hva er Python

Easy Efficient Interpreted Readable

Free as in beer Community



Extensible Cross-plattform Portable

Well documented Open Source

KOMPILERT KODE

- Raskere kode
- Klar til å kjøres når brukeren får programmet
- Et ekstra steg i utviklingen
- Låst til plattform
- Kildekoden er lukket
- Eks: C++, Java, Swift

TOLKET KODE

- Enklere å utvikle, enklere å teste
- Ikke låst til plattform
- Brukeren trenger en tolker for å kjøre programmet
- Kildekoden er åpen
- Eks: Ruby, Matlab, Python

```
Python 3.9.12 (main, Jun 1 2022, 06:36:29)
[Clang 12.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
```

If the implementation is easy to explain, it may be a good idea. Namespaces are one honking great idea -- let's do more of those!

>>> quit()
(base) kristianbotnen@c02vl0lphtdg ~ %

>>>

(base) kristianbotnen@c02vl0lphtdg ~ % python

PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan

<ncoghlan at gmail.com>

Status: Active
Type: Process
Created: 05-Jul-2001

Post-History: 05-Jul-2001, 01-Aug-2013

▶ Table of Contents

Introduction

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing style guidelines for the C code in the C implementation of Python.

This document and <u>PEP 257</u> (Docstring Conventions) were adapted from Guido's original Python Style Guide essay, with some additions from Barry's style guide [2].

This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.

Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.

A Foolish Consistency is the Hobgoblin of Little Minds

One of Guido's key insights is that code is read much more often than it is written. The guidelines provided here are intended to improve the readability of code and make it consistent across the wide spectrum of Python code. As <u>PEP</u> 20 says, "Readability counts".

A style guide is about consistency. Consistency with this style guide is important. Consistency within a project is more important. Consistency within one module or function is the most important.

However, know when to be inconsistent – sometimes style guide recommendations just aren't applicable. When in doubt, use your best judgment. Look at other examples and decide what looks best. And don't hesitate to ask!

In particular: do not break backwards compatibility just to comply with this PEP!



https://docs.conda.io/en/latest/miniconda.html



https://www.jetbrains.com/pycharm/



https://replit.com/



https://code.visualstudio.com/



https://codewith.mu/en/

Noen kule typer

1 = Tall

1.0 = Tall med desimal

a = Bokstav

hei = Ord

hei du = Setning

VERDIER

1 = int

1.0 = float

a = string hei = string

hei du = string

TYPER

Typer

:::python

Text Type: str

Numeric Types: int, float, complex Sequence Types: list, tuple, range

Mapping Type: dict

Set Types: set, frozenset

Boolean Type: bool

Binary Types: bytes, bytearray, memoryview

None Type: NoneType

Variabler

Variabelnavn

1

Variabelnavn

a

Variabelnavn

Kristian

variabel_a

1

variabel_b

"a"

variabel_c

"Kristian"

```
# Pythonkode

variabelnavn_a = 1
variabelnavn_b = «a»
variabelnavn_c = «Kristian»
```

```
# Pythonkode

variabelnavn_a = 1
variabelnavn_b = «a»
variabelnavn_c = «Kristian»

variabelnavn a = «Python er gøy»
```

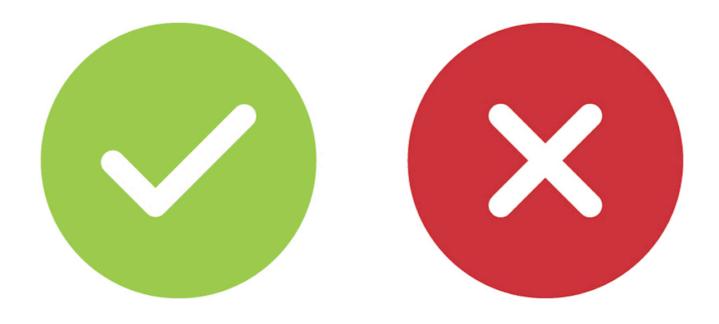
Løkker

Tilstandsløkke

Ta på lue <u>så lenge det snør ute</u>.

Så lenge det snør ute så ta på lue

Om det ikke snør ute, ikke ta på lue



Tilstandsløkke

```
i = 0
while (i < 5):
    print(i)
    i = i + 1</pre>
```

Telleløkke

```
for i in range(0, 5):
    print(i)
```

Turtle

```
forward()
backward()
right()
left()

speed()
shape()

pensize()
pencolor()
```

Ressurser og lenker

