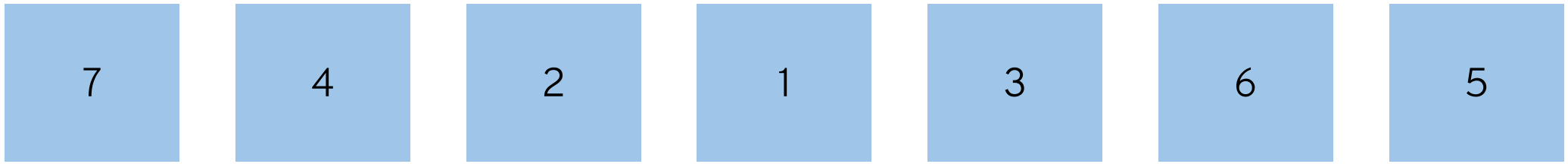
PROGRAMMERING I MATEMATIKK FOR UNGDOMSTRINNET

ALGORITMISK TENKING

Hvordan vil du gå fram for å sortere disse syv tallene i stigende rekkefølge?

Kan framgangsmåten du bruker generaliseres til en oppskrift?



HVORDAN?

Programmering er ikke bare et verktøy, men et eget fag.

Hva betyr dette for fagene? Hvordan kan programmering styrke fagopplæringen?

Fagene må ivaretas.

Hva gjør vi i Aschehoug?

Tilbakemeldinger?
programmering@aschehoug.no

(FORELØPIGE) KOMPETANSEMÅL

MATEMATIKK 4. TRINN

lage algoritmar og uttrykkje dei ved bruk av variablar, vilkår og lykkjer

MATEMATIKK 5. TRINN

lage og programmere algoritmar med bruk av variablar, vilkår og lykkjer

MATEMATIKK 6. TRINN

bruke variablar, lykkjer, vilkår og funksjonar i programmering til å utforske geometriske figurar og mønster

MATEMATIKK 7. TRINN

bruke variablar, lykkjer, vilkår og funksjonar i programmering til å utforske data i tabellar og reelle datasett

MATEMATIKK 8. TRINN

utforske korleis algoritmar kan skapast, testast og forbetrast ved hjelp av programmering

MATEMATIKK 9. TRINN

simulere utfall i statistiske fordelingar og berekne sannsynet for at noko skal inntreffe ved å bruke programmering

MATEMATIKK 10. TRINN

utforske matematiske eigenskapar og samanhengar ved å bruke programmering

HVORFOR PYTHON?

JAVA

```
public static void main(String[]
args) {
  int a = 3;
  int b = 5;

  System.out.println(a + b);
}
```

PYTHON

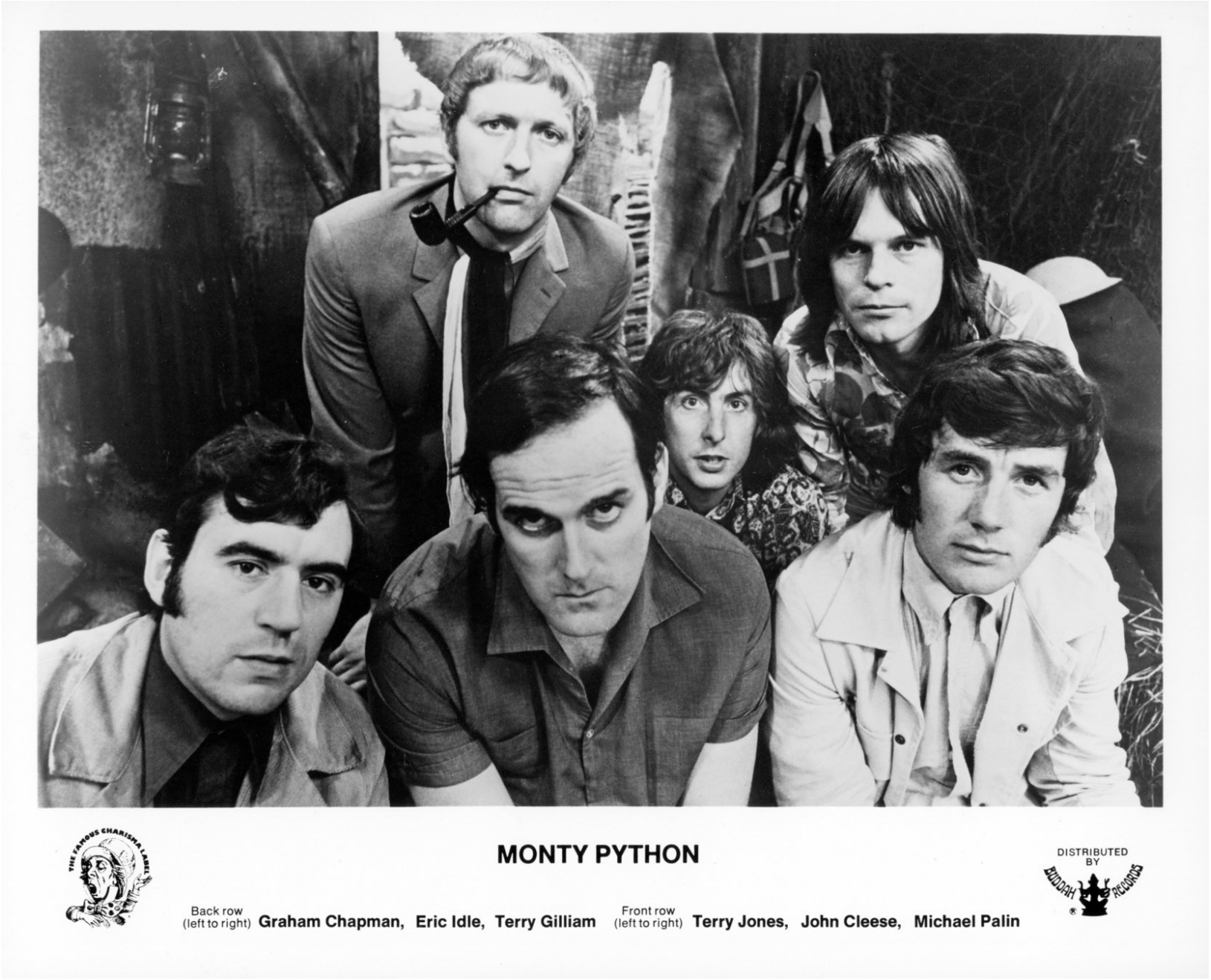
```
a = 3
b = 5

print(a + b)
```

JAVASCRIPT

```
<!DOCTYPE html>
<html>
<head>
  <title> ... </title>
</head>
<body>
<script>
var a = 3;
var b = 5;

console.log(a + b);
</script>
</body>
</html>
```



MONTY PYTHON

Fra Wikipedia: An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python

HVA TRENGER VI?

VI KOMMER VELDIG LANGT MED …

- **Variabler** (for å ta vare på verdier)
- **Løkker** (for å gjenta kode så mange ganger vi ønsker)
- **Beslutninger/valg** (for å styre hva vi ønsker at programmet skal gjøre)

I TILLEGG KAN DET VÆRE GREIT MED …

- **Lister** og **arrayer** (for å ta vare på mengder av verdier)
- **Graftegning**
- **Funksjoner** (som ikke behøver å være det samme som matematiske funksjoner)
- **Innlesing av data fra fil**

WHILE-LØKKER

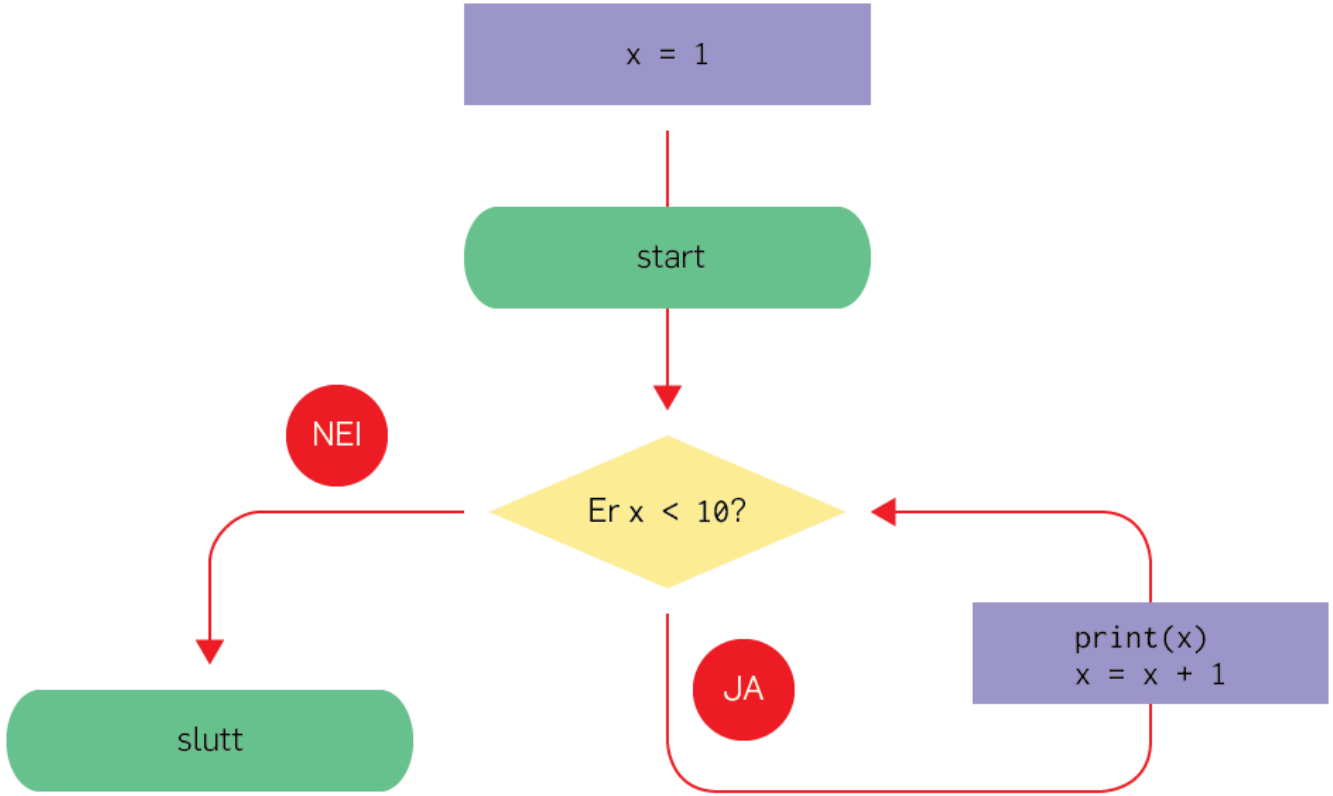
En **while**-løkke lar oss gjenta kode så lenge (*while*) en betingelse er sann/True.

```
while <betingelse>:
  # gjør noe
```

Eksempel:

```
x = 1

while x < 10:
  print(x)
  x = x + 1
```



Hva skjer hvis vi ikke skriver **x = x + 1**?

FOR-LØKKER

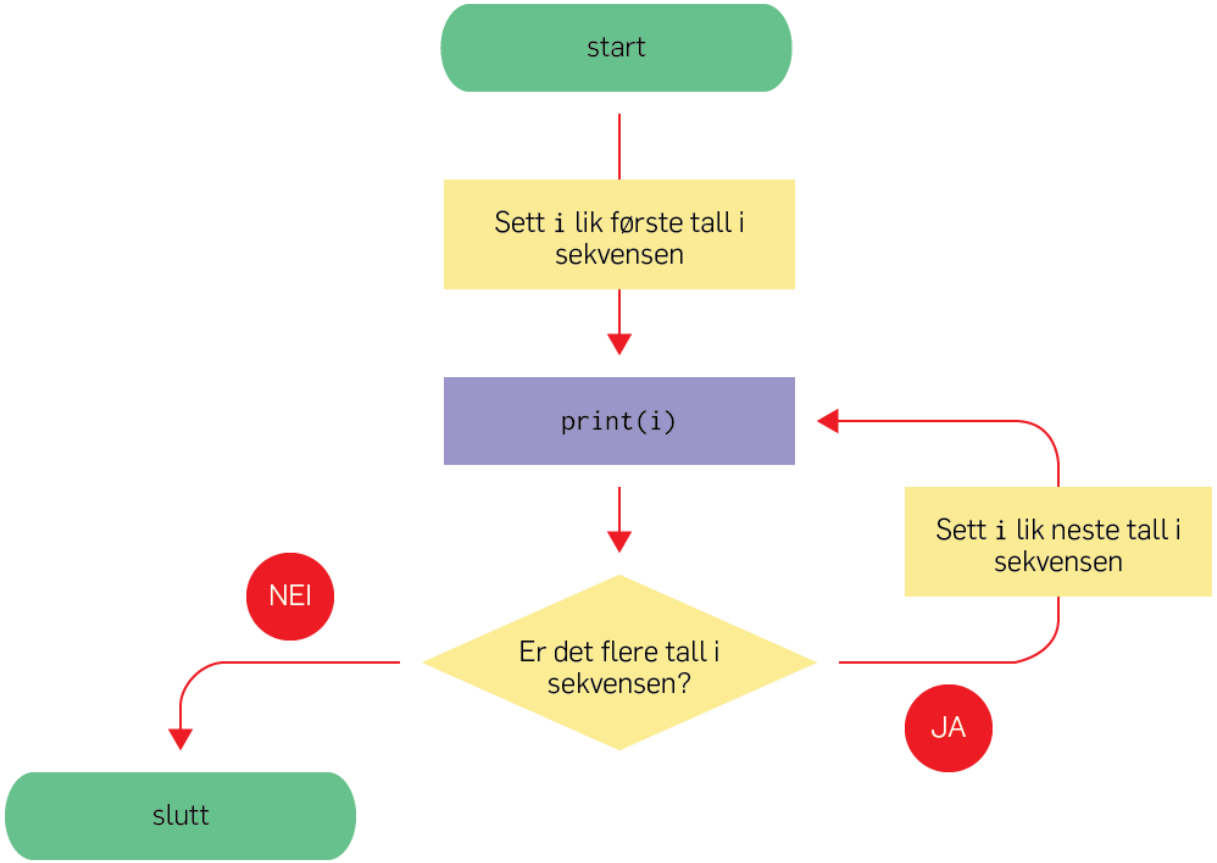
0	1	2	3	4	5	6	7	8	9

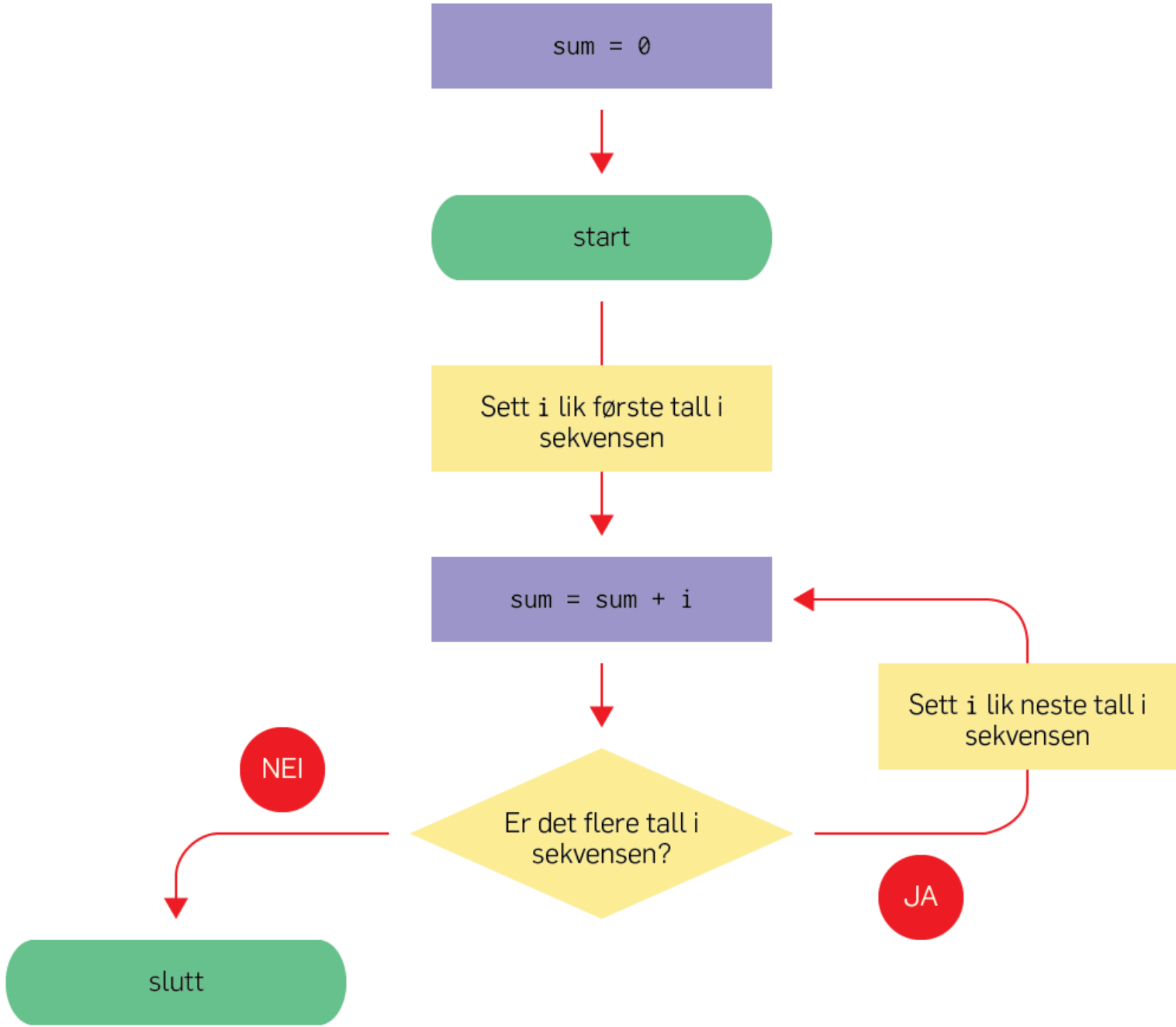
En **for**-løkke lar oss gjenta kode et bestemt antall ganger.

Vi (eller programmet) må vite på forhånd hvor mange gjentagelser vi ønsker.

Eksempel:

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





VALG OG BETINGELSER

Betingelser lar oss skrive programmer med flere ulike utfall.

Vi kan bruke **if** alene, vi kan legge til så mange ekstra-betingelser vi ønsker med **elif**, og vi kan legge til **else** på slutten for å samle opp alt som ikke fanges opp av de andre betingelsene.

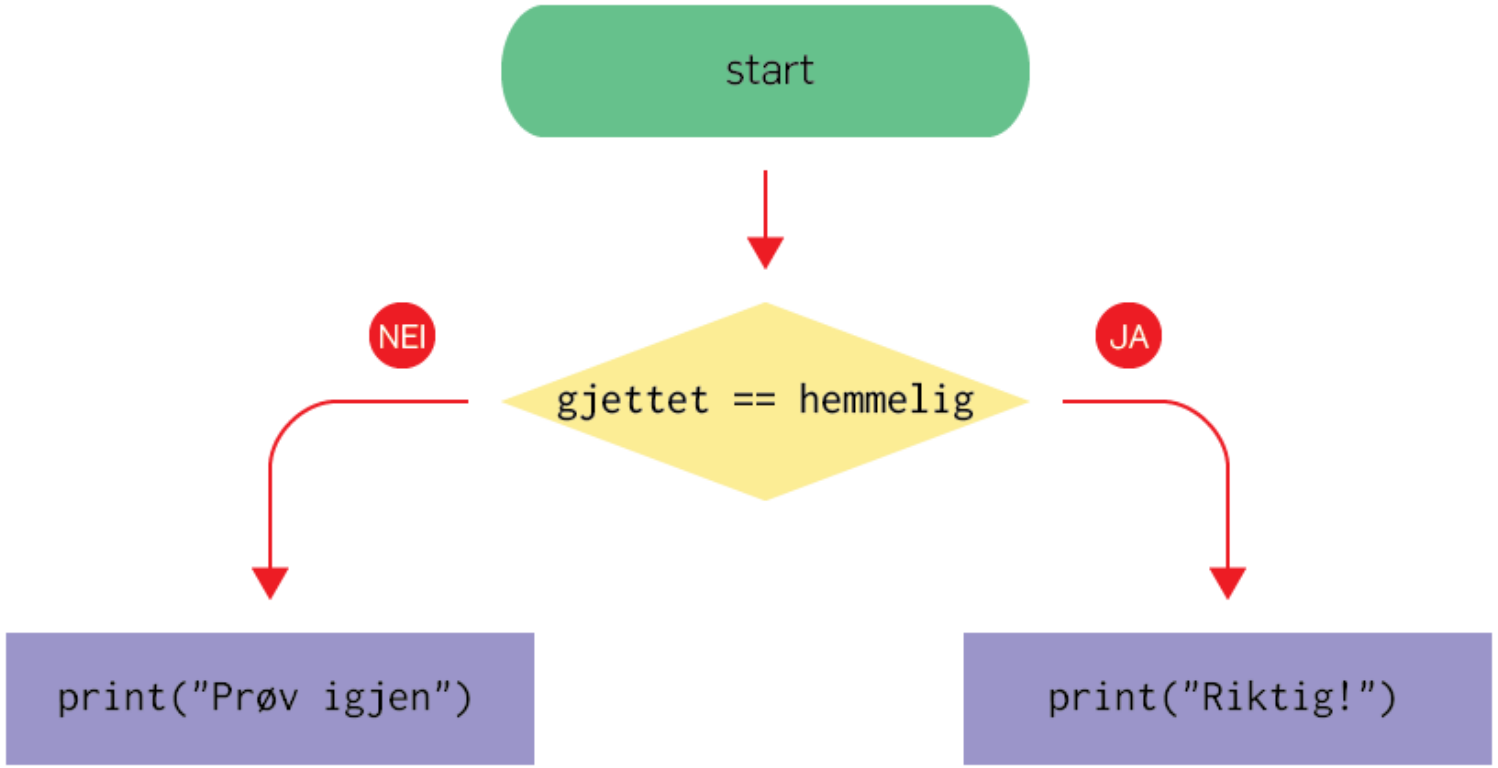
```
if <betingelse 1>:
  # kode

elif <betingelse 2>:
  # kode

else:
  # kode
```

VALG OG BETINGELSER

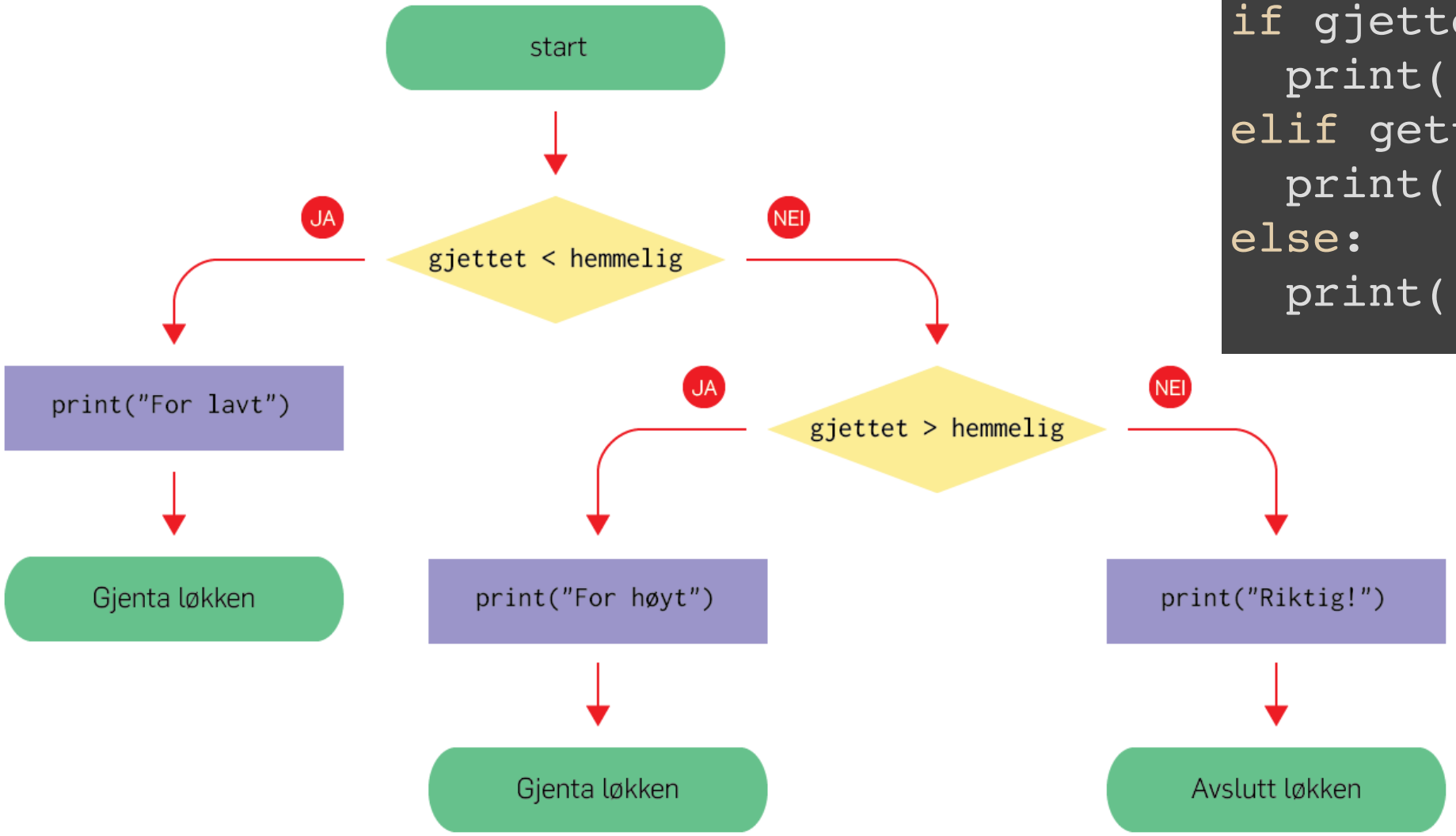
KAN DU GJETTE DET HEMMELIGE TALLET?




```
if gjettet == hemmelig:
  print("Riktig!")
else:
  print("Prøv igjen")
```

VALG OG BETINGELSER

KAN DU GJETTE DET HEMMELIGE TALLET?




```
if gjettet < hemmelig:
  print("For lavt")
elif gettet > hemmelig:
  print("For høyt")
else:
  print("Riktig")
```

ALGORITMISK TANKEGANG

Fra Wikipedia: **Algoritmisk tenkning** er den norske oversettelsen av det engelske *computational thinking*. En av mange definisjoner går ut på at algoritmisk tenkning er "tankeprosessen knyttet til å formulere et problem og dets løsning(er) på en slik måte at det kan løses av en informasjonsprosesser (menneske eller maskin)".

Det dreier seg altså om problemløsning.

Den viktigste delen av programmeringsarbeidet foregår utenfor skjermen.

Vi kan se på et eksempel:

Du skal lage et program som skriver ut primtallsfaktoriseringen til et vilkårlig tall.

8. TRINN

- utforske og beskrive primtalsfaktorisering og bruke det i brøkrekning
- utforske korleis algoritmer kan skapast, testast og forbetrast ved hjelp av programmering

PRIMTALLSFAKTORISERING AV 420

MED "PENN OG PAPIR"

420	**2**
210	**2**
105	**3**
35	**5**
7	**7**
1	

$$420 = 2 \cdot 2 \cdot 3 \cdot 5 \cdot 7$$

Hvordan kan vi generalisere denne prosessen?

Kan vi skrive en oppskrift (algoritme) som vi kan følge og som vil gi oss et faktorisert resultatet uansett hvilket tall vi starter med?

Hva må gjentas? Hvor mange ganger?

PRIMTALLSFAKTORISERING AV 420

ALGORITMEHINT

420	2	105	3	35	4	35	5	7	6	7	7
210	2	35	3			7	5				
105	2										

1 | 8

osv...

1 | 419

Vi kan prøve å dele på **alle** heltall (i stedet for å bare dele på primtallene som vi tror fungerer).

Hvorfor går det bra?