NTNU

Kunnskap for en bedre verden

Kunnskap for en bedre verden

TDT4110 Informasjonsteknologi grunnkurs:
Tema: Funksjoner med retur og moduler
Utgave 3: Kap. 5.7 - 5.10

Terje Rydland - IDI/NTNU

Auditorieøving

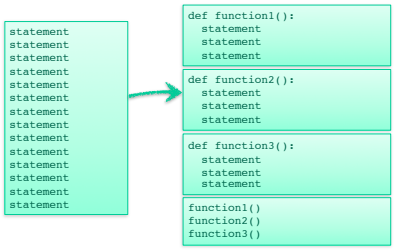
- Auditorieøvingstider for alle linjer er postet på forsiden av wikien (under «Siste nytt»).
- Studenter som ikke kan møte til oppsatt tidspunkt, kan
  - 1) sjekke forsiden igjen (det står instrukser om dette her)
  - 2) sjekke «Ofte stilte spørsmål»-siden på wikien.
- Det kommer mer informasjon om en oppsamlingsauditorieøving (for aud. øving 1) ila. neste uke, hvis noen absolutt ikke har mulighet til å møte til ordinær aud. øving i uke 40.

Læringsmål og pensum

- Mål
  - Beherske returverdier og funksjoner som returnerer verdi(er)
  - Bruke matematikkbibliotek i Python
  - Generere tilfeldige tallverdier
  - Lage egne moduler
- Pensum
  - Starting out with Pyton:
    - Kapitler 5.7-5.10: Value-Returning Functions and Modules

Funksjoner

- Enkel funksjon:
  - grupperte uttrykk for å utføre en viss oppgave



- Returverdifunksjoner:
  - Tilsvarende ovenfor, men returnerer en (eller flere) verdi(er)

Standard Library-funksjoner og import-uttrykket

- Standardbibliotek: bibliotek inneholdende ferdige funksjoner som kommer med Python
  - Biblioteksfunksjoner utfører oppgaver som programmerere ofte trenger
  - Eks: print(), input(), range()
  - Betraktes gjerne (men ikke nødvendigvis(!) som «black box»
- Noen bibliotekfunksjoner er bygd inn i Python-tolkeren
  - Bare kall funksjonen for å bruke den
  - Eksempler
    - print(), input(), int(), float(), range()

Standard Library-funksjoner og import-uttrykket

- En funksjon inne i et bibliotek, kan sees på som en svart boks, dvs. at vi bruker funksjonene uten å vite hva som foregår inne i den svarte boksen.

Eks (innebygd funksjon):

```
range(15)
```

Å generere tilfeldige tall

- Tilfeldig genererte tall er nyttige i mange programmeringsoppgaver
- Random-modulen
  - inneholder bibliotekfunksjoner til arbeid med tilfeldige tall
- «Dot notation»: notasjon for å kalle funksjoner som tilhører en gitt modul:
  - Format
    - module_name.function_name()
  - Eks
    - random.random()

Å generere tilfeldige tall (forts.)

- Funksjonen **randint**:
  - Genererer et **pseudo-tilfeldig heltall** i intervallet spesifisert gjennom argumentet
  - Returnerer det tilfeldige tallet til en annen del av programmet som kalte på denne funksjonen
  - Det returnerte tallet kan brukes hvor som helst et heltall kan brukes
  - Dette kan du eksperimentere med i interaktiv modus

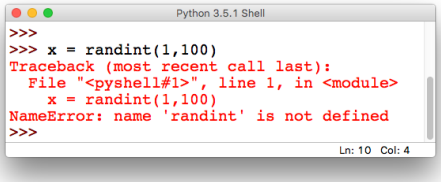
Eks (fra random-biblioteket):

```
randint(1,100)
```

Randint

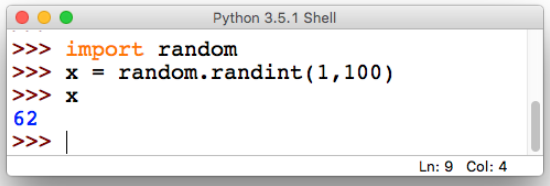
Feil:

Prøve å bruke randint direkte



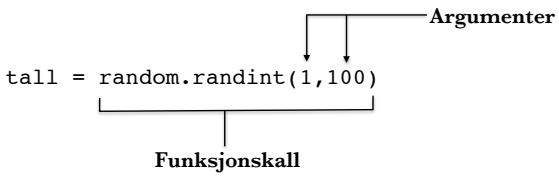
Korrekt:

Først importere random
Så bruke dot-notasjon



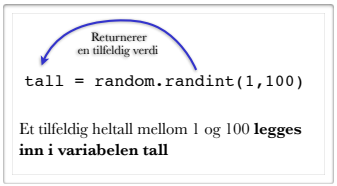
Å generere tilfeldige tall (forts.)

- Kall av random-funksjon med to argumenter:
  - To input argumenter
  - Returnerer tilfeldig heltall som typisk lagres i en variabel



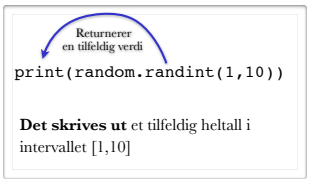
Å generere tilfeldige tall (forts.)

Retur av verdi fra random-funksjon:



Et tilfeldig heltall mellom 1 og 100 **legges inn i variabelen tall**

Vise et tilfeldig tall i konsollet vha. random-funksjon:



Det skrives ut et tilfeldig heltall i intervallet [1,10]

Å generere tilfeldige tall

- Funksjonen **randrange**
  - tilsvarende funksjonen range, men returnerer et tilfeldig valgt heltall fra den resulterende sekvensen
  - Samme argumenter som for funksjonen range
- Funksjonen **random**
  - returnerer et tilfeldig flyttall i [0.0, 1.0 >
- Funksjonen **uniform**
  - returnerer et tilfeldig flyttall, men tillater brukeren å spesifisere et intervall

`random.random()`	gir tilfeldig flyttall mellom 0 og 1 (f.eks. 0.15986478)
`random.randrange(0,101,4)`	gir tilfeldig heltall mellom 0 og 100 fra listen [0,4,8,12,16…92,96,100]
`random.uniform(1,10)`	gir tilfeldig flyttall mellom 0 og 10 (f.eks. 8.126987542)

Seeds for tilfeldige tall

- Tilfeldige tall laget av funksjoner i modulen random er egentlig **pseudo-tilfeldige** tall - de beregnes.
- Seed-verdi
  - initialiserer formelen som genererer de tilfeldige tallene
  - Du trenger å bruke forskjellige seeds for å få forskjellige serier med tilfeldige tall
    - Default-verdien for seeds er systemtiden
    - Du kan bruke funksjonen **random.seed**() for å sette ønsket seed-verdi

Oppgave: random

- Skriv Python-koden for å skrive ut 10 tilfeldige tall i 4 gangen til konsoll ved hjelp av random-biblioteket.

Oppgave: random

- Skriv Python-koden for å skrive ut 10 tilfeldige tall i 4 gangen til konsoll ved hjelp av random-biblioteket.

```
import random
for x in range(10):
    print(random.randrange(4,41,4))
```

kode: firegangen.py

Å skrive egne returverdi-funksjoner Kap. 5.7

- For å skrive en returverdifunksjon skriver du en *enkel funksjon* og legger til en eller flere return-uttrykk
  - Format: `return` uttrykk
    - Verdien til uttrykk vil returneres til den delen av programmet som kalte på funksjonen
    - Uttrykket i utrykket `return` kan være et komplisert uttrykk som summen av to *variable* eller resultatet av en annen returverdifunksjon

Untitled.py - /Users/terjery/Library/Mobile Do...

```
def mult(a,b):
    c = a*b
    return c

a = mult(2,4)
print(a)
```

Ln: 7 Col: 0

Untitled.py - /Users/terjery/Library/Mobile Do...

```
def mult(a,b):
    return a*b

a = mult(2,4)
print(a)
```

Ln: 2 Col: 14

Oppbygning av en funksjon.

- Når man definerer/lager en funksjon bruker man det reserverte ordet `def`
- All koden i funksjonen (kroppen) skrives med *innrykk*.
- Avslutter med `return` (når noe skal returneres)
- Funksjoner kan også skrives i den interaktive omgivelsen

Navn på funksjon

Parametere

```
def funksjonsnavn (para1, para2, para3):
    kode…
    kode…
    return variabel
```

Funksjonskropp

Avslutning

Kall av funksjoner

- Ved kall av en funksjon brukes ***funksjonsnavnet***, samt en liste av ***argumenter*** (verdier) som matcher navnet på funksjonen og lista av parametere:

Parametere

```
def arealRektangel (b,h):
    # Beregne areal av rektangel
    return b*h

>>> arealRektangel(20,4)
80
```

Argumenter

Resultatet

- Et program kan f.eks. se slik ut

*Untitled.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/...

```
def arealRekt(lengde,bredde):
    return lengde*bredde

l = int(input('Oppgi lengde: '))
b = int(input('Oppgi bredde: '))
areal = arealRekt(l,b)

print('Arealet blir:',areal)
```

Ln: 9 Col: 0

Oppgave: funksjon

- Lag funksjonen celFar som omregner fra celcius til Fahrenheit ved bruk av følgende formel:
  - F = c * (9/5) + 32
- Benytt deg av `def` og `return`
- Skriv også koden for følgende:
  - Spør brukeren om temperatur i Celcius
  - Skriv ut temperaturen i Fahrenheit ved bruk av funksjonen celFar
  - Bruk `input` og `print`

kode: farenheit.py

Oppgave: funksjon

- Lag funksjonen celFar som omregner fra celcius til Fahrenheit ved bruk av følgende formel:
  - F = c * (9/5) + 32
- Benytt deg av `def` og `return`
- Skriv også koden for følgende:
  - Spør brukeren om temperatur i Celcius
  - Skriv ut temperaturen i Fahrenheit ved bruk av funksjonen celFar
  - Bruk `input` og `print`

*farenheit.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/Keynote/U...

```
def celFar(celsius):
    # funksjonen omregner fra Celcius til Farhenheit
    fahrenheit = celsius*(9/5) + 32
    return fahrenheit

cel = int(input('Oppgi temperatur i celsius: '))
fahr = celFar(cel)
print ('Temperatur i Fahrenheit', fahr)
```

Ln: 3 Col: 35

Å bruke IPO-kart

- IPO-kart: beskriver input (inndata), processing (prosessering) og output (utdata) til en funksjon
  - Verktøy for å designe og dokumentere funksjoner
  - Typisk presentert i kolonner
  - Har normalt korte beskrivelser av inndata, prosessering og utdata uten å gå i detalj
    - Inkluderer ofte nok informasjon til å erstatte et flytdiagram

IPO_Chart		
Input	**Prosessering**	**Output**

Å bruke IPO-kart (forts.)

IPO_Chart for `get_regular_price`		
Input	**Prosessering**	**Output**
Ingen	Spør brukeren om salgsobjektets normale pris.	Objektets normale pris

```
def get_regular_price():
   price=int(input("Regular price? "))
   return price
```


Å bruke IPO-kart (forts.)

IPO_Chart for `discount` funksjonen		
Input	**Prosessering**	**Output**
Objektets normalpris	Beregner rabatten ved å multiplisere den normale prisen med den globale konstaten `DISCOUNT_PERCENTAGE`	Rabatten som skal gis

```
DISCOUNT_PERCENTAGE=0.20

def discount(price):
   discount = price*DISCOUNT_PERCENTAGE
   return discount
```

Å returnere *boolske verdier*

- Boolsk funksjon: returnerer enten `True` eller `False`
  - Brukes til å teste et kriterium for beslutningsstrukturer og repetitive strukturerer
    - F.eks å repetere handlingen i en løkke til en teller har nådd et visst tall
    - Vanlige beregninger, som å sjekke om et tall er partall kan enkelt repeteres gjennom å kalle på en funksjon
  - Brukes til å forenkle sjekk av inndata for gyldighet
    - Du kan spørre om verdier, og bare godta enkelte verdier ved å returnere `True` hvis inndata er en av disse
  - Vi ser på et eksempel!

kode: tre_gangen.py

tre_gangen.py

```
def i_tre_gangen(tall):
    if (tall%3==0):
        return True
    else:
        return False

print('\nTester 3:',i_tre_gangen(3))
print('Tester 5:',i_tre_gangen(5),'\n')
```

Ln: 7 Col: 36

Python 3.5.1 Shell

```
 RESTART: /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/Keynote/Uke 40/Python/tre_gangen.py

Tester 3: True
Tester 5: False

>>>
```

Ln: 22 Col: 0

Kan også skrives slik:

```
def i_tre_gangen(tall):
    return tall % 3 == 0

print('\nTester 3:',i_tre_gangen(3))
print('\nTester 5:',i_tre_gangen(5),'\n')
```

Ln: 5 Col: 8

Et annet eksempel

```
def i_tre_gangen(tall):
    return tall % 3 == 0

for i in range(1,11):
    print(i,'i 3-gangen? ',i_tre_gangen(i))
```

Ln: 6 Col: 0

Dette programmet

gir denne utskriften

Python 3.5.2 Shell

```
1 i 3-gangen?  False
2 i 3-gangen?  False
3 i 3-gangen?  True
4 i 3-gangen?  False
5 i 3-gangen?  False
6 i 3-gangen?  True
7 i 3-gangen?  False
8 i 3-gangen?  False
9 i 3-gangen?  True
10 i 3-gangen?  False
>>>
```

Ln: 70 Col: 0

Å returnere flere verdier

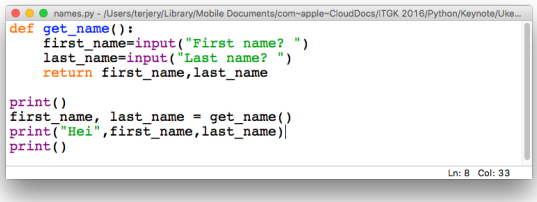
- I Python kan en funksjon returnere flere verdier
  - Spesifisert etter return-uttrykket, atskilt av kommaer
    - Format: `return uttrykk1, uttrykk2,` etc.
  - Når du kaller på en slik funksjon i et tilordningsuttrykk trenger du en egen variabel på venstresiden av = operatoren for å ta i mot hver returnert verdi:

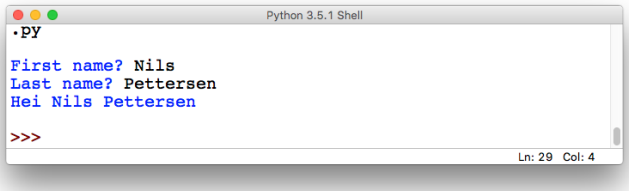
```
def get_name():
    first_name=input('First name? ')
    last_name=input('Last name? ')
    return first_name,last_name


first_name, last_name = get_name()
```

kode: names.py

names.py





Modulen `math`

Modulen math: er en del av et ***standardbibliotek*** som inneholder nyttige funksjoner for å utføre matematiske beregninger

- Typisk akseptere en eller flere verdier som argumenter, deretter å utføre matematiske operasjoner og returnere resultatet
- Bruk av modulen betinger et import math-uttrykk

Funksjoner i Modulen math:

`math` Module Function	Description
`acos(x)`	Returns the arc cosine of `x`, in radians.
`asin(x)`	Returns the arc sine of `x`, in radians.
`atan(x)`	Returns the arc tangent of `x`, in radians.
`ceil(x)`	Returns the smallest integer that is greater than or equal to `x`.
`cos(x)`	Returns the cosine of `x` in radians.
`degrees(x)`	Assuming `x` is an angle in radians, the function returns the angle converted to degrees.
`exp(x)`	Returns e^x
`floor(x)`	Returns the largest integer that is less than or equal to `x`.
`hypot(x, y)`	Returns the length of a hypotenuse that extends from (0, 0) to (`x`, `y`).
`log(x)`	Returns the natural logarithm of `x`.
`log10(x)`	Returns the base-10 logarithm of `x`.
`radians(x)`	Assuming `x` is an angle in degrees, the function returns the angle converted to radians.
`sin(x)`	Returns the sine of `x` in radians.
`sqrt(x)`	Returns the square root of `x`.
`tan(x)`	Returns the tangent of `x` in radians.

Modulen math (forts.)

- Modulen math definerer variablene pi og e, som er tilordnet de matematiske verdiene for **`pi`** og **`e`**
  - Kan brukes i likninger som fordrer disse verdiene for å få mer nøyaktige resultater
- Variabler må også kalles på ved å bruke *dot*-notasjonen:
  - Eks:

```
import math
radius = 12
circle_area = math.pi * radius**2
```


Å lagre funksjoner i moduler Kap 5.10

- I større, kompliserte programmer er det viktig å organisere koden godt
- Modularisering: å gruppere relaterte funksjoner i moduler
  - Gjør at programmene er lettere å forstå, teste og vedlikeholde
  - Gjør det lettere å bruke gode om igjen for flere programmer
    - Importer modulen som inneholder den fordrede funksjonen til hvert program som trenger den

Å lagre funksjoner i moduler (forts.)

- En Module er en fil som inneholder Python-kode
  - Inneholder funksjonsdefinisjoner, men inneholder ikke kall til disse funksjonene
    - Programmer vil importere modulene og kalle funksjonene
- Regler for modulnavn:
  - Filnavn skal slutte med .py
  - Kan ikke være det samme som et nøkkeluttrykk i Python
- Importer moduler med import-uttrykket

Menystyrte programmer

- Menystyrte programmer: Viser en liste av operasjoner på skjermen, som tillater brukeren å velge den ønskede handlingen
  - Liste av operasjoner som presenteres på skjermen kalles en *meny*
- Programmer bruker en beslutningsstruktur til å avgjøre hvilket menyvalg som velges, og gjennomfører den betingede operasjonen
- Typisk vel den gå i sløyfe til brukere avslutter

Funksjoner som bruker funksjoner – et større eksempel

- Romertallene fra 0 til 9 er som følger:
- 0=”, 1=I, 2=II, 3=III, 4=IV, 5=V, 6=VI, 7=VII, 8=VIII, 9=IX
- Det er en struktur på tallene som også viser seg i 10 og 100-posisjonen
  - 1 2 3 4 5 6 7 8 9: `I II III IV V VI VII VIII IX X`
  - 10 20 30 40 50 60 70 80 90 : `X XX XXX XL L LX LXX LXXX XC C`
  - 100 200 300 400…1000: `C CC CCC CD D DC DCC DCCC DM M`
- dette kan vi utnytte til å skrive et generelt program for å generere romertall ut fra et latinsk tall
- Vi starter med å skrive et program som kan konvertere et latinsk siffer (0…9) til et romertall

Funksjoner som bruker funksjoner – et større eksempel

- Vi skal lage funksjonen `romertall` som tar inn et tall mellom 0 og 9 og returnerer et romertall som svarer til tallet.
- Romertallene fra 0 til 9 er som følger:
- `0='', 1=I, 2=II, 3=III, 4=IV, 5=V, 6=VI, 7=VII, 8=VIII, 9=IX`
- Lag også koden slik at en bruker kan skrive inn et tall og få oversatt det til et romertall samt skrive det ut til konsollet så lenge at tallet brukeren skriver inn er større enn 0.
- Benytt `def`, `if`, `elif`, `else`, `input`, `print`, `while` og `return`.

romertall_1.py

romertall_1.py

romertall_1.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/Keyn...

```
def romertall(n):
    # returnerer romertall som svarer til n
    if n==0:   return ''
    elif n==1: return 'I'
    elif n==2: return 'II'
    elif n==3: return 'III'
    elif n==4: return 'IV'
    elif n==5: return 'V'
    elif n==6: return 'VI'
    elif n==7: return 'VII'
    elif n==8: return 'VIII'
    elif n==9: return 'IX'
    else: return 'Tallet er for stort eller lite'

n=0
while n>=0 :
    n = int(input('Tall mellom 0 og 9: '))
    r = romertall(n)
    print('Romertall for ',n,' er ', r)
```

Ln: 1 Col: 0

Funksjoner som bruker funksjoner – et større eksempel (fortsettelse)

- For å angi det siste sifferet i et heltall som et romertall bruker vi symbolene I=1 og V=5
- For å angi det nest siste sifferet i et heltall som et romertall bruker vi symbolene X=10 og L = 50
- For å angi det nest-nest siste sifferet i et heltall som et romer tall bruker vi symbolene, C=100 og D = 500 etc
- Romertall oppfører seg på samme måte for tall større en 10, men da med brukes X, L og C brukes i stedet for I, V og X

Funksjoner som bruker funksjoner – et større eksempel (fortsettelse)

- Vi skal utvide slik at vi kan skrive et romertall avhengig av hvilket siffer vi skal angi.
- Lag funksjonen `romersiffer` som tar inn 4 parametere:
  - n: heltall mellom 0 og 9 som skal oversettes til romertall.
  - entegn: Angir hvilket romertall som skal brukes for enerplassen
  - femtegn: Angir hvilket romertall som skal brukes for femmerplassen
  - titegn: Angir hvilket romertall som skal brukes for tierplassen
- Bruk:
  - For å angi bakerste siffer vil du gjøre følgende funksjonskall:

```
_r = romersiffer(n,'I','V','X')
```

  - For å angi neste bakerste siffer vil du gjøre følgende funksjonskall:

```
_r = romersiffer(n,'X','L','C')
```

- Benytt `def`, `if`, `elif`, `print` og `return`.

romertall_2.py

romertall_2.py

romertall_2.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/Keynote/Uke 40/Pytho...

```
def romersiffer (n,entegn,femtegn,titegn):
    # returnere romertall som svarer til n
    if n==0: return ''
    elif n==1: return entegn
    elif n==2: return entegn+entegn
    elif n==3: return entegn+entegn+entegn
    elif n==4: return entegn+femtegn
    elif n==5: return femtegn
    elif n==6: return femtegn+entegn
    elif n==7: return femtegn+entegn+entegn
    elif n==8: return femtegn+entegn+entegn+entegn
    elif n==9: return entegn+titegn
    else: return 'ERROR'

n=0
while n>=0:
    n = int(input('Tall mellom 0 og 9 paa tierplassen: '))
    r = romersiffer(n,'X','L','C')
    print('Romertall',r)
```

Ln: 14 Col: 0

Funksjoner som bruker funksjoner – et større eksempel (fortsettelse)

- Vi skal utvide slik at vi kan skrive et romertall avhengig av hvilket siffer vi skal angi.
- Lag funksjonen `romersiffer` som tar inn 4 parametere:
  - n: heltall mellom 0 og 9 som skal oversettes til romertall.
  - onechar: Angir hvilket romertall som skal brukes for enerplassen
  - fivechar: Angir hvilket romertall som skal brukes for femmerplassen
  - tenchar: Angir hvilket romertall som skal brukes for tierplassen
- Bruk:
  - For å angi bakerste siffer vil du gjøre følgende funksjonskall:

```
_r = romersiffer(n,'I','V','X')
```

  - For å angi neste bakerste siffer vil du gjøre følgende funksjonskall:

```
_r = romersiffer(n,'X','L','C')
```

- Benytt `def`, `if`, `elif`, `print` og `return`.

romertall_3.py

romertall_3.py

romertall_3.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/Keynote/Uke 40/Python/romertall_3.py (3.5.1)

```
def romersiffer (n, entegn, femtegn, titegn):
    # returnerer romertall som svarer til n med spesifiserte tegn for 1, 5 og 10
    if n==0: return ''
    elif n==1: return entegn
    elif n==2: return entegn+entegn
    elif n==3: return entegn+entegn+entegn
    elif n==4: return entegn+femtegn
    elif n==5: return femtegn
    elif n==6: return femtegn+entegn
    elif n==7: return femtegn+entegn+entegn
    elif n==8: return femtegn+entegn+entegn+entegn
    elif n==9: return entegn+titegn
    else: return 'ERROR!!!!!'

def romertall(n):
    # returnere romertall fra 0 til 999
    r = '' # romertallet som streng
    if n>999 or n<0: return 'Tallet for stort eller lite'
    elif n>99:
        r = r +romersiffer(n//100,'C','D','M')
        n = n - (n//100)*100 # Fjerner hundeplassen
    r = r + romersiffer(n//10,'X','L','C') # Tierplassen
    r = r + romersiffer(n%10,'I','V','X') # Enerplassen
    return r

n = 0
while n>=0:
    n = int(input('Tall mellom 0 og 999 '))
    r = romertall(n)
    print('Romertall for ',n,' er ',r)
```

Ln: 1 Col: 0

NTNU

Oppsummering

- Dette kapittelet dekket:
  - Returverdifunksjoner, inkludert
    - Å skrive returverdifunksjoner
    - Å bruke returverdifunksjoner
    - Å returnere flere verdier fra en funksjon
  - Å bruke bibliotekfunksjoner og import-uttrykket
  - Moduler, inkludert
    - Modulene random og math
    - Å gruppere dine egne funksjoner i moduler

NTNU