

Lær Python, dag 1 - modul 1

Institut for Matematik og Datalogi, Syddansk Universitet

Mikkel Juul Vestergaard

<https://moggel12.github.io/PythonWorkshop2021/>

14. november, 2021

SDU



Indhold

Velkomst og introduktion

Hvad er programmering?

Programmering i Python

Typer, variabler og udtryk



Velkomst og Introduktion



Hvem er vi?



Mikkel Juul Vestergaard

- ▶ Datalog på 7. semester (kandidat)
- ▶ BSc. i datalogi
- ▶ 22 år gammel
- ▶ Fritid: Computer-halløj, film, kampsport

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.



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

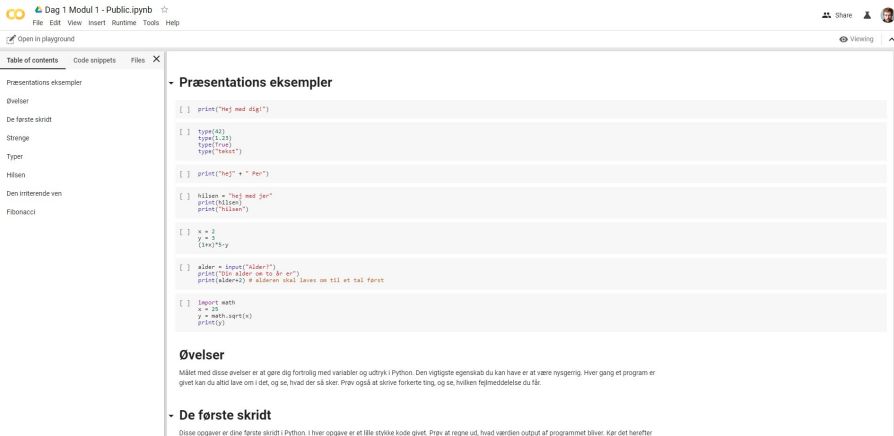
Vigtigst af alt, husk at have det sjovt og at stille de spørgsmål i vil!
Vi er her for at i kan lære.



Hvilke værktøjer skal vi bruge? (Anbefales)

Google Colaboratory

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



Google Colaboratory interface showing a notebook titled "Dag 1 Modul 1 - Public.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a sidebar with a "Table of contents" tab. The main content area displays a list of sections: "Præsentations eksempler", "Øvelser", "De første skridt", "Strøge", "Typer", "Hilsen", "Den irriterende ven", and "Fibonacci". The "Præsentations eksempler" section is expanded, showing several code snippets in Python, including basic printing, type conversions, string concatenation, and simple calculations. The "Øvelser" section is also visible, containing a paragraph of text about the goal of the exercises. The "De første skridt" section is partially visible at the bottom.

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 nødvendig (anbefales ikke for nye).

```

+ = python example.py
File Edit Selection View Go Run Terminal Help
example.py X
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1
```

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?

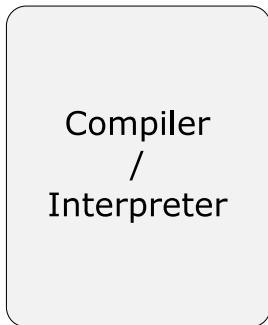
Programmer skrives i et højniveau-sprog, så som C, Java eller Python. Et program skal oversættes (compiles) før computeren kan forstå det.

Inputprogram



Programinput

- Tekster
- Filer
- Billeder



Output

- Tekst/tal
- Filer
- Billeder

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):

and, exec, 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.



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 ' >
```



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 hinaden	$50 - 8$	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
```



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 meningsfyldte 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, variabler 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 variablers 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 variablers 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
```



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!



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!

