

i Informasjon om eksamenen

Eksamen i: DAT120 grunnleggende programmering

Dato: 11.03 2022

Varighet: 4 timer

Tillatte hjelpemidler:

Du har lov til å ta med ei av de to anbefalte lærebøkene på eksamen, enten "Starting out with Python av Tony Gaddis (3., 4. og 5. utgave er alle OK)" eller "Python for realfag av Finn Aakre Haugen og Marius Lysaker (1. og 2. utgave er begge OK)". Du har lov til å ta med to A4-ark med egne notater på eksamen. Disse notatene kan være enten håndskrevne eller maskinskrrevne. Du kan skrive på begge sidene av A4-arket.

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 hjelpemetoder 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.

Ved spørsmål:

Kontakt faglærer Erlend Tøssebro på telefon +47 481 07 119

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 ei for-løkke eller ei while-løkke

continue

Hopper tilbake til starten av løkka den står i og tar neste verdi for ei 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 operatører

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 operatører

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(<tail>)

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.

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.

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 stenglengde 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.

Spesialmetoder brukt i temaet 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
clone <lenke til eksternt repo>	Lager et lokale repo og en lokal filstruktur som en kopi av et eksternt 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

Bokmål: Beregn arealet til en sirkel
 Nynorsk: Rekn ut arealet til ein sirkel
 English: Compute the area of a circle

```
import math

radius_streng = input("Skriv inn radius til sirkelen: ")
radius = float(radius_streng)
areal = math.pi*radius*radius
omkrets = 2.0*math.pi*radius
print("Areal: ", end="")
print(areal)
print("Omkrets: ", end="")
print(omkrets)
```

Bokmål: Er personen en tenåring?
 Nynorsk: Er personen ein tenåring?
 English: Is the person a teenager

```
tall_streng = input("Skriv inn en alder: ")
alder = int(tall_streng)
if alder >= 13 and alder < 18:
    print("Personen er tenåring")
else:
    print("Personen er ikke tenåring")
```

Bokmål: Er dette tallet for måned gyldig?
 Nynorsk: Er dette talet for månad gyldig?
 English: Is this number for month valid?

```
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")
```

Bokmål: Løsning av andregradslikninger
 Nynorsk: Løysing av andregradslikningar
 English: Solutions to quadratic equations

```
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")
```

Bokmål: Fakultet, med sjekk av lovlig input
 Nynorsk: Fakultet, med sjekk av lovleg input
 English: Factorial, with check for legal input

```
tall = int(input("Skriv inn tallet du ønsker fakultet av: "))
while tall < 0:
    print("Fakultet for negative tal 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-løkka er ferdig")
print(resultat)
```

Bokmål: La brukeren skrive inn flere linjer tekst. Avslutt med tom linje

Nynorsk: La brukaren skrive inn fleire linjer tekst. Avslutt med tom linje

English: Let the user enter several lines of text. The program stops and prints the text when the user enters an empty line

```
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)

Bokmål: Regn ut bokstavkarakter fra prosentsscore
Nynorsk: Rekn ut bokstavkarakter frå prosentsscore
English: Compute a grade from a percentage 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")

Bokmål: For-løkke som teller fra 10 og ned til 0
Nynorsk: For-løkke som tel frå 10 og ned til 0
English: For-loop that counts from ten and down to 0

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

Bokmål: Skriv ut en firkant av stjerner.
Eksempelutskrift med høyde 3 og bredde 5:
*****
*****
*****

Nynorsk: Skriv ut en firkant av stjerner.
Døme på utskrift med høgde 3 og bredde 5:
*****
*****
*****

English: Print a square of stars
Example printput with height 3 and width 5:
*****
*****
*****

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

for j in range(hoyde):
    for i in range(bredde):
        print("*", end="")
    print()

Bokmål: Areal og omkrets av en sirkel som funksjon
Nynorsk: Areal og omkrets av ein sirkel som funksjon
English: Area and circumference of a circle as a function

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"Arealet ble: {areal_global:8.2f}")
omkrets = omkrets_sirkel(radius Bruker)
print(f"Omkretsen ble: {omkrets:8.2f}")

Bokmål: Skriv ut en firkant av tegn som funksjon
Nynorsk: Skriv ut ein firkant av teikn som funksjon
English: Print a square of characters as a function

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")

```

Bokmål: Funksjon som lar brukeren skrive inn et positivt heltall. Brukeren får prøve på nytt helt til brukeren skriver

Nynorsk: Funksjon som lar brukaren skrive inn eit positivt heiltal. Brukaren får prøve på nytt heilt til brukaren sk

English: Function that reads positive integer from the user and lets the user try again if the user enters 0 or a ne

```

def les_positivt_heltall(beskjed):
    tall = int(input(beskjed))
    while tall < 0:
        print("Du må skrive inn et ikke-negativt heltall!")
        tall = int(input(beskjed))
    return tall

```

Bokmål: Øving 4 oppgave a-b: Lag et program som lar brukeren skrive inn en alder og så skriver ut prisen for en bussl

Nynorsk: Øving 4 oppgave a-b: Lag eit program som lar brukaren skrive inn ein alder og så skriv ut prisen for ein bu

English: Exercise 4 task a-b: Write a program that lets the user enter an age, and the program then prints the price

```

# Oppgave a)
def pris_bussbillett(alder):
    if alder < 18 or alder >= 67:
        pris = 20
    else:
        pris = 40
    return pris

```

```

# Oppgave b)
if __name__ == "__main__":
    alder_fra_bruker = les_positivt_heltall("Alder: ")
    pris = pris_bussbillett(alder_fra_bruker)
    print(f"Pris: {pris} kr")

```

Bokmål: Øving 4 oppgave d-e: Lag en funksjon som regner ut kinetisk energi til et objekt fra massen og farten til ob

Nynorsk: Øving 4 oppgave d-e: Lag ein funksjon som rekner ut kinetisk energi til eit objekt frå massen og farten til

English: Exercise 4 task d-e: Write a function that computes the kinetic energy of an object from the mass and veloc

```

# Oppgave d)
def kinetisk_energi(masse, fart):
    return 0.5*masse*fart*fart

```

```

# Oppgave e)
if __name__ == "__main__":
    print("Beregner kinetisk energi")
    masse = float(input("Massen til legemet: "))
    fart = float(input("Farten til legemet: "))
    energi = kinetisk_energi(masse, fart)
    print(f"Med masse {masse} og fart {fart} blir den kinetiske energien" +
          f"{energi:8.2f}")

```

Bokmål: Leser inn ei tekstfil og skriver ut innholdet

Nynorsk: Les inn ei tekstfil og skriv ut innhaldet

English: Reads a text file and prints the contents

```

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

```

Bokmål: Løsningsforslag for øving 5: Øvingen gikk ut på å lese inn ei tekstfil med eposter og plukke ut alle avsende

Nynorsk: Løysingsforslag for øving 5: Øvingen gjekk ut på å lese inn ei tekstfil med epostar og plukke ut alle avsend

English: Solution to exercise 5: Read a text file containing email addresses and find the email addresses of all the

```

epost_filnavn = input("Skriv inn navnet på epostfila: ")
avsendere_filnavn = input("Skriv inn navnet på fila med avsendere som skal skrives ut: ")
try:
    with open(epost_filnavn, "r") as epost_fil:
        with open(avsendere_filnavn, "w") as avsendere_fil:
            for linje in epost_fil:
                linje = linje.strip()
                if linje.find("From:") == 0:

```

```

        start = linje.find("<") + 1
        slutt = linje.find(">")
        epost = linje[start:slutt]
        avsendere_fil.write(epost + "\n")
except FileNotFoundError:
    print("Klarer ikke å åpne en av disse filene!")
except IOError as e:
    print("En feil oppsto under filbehandling: " + str(e))

Bokmål: Løsningsforslag for øving 6: Øvingen gikk ut på å lese inn en semikolondelt CSV fil med data fra en tempera
Nynorsk: Løysingsforslag for øving 6: Øvingen gjekk ut på å lese inn ein semikolondelt CSV fil med data frå ein temp
English: Solution to exercise 6: The task here is to read a CSV file of the following format: "date and time;Time si

# Konstant til del f
ANTALL_MAALINGER_MIDDLE_OVER = 60

# Til del f
def standardavvik(gjennomsnitt, liste_av_verdier):
    sum_av_kvadrater = 0.0
    for element in liste_av_verdier:
        sum_av_kvadrater += (element-gjennomsnitt)**2
    avvik = math.sqrt(sum_av_kvadrater/len(liste_av_verdier))
    return avvik

# Til del f
def lopende_middel(tidspunkter, temperaturer, antall_aa_midle):
    lopende_sum = 0
    resultat_tidspunkter = list()
    resultat_temperaturer = list()
    resultat_avvik = list()
    for i in range(antall_aa_midle):
        lopende_sum += temperaturer[i]
        resultat_tidspunkter.append(tidspunkter[i-(antall_aa_midle//2)])
        resultat_temperaturer.append(lopende_sum/antall_aa_midle)
        resultat_avvik.append(standardavvik(lopende_sum/antall_aa_midle, temperaturer[0:antall_aa_midle]))
        i += 1
    while i < len(tidspunkter):
        lopende_sum += temperaturer[i]
        lopende_sum -= temperaturer[i-antall_aa_midle]
        resultat_tidspunkter.append(tidspunkter[i-(antall_aa_midle//2)])
        resultat_temperaturer.append(lopende_sum/antall_aa_midle)
        i += 1
        resultat_avvik.append(standardavvik(lopende_sum/antall_aa_midle, temperaturer[i-antall_aa_midle:i]))
    return resultat_tidspunkter, resultat_temperaturer, resultat_avvik

# Definerer listene over verdier, til del a
tider = list()
tidspunkter = list()
temperaturer = list()
absolutt_trykk = list()

# Definerer lister til del e
tidspunkter_barometrisk_trykk = list()
barometrisk_trykk = list()

# Leser fila, del a
with open("trykk_og_temperaturlogg.csv") as fila:
    fila.readline() # Leser kolonnetitlene, som jeg ikke bruker
    for linje in fila:
        linje = linje.replace(",", ".")
        kolonner = linje.split(";")
        tider.append(kolonner[0])
        tidspunkter.append(float(kolonner[1]))
        try: # Denne try...except blokken er til del e
            bar_trykk = float(kolonner[2])
            barometrisk_trykk.append(bar_trykk)
            tidspunkter_barometrisk_trykk.append(float(kolonner[1]))
        except ValueError:
            pass
        absolutt_trykk.append(float(kolonner[3]))
        temperaturer.append(float(kolonner[4]))

# Del b
plt.subplot(2, 2, 1)
plt.title("Temperaturmålinger")
print(f"Antall elementer i listene: tidspunkter: {len(tidspunkter)}, Temperaturer: {len(temperaturer)}")
plt.plot(tidspunkter, temperaturer)
plt.xlabel("Tidspunkt, sekunder etter start")
plt.ylabel("Temperatur")

# Del b og e
plt.subplot(2, 2, 2)
plt.title("Trykkmålinger")
plt.plot(tidspunkter_barometrisk_trykk, barometrisk_trykk, label="Barometrisk Trykk")

```

```

plt.plot(tidspunkter, absolutt_trykk, label="Absolutt trykk")
plt.xlabel("Tidspunkt, sekunder etter start")
plt.ylabel("Trykk i bar")
plt.legend()

# Del b og f
plt.subplot(2, 2, 3)
middel_tidspunkter, middel_temperaturer, middel_avvik = lopende_middel(tidspunkter, temperaturer, ANTALL_MAALINGER_M:
plt.title("Temperaturmålinger med midling")
plt.plot(tidspunkter, temperaturer, label="Temperaturer")
plt.plot(middel_tidspunkter, middel_temperaturer, label="Midleste temperaturer")
plt.plot(middel_tidspunkter, middel_avvik, label="Avvik mellom middel og målt temperatur")
plt.xlabel("Tidspunkt, sekunder etter start")
plt.ylabel("Temperatur")
plt.legend()

# Del c
plt.subplot(2, 2, 4)
plt.title("Histogram over temperaturer")
plt.hist(temperaturer, bins=np.arange(8, 24, 0.25))
plt.show()

# Del d
print("Tidspunkter hvor målingene stopper opp")
for i in range(len(tidspunkter)-1):
    if tidspunkter[i+1]-tidspunkter[i] > 10:
        print(f"Målingene stopper mellom {tidspunkter[i]} og {tidspunkter[i+1]}. Tider: {tider[i]} til {tider[i+1]}")

Bokmål: Klasse for et punkt

Nynorsk: Klasse for eit punkt

English: Class for a point

import math

class Punkt:
    # Konstruktør
    def __init__(self, start_x=0, start_y=0):
        self.__x_koordinat = start_x
        self.y_koordinat = start_y

    @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

    # 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)

    # 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
    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
def __repr__(self):
    return f"Punkt ({self.x_koordinat}, {self.y_koordinat})"

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

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}")

```

Bokmål: Løsningsforslag for øving 8 oppgave b-c: Klasse for et spørsmål til et spørrespill

Nynorsk: Løysingsforslag for øving 8 oppgave b-c: Klasse for eit spørsmål til eit spørjespel

English: Solution to exercise 8 task b and c: Class for a question in a questionnaire game:

```

class Sporsmaal:
    def __init__(self, sporsmaal, alternativer, korrekt_svar=0):
        self.sporsmaal = sporsmaal
        self.alternativer = alternativer
        self.korrekt_svar = korrekt_svar

    def __str__(self):
        resultat = self.sporsmaal + "\nSvaralternativer:\n"
        for index, verdi in enumerate(self.alternativer):
            resultat += f"{index}: {verdi}\n"
        return resultat

    def sjekk_svar(self, svaret):
        if svaret == self.korrekt_svar:
            return True
        else:
            return False

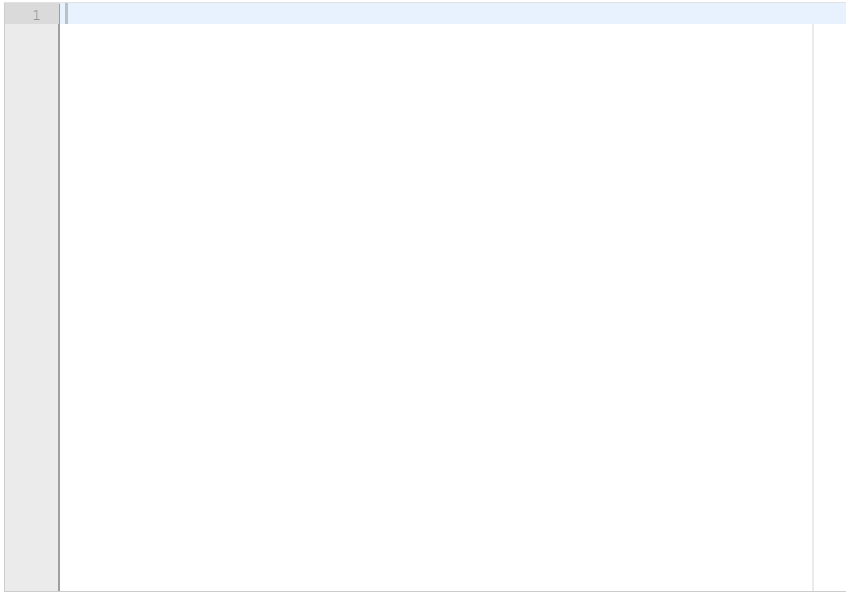
if __name__ == "__main__":
    sp1 = Sporsmaal("Hva heter hovedstaden i Sverige?", ["Oslo", "Stockholm", "Göteborg", "Norge"], 1)
    sp2 = Sporsmaal("Hva er det største universitetet i Norge?", ["UiO", "UiS", "NTNU", "UiB", "UiT"], 2)
    print("Spørsmål 1: ")
    print(sp1)
    svar = int(input("Ditt svar: "))
    if sp1.sjekk_svar(svar):
        print("Korrekt")
    else:
        print("Feil")
    print()
    print("Spørsmål 2: ")
    print(sp2)
    svar = int(input("Ditt svar: "))
    if sp2.sjekk_svar(svar):
        print("Korrekt")
    else:
        print("Feil")

```

1 Basis Python script

Skriv et Python script som lar brukeren skrive inn et flyttall, regner ut kvadratet av flyttallet (flyttallet opphøyd i andre potens), og skriver ut resultatet. (6%)

Skriv ditt svar her



Maks poeng: 6

2 Aritmetisk tallfølge

En aritmetisk sekvens er en serie med heltall som starter med et oppgitt tall a_0 , inneholder n tall, og hvert tall er d høyere enn det forrige tallet. Et eksempel med $a_0=1$, $n=10$ og $d=3$ er oppgitt her:

1
4
7
10
13
16
19
22
25
28

Skriv et Python script hvor du bruker input-setninger for å la brukeren skrive inn a_0 , d og n , og så skriver du ut den resulterende tallfølgen. (8%)

Skriv ditt svar her

1	
---	--

Maks poeng: 8

3 Kodeforståelse, løkker

```
def funksjon(tall):  
    for i in range(tall):  
        print(i, end="")  
        print(" ", end="")  
        for j in range(i):  
            print("#", end="")  
        print()
```

Hva blir skrevet ut hvis denne funksjonen blir kalt med parameteren 6? (8%)

Skriv ditt svar her

Format | **B** | *I* | U | x_2 | x^2 | I_x | | | | | | | | | | |

Words: 0

Maks poeng: 8

4 Flervalgsoppgave

For hver oppgave, velg korrekt svaralternativ: (3% for hver deloppgave)

Deloppgave 1: Hva blir skrevet ut av følgende Python script?

```
a = 15
if a < 10:
    b = 1
else:
    b = 0
if a < 20:
    b = 2
if a < 30:
    b = 3
print(b)
```

Velg ett alternativ

- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 2
- ☐ 15

Deloppgave 2: Hva blir skrevet ut av følgende Python script?

```
import math
a = "Test"
b = "B"
resultat = "Resultat"
def slaar_sammen(a, b):
    resultat = a + b*2
```

```
slaar_sammen(a, b)
print(resultat)
```

Velg ett alternativ:

- ☐ TestBB
- ☐ Du får en exception siden * operatoren ikke fungerer på en streng og et heltall
- ☐ Resultat
- ☐ None
- ☐ TestB

Deloppgave 3: Hva blir skrevet ut av følgende Python script:

```
def funksjon(verdi):
    tall = 1
    while tall < verdi:
        print(tall, end="; ")
        tall += 3
    if tall > 10:
        break
    print(23, end="; ")
```

```
if __name__ == "__main__":
    funksjon(20)
    print(26, end="; ")
```


Velg ett alternativ

- ☐ 1; 4; 7; 10;
- ☐ 1; 4; 7; 10; 23; 26;
- ☐ 1; 4; 7; 10; 13; 16; 19; 23; 26;
- ☐ 1; 4; 7; 10; 13; 16; 19; 21; 23; 26;
- ☐ 1; 4; 7; 10; 26;

Deloppgave 4: Hva blir skrevet ut av følgende kode hvis brukeren skriver inn tallet 5 for lengde og 3 for bredde?

```
lengde = input("Skriv inn lengden til firkanten: ")
bredde = input("Skriv inn bredden til firkanten: ")
omkrets = lengde*2 + bredde*2
print(omkrets)
```

Velg ett alternativ

- ☐ None
- ☐ 16
- ☐ 5533
- ☐ 53
- ☐ Du får en exception i linje 3 siden du ikke kan gange en streng med et tall

Deloppgave 5: Koden under er løsningsforslag til øving 1 oppgave 5. Hva skjer hvis brukeren skriver inn noe som ikke er et lovlig flyttall?

```
TYNGDE_AKSELLERASJON = 9.81
print("Fallende objekt")
tallstreng = input("Hvor mange sekunder har objektet falt? ")
tid_sekunder = float(tallstreng)
fart = tid_sekunder*TYNGDE_AKSELLERASJON
distanse = 0.5*fart*tid_sekunder
print(f"Etter {tid_sekunder:6.2f} sekunder har objektet fart {fart:6.2f}m/s"+
      f" og har falt {distanse:6.2f} meter")
```

Velg ett alternativ

- ☐ Du får en ValueError i linje 4 og programmet stopper der
- ☐ tid_sekunder blir tilordnet verdien 0.0 i linje 4, og programmet fortsetter med denne verdien
- ☐ tid_sekunder blir tilordnet spesialverdien NaN (not a number) i linje 4, og dette blir resultatet av samtlige av de matematiske operasjonene i programmet
- ☐ tid_sekunder blir tilordnet verdien None, og du får en TypeError i linje 5 siden None ikke kan ganges med flyttallet TYNGDE_AKSELLERASJON

Deloppgave 6: Gitt følgende 6 script. Hvilket av scriptene er en korrekt løsning av følgende oppgave:

La brukeren skrive inn radius og høyde til en sylinder. Programmet skal regne ut og deretter skrive ut volumet til sylinderen. Volumet til en sylinder er gitt med formelen

$$volum = \pi * høyde * radius^2$$

```
# Kode 1
print("Volumet til en sylinder")
radius = input("Radius: ")
hoyde = input("Høyde: ")
```

```
volum = math.pi*hoyde*radius**2
print("Volumet er: {volum:8.2f}")
# Kode 2
print("Volumet til en sylinder")
radius = float(input("Radius: "))
hoyde = float(input("Høyde: "))
volum = math.pi*hoyde*radius**2
# Kode 3
print("Volumet til en sylinder")
radius = float(input("Radius: "))
hoyde = float(input("Høyde: "))
volum = math.pi*hoyde*radius**2
print("Volumet er: {volum:8.2f}")
# Kode 4
print("Volumet til en sylinder")
radius = float(input("Radius: "))
hoyde = float(input("Høyde: "))
volum = math.pi*hoyde*radius**2
print("Volumet er: {volum:8.2f}")
# Kode 5
print("Volumet til en sylinder")
radius = float(input("Radius: "))
hoyde = float(input("Høyde: "))
volum = math.pi*hoyde*radius**2
print("Volumet er: {volum:8.2f}")
# Kode 6
print("Volumet til en sylinder")
radius = float(input("Radius: "))
hoyde = float(input("Høyde: "))
print("Volumet er: {radius:8.2f}")
volum = math.pi*hoyde*radius**2
```

Velg ett alternativ

- ☐ Kode 5
- ☐ Kode 3
- ☐ Kode 6
- ☐ Kode 2
- ☐ Kode 4
- ☐ Kode 1

Maks poeng: 18

5 Funksjon

Skriv en funksjon som bruker Turtle Graphics for å tegne en taggete linje. Et eksempel på en slik linje er gitt under. Funksjonen skal ta to parametere: Antall tagger og størrelsen til hver tagg. Eksemplet under viser ei linje med 10 tagger og taggstørrelse 30. Taggstørrelsen skal ha default verdi 15. (6%)

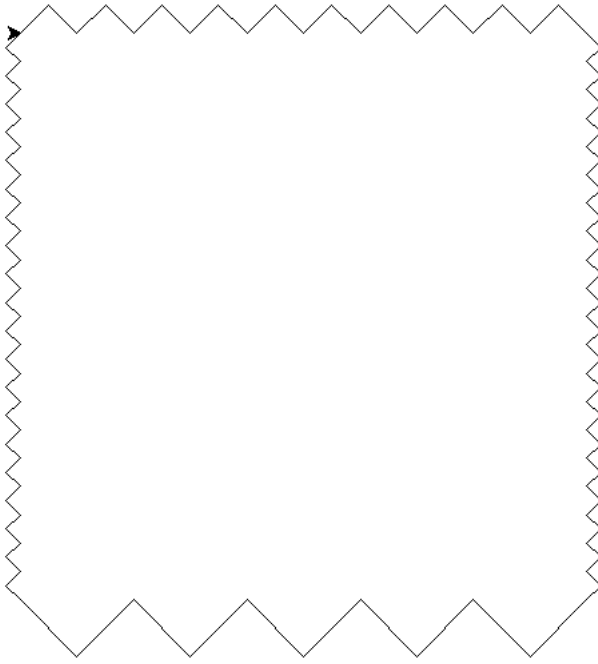
Etterpå, skriv en if `__name__ == "__main__"` blokk som bruker denne funksjonen for å tegne en firkant av taggete linjer, liknende men ikke nødvendigvis helt lik den som er oppgitt under. (4%)

Skriv ditt svar her

Eksempel på taggete linje med taggstørrelse 30 og 10 tagger:



Eksempel firkant med taggete linjer. Den på toppen har 10 tagger og taggstørrelse 30. De på sidene har default taggstørrelse og 20 tagger. Den på bunnen har 5 tagger og taggstørrelse 60.



Maks poeng: 10

6 Problemløsning

Skriv en funksjon som tar som parameter ei liste som inneholder heltall mellom 1 og 6. Funksjonen din skal finne ut om det er et tall som forekommer tre eller flere ganger i lista. Hvis det er et tall som forekommer minst tre ganger skal funksjonen returnere en verdi lik $3 \cdot \text{tallet}$. Hvis det er flere enn ett tall som forekommer minst tre ganger skal funksjonen beregne returverdien basert på det høyeste av dem. Hvis det ikke er noen tall som forekommer minst tre ganger i lista skal funksjonen returnere 0. (10%)

Skriv ditt svar her

1	
---	--

Maks poeng: 10

7 Lesing av fil

Gitt ei fil med historiske værdata på følgende format:

Navn på værstasjon; Kode for værstasjon; dato; middeltemperatur; nedbør

1. Skriv et Python script som leser gjennom denne fila og finner total nedbør for hele perioden som fila dekker. Scriptet skal skrive ut total nedbør til konsollet (8%)
2. Utvid scriptet fra oppgave 1 til å også finne høyeste og laveste middeltemperatur. Scriptet skal skrive ut høyeste middeltemperatur, datoen som hadde høyeste middeltemperatur, laveste middeltemperatur og datoen som hadde laveste middeltemperatur. (4%)
3. Fila inneholder informasjon fra flere værstasjoner. La brukeren skrive inn en søkestreng og tell med bare linjer i fila hvor navnet på værstasjonen passer med søkestrengen, det vil si at søkestrengen er med i navnet på værstasjonen. (4%)
4. Finn informasjonen i oppgave 1-2 (total nedbør, høyeste middeltemperatur med dato, laveste middeltemperatur med dato) for hver værstasjon i fila. Du vet ikke på forhånd hvilke værstasjoner fila inneholder. Skriv resultatene for alle værstasjonene, ei linje for hver værstasjon, til ei ny fil. Hver linje skal inneholde navnet til værstasjonen, total nedbør for værstasjonen, høyeste middeltemperatur for værstasjonen, dato for høyeste middeltemperatur for den værstasjonen, laveste middeltemperatur for værstasjonen, og dato for laveste middeltemperatur for den værstasjonen. (6%)
5. Håndter feil. Du skal både håndtere feil i fillesinga og håndtere feil format på linjene slik som at de ikke kan splittes i riktig antall kolonner eller at de kolonnene som skal være flyttall ikke kan tolkes som dette (4%)

Skriv ditt svar her

1	
---	--

Her er de første linjene i ei potensiell datafil for denne oppgaven:

Navn;Stasjon;Tid(norsk normaltid);Middeltemperatur (døgn);Nedbør (døgn)

Stavanger - Våland;SN44640;30.12.2021;7,3;1,6

Stavanger - Våland;SN44640;31.12.2021;6,9;6,2

Stavanger - Våland;SN44640;01.01.2022;6,5;0,3

Stavanger - Våland;SN44640;02.01.2022;7,5;2,8

Stavanger - Våland;SN44640;03.01.2022;4,9;3

Stavanger - Våland;SN44640;04.01.2022;3,5;3

Stavanger - Våland;SN44640;05.01.2022;1,6;2,2

Stavanger - Våland;SN44640;06.01.2022;0,4;0

Maks poeng: 26

8 Objektorientert programmering

Gitt klassene kort og kortstokk oppgitt under. Skriv en funksjon som tar inn som parameter en instans av klassen Kortstokk. Funksjonen skal returnere et heltall som sier hvor mange poeng denne kortsatokken representerer.

Deoppgaver:

- Poengene til en kortstokk er summen av poengene til hvert kort i stokken. Ess gir 15 poeng. Bildekort (knekt, dame, konge) gir 10 poeng. Andre kort gir poeng lik verdien til kortet. (10%)
- For å gi poeng må alle kortene i kortstokken være av samme korttype (hjerter, spar, ruter, kløver). Hvis ikke alle kortene i kortstokken er av samme type skal funksjonen returnere 0 poeng. (4%)

Skriv ditt svar her

```
import random
```

```
# Korttypene som konstanter
KLOVER = 1
RUTER = 2
SPAR = 3
HJERTER = 4
```

```
class Kort:
    def __init__(self, korttype, verdi):
        self.korttype = korttype
        self.verdi = verdi

    def __str__(self):
        korttypene = ["ingen", "Kløver", "Ruter", "Spar", "Hjerter"]
        if self.verdi == 1:
            return korttypene[self.korttype] + " ess"
        elif self.verdi <= 10:
            return f"{korttypene[self.korttype]} {self.verdi}"
        elif self.verdi == 11:
            return korttypene[self.korttype] + " knekt"
        elif self.verdi == 12:
            return korttypene[self.korttype] + " dame"
        elif self.verdi == 13:
            return korttypene[self.korttype] + " konge"
        else:
            return "Ugyldig kort!"
```

```
class Kortstokk:
    def __init__(self):
        self.kortene = list()

    def lag_kortene(self):
        for t in range(1, 5):
            for v in range(1, 14):
                self.kortene.append(Kort(t, v))

    def trekk(self):
        kortet = self.kortene[-1]
        del self.kortene[-1]
        return kortet
```

```
def stakk(self):
    random.shuffle(self.kortene)
def __str__(self):
    resultat = f"Kortstakk med {len(self.kortene)} kort\n"
    for kort in self.kortene:
        resultat += str(kort) + "\n"
    return resultat
```

Maks poeng: 14