

Introduktion til Programmering i Python

Studiepraktik 2019

Jonas Bamse Andersen

Syddansk Universitet

1. Programmering generelt
2. Programmering i Python

Programmering generelt

Få en computer til at udføre ønskede handlinger.

Computer = desktop, laptop, smartphone, vaskemaskine-controller...

Handlinger: Databehandling og interaktion med hardware.

Eksempler:

- Tekstbehandling
- WWW
- Databaser
- Computerspil
- Styring af andre maskiner (industrimaskiner, måneraketter, ...).
- Regneopgaver, videnskabelige udregninger

Mennesker	Computere
Uformel beskrivelse	Entydige kommandoer
Naturlige sprog	Fast format
Højniveausprog	Maskinkode
if... then... else...	01110111
Tæt på engelsk(?)	Binære koder
Bedre overblik	Dårligt overblik
Sværere at lave fejl	Nemt at lave fejl
Hurtig programmering	Langsom programmering

Oversætter-program: Højniveau sprog → maskinkode.

I alle programmeringssprog foregår problemløsning/programmering således:

1. Inddel problemet i mindre/simplere del-problemer.
2. Inddel de mindre del-problemer i endnu mindre del-problemer.
3. ...
4. Indtil du kan løse del-problemerne "nemt".

De fleste programmeringssprog er af den “imperative” type, dvs. er baseret på dataceller (kasser) hvis indhold kan ændres undervejs.

Eksempler:

Java, C, C++, C#, Pascal, Delphi, Basic, Fortran, Cobol, Ada, Perl, Python, Ruby, ...

Essensen i disse er den samme. Den gennemgås på de følgende sider. Vi bruger Python som eksempel.

Programmering i Python

Hurtig start:

1. Åbn `repl.it/X3G` og klik 'Start Now'.
2. Skriv kode i tekstfeltet.
3. Udfør dit program ved at klikke på 'Run' øverst.

Prøv tingene af undervejs. Vær nysgerrige.

Data og ændring af data:

1. Typer
2. Variable
3. Operatorer

Rækkefølge af udførsel af kommandoer:

5. Sekvens
6. Gentagelse
7. Betinget udførsel
8. Modularisering

Hvert sprog har desuden en **syntaks** (regler for opbygning af kode), som kan variere noget.

```
# The very basic example of Python: Hello World  
print("Hello World")
```

```
forbrug = int(input("Hvad er dit elforbrug?"))  
pris = 1020 + 1.65 * forbrug  
print("Din elregning er:")  
print(pris)
```

1. Typer

Al data i et program har en **type**.

Nogle typer i Python:

Datatype	Eksempel
String (tekst)	"Hej"
Integer (heltal)	42
Float (kommatal)	42.0
Boolean (sandhedsværdi)	True

2. Variable

Variabel = Navngiven beholder med data af en bestemt type.

Variabel oprettes, når de bruges første gang.

Generelt:

```
<variabelnavn> = <værdi>
```

Eksempler:

```
counter = 27  
greeting = "Hello World!"  
temperature = 44.7
```

Variable tildeles værdier med lighedstegn (=):

Bemærk at et variabelnavn på **højre** side henter værdien gemt i variablen, mens det på **venstre** side gemmer en ny værdi i variablen.

3. Operatorer: Nye data fra gamle

Almindelig matematik fungerer nogenlunde som det plejer:

Operator	Beskrivelse	Eksempel	Resultat
+	Læg to operander sammen	$40 + 2$	42
−	Træk to operander fra hinanden	$50 - 8$	42
*	Gang to operander	$6 * 7$	42
/	Division mellem to operander	$12.6/3$	4.2

3. Operatorer: Nye data fra gamle

Derudover har vi operatorer som giver en sandhedsværdi:

Operator	Beskrivelse	Eksempel	Resultat
<code>==</code>	Lig med	<code>42 == 3</code>	False
<code>!=</code>	Ikke lig med	<code>42 != 3</code>	True
<code><</code>	Mindre end	<code>6 < 7</code>	True
<code><=</code>	Mindre end eller lig med	<code>6 <= 6</code>	True
<code>></code>	Større end	<code>21 > 5</code>	True
<code>>=</code>	Større end eller lig med	<code>21 >= 5</code>	True

Operator	Beskrivelse	Eksempel	Resultat
<i>and</i>	Boolsk og	<code>(4 <= 6) and (3 < 2)</code>	False
<i>or</i>	Boolsk eller	<code>(4 <= 6) or (3 < 2)</code>	True
<i>not</i>	Bolsk ikke	<code>not(4 < 6)</code>	False

4. Sekvens

Som udgangspunkt udføres kommandoer én efter én i rækkefølge:

```
a = 7
```

```
b = a * a
```

```
a = 8
```

```
b = b + 1
```

Hvad ligger der i a og b?

4. Sekvens

Som udgangspunkt udføres kommandoer én efter én i rækkefølge:

```
a = 7  
b = a * a  
a = 8  
b = b + 1
```

Hvad ligger der i a og b ?

$$a = 8$$

$$b = 50$$

5. Gentagelse

Generel `while`-kommando:

```
while <sand/falsk betingelse>:  
    <instruktion>  
    <instruktion>  
    <instruktion>  
    ...
```

Kun den indrykkede kode gentages.

Eksempel (find kvadratroden af 25):

```
r = 0  
while r * r < 25:  
    r = r + 1  
print("The square root of 25 is :")  
print(r)
```

6. Betinget udførsel

Generel if-else-kommando:

```
if <sand/falsk-betingelse>:  
    Kode, der udføres, hvis betingelsen er sand  
else :  
    Kode, der udføres, hvis betingelsen er falsk
```

Kun den indrykkede kode udføres.

Eksempel:

```
if a > b:  
    print("a is larger")  
else :  
    print("b is at least as large")
```

7. Modularisering

Metoder (funktion, rutine, procedure): **opdeling** og **genbrug** af kode.

Eksempel:

```
def regning(forbrug):  
    pris = 1020 + 1.65 * forbrug  
    print("Hvis dit forbrug er:")  
    print(forbrug)  
    print(" skal du betale")  
    print(pris)
```

```
regning(40)
```

```
regning(50)
```

Python kommer med et meget stort bibliotek af færdiglavede metoder. De bruges til eksempelvis webudvikling, forskning og hobbyprojekter.

Se for eksempel:

`https://wiki.python.org/moin/UsefulModules`

Input og output

Udskriv til brugeren:

```
print(Det der skal skrives )
```

Vi kan både udskrive konkrete tal og strenge, men også variabler!

Læs fra tastatur (skriv `int()` omkring `input`, hvis du vil læse et tal):

```
navn = input("Hvad hedder du?")  
forbrug = int(input("Hvad er dit elforbrug?"))
```

```
# The very basic example of Python: Hello World  
print("Hello World")
```

```
forbrug = int(input("Hvad er dit elforbrug?"))  
pris = 1020 + 1.65 * forbrug  
print("Din elregning er:")  
print(pris)
```

1. Åbn <https://repl.it/X3G> og klik 'Start Now'.
2. Skriv kode i tekstfeltet.
3. Udfør dit program ved at klikke på 'Run' øverst.