

i Informasjon om eksamenen

Emnekode: DAT120

Emnenavn: Grunnleggende Programmering

År og semester: 2024 Kontinuasjoneksamen

Eksamensdato: 7. mars 2024

Klokkeslett: 09.00 - 13.00

Tillatte hjelpemiddel:

- Godkjent kalkulator
- to A4-ark med egne notater på eksamen. Disse notatene kan være enten håndskrevne eller maskinskrrevne. Du kan skrive på begge sidene av A4-arkene.

Faglig ansvarlig: Erlend Tøssebro

Telefonnummer: +47 48107119

Om eksamenen:

Du bør lese instruksjonene før du starter med å løse eksamensoppgavene!

For oppgaver som krever programmering: Det viktigste er at du viser hvordan en løsning kan programmeres i Python. Enkle skrivefeil i navn på funksjoner vil ikke trekke ned. Å bruke et annet navn på en funksjon vil heller ikke trekke ned så lenge det går klart fram hva du prøver å gjøre og hvilken funksjon du prøver å kalle.

Det er lov å skrive hjelpefunksjoner og hjelpeklasser utover det oppgaven ber om hvis du mener det er den beste måten å løse oppgaven på.

Evaluerings:

Merk at all kode blir evaluert ikke bare på om den virker men også på hvor ryddig og lesbar den er. Du vil bli trukket for uleselig kode selv om den gjør alt oppgaven ber om.

i Oversikt over Python kommandoer og nøkkelord

Nøkkelord

Nøkkelord (engelsk Keyword) er reserverte ord med fast betydning i Python

Definisjoner

def <navn på funksjon eller metode>(<Liste med parametere>):

Definerer en funksjon. Hvis denne står inne i blokka til en klasse så er det en metode og første parameter skal være self.

class <navn på klassen>:

Definerer en klasse

Import av kode

from <navn på fil eller pakke> **import** <navn på funksjon eller klasse>

Importerer navngitte funksjoner eller klasser fra en pakke eller fil. Du kan oppgi flere med komma mellom. Du kan skrive * for å importere alle.

import <navn på fil eller pakke> **as** <alias>

Importerer en pakke eller ei Python fil. Den importerer Python filer gjennom å kjøre dem. Den siste delen «as <alias>» er valgfri, og brukes for å gi pakken eller fila et kortere navn enn den ellers ville ha hatt.

Kontrollstrukturer

break

Avbryter løkka som den står i. Den må stå inne i en for-løkke eller ei while-løkke

continue

Hopper tilbake til starten av løkka den står i og tar neste verdi for en for-løkke.

for <variabel> **in** <liste av verdier>:

<blokk, som utføres en gang for hver verdi i lista, med variabel satt lik verdien den utføres for nå>

if <betingelse>:

<blokk, som utføres hvis betingelsen er sann (True)>

elif <betingelse>:

<blokk, som utføres hvis alle foregående betingelser i denne if-setningen er usanne (False) men denne betingelsen er True>

else:

<blokk, som utføres hvis alle betingelsene i denne if-setningen er usanne>

while <betingelse>:

<blokk, som utføres gang etter gang helt til betingelsen er usann>

with <ressurs> **as** <variabelnavn>:

<blokk som bruker ressursen. Ressursen lukkes alltid når blokken fullfører, uansett hvordan den fullfører. I DAT120 har det bare vært filer som har vært brukt som slike ressurser>

Boolske operatorer

Disse opererer på de boolske verdiene False og True, og kan brukes for å lage mer kompliserte betingelser for if- og while-setninger.

a **and** b: Er sann bare hvis både a og b er sanne.

a **or** b: Er sann hvis minst en av a og b er sanne. Er usann bare hvis både a og b er usanne.

not a: Er sann hvis a er usann og usann hvis a er sann.

Matematiske operatorer

a + b: Fungerer som vanlig matematisk + for tall. Slår sammen to strenger eller lister

a – b: Matematisk subtraksjon

a * b: For tall gjør denne multiplikasjon. For strenger og lister lager den en ny streng eller liste med b antall duplikater av a.

a ** b: Opphøying, a^b .

a / b: Flyttallsdivisjon. Returnerer alltid et flyttall.

a // b: Heltallsdivisjon. Kutter desimalen hvis resultatet hadde vært et flyttall. 11 // 3 gir resultatet 3

a % b: Modulo. Returnerer resten etter en heltallsdivisjon. 11 % 3 gir resultatet 2 siden det høyeste tallet under 11 som går opp i 3 er 9, og 11 er to høyere enn 9. 9 % 3 gir resultatet 0.

Sammenlikningsoperatorer

a == b Er a lik b?

a != b Er a ulik b?

a < b Er a mindre enn b?

a > b Er a større enn b

a <= b Er a mindre enn eller lik b?

a >= b Er a større enn eller lik b?

Innebygde kommandoer

Dette er kommandoer (funksjoner) som er bygget inn i Python språket.

chr(<tall>)

Returnerer tegnet som er representert med dette tallet

float(<verdi>)

Prøver å konvertere verdien til et flyttall. Kan konvertere både heltall og strenger. Strengen må inneholde bare et flyttall, ellers risikerer du en ValueError.

input(<spørsmål>)

Skriver ut spørsmålet og ber deretter brukeren om å skrive inn en verdi. Returnerer verdien brukeren skreiv inn som en streng.

int(<verdi>)

Prøver å konvertere verdien til et heltall. Kan konvertere både flyttall og strenger. Kutter desimalen til et flyttall. En streng må inneholde bare et heltall, ellers risikerer du en ValueError.

len(<samling>)

Returnerer antall elementer i samlingen som et heltall. For et dictionary er det antall nøkler som er lagt inn.

max(<liste med sammenliknbare verdier>)

Returnerer det høyeste elementet i lista

min(<liste med sammenliknbare verdier>)

Returnerer det laveste elementet i lista

open(<filnavn>, <modus>, encoding=<navn på kodetabell>)

Åpner fila med oppgitt navn og returnerer en referanse til denne fila. Modus har default «r» for lesing, men kan også være «w» for skriving. Man kan bruke den navngitte parameteren encoding for å si hvilken kodetabell tekst er kodet på. Filreferansen som returneres kan man bruke til å lese og skrive til fila. Skriving skjer med write() metoden. Lesing skjer med read() eller readline() metodene, eller at du bruker filreferansen som lista med verdier i en for-løkke.

ord(<streng med ett tegn>)

Returnerer tallkoden til dette tegnet

print(<verdi>)

Skriver ut en streng. Konverterer andre verdier til strenger på samme vis som **str**

range(start, slutt, steglengde)

Lager en slags liste av verdier. Denne kan brukes i for-løkker men er ikke ei Python liste. Lista starter med verdien «start», slutter før verdien «slutt» og går «steglengde» langt mellom hver verdi. Range-funksjonen tar bare heltall. Ønsker du å bruke flyttall for eksempel steglengde, må du bruke numpy.arange eller numpy.linspace i stedet. Range med en parameter oppgir bare slutt, med start 0. Range med to parametere oppgir start og slutt, med stengelengde 1.

round(<flyttall>, <valgfri: antall desimaler>)

Runder av et flyttall. Uten den andre parameteren returnerer den et heltall. Ellers runder den av til det oppgitte antallet desimaler og returnerer et flyttall.

str(<verdi>)

Konverterer verdien til en streng. Er verdien et objekt gjør den dette ved å kalle `__str__` metoden til objektet.

sum(<liste med tall>)

Regner ut summen av alle elementene i lista.

Strengmetoder

Disse metodene kan man kalle på et str objekt:

find(<delstreng>, <valgfri: start>, <valgfri: slutt>)

Leiter etter en delstreng av strengen. Returnerer indeksen i strengen til første tegn i delstrengen hvis delstrengen fins i strengen. Ellers returnerer den -1. Parameterne start og slutt er indeksene hvor den skal starte og slutte med å leite.

lower()

Returnerer strengen med alle bokstavene gjort om til små bokstaver.

replace(<delstreng den skal se etter, a>, <delstreng den skal erstatte med, b>)

Returnerer en ny streng der alle forekomster av delstrengen a er erstattet av delstrengen b.

split(<streng, valgfri>)

Splitter strengen basert på den oppgitte delstrengen, eller på whitespace (åpenrom, tabulatorer og linjeskift) hvis denne parameteren ikke er oppgitt. Returnerer ei liste med strenger.

upper():

Returnerer strengen med alle bokstavene gjort om til store bokstaver.

Formatering av strenger

Formateringskoder

Disse formateringskodene kan brukes i f-strenger for å formatere dem

Kode	Betydning	Eksempel
d	Heltall i titallssystemet (det normale tallsystemet). Standard for heltall. Antall siffer den skal sette av plass til oppgis rett før.	Tall = 7 Print(f«tallet er {tall:d}») Print(f«tallet er {tall:10d}»)
c	Konverterer heltallet til et tegn med den nåværende kodetabellen	
x	Heltall i det heksadesimale tallsystemet	
b	Heltall i det binære tallsystemet	
f	Flyttall med komma-notasjon. Tar to parametere. I eksemplet skal den sette av 5 siffer hvorav 2 bak komma. Merk at komma-tegnet regnes som ett av sifrene!	Flyttall = 7.3 Print(f«tallet er {flyttall:5.2f}»)
e	Flyttall på eksponentialform (e-notasjon)	
%	Flyttall som prosent. Ganger tallet med 100 og viser det med et prosent-tegn etter. Parametere som for «f»	

Spesialtegn

Man kan bruke disse kodene i en streng for å lage spesialtegn

Kode	Betydning
\n	Linjeskift
\t	Tabulator
\	Backslash
\"	Anførselstegn (slik inkluderer du anførselstegn i strenger)

Metoder for lister

Man kan bruke disse metodene på lister

append(<verdi>)

Legger verdien til på slutten av lista

index(<verdi>)

Leiter etter verdien og returnerer første indeks hvor den fins. Lager en `AttributeError` hvis verdien ikke fins i lista.

insert(<indeks>, <verdi>)

Setter inn verdien på oppgitt indeks. Alle elementer som lå på eller etter denne indeksen blir forskjøvet ett hakk bakover i lista.

remove(<verdi>)

Fjerner første forekomst av verdien i lista, alle elementer bak blir flyttet ett hakk fram. Lager en `ValueError` hvis verdien ikke fins i lista.

sort()

Sorterer lista. Alle elementene i lista må være sammenliknbare med hverandre, ellers får du en `TypeError`. Returnerer ingen ting. Modifiserer lista.

Spesialmetoder brukt i temæet klasser og objekter

Disse metodene brukes av selve Python språket for å håndtere objekter av klassen.

__init__(self, <parameterliste>)

Konstruktøren for klassen. Denne kalles når du lager nye objekter av klassen. Den tar de parameterne som du definerer. Når man lager objektene så må man da oppgi disse parameterne.

__repr__(self)

Lager og returnerer en strengrepresentasjon mer ment til internt bruk

__str__(self)

Lager og returnerer en strengrepresentasjon av objektet, ment for brukere å lese.

Matplotlib

Her er en oversikt over funksjoner i Matplotlib som er brukt i dette faget.

```
import matplotlib.pyplot as plt
```

Standard setning for å importere matplotlib

```
plt.subplot(<Antall rader>, <Antall kolonner>, <Hvilken av sub-figurene du skal starte på nå>)
```

Brukes for å tegne opp flere sub-plott i ett vindu. Antall rader og antall kolonner bør være like for alle kallene til plt.subplot, mens den siste parameteren bør være ulik. Den siste parameteren starter å telle på 1, og går først horisontalt, deretter vertikalt.

```
plt.plot(<liste av x-koordinater>, <liste av y-koordinater>, <tegnestil>, label=<hvilken kurve er dette?>)
```

Tegner et standard linje-plott. Listene bør være like lange. Den starter i punktet (x[0], y[0]), tegner ei linje til (x[1], y[1]), deretter til (x[2], y[2]) og så videre til den når slutten av listene. Tegnestil og label er valgfrie parametere. Tegnestil er en streng som inneholder ett eller flere av disse tegnene: «o» for å tegne sirkler for hver verdi, «-» for å tegne linje mellom verdiene, «*» for å tegne stjerner for hver verdi, og flere.

Flere plot-kommandoer etter hverandre kan brukes for å tegne flere linje-plott i samme figur.

```
plt.bar(<liste av verdier langs x-aksen>, <liste av høydene for hver stolpe>, color=<liste av farger for stolpene>)
```

Tegner et stolpediagram. Parameteren «color» er valgfri. De to listene må ha indekser som korresponderer med hverandre. Så høyden til stolpen til den første verdien langs x-aksen må ligge i første element i lista av høyder.

```
plt.pie(<Liste av verdier>, labels=<Liste av etiketter, en for hver verdi>)
```

Tegner et kakediagram. Den regner ut summen av verdiene og regner ut vinklene slik at vinkelen til hvert kakestykke korresponderer med hvor mange prosent av summen hver verdi representerer.

```
plt.hist(<liste av verdier>, <Antall stolper>)
```

Tegner et histogram av verdier.

```
plt.title(<Tittel>)
```

Gir plottet en tittel, som tegnes øverst, sentrert.

```
plt.xlabel(<beskrivelse>)
```

Gir en beskrivelse av hva x-koordinatene er evt. enhet langs x-aksen. Tegnes under x-aksen.

```
plt.ylabel(<beskrivelse>)
```

Gir en beskrivelse av hva y-koordinatene er evt. enhet langs y-aksen. Tegnes vertikalt til venstre for y-aksen.

```
plt.legend()
```

Ber matplotlib lage en boks som inneholder label for alle plottene i figuren

```
plt.grid(True)
```

Viser et rutenett

```
plt.savefig(<Filnavn>)
```

Lagrer figuren. Filtypen avgjøres av etternavnet til filnavnet du oppgir, for eksempel .pdf.

```
plt.show()
```

Viser figuren

Numpy

Her er en oversikt over funksjoner i numpy som er brukt i dette faget

```
import numpy as np
```

Importerer numpy og gir den dens valnlige alias

```
np.arange(<start>, <slutt>, <steg>)
```

Lager og returnerer en numpy array som starter med tallet «start», slutter rett før den når «slutt» og med «steg» mellom tallene. Arrayen blir så lang som den trenger å være. I motsetning til den innebygde range-funksjonen så aksepterer np.arange flyttall som parametere.

```
np.array(<liste>)
```

Lager og returnerer en numpy array som inneholder de samme elementene som den oppgitte lista.

```
np.linspace(<start>, <slutt>, <antall>)
```

Lager og returnerer en numpy array som starter med tallet «start», slutter med tallet «slutt» og med «antall» elementer. Elementene har lik avstand mellom hverandre. Merk at denne tar med verdien «slutt», i motsetning til numpy.arange og den innebygde range funksjonen.

```
np.ones(<form>)
```

Lager og returnerer en numpy array med oppgitt form og hvor alle elementene er 1. Hvis form er et heltall lager den en endimensjonal array med den lengden. For eksempel med form=7 så lager den en endimensjonal array som er 7 lang. Hvis form er et tuppel lager den en flerdimensjonal array (matrise) med antall dimensjoner lik antall elementer i tuppelet og antall elementer i hver retning lik tallene i tuppelet. For eksempel med form=(3, 4) lager den en todimensjonal matrise som er 3 ganger 4 stor.

```
np.zeros(<form>)
```

Lager og returnerer en numpy array med oppgitt form og hvor alle elementene er 0. Form er som for np.ones.

Turtle Graphics

Her er en oversikt over funksjoner i Turtle Graphics som er brukt i dette faget.

```
import turtle
```

Importerer Turtle Graphics

`turtle.forward(<lengde>)`

Går forover. Tegner hvis penna er nede. Penna starter nede

`turtle.backward(<lengde>)`

Går bakover. Tegner hvis penna er nede. Penna starter nede

`turtle.right(<vinkel i grader>)`

Snur seg den oppgitte vinkelen til høyre

`turtle.left(<vinkel i grader>)`

Snur seg den oppgitte vinkelen til venstre

`turtle.penup()`

Tar opp penna

`turtle.pendown()`

Tar ned penna

`turtle.circle(<radius>)`

Tegner en sirkel med oppgitt radius. Posisjonen til skilpadda nå er der den starter å tegne, og vil derfor være langs sirkelen og ikke i midten.

`turtle.speed(<fart>)`

Setter farta til skilpadda. Speed 0 er så raskt som mulig. Speed 1 er så tregt som mulig. Så øker hastigheten opp til speed 10. Deretter er den fast.

`turtle.fillcolor(<farge>)`

Setter fyllfargen. Fargen er en streng med et fargenavn på engelsk.

`turtle.pencolor(<farge>)`

Setter farge på penna

`turtle.pensize(<tjukkelse>)`

Setter tjukkelse på penna

`turtle.begin_fill()`

Starter å tegne mønsteret som skal fylles

`turtle.end_fill()`

Fyller mønsteret som er tegnet med fyllfargen

`turtle.done()`

Sier fra at den er ferdig med å tegne. Brukes i script for å sørge for at turtle-vinduet forblir oppe selv etter at scriptet er ferdig.

Git

Basis håndtering av repo-et

Kommando	Bruk
<code>clone <lenke til></code>	Lager et lokale repo og en lokal filstruktur som en kopi av et eksternt

eksternt repo>	repo.
branch <navn>	Lager en ny branch i ditt lokale repo
checkout <navn på branch>	Bytter til angitt branch. Oppdaterer filstrukturen din med innholdet av denne branchen
push	Sender endringer (gjort med en commit) til det eksterne report som denne er klonet fra. Brukes til å oppdatere Github med endringer. For å oppdatere Github med en ny branch, bruk push --set-upstream origin <navn på branch>
Pull	Henter endringer gjort på tjeneren ned til lokalt repo. Må muligens bruke git pull origin main for å hente ned ny main-branch etter en merge konflikt.

Basis håndtering av filer

Kommando	Bruk
add <mappenavn>	Registrerer endringer og nye filer før en commit. Kan brukes for å legge til nye filer og å si fra om oppdateringer du har gjort på eksisterende filer. Du kjører typisk git add på hele mapper, men den kan også kjøres på enkeltfiler. Kjør alltid git add før en git commit. Du kan fint kjøre flere add før en enkelt commit, også på de samme filene.
status	Sier fra hvilke filer som er registrert for å committe, det vil si hvilke filer du har lagt til med git add siden sist commit.
diff	Kan brukes på flere måter, den enkleste, uten ekstra parametere, sjekker filene dine for endringer og sier fra hva du burde legge til med add før en commit
Commit -m <melding>	Oppdaterer ditt lokale repository med endringene du har lagt til med add, og legger til meldingen slik at andre som bruker repositoryet kan se hva endringen er
restore <filnavn>	Setter den oppgitte fila tilbake slik den var ved siste commit

i Kode-eksempler

Se vedlagte .pdf fil.

1 Datatyper

Hva blir datatypen for variabelen "verdi" etter følgende Python-setninger?

Finn de som passer sammen:

	int	float	boolean	list	str	Du får en feilmelding
verdi = 3.2 * 7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
verdi = 5/2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
verdi = [5, 3, 7, 5] + 5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
verdi = "Tallet er: " + 7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
verdi = input("Skriv inn et tall mellom 1 og 10: ")	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
verdi = [5, 3, 7, 5] * 5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 12

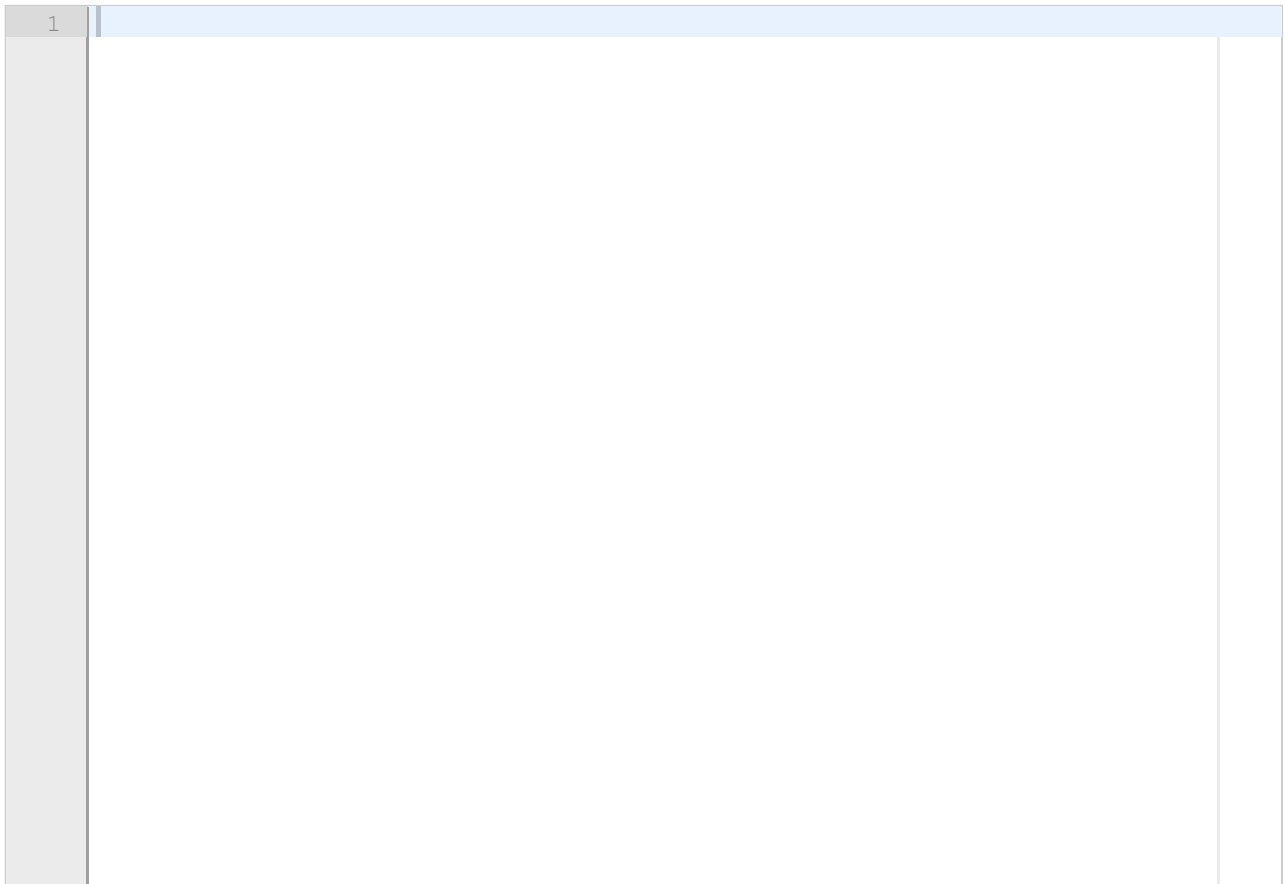
2 Basis Python script

Skriv et Python script som regner ut volumet til en pyramide. Programmet skal lese inn lengde, bredde og høyde til pyramiden som flyttall, regne ut volumet, og skrive ut volumet.

Formelen for volumet av en pyramide er $\frac{1}{3} \cdot \text{bredde} \cdot \text{lengde} \cdot \text{høyde}$

Merk: Det er ikke noe behov for å sjekke for lovlig input fra brukeren i denne oppgaven.

Skriv ditt svar her



Maks poeng: 5

3 Løkke

Lag et Python-script som regner ut arealet til en leilighet. For hvert rom skal brukeren skrive inn lengde og bredde til rommet. Arealet til et rom er $\text{lengde} \times \text{bredde}$. Arealet til hele leiligheten er summen av arealene av alle rommene. Ulike leiligheter kan ha ulikt antall rom. Du kan velge en av to måter å angi antall rom på: (7%)

1: Brukeren oppgir antall rom før hen oppgir lengde og bredde til første rom.

2: Brukeren oppgir et negativt tall for lengde når brukeren er ferdig med å oppgi rom.

Skriv ditt svar her

1	
---	--

Maks poeng: 7

4 Lister

Gitt følgende Python-kode

```
liste = [2, 4, 6, 8, 10]  
liste.append(7)  
liste.insert(2, 5)  
liste.remove(8)
```

Skriv hva som skrives ut av følgende setninger. Alle setningene kjøres etter koden oppgitt over. Skriv "Feil" hvis du mener denne setningen lager en exception.

print(liste[1])

print(liste[4])

print(liste[-1])

print(liste[6])

Score: 2% pr. riktig svar.

Maks poeng: 8

5 Plotting

Gitt følgende program som lager ei liste:

```
liste = list()
liste.append(1)
liste.append(1)
for i in range(2, 10):
    liste.append(liste[i-1] + liste[i-2])
```

Fortsett dette programmet slik at det bruker matplotlib til å plote verdiene i lista "liste". Den første verdien i lista skal plottes på x-koordinat 1, det neste på x-koordinat 2 og så videre fram til slutten av lista. 6%

Skriv ditt svar her

1	
---	--

Maks poeng: 6

6 Kodeforståelse

Du har følgende oppgavetekst og følgende forslag til programkode. Forslaget til programkoden er oppgitt under der du skriver inn svaret og i vedlagt .pdf fil. Beskriv hva som er problemene med den foreslåtte koden og hva som er løsningene, enten som programkode eller som tekst. 10%

Oppgavetekst:

"Du har et system med tre brytere som kan være enten av (0) eller på (1). Bryterne er nummerert 1, 2 og 3. Programmet ditt skal registrere historikken til dette systemet i ei liste. For hver gang en bryter blir slått av eller på skal det legges inn et nytt innslag i denne lista. Hvert innslag skal være ei liste som inneholder tidspunktet i indeks 0 og tilstanden til hver bryter i de neste tre plassene i lista. Programmet skal gå i løkke. For hver iterasjon skal programmet be brukeren skrive hvilken bryter som bytter tilstand og når det skjedde. For de andre bryterne skal tilstanden i det nye innslaget være den samme for i forrige innslag i historikken. Programmet skal avslutte og skrive ut historikken når brukeren skriver ei tom linje for hvilken bryter som skifter tilstand. Brukeren skal få en feilmelding hvis brukeren skriver inn ei tid som er tidligere enn sist innslag eller skriver inn en ugyldig bryter."

Skriv ditt svar her

1	
---	--

Foreslått løsning:

```
historikk = list()
historikk.append([0, 0, 0, 0])
fortsetter = True
while fortsetter:
```



```

hvilken_bryter = input("Skriv inn hvilken bryter (1, 2 eller 3) som skifter verdi: ")
if hvilken_bryter == "":
    fortsetter = False
    break
tid = int(input("Skriv inn tidspunktet innslaget skjedde: "))
forrige_innslag = historikk[-1]
dette_innslaget = forrige_innslag
dette_innslaget[0] = tid
bryter_indeks = int(hvilken_bryter)
if dette_innslaget[bryter_indeks] == 0:
    dette_innslaget[bryter_indeks] = 1
else:
    dette_innslaget[bryter_indeks] = 0
historikk.append(dette_innslaget)
print("\n Resultat:")
print(historikk)

```

Utskrift fra foreslått løsning:

```

"
Skriv inn hvilken bryter (1, 2 eller 3) som skifter verdi: 1
Skriv inn tidspunktet innslaget skjedde: 5
Skriv inn hvilken bryter (1, 2 eller 3) som skifter verdi: 2
Skriv inn tidspunktet innslaget skjedde: 8
Skriv inn hvilken bryter (1, 2 eller 3) som skifter verdi: 3
Skriv inn tidspunktet innslaget skjedde: 12
Skriv inn hvilken bryter (1, 2 eller 3) som skifter verdi: 2
Skriv inn tidspunktet innslaget skjedde: 15
Skriv inn hvilken bryter (1, 2 eller 3) som skifter verdi:

```

Resultat:

```

[[15, 1, 0, 1], [15, 1, 0, 1], [15, 1, 0, 1], [15, 1, 0, 1], [15, 1, 0, 1]]
"

```

Dette resultatet burde ha vært:

```

[[0, 0, 0, 0], [5, 1, 0, 0], [8, 1, 1, 0], [12, 1, 1, 1], [15, 1, 0, 1]]

```

Maks poeng: 10

7 Funksjoner

- a) Skriv en funksjon som sjekker at et tall er innenfor et lovlig spenn. Funksjonen skal ta inn et tall som parameter og sjekke om tallet er mellom -100 og +100. Hvis tallet er mellom -100 og +100 skal funksjonen returnere True ellers False (4%)
- b) Skriv et program som leser inn en temperatur fra brukeren, bruker funksjonen fra deloppgave a) til å sjekke om den er rimelig, og skriver ut svaret. (3%)
- c) Du skal skrive en funksjon som tar inn ei liste med tall som parameter og finner toppene i lista. En topp er et element som er høyere enn de to foregående og de to neste elementene i lista. For eksempel i lista [1, 3, 2, 3, 5, 3, 1, 1, 5, 4, 6, 4, 3, 2] så er elementet 5 på index 4 en topp. Funksjonen din skal returnere ei liste som inneholder indeksene til alle toppene i lista som kommer inn. (6%)
- d) Fortsett programmet oppgitt under slik at det tar lista "lista", fjerner alle ulovlige verdier gjennom å bruke funksjonen fra oppgave a), deretter finner alle toppene med funksjonen i oppgave c) og til slutt skriver ut indeks og verdi for alle toppene. (5%)

Skriv ditt svar her

1	
---	--

```
lista = [23, 25, 22, 29, 34, 32, 175, 26, 15, 12, 10, 12, 15, 13, -123, 14, 17, 16, 11]
```

```
if __name__ == "__main__":  
    # Din kode for oppgave d) her
```

Maks poeng: 18

8 Bruk av filer

Lag et Python-script som konverterer beløp i ei fil til en annen valuta og skriver dette ut i ei anna fil. Begge filene skal ha navn på valutaen i første linje og deretter ett beløp i hver linje. Programmet ditt skal:

- Be brukeren om navn på den første fila
- Lese inn valuta-navnet i den første fila og skrive ut dette til brukeren
- Be brukeren om navn på den andre valutaen
- Be brukeren om kursen mellom de to valutaene
- Be brukeren om navn på den andre fila
- Skrive ut navnet på den andre valutaen til den andre fila
- For hver linje etter den første i den første fila, les inn beløpet, gang det med kursen mellom de to valutaene, og skriv ut resultatet til den andre fila.

Evaluerings:

1. Lesing og skriving av filene 6 poeng
2. Konvertering av valutaer 4 poeng
3. Håndtering av feil 4 poeng. Du skal håndtere både feil i selve filbehandlingen slik som at brukeren har oppgitt en første fil som ikke eksisterer eller en andre fil som ikke kan skrives til, og feil format i selve fila. Linjer med feil format i den første fila skal du hoppe over. 4 poeng

Skriv ditt svar her

1	
---	--

Et eksempel på filformatet:

NOK
152.5
134.5
97.9
78
4563.3
234.34
45.6
343.4
23.5
45.7
33.6

Maks poeng: 14

9 Klasser og objekter

Gitt klassene Punkt og Linje oppgitt under og i vedlagt fil, løs følgende deloppgaver. 5% pr. deloppgave:

- Skriv en `__str__` metode for klassen Linje. Metoden skal skrive ut at det er ei linje og skrive ut koordinatene til alle punktene i linja.
- Skriv en metode "lengde" for klassen Linje. Metoden skal returnere lengden til linja. Lengden til ei linje er summen av avstanden mellom alle naboer i punktlista. Er det for eksempel tre punkter i punktlista så lengden være avstanden mellom element 0 og 1 i punktlista pluss avstanden mellom element 1 og 2 i punktlista.
- Bruk klassen Punkt som utgangspunkt for en klasse for en fri 3D vektor. En fri vektor er en vektor som går fra koordinat 0,0,0 til et annet koordinat. Klassen trenger derfor bare å inkludere x, y og z verdier for ett koordinat (punkt). Du skal lage konstruktør, `__str__` metode og en metode som regner ut lengden til vektoren, Lengden til en 3D vektor er $lengde = \sqrt{x^2 + y^2 + z^2}$
- Skriv en funksjon som legger sammen to 3D vektorer og returnerer resultatet som en ny 3D vektor. For å legge sammen to vektorer så legger man sammen koordinatene separat.

Skriv ditt svar her

class Punkt:

```
def __init__(self, x_koordinat, y_koordinat):
    self.x_koordinat = x_koordinat
    self.y_koordinat = y_koordinat
```

```
def avstand(self, annet_punkt):
    avstand = ((self.x_koordinat - annet_punkt.x_koordinat)**2 +
```

```
        (self.y_koordinat - annet_punkt.y_koordinat)**2)**0.5  
    return avstand
```

```
def __str__(self) -> str:  
    return f'({self.x_koordinat}, {self.y_koordinat})'
```

```
class Linje:  
    def __init__(self, startpunkt, sluttpunkt):  
        self.punktliste = list()  
        self.punktliste.append(startpunkt)  
        self.punktliste.append(sluttpunkt)  
  
    def legg_til_punkt(self, punkt):  
        self.punktliste.append(punkt)
```

```
if __name__ == "__main__":  
    punkt1 = Punkt(1, 2)  
    punkt2 = Punkt(1, 6)  
    print(punkt1.avstand(punkt2))  
    linje = Linje(punkt1, punkt2)  
    punkt3 = Punkt(8, 6)  
    punkt4 = Punkt(1, 2)  
    linje.legg_til_punkt(punkt3)  
    linje.legg_til_punkt(punkt4)  
    print(linje)
```

Maks poeng: 20

Document 3

Attached



Kode-eksempler til eksamen i DAT120

Kontrollstrukturer

Er dette tallet for en måned gyldig?

```
tall_streng = input("Skriv inn et tall for en måned: ")
maaned = int(tall_streng)
if maaned < 1 or maaned > 12:
    print("ugyldig måned")
else:
    print("Gyldig måned")
```

Løsninger av en andregradslikning

```
import math

a = int(input("Skriv inn tallet a: "))
b = int(input("Skriv inn tallet b: "))
c = int(input("Skriv inn tallet c: "))

verditest = b**2 - 4*a*c

if verditest > 0:
    losning1 = (-b + math.sqrt(verditest))/(2*a)
    losning2 = (-b - math.sqrt(verditest))/(2*a)
    print(f"Likningen har to løsninger: {losning1} og {losning2}")
elif verditest == 0:
    losning = (-b)/(2*a)
    print(f"Likningen har en løsning: {losning}")
else:
    print("Likningen har ingen løsninger")
```

Lar brukeren skrive inn flere linjer tekst og avslutter når brukeren skriver inn ei tom linje

```
teksten = ""
tekstlinje = input("Skriv inn første linje: ")
while tekstlinje != "":
    teksten += tekstlinje + "\n"
    tekstlinje = input("Skriv inn neste linje: ")
print("Den endelige teksten ble:")
```



```
print(teksten)
```

Regner ut bokstavkarakter fra prosentsscore, avslutter hvis brukeren skriver inn en negativ score

```
fortsetter = True

while fortsetter:
    prosentsscore = int(input("Skriv inn en prosentsscore: "))
    if prosentsscore < 0:
        fortsetter = False
        break
    if prosentsscore >= 90:
        print("A")
    elif prosentsscore >= 80:
        print("B")
    elif prosentsscore >= 60:
        print("C")
    elif prosentsscore >= 50:
        print("D")
    elif prosentsscore >= 40:
        print("E")
    else:
        print("F")
```

Regner ut fakultet

```
tall = int(input("Skriv inn tallet du ønsker fakultet av: "))
while tall < 0:
    print("Fakultet for negative tall finnes ikke!")
    tall = int(input("Skriv inn tallet du ønsker fakultet av: "))

resultat = 1
for i in range(1, tall+1):
    print(i)
    resultat *= i
print("Etter for-løkke er ferdig")
print(resultat)
```

For-løkke som teller nedover

```
for i in range(10, 0, -1):  
    print(i)  
print("Etter for-løkka er ferdig")
```

Nøstet for-løkke som skriver ut en firkant av ASCII stjerner

```
hoyde = int(input("Høyde: "))  
bredde = int(input("Bredde: "))  
  
for j in range(hoyde):  
    for i in range(bredde):  
        print("*", end="")  
    print()
```

Funksjoner

Areal og omkrets av en sirkel

```
import math  
  
def areal_sirkel(radius):  
    areal = math.pi*radius*radius  
    return areal  
  
def omkrets_sirkel(radius):  
    return 2.0*math.pi*radius  
  
radius_streng = input("Skriv inn radius til sirkelen: ")  
radius Bruker = float(radius_streng)  
areal_global = areal_sirkel(radius Bruker)  
print(f"Aralet ble: {areal_global:8.2f}")  
omkrets = omkrets_sirkel(radius Bruker)  
print(f"Omkretsen ble: {omkrets:8.2f}")
```

Funksjon med flere parametere: Skriv ut en ASCII firkant av tegn

```
def skriv_firkant(hoyde=5, bredde=5, tegn="*"):  
    for j in range(hoyde):  
        for i in range(bredde):
```

```

        print(tegn, end="")
    print()

    hoyde = int(input("Høyde: "))
    bredde = int(input("Bredde: "))

    skriv_firkant(2, 4)
    print("\n ny firkant \n")
    skriv_firkant(hoyde, bredde, "#")
    print("Ferdig")

```

Filer og exceptions

Les inn ei tekstfil og skriv ut innholdet

```

filnavn = input("Hvilken fil skal leses? ")
fila = open(filnavn, "r", encoding="UTF8")
for linje in fila:
    print(linje, end="")
fila.close()

```

Lar brukeren skrive inn flere linjer tekst og lagrer dette til fil. Avslutter når brukeren skriver inn ei tom linje

```

tekstlinje = input("Skriv inn første linje: ")
fila = open("tekstfil.txt", "w", encoding="UTF8")
while tekstlinje != "":
    fila.write(tekstlinje + "\n")
    tekstlinje = input("Skriv inn neste linje: ")
fila.close()

```

Eksempel på håndtering av feil brukerinput

```

fortsetter = True
while fortsetter:
    try:
        tall = int(input("Skriv inn tallet du ønsker fakultet av: "))
    except:
        print("Fakultet for negative tall finnes ikke!")
    else:
        if tall < 0:
            print("Fakultet for negative tall finnes ikke!")
        else:
            # Beregning av fakultet
            pass

```

```

        fortsetter = False
    except ValueError:
        print("Du må skrive inn et tall!")

resultat = 1
for i in range(1, tall+1):
    print(i)
    resultat *= i
print("Etter for-løkken er ferdig")
print(resultat)

```

Eksempel på with-setningen og håndtering av feil med filer

```

filnavn = input("Hvilken fil skal leses? ")
try:
    with open(filnavn, "r", encoding="UTF8") as fila:
        summen = 0
        for linje in fila:
            summen += int(linje)
        print(f"Summen ble: {summen}")
except FileNotFoundError:
    print("Finner ikke denne fila!")
except ValueError:
    print("Fila har feil format")
except:
    print("En annen feil har oppstått")

```

Samlingsobjekter

Lister

```

# Lager ei tom liste
liste = list()

# Lager ei liste med oppgitte elementer, her tallene 1, 2, 3, 4 og 5
liste2 = [1, 2, 3, 4, 5]

# Skriver ut lengden til lista
print(len(liste2))

```

```
# Legger til elementet 4 i lista "liste"
liste.append(4)

# Legger til elementet 7 i lista "liste2"
liste2.append(7)

# Henter ut elementet med indeks 3 og skriver det ut. Merk
# at Python teller fra 0, første elementet i lista har indeks 0!
print(liste2[3])

# Elementene i ei liste kan være hva som helst
liste.append("En streng")

# Inkludert andre lister
liste.append([9, 8, 7])

# Kan overskrive verdien på en bestemt indeks med tilordning
liste2[2] = 9

# Negative indekser teller fra slutten
print(liste2[-1])

# Setter inn elementet 10 på indeks 2. Dette forskyver
# alle elementer som er etterpå en plass lengre ut i lista
liste2.insert(2, 10)

# For lister inni lister, bruk flere indekser etter hverandre.
# For eksempel denne, som kan leses
# Hent ut elementet med indeks 2 i liste. Hent ut elementet
# med indeks 1 i lista som ligger i indeks 2 i liste.
print(liste[2][1])

# Lager ei liste hvor innholdet er lik indeksene
liste_indekser = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15]

# Lager ei ny liste so består av elementene fra og med
# indeks 3, til men ikke med indeks 8, i liste_indekser
```

```
test = liste_indekser[3:8]
print(test)

# Starter fra starten av lista og går til men ikke med indeks 8
print(liste_indekser[:8])

# Starter på indeks 8 og går til slutten av lista
print(liste_indekser[8:])

# Starter på indeks 1 og tar med bare annethvert element.
# Stopper rett før den når indeks 10
print(liste_indekser[1:10:2])

# Finner det minste elementet
print(min(liste_indekser))

# Finner det største elementet
print(max(liste_indekser))

liste3 = [1, 2, 3, 4, 5]
liste4 = [9, 10, 11, 12]

# Kan bruke + operatoren til å slå sammen lister
liste5 = liste3 + liste4

# Kan bruke * operatoren til å repetere ei liste. Denne lager
# ei liste med elementene 0,1 repetert 5 ganger
liste6 = [0, 1]*5

# Skriver ut indeks til første element som inneholder verdien 10
print(liste4.index(10))

# Fjerner første forekomst av elementet 9 fra lista
liste5.remove(9)

# Fjerner elementet som ligger på indeks 5 fra lista
del liste5[5]
```

```
# Sorterer lista. Denne modifierer lista slik at den er sortert
liste5.sort()

# Reverserer lista slik at den får motsatt rekkefølge
liste2.reverse()
```

Bruk av lister i ei for-løkke

```
liste = [1, 3, 5, 7, 9]

for element in liste:
    element = element*2
    print(element)

print(liste)

element = input("Skriv inn et tall for å sjekke om det er med i
lista: ")
element = int(element)
if element in liste:
    print(f"{element} er med i lista")
else:
    print(f"{element} er ikke med i lista")

for indeks, element in enumerate(liste):
    print(f"Element {element} ligger på indeks {indeks}")
```

List comprehensions

```
# Lager en liste med kvadratene av tallene fra 1 til 10
liste = list()
for verdi in range(1, 11):
    liste.append(verdi**2)
print(liste)

# Lager samme liste med en list comprehension
liste2 = [verdi**2 for verdi in range(1, 11)]
print(liste2)

# Lager en liste med kvadratene av oddetallene fra 1 til 20
```

```

liste3 = [verdi**2 for verdi in range(1, 20) if verdi%2 == 1]
print(liste3)

# Lager samme liste uten list comprehension
liste4 = list()
for verdi in range(1, 20):
    if verdi%2 == 1:
        liste4.append(verdi**2)
print(liste4)

```

Numpy arrayer

```

import numpy as np

# Lager en numpy array med 5 0-ere
nullere = np.zeros(5)

# En for-loop med steglengde 0.1
for i in np.arange(1, 5, 0.1):
    print(f"{i:5.2f}")

# Lager en numpy array med 4 elementer. Det første er 1,
# det siste er 10, og de to mellom er jevnt spreidd mellom
# 1 og 10
test = np.linspace(1, 10, 4)
for element in test:
    print(test)

# Lager en numpy array fra ei liste
oddetall = np.array([1, 3, 5, 7, 9])

# Lager ei liste fra en numpy array
liste = list(test)

# Kan lage matrise med å oppgi et tuppel til np.zeros. Ekseplet
# lager en 4x3 matrise
matrise = np.zeros((4, 3))

```



```
# Lan lage en matrise fra ei liste av lister så lenge
# de indre listene alle er like lange
matrise2 = np.array([[1, 2, 3], [2, 2, 2], [3, 2, 1]])
```

Mengder

```
mengde1 = set()
mengde1.add("Henrik")
mengde1.add("Nils")
mengde1.add("Olav")
mengde1.add("Anders")
mengde1.add("Henrik")

mengde1.add("Åsmund")

for element in mengde1:
    print(element)
print("\n Mengde 2: ")
mengde2 = set()
mengde2.add("Nils")
mengde2.add("Andreas")
mengde2.add("Tomas")
mengde2.add("Olav")
mengde2.add("Mads")
for element in mengde2:
    print(element)

# Mengden av alle som er med i minst en av de to
m3 = mengde1.union(mengde2)
print("\n Union: ")
for element in m3:
    print(element)

# Mengden av alle som er med i begge
m3 = mengde1.intersection(mengde2)
print("\n Snitt: ")
for element in m3:
    print(element)
```

```
# Mengden av alle som er med i mengde1 men ikke i mengde2
m3 = mengde1.difference(mengde2)
print("\n m1 minus m2: ")
for element in m3:
    print(element)
```

Dictionaries

```
# Ordteller for tekstfil:
#   Les inn ei tekstfil
#   Finn alle ordene
#   Tell antall forekomster av hvert ord

ordliste = dict()
with open("oving_1_rein_tekst.txt", "r", encoding="UTF8") as fila:
    for linje in fila:
        ordene = linje.split()
        for ordet in ordene:
            ordet = ordet.lower()
            if ordet in ordliste:
                teller = ordliste[ordet]
                teller += 1
                ordliste[ordet] = teller
            else:
                ordliste[ordet] = 1
        for ordet in ordliste:
            print(f"Ordet {ordet} forekommer {ordliste[ordet]} antall ganger")
```

Klasser og objekter

Klassen Punkt

```
import math

class Punkt:
    # Konstruktør
```

```

def __init__(self, start_x=0, start_y=0):
    self.__x_koordinat = start_x
    self.y_koordinat = start_y

# Eksempel på bruk av @property og @setter for å lage en
# egenskap med kontroll på lovlige verdier
@property
def x_koordinat(self):
    return self.__x_koordinat

@x_koordinat.setter
def x_koordinat(self, ny_verdi):
    if ny_verdi < 0:
        raise ValueError("Punktet kan ikke ha negativt x-
koordinat!")
    self.__x_koordinat = ny_verdi

# Eksempel på en beregnet egenskap: r
@property
def r(self):
    return self.avstand_origo()

@r.setter
def r(self, ny_verdi):
    theta = self.theta
    self.x_koordinat = ny_verdi*math.cos(theta)
    self.y_koordinat = ny_verdi*math.sin(theta)

# Eksempel på en beregnet egenskap: Theta
@property
def theta(self):
    return math.acos(self.x_koordinat/self.r)

@theta.setter
def theta(self, ny_verdi):
    r = self.r
    self.x_koordinat = r*math.cos(ny_verdi)
    self.y_koordinat = r*math.sin(ny_verdi)

```

```

# Mutator
def flytt(self, delta_x, delta_y):
    self.x_koordinat += delta_x
    self.y_koordinat += delta_y

# Query
def avstand_origo(self):
    return (self.x_koordinat**2 + self.y_koordinat**2)**0.5

# Avstand mellom to punkter, antar at det som kommer inn er
# et Punkt-objekt
def avstand(self, annet_punkt):
    xdiff = self.x_koordinat - annet_punkt.x_koordinat
    ydiff = self.y_koordinat - annet_punkt.y_koordinat
    return (xdiff**2 + ydiff**2)**0.5

# Gir en strengrepresentasjon av objektet som skal gi mening for
# en bruker
def __str__(self):
    return f"Punkt: ({self.x_koordinat}, {self.y_koordinat})"

# Gir en strengrepresentasjon til mer internt bruk. Du skal være
# i stand til å rekonstruere objektet fra denne
def __repr__(self):
    return f"Punkt ({self.x_koordinat}, {self.y_koordinat})"

# Eksempel på en funksjon som tar inn to punkter: Regner ut
# avstanden mellom dem. Denne gjør det samme som avstand-metoden i
# klassen Punkt og viser en annen måte å formulere samme funksjon
# på.
def avstand_punkter(punkt1, punkt2):
    xdiff = punkt1.x_koordinat - punkt2.x_koordinat
    ydiff = punkt1.y_koordinat - punkt2.y_koordinat
    return (xdiff**2 + ydiff**2)**0.5

# Eksempel på en funksjon som modifierer punktene den får inn

```

```
def flytt_til_midten(punkt1, punkt2):
    snitt_x = (punkt1.x_koordinat + punkt2.x_koordinat)/2
    snitt_y = (punkt1.y_koordinat + punkt2.y_koordinat)/2
    punkt1.x_koordinat = snitt_x
    punkt1.y_koordinat = snitt_y
    punkt2.x_koordinat = snitt_x
    punkt2.y_koordinat = snitt_y

if __name__ == "__main__":
    punkt1 = Punkt(3, 4)
    print(punkt1)
    print(punkt1.avstand_origo())
    punkt2 = Punkt()
    print(punkt2)
    print(punkt2.avstand_origo())
    punkt1.flytt(2, -1)
    print(punkt1)
    print(punkt1.avstand_origo())
    punkt3 = Punkt(5, 10)
    avstanden = punkt1.avstand(punkt3)
    print(f"Avstanden mellom {punkt1} og {punkt3} er {avstanden}")
```

Question 6

Attached



```
historikk = list()
historikk.append([0, 0, 0, 0])
fortsetter = True
while fortsetter:
    hvilken_bryter = input("Skriv inn hvilken bryter (1, 2 eller 3) som
skifter verdi: ")
    if hvilken_bryter == "":
        fortsetter = False
        break
    tid = int(input("Skriv inn tidspunktet innslaget skjedde: "))
    forrige_innslag = historikk[-1]
    dette_innslaget = forrige_innslag
    dette_innslaget[0] = tid
    bryter_indeks = int(hvilken_bryter)
    if dette_innslaget[bryter_indeks] == 0:
        dette_innslaget[bryter_indeks] = 1
    else:
        dette_innslaget[bryter_indeks] = 0
    historikk.append(dette_innslaget)
print("\n Resultat:")
print(historikk)
```

Question 9

Attached




```

class Punkt:
    def __init__(self, x_koordinat, y_koordinat):
        self.x_koordinat = x_koordinat
        self.y_koordinat = y_koordinat

    def avstand(self, annet_punkt):
        avstand = ((self.x_koordinat - annet_punkt.x_koordinat)**2 +
                    (self.y_koordinat - annet_punkt.y_koordinat)**2)**0.5
        return avstand

    def __str__(self) -> str:
        return f"({self.x_koordinat}, {self.y_koordinat})"

class Linje:
    def __init__(self, startpunkt, sluttpunkt):
        self.punktliste = list()
        self.punktliste.append(startpunkt)
        self.punktliste.append(sluttpunkt)

    def legg_til_punkt(self, punkt):
        self.punktliste.append(punkt)

if __name__ == "__main__":
    punkt1 = Punkt(1, 2)
    punkt2 = Punkt(1, 6)
    print(punkt1.avstand(punkt2))
    linje = Linje(punkt1, punkt2)
    punkt3 = Punkt(8, 6)
    punkt4 = Punkt(1, 2)
    linje.legg_til_punkt(punkt3)
    linje.legg_til_punkt(punkt4)
    print(linje)

```