Hemmelige koder

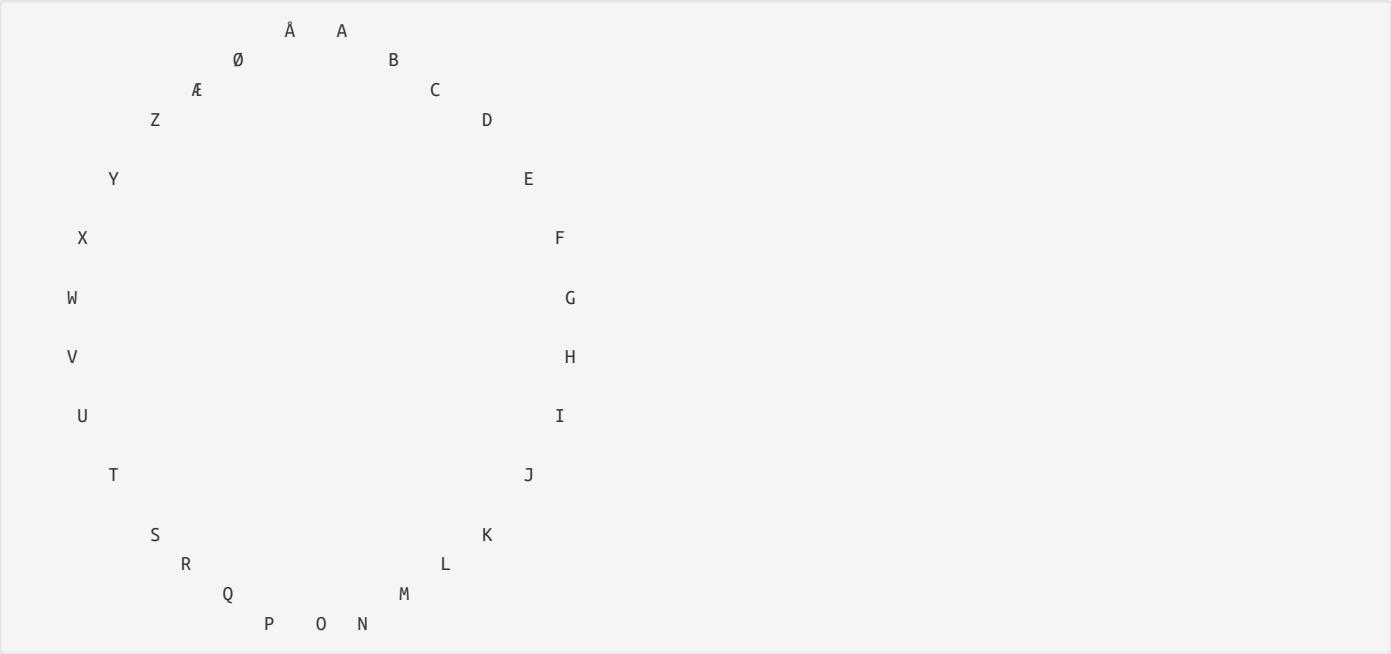
Introduksjon

Legg bort skilpaddene dine, i dag skal vi lære hvordan vi kan sende hemmelige beskjeder!

Kodeklubb-koden

Et chiffer er et system for å gjøre om vanlig tekst til kode som ikke andre skal kunne lese. Vi skal bruke et av de eldste og mest berømte chifferene, Cæsar-chifferet eller Cæsars kode - oppkalt etter Gaius Julius Cæsar som sannsynligvis brukte det til å sende hemmelige beskjeder. Det er neppe den beste måten å hindre andre i å lese beskjedene dine, men det kommer vi tilbake til. Det finnes ferdige moduler til Python du kan bruke hvis du vil lage noe som skal være vanskelig å knekke, men nå skal vi forsøke å lage Cæsar-chifferet selv.

Start med å tegne alle bokstavene i en sirkel.



For å lage en hemmelig bokstav fra en vanlig bokstav, trenger vi et tall vi kan bruke som hemmelig nøkkel. Jeg liker tallet 3, det er et magisk tall, så vi bruker det.

```
A + 3 = D      T + 3 = W      Å + 3 = C
```

Vi begynner med A og teller fremover 3 bokstaver: B, C, D. Så bokstaven A blir til bokstaven D. For å dekode gjør vi det samme, men baklengs. Vi begynner med D og teller bakover for å få A.

Steg 1: Alfabetet

Her kan du få trøbbel med norske bokstaver om du ikke har Python 3. Du har Python 2 dersom det står 2.6 eller 2.7 i IDLE. I så fall må du legge en `u` foran tekst som er inni `""`. For eksempel blir alfabetet under `u"abcdefghijklmnopqrstuvwxyzæøå"`. `u`-en betyr at teksten er av typen *Unicode* som støtter alle norske bokstaver.

Sjekkliste

- [ ] Først må vi lære python alfabetet. Åpne IDLE og lag en ny fil med koden under:

```
alphabet = "abcdefghijklmnopqrstuvwxyzæøå"

print(len(alphabet))
```

- [ ] Når du kjører dette programmet skal det skrive ut 29. Pass på at du har med alle bokstavene, ellers kommer ikke den hemmelige koden din til å virke.

  Hvis du er fornøyd med alfabetet ditt kan vi begynne å kode en bokstav.

Steg 2: Kode en bokstav

Sjekkliste

- [ ] Akkurat som vi gjorde med hjulet ovenfor kan vi finne posisjonen til en bokstav ved å telle forover, og så bruke bokstaven vi ender opp med.

  Skriv inn koden under og kjør den:

```
alphabet = "abcdefghijklmnopqrstuvwxyzæøå"

letter = "a"
secret = 3

pos = alphabet.find(letter)

newpos = (pos + secret)

if newpos >= 29:
    newpos = newpos - 29

secretletter = alphabet[newpos]

print(secretletter)
```

  Vi slår opp hvor “a” er i alfabetet og legger til det hemmelige tallet vårt for å telle fremover. Vi sjekker om vi har gått rundt, hvis vi har det må vi gå en hel runde tilbake igjen ved å trekke fra 29 (dette er litt som med gradene, å trekke fra 360 gjør at vi er akkurat der vi var). Så slår vi opp i alfabetet igjen for å se hvilken hemmelige bokstav vi fikk.

- [ ] Kjør koden og se hva som skjer.

- [ ] La oss ta en titt på koden igjen, men vi tar det sakte.

  Du trenger ikke å skrive dette! Alt som står bak firkant-tegnet bryr python seg vanligvis ikke om, det er bare kommentarer til mennesker som skal lese koden.

```
# alphabet er navnet på teksten fra a til å
alphabet = "abcdefghijklmnopqrstuvwxyz"

# Den hemmelige bokstaven (letter) og det hemmelige tallet
# (secret) vi bruker for å kode det
letter = "a"
secret = 3

# Finn posisjonen til bokstaven. Python vil gi oss et
# tall fra 0 til 28 (python teller fra 0)
pos = alphabet.find(letter)

# Gå like langt fremover som det hemmelige tallet sier
newpos = (pos + secret)

# Hvis vi har telt for langt, må vi gå en runde tilbake
# for å få et tall mellom 0 og 28
if newpos >= 29:
    newpos = newpos - 29

# Slå opp denne posisjonen for å se hvilken bokstav
# i alfabetet som står der
secretletter = alphabet[newpos]

# Skriv denne bokstaven ut på skjermen
print(secretletter)
```

Det er mye python-ting som skjer her, men ikke bli skremt om du ikke forstår alt til å begynne med. Mye av dette er akkurat som i scratch. `if newpos >= 29` er bare en `if`-setning, en ting som bare kjører koden under hvis det som står etter `if` er sant. En `if`-setning bruker en innrykksblokk, akkurat som `for` og `def` som vi har sett tidligere.

Nå som vi kan kode en bokstav, hva med å dekode en?

Steg 3: Finne tilbake bokstavene

Akkurat som i koden fra den forrige oppgaven skal vi finne posisjonen til bokstaven, men denne gangen skal vi gå bakover i alfabetet for å dekode.

Sjekkliste

- [ ] Forsøk å skriv inn denne koden og kjør den:

```
alphabet = "abcdefghijklmnopqrstuvwxyzæøå"

secret = 17
secretletter = "r"

pos = alphabet.find(secretletter)

newpos = pos - secret

if newpos < 0:
    newpos = newpos + 29

letter = alphabet[newpos]

print(letter)
```

Steg 4: Bygge funksjoner

La oss ta den første programkoden (som laget Cæsar-kode av bokstaver) og gjøre den om til en funksjon `encode` og den andre programkoden til en funksjon `decode`. I modul to snakket vi om å bruke prosedyrer for å unngå gjentagelser, denne gangen skal vi lage funksjoner i stedet. Ved første øyekast er prosedyrer og funksjoner veldig like (og Python bryr seg strengt tatt ikke om forskjellen), men når man ser nærmere etter har

de ulike egenskaper. Dette spiller ikke så stor rolle nå, men det er like greit å lære seg forskjellen med en gang.

Prosedyrer bare gjør ting, mens funksjoner bare beregner ting. Noen ganger blander man de to tingene sammen og lager prosedyrer som både beregner og gjør ting, men jo mer du kan skille disse fra hverandre jo enklere blir programmet ditt både for deg selv og for andre. Grunnen til at vi kaller det vi lager nå for funksjoner, er at de bare beregner en verdi - de skriver ingenting ut, de tegner ingenting på skjermen og resultatet blir likt hver eneste gang hvis man gir inn samme bokstav og hemmelige tall.

For å få en funksjon (eller en prosedyre som beregner noe) til å returnere en verdi som vi kan bruke senere, bruker vi kommandoen `return`.

Sjekkliste

- [ ] Lag en fil som ser slik ut:

```
alphabet = "abcdefghijklmnopqrstuvwxyzæøå"

def encode(letter, secret):
    pos = alphabet.find(letter)

    newpos = (pos + secret)

    if newpos >= 29:
        newpos = newpos - 29

    return alphabet[newpos]


def decode(letter, secret):
    pos = alphabet.find(letter)

    newpos = (pos - secret)

    if newpos < 0:
        newpos = newpos + 29

    return alphabet[newpos]

print(encode("a", 17))
print(decode("r", 17))
```

  Husk at du kan bruke 'Tab' i IDLE for å få innrykk. Du kan også merke deler av koden og rykke alt inn på en gang.

- [ ] Prøv å kode og dekode noen bokstaver!

Steg 5: Send et hemmelig ord eller to, og finn dem tilbake igjen

Nå har vi noen funksjoner, la oss bruke dem til å kode ord. Vi kommer til å gå igjennom hver bokstav i ordet og kode det hvis det finnes i alfabetet (vi hopper over tegn som punktum og mellomrom).

Sjekkliste

- [ ] Under de nye funksjonene fra forrige oppgave kan du skrive inn koden under (med andre ord: behold det du gjorde i oppgave 4, og legg til koden under).

```
secret = 17
message = "hello world"

output = ""

for character in message:
    if character in alphabet:
        output = output +  encode(character, secret)
    else:
        output = output + character


print(output)

secret = 17
message = "yvåac kcfåu"
output = ""

for character in message:
    if character in alphabet:
        output = output + decode(character, secret)
    else:
        output = output + character

print(output)
```

- [ ] Kjør programmet og se hva som skjer.

  Den første delen av koden burde skrive ut “yvåac kcfåu”, som er den hemmelige versjonen av “hello world”. Den andre delen dekoder det igjen og skriver ut “hello world”

Steg 6: Dekoding av noen hemmelige beskjeder

Her er noen hemmelige beskjeder, forsøk å dekode dem!

- [ ] `daczj ym cgyzcdmwwzf?`, hemmeligheten er 21.

- [ ] `æxkxånwn næ bnwwnwn mrwn`, hemmeligheten er 9.

Prøv å sende noen beskjeder til vennene dine! Hva med å lage et Python-program som forsøker seg på alle mulige hemmelige tall og forsøker å knekke koder selv om du ikke kan det hemmelige tallet?