Øvingsforelesning 5 Python (TDT4110)

Repetisjon av løkker og funksjoner

Ole-Magnus Pedersen

Kunnskap for en bedre verden

Oversikt

- Praktisk Info
- Gjennomgang av Øving 3
- Repetisjon

Praktisk info

- Prosjekter i PyCharm må startes med Python 3.x
- Idle på mac:
  - Installer ny versjon av Tcl (for eksempel ActiveTcl, som omtalt her)
  - Problemer med backslash (løsning fra StackOverflow):
    - *Preferences* → *Keys*
    - Under *Custom Key Bindings*, finn *expand-word*
    - Endre kombinasjonen til noe annet (f.eks. Control-Option-Key-Slash)

QUEUEME

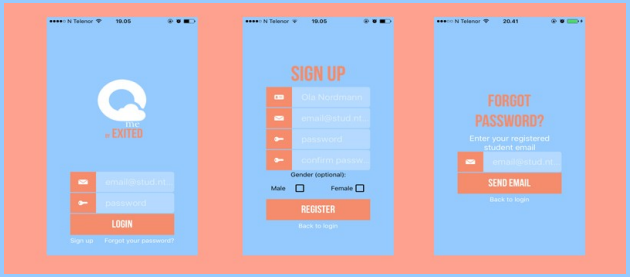
STOP WASTING TIME IN LINE.

DOWNLOAD THE APP OR VISIT QUEUEME.NO

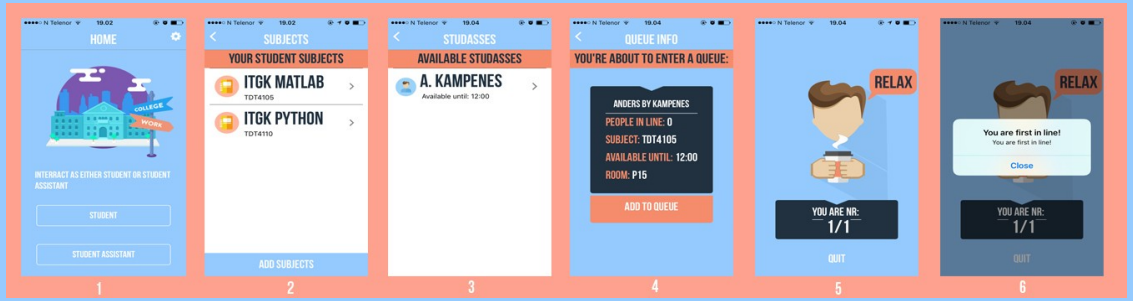
Available on the iPhone App Store

ANDROID APP ON Google play

Innlogging


N Telenor
10.05
email@stud.ntnu...
password
LOGIN
Sign up
Forgot your password?
SIGN UP
Ola Nordmann
email@stud.ntnu...
password
confirm passw...
Gender (optional):
Male
Female
REGISTER
Back to login
N Telenor
20.41
FORGOT PASSWORD?
Enter your registered student email
email@stud.ntnu...
SEND EMAIL
Back to login

Student


HOME
COLLEGE
WORK
INTERRACT AS EITHER STUDENT OR STUDENT ASSISTANT
STUDENT
STUDENT ASSISTANT
1
SUBJECTS
YOUR STUDENT SUBJECTS
ITGK MATLAB
TDT4105
ITGK PYTHON
TDT4110
ADD SUBJECTS
2
STUDASSES
AVAILABLE STUDASSES
A. KAMPENES
Available until: 12:00
3
QUEUE INFO
YOU'RE ABOUT TO ENTER A QUEUE:
ANDERS BY KAMPENES
PEOPLE IN LINE: 0
SUBJECT: TDT4105
AVAILABLE UNTIL: 12:00
ROOM: P15
ADD TO QUEUE
4
RELAX
YOU ARE NR:
1/1
QUIT
5
RELAX
You are first in line!
You are first in line!
Close
YOU ARE NR:
1/1
QUIT
6

Oversikt

- Praktisk Info
- Gjennomgang av Øving 3
- Repetisjon

Gjennomgang av Øving 3

- Alternerende sum
- Doble løkker

Oversikt

- Praktisk Info
- Gjennomgang av Øving 3
- Repetisjon

Variabler

- Peker på en minneplassering med data
- Kan sette verdien, bruke verdien og sette verdien på nytt
- Har en type (f.eks. `int`, `float`, `boolean`, `str`)
- NB! Må initialiseres før den brukes

```
# Dette går ikke
if a > 5:
  pass

# Vi må initialisere a
   først
a = 3
if a > 5:
  pass
```

Betingelser og if-setninger

- Betingelser er boolske uttrykk eller variabler, og har verdi `True` eller `False`
- Kan sammenligne variabler/konstanter med `==`, `!=`, `>`, `>=`, `<`, `<=`
- Spesialord som `and`, `or`, `not`, `is`, `in`
- Ved å bruke if-setninger utføres kode kun hvis betingelsen stemmer

```
if condition:
  pass
elif condition_2:
  pass
else:
  pass
```

Løkker

- Kjører koden flere ganger
- For-løkker kjører et gitt antall ganger, f.eks. bestemt ved å bruke `range()`
- While-løkker kjører så lenge en betingelse er oppfylt
- `break` og `continue` kan brukes for å avslutte enten hele løkka eller en iterasjon

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

x = 0
while x < 10:
  x += 1
```

Løkker 1 (for-løkke)

— Lag et program som tar inn en streng (*s*) og et tall (*x*). Programmet skal skrive ut hver *x*te bokstav i *s*, fra og med den første.

Eksempel på kjøring

Skriv inn en streng: abcDeFGhiJKL
Skriv inn et tall: 3

a
D
G
J

Løkker 2 (while-løkke)

— Lag et program som spør om et passord fra bruker. Hvis passordet er likt et passord du bestemmer, skal brukeren få beskjed om at passordet er riktig. Hvis ikke skal programmet spørre om et nytt passord.

Eksempel på kjøring

```
# passordet er ”qwerty”
Passord: abcd
Passordet er feil. Prøv igjen.
Passord: qwerty
Passordet er riktig!
```

Løkker 3 (while-løkke)

— Lag et program som tar inn to tall, n og k. Programmet skal summere alle tallene fra 0 til n, eller fram til summen er større enn k. Programmet skal skrive ut ”Sum ved i=<iterasjon>: <nåværende sum>” i hver iterasjon, og skal bruke en while-løkke.

— Hvis programmet avslutter fordi $i > n$ skal programmet skrive ut ”Etter <n> iterasjoner er summen <sum>”. Ellers skal programmet skrive ut ”Etter <antall iterasjoner> overgikk summen <k>”.

Eksempel på kjøring

n: 10
k: 7
Sum ved i=0: 0
Sum ved i=1: 1
Sum ved i=2: 3
Sum ved i=3: 6
Sum ved i=4: 10
Etter 5 iterasjoner overgikk summen 7.

Løkker 4 (sammensatte løkker)

— Lag et program som tar inn et heltall fra brukeren (kalt x her). Deretter skal programmet spørre brukeren om svaret på $x \times i$ for alle i fra 1 til 10. Hvis brukeren svarer riktig skal programmet skrive ut det, ellers skal det spørre igjen til brukeren har riktig svar.

Eksempel på kjøring

Skriv inn et tall: 10
Hva er 1×10? 10
Riktig!
Hva er 2×10? 21
Prøv igjen. Hva er 2×10? 20
Riktig!
...

Funksjoner

- Definer kode som kjøres senere i programmet
- Parametre er input-verdier til funksjonen
- Kan returnere en eller flere output-verdier

```
def func(param1, param2):
  # Do something with params, here we
    add them and return the result
  res = param1 + param2
  return res

# Now we can use the function multiple
    times
many_adds = func(func(1, 3),
    func(func(2, 4), 5))
print(many_adds) # prints the result of
    (1 + 3) + ((2 + 4) + 5)
```

Scopes

- Variabler er gjeldende ut i fra hvor de defineres
- Variabler definert i en funksjon kan bare brukes i den funksjonen
  - Overskriver også variabler med samme navn i fra utenfor funksjonen
- Variabler definert utenfor funksjoner kan refereres i funksjoner, men man må bruke `global`-nøkkelordet for å endre verdien

```
g = 5
def func1():
  g = 3
  print(g)
def func2(g):
  print(g)
def func3():
  print(g)
def func4():
  global g
  g = 2
```

```
print(g) # 5
func1()  # 3
func2(4) # 4
func3()  # 5
print(g) # 5
func4()
print(g) # 2

def func5():
  h = 1
func5()
print(h) # Gir
   feilmelding
```

Funksjoner 1

— Utvid programmet fra *Løkker 4* til å kjøre i en funksjon som tar inn x som et parameter. Utvid programmet til å også ta inn y som et parameter og spør om svar på $x \times i$ for alle i fra 1 til y.

Funksjoner 1

— Utvid programmet fra *Løkker 4* til å kjøre i en funksjon som tar inn x som et parameter. Utvid programmet til å også ta inn y som et parameter og spør om svar på $x \times i$ for alle i fra 1 til y.

— Del koden opp i to funksjoner, en som kjører for-løkka, og en som spør bruker om svaret på regnestykket (til svaret er riktig).

Funksjoner 2

— Lag en funksjon som tar inn to strenger. Funksjonen skal returnere `True` hvis strengene er like, eller hvis den ene strengen er lik den reverserte andre strengen.
  - Hint: For å sjekke om strengene er reversert av hverandre kan du sjekke om de er like lange og sjekke at `str1[i] == str2[-i - 1]` i en for-løkke fra 0 til (men ikke med) `len(str1)`

— Lag så et program som kontinuerlig tar inn to strenger fra brukeren, og hvis de er like eller reversert skriver ut ”Dette var en match”, og ellers skriver ut ”Dette var ikke en match”. Programmet skal avslutte hvis begge strengene er lik ”q”.

Eksempel på kjøring

Str1: abc
Str2: abc
Dette var en match
Str1: abc
Str2: cba
Dette var en match
Str1: abc
Str2: abcd
Dette var ikke en match
Str1: q
Str2: q

Funksjoner 3

— Lag et program som gjør to ting: Først spør den brukeren om et adjektiv, og skriver ut komparativ-formen av dette. Deretter ber den brukeren om å skrive inn komparativ-formen av noen adjektiv (snill, pen, ung, gammel, lik), og skriver ut om brukeren skriver inn riktig svar.

— For de fleste norske adjektiv finnes komparativ ved å legge til -ere på slutten (snill - snill*ere*), men programmet skal også ta hensyn til ordene ung (yngre) og gammel (eldre).

Eksempel på kjøring

Skriv inn et adjektiv: rik
Komparativ av rik er: rikere
Hva er komparativ av snill? snillere
Riktig!
Hva er komparativ av pen? penn
Feil. Komparativ av pen er penere
Hva er komparativ av ung? yngre
Riktig!
...

Funksjoner 4

— Lag et program som sjekker om du kommer med på en kino. Dette er avhengig av 2 ting:
  - Har du råd? Lag en funksjon som tar inn hvor mange penger du har, og hva biletten koster, og returnerer true om du har råd og false om du ikke har det. Anta at filmen koster 120kr
  - Kommer du tidsnok? Lag en funksjon som tar inn antall personer i køen, og antall minutter til filmen starter. Funksjonen skal returnere True om du kan rekke filmen. Anta at man bruker et halvt minutt per person i køen, og at det tar et minutt å gå fra køen til salen.

— Programmet skal kontinuerlig spørre bruker om relevante tall, og si om brukeren kommer med på kinoen.

Eksempel på kjøring

```
Hvor mye penger har du? 50
Du kommer ikke med!
Hvor mye penger har du? 120
Hvor lenge er det til filmen starter?
20
Hvor mange er foran deg i køen?
35
Du kommer med!
```

Funksjoner 5

— Lag en funksjon som tar inn et tall, *n*, som parameter. Programmet skal returnere det *n*te fibonacci-tallet.
  - Fibonacci-tallene er definert på denne måten: $f_0 = 0, f_1 = 1, f_n = f_{n-1} + f_{n-2}$

— Bruk funksjonen i et program som tar inn to tall fra brukeren, *n* og *x*, der *n* er hvilket fibonacci-tall vi er interessert i og *x* er brukerens gjettning på det *n*te fibonacci-tallet. Hvis brukeren gjetter riktig skal programmet printe ut ”Det var riktig”. Ellers skal det printe ”Feil. Det <n-te> fibonacci-tallet er <n-te fibonacci tall>”.

Eksempel på kjøring

n:15
Hva tror du det 15. fibonacci tallet er? 610
Dette var riktig
#—- ny kjøring —-
n:15
Hva tror du det 15. fibonacci tallet er? 500
Feil. Det 15. fibonacci tallet er 610

Debugging

- Måten å gå fram på for å finne feil i koden
- To problemstillinger:
  - Programmet kjører ikke, og vi får en feilmelding
    - Les feilmelding tydelig
    - Vi får typisk beskjed om hva som er feil, og hvor i koden feilen er
    - Tips: Noen ganger må vi se på linjen før eller etter det man får beskjed om
  - Programmet kjører, men gir feil resultat
    - Logg relevante verdier flere steder i koden (i dette faget kan dere printe ut verdien)
    - PyCharm (eller andre IDE-er) har debuggingsverktøy der du kan stoppe koden og se verdien av variabler på det tidspunktet