

Lær Python, dag 1 - modul 1

Institut for Matematik og Datalogi, Syddansk Universitet

Team IMADA

<https://moggel12.github.io/PythonWorkshop2022/>

13. november, 2022

SDU



Indhold

Velkomst og introduktion

Hvad er programmering?

Programmering i Python

Typer, variabler og udtryk



Velkomst og Introduktion



Om kurset

Dette er et introducerende kursus til Python

- ▶ Forvent ikke at lave den næste Facebook efter 3 · 6 timers Python programmering.
- ▶ Vi tager forbehold til at mange ikke har programmeret før.
- ▶ Programmering, som så meget andet, kræver tid.

SDU



Kursets opbygning

Vi kommer til at arbejde med to moduler per afsat dag. Ét modul består af:

- ▶ En gennemgang af et aspekt af Python-programmering.
- ▶ En 15min pause til at få lidt frisk luft.
- ▶ Opgaver med et niveau tilsvarende hvad i har set hidtil.

Derudover vil vi holde en spisepause mellem hvert modul (30min)



Kursets opbygning

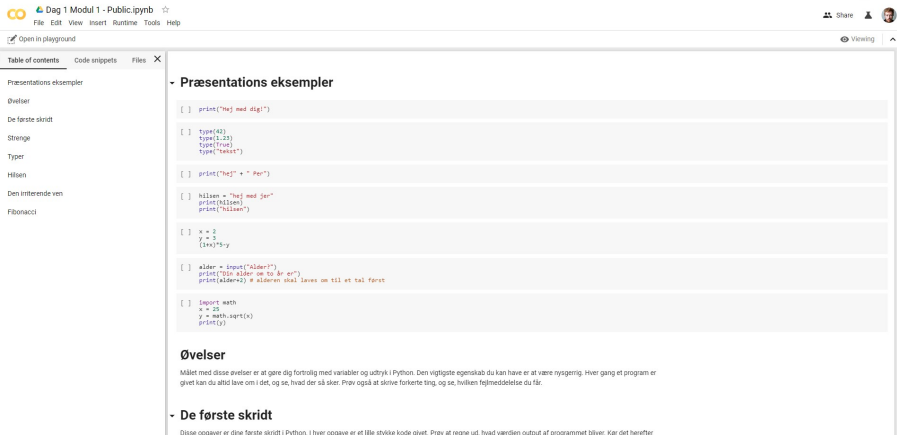
Vigtigst af alt, husk at have det sjovt og at stille de spørgsmål i vil!
Vi er her for at i kan lære.



Hvilke værktøjer skal vi bruge? (Anbefales)

Google Colaboratory

Det primære miljø vi kommer til at arbejde i er Google Colaboratory.



The screenshot shows the Google Colaboratory web interface. At the top, there is a navigation bar with the Google Colaboratory logo, the text "Dag 1 Modul 1 - Public.ipynb", and a star icon. Below this is a menu with options: File, Edit, View, Insert, Runtime, Tools, and Help. A "Share" button and a user profile icon are also visible in the top right corner.

The main content area is titled "Præsentations eksempler" and contains a list of code snippets. Each snippet is enclosed in a light gray box with a small "[]" icon in the top left corner. The snippets demonstrate various Python operations: printing a message, type checking, string concatenation, variable assignment and printing, arithmetic operations, user input, and using the math module for square roots.

On the left side of the interface, there is a "Table of contents" panel with a close button (X). It lists several sections: "Præsentations eksempler", "Øvelser", "De første skridt", "Strange", "Typer", "Hilsen", "Den irriterende ven", and "Fibonacci".

Below the code snippets, there is a section titled "Øvelser" (Exercises) with a paragraph of text: "Målet med disse øvelser er at gøre dig fortrolig med variabler og udtryk i Python. Den vigtigste egenskab du kan have er at være rygserrig. Hver gang et program er givet kan du altid lave om i det, og se, hvad der så sker. Prøv også at skrive forkerte ting, og se, hvilken fejlmeddelelse du får."

Below the exercises, there is a section titled "De første skridt" (The first steps) with a paragraph of text: "Disse opgaver er dine første skridt i Python. I hver opgave er et lille stykke kode givet. Prøv at regne ud, hvad værdien output af programmet bliver. Kar det herefter".

Hvilke værktøjer skal vi bruge? (Valgfrit)

(IPython)

Ikke nødvendigt, men en mulighed for de gængse programmører (anbefales ikke for nye).

```
→ ~ ipython
Python 3.9.4 (default, Apr 20 2021, 15:51:38)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.22.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: a = 42

In [2]: b = "hello there"

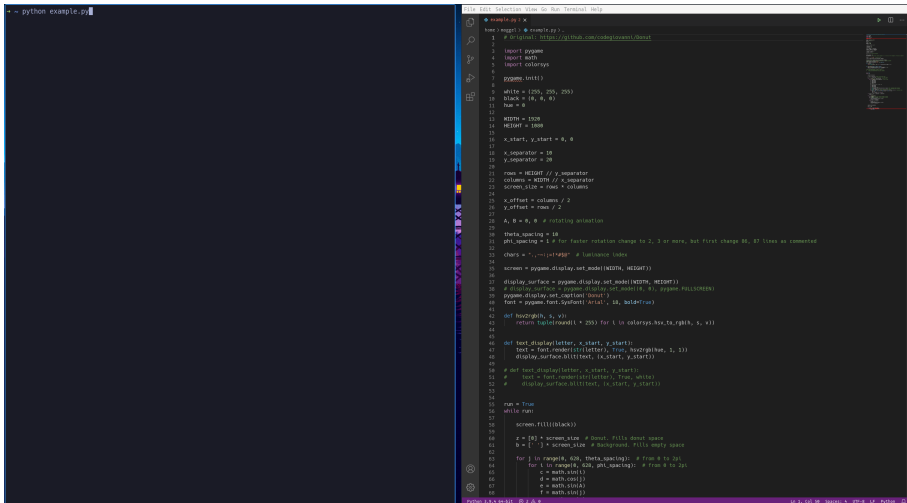
In [3]: c = (b*(a-22) + b[a%11])[6:17]

In [4]: □
```


Hvilke værktøjer skal vi bruge? (Valgfrit)

(Editor + terminal)

Heller ikke nødvendigt (anbefales ikke for nye).



```
python example.py

1 # Original: https://github.com/colseowang/doust
2
3 import pygame
4 import math
5 import colorsys
6
7 pygame.init()
8
9 white = (255, 255, 255)
10 black = (0, 0, 0)
11 hue = 0
12
13 WIDTH = 1024
14 HEIGHT = 1024
15
16 x_start, y_start = 0, 0
17
18 x_separator = 32
19 y_separator = 32
20
21 rows = HEIGHT // y_separator
22 columns = WIDTH // x_separator
23 screen_size = rows * columns
24
25 x_offset = columns // 2
26 y_offset = rows // 2
27
28 A, B = 0, 0 # starting orientation
29 theta_spacing = 18
30 phi_spacing = 1 # for faster rotation change to 2, 3 or more, but first change 86, 87 lines as commented
31
32 chars = ".,-:;=+*!@#%&^_{}~|'\">
33
34 screen = pygame.display.set_mode(WIDTH, HEIGHT)
35
36 display_surface = pygame.display.set_mode(WIDTH, HEIGHT)
37 # display_surface = pygame.display.set_mode(0, 0), pygame.FULLSCREEN)
38 pygame.display.set_caption('doust')
39 font = pygame.font.SysFont('Arial', 18, bold=True)
40
41 def hex2rgb(h, s, v):
42     return tuple(round(i * 255) for i in colorsys.hsv_to_rgb(h, s, v))
43
44
45 def text_display(letter, x_start, y_start):
46     text = font.render(str(letter), True, hex2rgb(hue, 1, 1))
47     display_surface.blit(text, (x_start, y_start))
48
49
50 # def text_display(letter, x_start, y_start):
51 #     text = font.render(str(letter), True, white)
52 #     display_surface.blit(text, (x_start, y_start))
53
54
55 run = True
56 while run:
57
58     screen.fill(black)
59
60     z = [0] * screen_size # Doust: fills doust space
61     b = [1] * screen_size # Background: fills empty space
62
63     for j in range(0, 628, theta_spacing): # from 0 to 2pi
64         for i in range(0, 628, phi_spacing): # from 0 to 2pi
65             c = math.cos(i)
66             d = math.cos(j)
67             e = math.sin(d)
68             f = math.sin(j)
```

Hvad er programmering?



Hvad er et program?

Program = sekvens af instruktioner

Instruktioner beskriver hvad der skal ske / hvad der skal beregnes

Termer:

- ▶ Instruktioner: Hvad selve programmet består af
- ▶ Input: Data som programmet skal forholde sig til
- ▶ Output: Resultatet af programmet

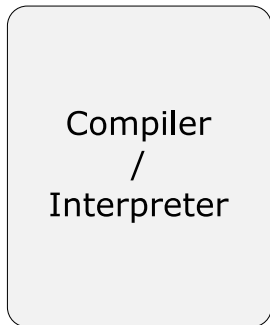
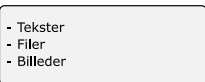
Hvordan forstår computeren vores programmer?

Programmer skrives i et højniveau-sprog, så som C, Java eller Python. Et program skal oversættes (compiles) før computeren kan forstå det.

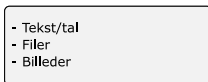
Inputprogram



Programinput



Output



Givet et program og et input vil computeren udføre nogle instruktioner der giver et ønsket (eller uønsket *sad face*) output.

Flow: Et program udføres indstruktion for instruktion, fra top til bund.

Tænk først, programmér bagefter

Det er ikke unormalt at selv de mest gængse udviklere skal bearbejde et problem før de skriver koden. Der er ingen skam i ikke at se koden for øjnene med det samme!

Computere er dumme! (men hurtige). De gør kun som de får besked på...

First, solve the problem. Then write the code.

- John Johnson

Programmering i Python



Hvorfor Python når vi har *indsæt populært sprog her*?

Fordele

- ▶ Simpelt sprog - nemt at lære
- ▶ Stort standardbibliotek (allerede implementerede funktioner)
- ▶ Kan køre på mange platforme
- ▶ Udbredt brugt både i industri og forskning

Ulemper

- ▶ Langsomt sammenlignet med mere avancerede sprog som C/C++
- ▶ Dynamiske typer kan give upraktiske problemer ved køretid (tænk ikke for meget over dette hvis du er ny)

Hvad bruges Python til i det virkelige liv?

- ▶ Scripting - små hurtige programmer
- ▶ Data behandling og visualisering
- ▶ Machine Learning og Deep Learning
- ▶ Backend-development (fx brugt i Instagrams servermiljø)
- ▶ Hardwareprojekter



Keywords

Visse ord i Python er reserveret til sproget selv (i vil løbende støde på nogle af disse), her ser i et uddrag af dem:

and, not, assert, finally, or, break, for, pass, class, from,
print, continue, global, raise, def, if, return, del, import,
try, elif, in, while, else, is, with, except, lambda, yield



Typer, variabler og udtryk



Værdier

Når vi arbejder med problemer arbejder vi typisk med værdier af forskellige slags.

- ▶ Hvis vi plusser to tal sammen arbejder vi med talværdier.
- ▶ Hvis vi leder efter bandlyste ord i en Twitch-chat kigger vi efter tekstværdier.



Typer — Fordi vi skal have noget at arbejde med!

Nogle typer kalder vi for *primitive* typer:

Datatyper	Eksempel
String (tekst strenge)	"Hej"
Integer (heltal)	42
Float (kommatal)	42.0
Boolean (sand/falsk)	True

Men der findes mange andre (mere avancerede) typer også! Vi vil støde på et par stykker senere.

SDU



Typer — Et lille fif

Er man i tvivl om typen af den værdi man arbejder med kan man skrive:

```
type(< type_der_skal_tjekkes >)
```

Som et eksempel:

Program:

```
type(42.0)
```

Output:

```
<class 'float' >
```

SDU



Type konvertering (type casting)

Nogle typer kan konverteres/tvinges til at blive andre typer.

Konvertering til kommatall:

```
float(4)
```

```
4.0
```

Konvertering til heltal:

```
int(4.3)
```

```
4
```

Konvertering til streng:

```
str(4 + 3)
```

```
"7"
```



Operatorer

Følgende beskriver de basale operatorer anvendt på tal:

Operatorer	Beskrivelse	Eksempel	Resultat
+	Læg to operanter sammen	$40 + 2$	42
-	Træk to operanter fra hinanden	$50 - 8$	42
*	Gang to operanter	$6 * 7$	42
/	Division mellem to operanter	$126/3$	42
//	Heltalsdivision mellem to operanter	$126.5//3$	42
**	Eksponentiering	$2 * * 3$	8

SDU



Operatorer — Buyer beware

Nogle af de forgående operatorer kan også bruges på andet end talværdier.



Operatorer — Buyer beware

Addition?

Kode:

```
print("hej" + " Per")
```

Output:

```
hej Per
```



Operatorer — Buyer beware

Subtraktion?

Kode:

```
print("hej" - "ej")
```

Output:

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```



Operatorer — Buyer beware

Multiplikation?:

Kode:

```
print("hej" * 3)
```

Output:

```
hejhejhej
```

SDU



Operatorer — Buyer beware

Division?:

Kode:

```
print("hej" / 3)
```

Output:

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```



Variabler — Fordi nogle ting er værd at huske!

En variabel er en "beholder" som kan gemme en værdi. I hukommelsen bliver der reserveret plads til den givne variabel.

En variabel har et navn som udvikleren selv kan angive. Der er dog nogle restriktioner

Tilladte variabelnavne

x
et_navn
TEST
var2
_hej

Forbudte variabelnavne

1var
en.var
-var

Giv meningsfulde variabelnavne!

Variabler

Man kan læse op på præcist hvordan navnet på en variabel i Python må se ud, men det er ikke noget vi kommer til at snakke om her. Prøv jer frem og lad lysten drive jer!

Man må heller ikke gøre brug af reservede **nøgleord** som variabelnavne!



Variabler

En tildeling gemmer noget i den givne beholder/variabel.

Tildeling til en variabel sker på følgende vis:

```
<navn> = <værdi>
```

Tildeling af et heltal:

```
x = 42
```

Tildeling af en streng:

```
hilsen = "hej med jer"
```



Udtryk

Et udtryk er en kombination af værdier, variable og operatorer

Eksempler på udtryk:

5

$x = 3$

$x + 5$

$x = 2$

$y = 3$

$(1 + x) * 5 - y$

Regnereglernes hieraki overholdes.

Print

Udskriv udtryk og variabelers værdier med `print`-funktionen:

Program:

```
print("hej")  
print(42)  
x = 23  
print(x)
```

Output:

```
hej  
42  
23
```

Print kan bruges til output fra et program, men også til fejlfinding af ens program (debugging).

Bemærk! I Google Colaboratory og IPython vil værdien af en variabel også typisk printes ved bare at skrive variabelnavnet i sig selv (se live-kodning).

Print

Udskriv udtryk og variabelers værdier med `print`-funktionen:

Program:

```
print("hej")  
print(42)  
x = 23  
print(x)
```

Output:

```
hej  
42  
23
```

Print kan bruges til output fra et program, men også til fejlfinding af ens program (debugging).

Input

Input fra brugeren tages på følgende vis:

```
<variabel> = input(<Beskrivende streng>)
```

Eksempel 1:

```
navn = input("Hvad er dit navn?")
```



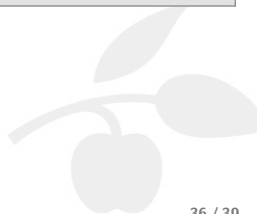
Input

Eksempel 2:

```
alder = input("Hvad er din alder?")  
print("Din alder om 2 år: ")  
print(alder + 2)
```

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: must be str, not int
```

SDU



Kommentar-streng

Kommentarer i koden er tekst som ignoreres når koden eksekveres. Disse er kun til ære for den der læser koden.

```
print("hej") # en kommentar som beskriver denne instruktion  
  
# en fritstående kommentar som beskriver den følgende kode  
print("whatup")
```

Gode kommentarer kan hjælpe afsindigt meget med at forstå kompleks kode!

SDU



Moduler

Mange ting er allerede implementeret i python af dygtige programmører. Disse funktioner er tilgængelige via moduler (aka. biblioteker).

Fx kan vi benytte `math`-biblioteket til at lave klassiske matematiske operationer:

Program:

```
import math
x = 64
y = math.sqrt(x)
print(y)
```

Output:

```
8.0
```

Se dokumentation for alle matematikfunktioner:

<https://docs.python.org/3/library/math.html>

First steps

Det var alt for første modul!

Nu skal i afprøve disse ting for jer selv, og kom endelig med eventuelle spørgsmål!

