TDT4109 - s2024 - Forside

Institutt for datateknologi og informatikk

Eksamensoppgåve i TDT4109 - Informasjonsteknologi, grunnkurs

Eksamensdato: 6. august 2024

Eksamenstid (frå-til): 09:00-13:00

Hjelpemiddelkode/Tillatte hjelpemidler: D / Ingen trykte eller handskrivne hjelpemiddel tillate. Bestemt, enkel kalkulator tillate.

Fagleg kontakt under eksamen: Børge Haugset
Tlf.: 934 20 190

Fagleg kontakt møter i eksamenslokalet: NEI

ANNA INFORMASJON

Skaff deg eit overblikk over oppgåvesettet før du byrjar å svare på oppgåvene.

Les oppgåvene nøye, gjer deg opp dine eigne meiningar og presiser i svara dine kva for føresetnadar du har lagt til grunn i tolking/avgrensing av oppgåva. Fagleg kontaktperson kontaktast berre dersom du meiner det er direkte feil eller manglar i oppgåvesettet. Vend deg til ei eksamensvakt om du meiner det er feil eller manglar. Noter spørsmålet ditt på førehand.

Ingen handteikningar: Denne eksamenen tillèt ikkje bruk av handteikningar. Om du likevel har fått utdelt skanne-ark, er dette en feil. **Arka vil ikkje bli akseptert for innlevering, og dei vil difor heller ikkje sendast til sensur.**

Vekting av oppgåvene: Vektinga til oppgåvene står i eksamen. Poeng er ca. lik prosent, med atterhald om at dette kan endrast under sensur.

Varslingar: Dersom det oppstår behov for å gje beskjedar til kandidatane medan eksamen er i gang (t.d. ved feil i oppgåvesettet), vil dette bli gjort via varslingar i Inspera. Eit varsel vil dukka opp som ein dialogboks på skjermen i Inspera. Du kan finna att varselet ved å klikka på bjølla i øvre høgre hjørne på skjermen.

Trekk frå/avbroten eksamen: Blir du sjuk under eksamen, eller av andre grunnar ynskjer å levere blankt/avbryte eksamen, gå til “hamburgermenyen” i øvre høgre hjørne og vel «Lever blankt». Dette kan ikkje angrast sjølv om prøven framleis er open.

Tilgang til svara dine: Etter eksamen finn du svara dine i arkivet i Inspera. Merk at det kan ta ein vyrkedag før eventuelle handteikningar vert tilgjengelege i arkivet.

Eksamensstruktur og råd

Vekting av oppgåvene: Det er angitt i prosent kor mykje kvar deloppgåve i eksamenssettet tel ved sensur. Poengdelen er ca. lik prosent. Altså 5 poeng ~= 5% av eksamen. Me reserverer for oss retten til å kunna avvika frå desse andelane i sensurarbeidet viss dette skulle bli nødvendig. Les gjennom heile oppgåvesettet før du byrjar å løysa oppgåva. Disponer tida godt!

Eksamen er delt i tre seksjonar:

- Del 1: Teoridel som tel ca. **25%** av den totale vurderinga. I denne seksjonen blir 25 poeng fordelte mellom 5 og 20 rette, sidan ein med gjetting kan få 25% korrekt. Ein får derimot uansett aldri mindre enn 0 poeng på del 1
- Del 2 (oppgaver 2.1 - 4.6): Ulike automatisk retta programmeringsoppgåver med multiple choice, drag and drop og innfylling av resultat. Desse tel til saman ca. **45%**.
- Del 3: Programmering. Desse tel til saman ca. **30%**. Du kan ikkje køyra koden.

Struktur, info og tips:

- Det er **ikkje** minuspoeng for å svara feil i del 2, så gjett heller enn å hoppa over oppgåva.
- Les oppgåvene nøye (som det står på førre side), så du ikkje misforstår dei. Nokre oppgåver spør m.a. kva alternativ som er feil, ikkje rett.
- Svar det du får til på programmeringsoppgåvene. Hugsar du ikkje korleis å løysa heile oppgåva, men deler av ho? Gjer det. Skriv du ingenting får du 0 poeng. Skriv du litt, kan du få litt poeng. Og inkluder gjerne korte kommentarar i koden du skriv i programmeringsseksjonen, så det er lettare å skjønna kva du gjer eller prøver å gjera. **NB!** Viss du løyser oppgåvene på fleire måtar, vil me ta gjennomsnittet av delte ut poeng, heller enn å velja den beste. Det er altså *ikkje* nødvendigvis ein fordel å visa alle måtane ein kan løysa han på.
- Det er mange oppgåver på eksamen. Dei siste tek sannsynlegvis lengst tid. Fordel tida smart.
- Inspera liker ikkje kode i oppgåveteksten, og kodeblokkene har nokre gonger hoppa opp til toppen av oppgåva. Derfor har eg lagt til referense i tekst og i kodeblokka. Dette kan t.d. sjå sånn ut: *"I koden under (#2) ser me blabla..."*. Dette vil seia kodensnutten som startar med *"#2"*, som sikkert blir viste i toppen av oppgåva.

PC-tips:

- På vanleg **Windows**-tastatur kan me skriva følgjande teikn på følgjande måte:
  - (): shift + 8/9
  - []: alt.gr. + 8/9
  - {}: alt.gr + 7/0
  - /: shift + 7
  - \: tasten til venstre for backspace
  - ': tasten til venstre for enter
- På **Macbook** kan me skriva det same på følgjande måte:
  - (): shift + 8/9
  - []: option + 8/9
  - {}: option + shift + 8/9
  - /: shift + 7
  - \: option + shift + 7
  - ': tasten til venstre for 1
- Alt.gr. (Windows) finst rett til høgre for space/mellomrom
- option (Mac) er neders, litt til høgre og venstre for space/mellomrom
- Shift er pila opp, nesten nede i venstre hjørne

Lykke til!!

1 Teori (25%)

Marker det du meiner er det mest rette alternativet. Svaralternativa kjem i tilfeldig rekkjefølgje. Ved gjetting vil ein i snitt få rett på 5 av 20 oppgåver, gitt 4 alternativ. Poenga blir dermed distribuerte mellom 5 og 20 rette. Under 5 rette gir ikkje minuspoeng på eksamen, så gjett ivei!

1. Sekundærminne...

- er dyrare per byte enn primærminne.
- må ha straum for å lagra informasjon.
- lagrar informasjonen også etter at straumen er skrudd av. ✔
- er raskare enn primærminne.

2. Kva er binærrepresentasjonen til HEX-talet '1E'?

- 00111110
- 00011110 ✔
- 00011111
- 00011100

3. Kor mange byte treng ein for å representera eit 4K-bilde (3840x2160) viss biletet er ren svart-kvitt?

- 8294400
- 1036800 ✔
- 4147200
- 16588800

4. Kva påstand er korrekt for transistorar?

- Dei er ein slags brytarar som blir styrte ved hjelp av straum. ✔
- Dei erstatta radiorøyr i gamle dagar, men er no erstatta av ny teknologi.
- Dei blir brukte til å lagra informasjon permanent, til dømes i ein SSD.
- Dei kan gjera om digitale signal om til analoge signal, men ikkje motsett.

5. Kva er hovudoppgåva til ein ALU?

- Utføra rekneoperasjonar. ✔
- Henta og utføra instruksjonar.
- Styre programteljaren (Program Counter).
- Styre dataflyten mellom prosessoren og co-prosessoren.

6. Viss ascii-strengen 'berre' blir representert på binærformen 0110 0010 0110 0001 0111 0010 0110 0101 - korleis blir representert ascii-strengen 'bra'?

- 0110 0010 0111 0010 0110 0001 ✔
- 0110 0010 0111 0011 0110 0001
- 0110 0010 0111 0010 0110 0010
- 0110 0010 0111 0001 0110 0001

7. Kva ved kompilering og tolking av kode er korrekt:

- Kompilering betyr at éin og ei kodelinje blir omsett til datamaskinspråk, og blir køyrd.
- Tolking betyr at ein omset eit heilt program til eit anna og meir datamaskinnært språk som datamaskina så køyrer.
- Tolking betyr at éin og ei kodelinje blir omsett til datamaskinspråk, og blir køyrd. ✔
- Tolka kode er raskare å køyra enn kompilert kode.

8. Viss eg gir deg den heksadesimale koden "#ffff00" - kva burde du kalla fargen?

- blå
- gul ✔
- rød
- grønn

9. Kva er pipelining?

- Ein teknikk som gjer at ein CPU kan utføra fleire prosessar i parallel. ✔
- Ein teknikk som gjer at ein kan senda data mellom ulike delar av datamaskina (til dømes frå nettverksskort til CPU til SSD).
- Ein teknikk som blir brukt for å oppnå eit trygt samband mellom to datamaskiner (til dømes i bruk i VPN).
- Ein teknikk der lagringsmedium kan brukast i parallel for å bli raskare (til dømes i RAID).

10. Gitt UNICODE - kor mange byte er det minste ein kan bruka for å representera eit teikn?

- 2
- 1
- 3
- 4 ✔

11. Viss samplingsfrekvensen i ei sampling blir sett ned, kva slags endring medfører det?

- Spørsmålet er feil stilt - samplingsfrekvensen er definert ved hjelp av Nyquists teorem, og er 44.1kHZ.
- Ein kan introdusera meir støy.
- Ein kan mista lydinformasjon. ✔
- Volumet på signalet blir lågare.

12. Kva informasjonssikkerheitsbehov har me for data, i samsvar med McCumbers kube?

- Lagring og prosessering
- Lagring, overføring og prosessering
- Langtidslagring, lagring på RAM og prosessering
- Konfidensialitet, integritet og tilgjengelegheit ✔

13 - Kva seier Moores lov?

- Talet på transistorar i ein integrert krins blir omtrent dobla kvart anna år, på grunn av teknologiske framsteg. ✔
- Klokkefarten til ein prosessor aukar proporsjonalt med talet på transistorar prosessoren har.
- Jo meir diskplass ein har, jo meir plass tek spela og programma ein installerer.
- Samplingsfrekvensen er direkte knytt til volumet på signalet.

14. Organiser etter fart, raskast til tregast:

- Radiorøyr, relé, transistor
- Transistor, radiorøyr, relé ✔
- Transistor, relé, radiorøyr
- Relé, transistor, radiorøyr

15. Kontrolleininga har to register, den eine er instruksjonsregisteret. Kva heiter det andre?

- Seleksjonsregisteret
- Programtelleren ✔
- Dataregisteret
- Minnekontrollen

16. Me skal laga oss eit nytt git repository, og står inni eit terminalvindauge. Kva er kommandoen me skriv?

- git -i -s .
- git init ✔
- git repo .
- git make

17. Kva ligg i omgrepet konfidensialitet når me snakkar om informasjon?

- Når ein har brote seg inn i eit datasystem så har ein oppnådd konfidensialitet.
- Informasjonen er av ein art som gjer at ein ikkje ønskjer at kven som helst skal få endra på.
- Det er ikkje nokon andre enn spesielt definerte entitetar som får sjå informasjon. ✔
- Informasjon er av ein art som ein ikkje ønskjer at kven som helst skal få sjå.

18. Kva er eit register? (I konteksten datamaskinarkitektur)

- Ei eining for langsiktig lagring av data i hovudminnet til datamaskina.
- Ein tabell over korleis ein utfører rekneoperasjonar som AND, XOR og liknande som blir nytta av prosessoren.
- Eit spesielt område i RAM som held oversikt over alle filene på datamaskina.
- Ei rask og lita datalagringseining inne i CPU som blir brukt til å lagra og henta data for enkle operasjonar. ✔

19. Innhenting av data frå RAM til registeret blir gjord i ein spesiell fase, men kva for ein...

- Execute
- Fetch ✔
- Command
- Decode

20. Viss ein funksjon i Python tek inn ei liste som parameter, og ein gjer endringar på lista inni funksjonen, korleis kan endringane blir behaldne etter at funksjonen blir avslutta?

- Sidan dette er aaaaller siste gong eg skriv noka form for teorispørsmål til ITGK-eksamen så skal de som ein premie få eitt alternativ mindre på den siste oppgåva. Til gjengjeld kjem nokre av dykk til å lesa gjennom heile denne teksten, og dermed tapa bitte litt tid. Det trur eg går fint. Lykke til vidare! - Børge
- Viss ein skriv 'global liste' inni funksjonen så vil tolkaren skjønna at endringane skal behaldast.
- Viss ein i ein funksjon endrar på ei liste ein får inn som parameter så vil endringane i den originale lista behaldast sjølv utan at lista eksplisitt blir returnert. ✔
- Einaste moglegheit er å returnera den oppdaterte lista.

Maks poeng: 33.3

2.2 Kodeforståelse 2

```
# A
def read_file(filename):
    '''Lese inn filen `filename`.'''
    with open(filename, 'r') as file:
        text = file.read()
    return text

# B
def read_savegame():
    '''Bruker funksjonen `read_file`.'''
    filename = 'savegame.txt'
    text = read_file(filename)
    return text

# 1
def read_savegame():
    '''Bruker funksjonen `read_file`.'''
    filename = 'savegame.txt'
    try:
        text = read_file(filename)
    except FileNotFoundError:
        text = 'Save not found.'
    return text

# 2
import os
def read_savegame():
    '''Bruker funksjonen `read_file`.'''
    filename = 'savegame.txt'
    text = 'Save not found.'
    if os.path.exists(filename):
        text = read_file(filename)
    return text

# 3
import sys
def read_savegame():
    '''Bruker funksjonen `read_file`.'''
    filename = 'savegame.txt'
    if sys.exists(filename):
        text = read_file(filename)
    else:
        text = 'Save not found.'
    return text

# 4
import os
def read_savegame():
    '''Bruker funksjonen `read_file`.'''
    filename = 'savegame.txt'
    if filename in os.listdir():
        text = read_file(filename)
    else:
        text = 'Save not found.'
    return text
```

Me har funksjonane `read_file` (#A) og `read_savegame` (#B).

Det er ei moglegheit at fila (fila path/namn/lokasjon) me sender inn i `read_file` ikkje finnast. Då vil programmet krasja. Her er fire forslag på metodar å løysa dette på. Men éin av dei vil **ikkje** fungera. Kva fungerer **ikkje**?

Hint: Bruk sjå Python-dokumentasjonen for meir om try-except og modulane os/os.path og sys.

Vel eitt alternativ

- # 3 ✔
- # 1
- # 2
- # 4

Maks poeng: 3

2.3 Kodeforståelse 3

```
# 1
def hent_pris(kategori):
    '''Returner prisen på billetten, gitt alderen.'''
    if kategori == 'barn':
        return 0
    elif kategori == 'student':
        return 30
    elif kategori == 'pensjonist':
        return 20
    elif kategori == 'voksen':
        return 30

kategori = input('Velg kategori (barn, student, pensjonist eller voksen): ')
pris = hent_pris(kategori)
print(pris)
```

Kva påstand er sann?

Vel eitt alternativ

- Vi lager en string (med ''') i funksjonen, men den lagres ikke. Det er dårlig kodepraksis.
- Me lagar ein string (med ''') i funksjonen, men han blir ikkje lagra. Det gjer at programmet vil krasja.
- Viss brukaren skriv inn noko anna enn ein av kategoriane, vil `None` returnerast. D[illegible]` er dårleg kodepraksis. ✔
- Viss brukaren skriv inn noko anna enn ein av kategoriane, vil programmet krasja.

Maks poeng: 2

2.4 Kodeforståelse 4

```
# 1
prosjektmappe
- program.py
- undermappe
    - init.py
    - modul.py
```

Me har følgjande mappestruktur (#1) i prosjektet vårt.

Prosjektmappa vår heiter `prosjektmappe`. Inni ho har me programmet vårt `program.py`. Me har også ei undermappe kalla `undermappe` i prosjektmappa. Den har filene `init.py` og `modul.py` inni seg. Me vil importera koden frå `modul.py` inn i `program.py`, ved bruk av relativ import. (`prosjektmappe` er ikkje ein installert modul.) Kva skriv me i `program.py`?

Kva importering fungerer frå `program.py`?

Vel eitt alternativ

- import undermappe.modul ✔
- import prosjektmappe.undermappe.modul
- import modul
- import prosjektmappe.modul

Maks poeng: 2

2.5 Kodeforståelse 5

```
def f(a, b):
    return a % b == a * b * 3,14

type(f(3, 2))
```

Kva datatype(r) blir returnert? Eller krasjar han?

Vel eitt alternativ

- int eller float (avhengig av a og b sine datatyper)
- Error
- bool
- tuple ✔

Maks poeng: 2

2.6 Kodeforståelse 6

```
# 1
x = True or False
```

I denne oppgåva skal me finna ut kva kodelinjer som gir `True` eller `False`.

Kva er verdien til `x`? (#1)
Vel eitt alternativ

True ✔ False ○

```
# 2
y = True and 3 < 4
```

Kva er verdien til `y`? (#2)
Vel eitt alternativ

True ✔ False ○

```
# 3
z = (not True) and (not False)
```

Kva er verdien til `z`? (#3)
Vel eitt alternativ

False ✔ True ○

```
# 4
r = (True and False) or (True and True)
```

Kva er verdien til `r`? (#4)
Vel eitt alternativ

True ✔ False ○

```
# 5
s = True and True in [False, True, False, False]
```

Kva er verdien til `s`? (#5)

Vel eitt alternativ

False	True
○	✔

Maks poeng: 3

2.7 Kodeforståelse 7

Me har dei følgjande setta (#1).

```
# 1
FB = {'paul', 'obama', 'erna', 'trump'}  # Facebook
LI = {'paul', 'støre', 'vedum'}          # LinkedIn
IG = {'paul', 'biden', 'erna'}           # Instagram
TX = {'trump', 'obama', 'biden'}         # Twitter/X
```

Loyale følgjarar av Mark Zuckerberg er på alle Facebook/Meta plattformene sine (Facebook og Instagram, i dette tilfellet) og ingen andre (LinkedIn eller Twitter/X, i dette tilfellet). Kva linje med kode skriv ut dei loyale følgjarane?

1. `print(FB.intersection(IG).difference(LI).difference(TX))`
2. `print(FB.intersection(IG).symmetric_difference(LI).symmetric_difference(TX))`
3. `print(FB.union(IG).intersection(LI).intersection(TX))`
4. `print(FB.symmetric_difference(IG).union(LI).union(TX))`

Vel eitt alternativ

- ○ 2
- ○ 4
- ○ 3
- ○ 1 ✔

Maks poeng: 3

math.e
The mathematical constant e = 2.718281…, to available precision.

3.1 Fyll inn 1

```
# 1
tekst = 'Eksamen 2023, TD41'
tekst = tekst.lower()
tekst = tekst.replace(',', '-')
tekst = tekst.replace(' ', '.')
ny_tekst = ''
for tegn in tekst:
    if tegn in '0123456789':
        ny_tekst += str(int(tegn) + 1)
    else:
        ny_tekst += tegn
print(ny_tekst)
```

Her skal du skriva inn svaret. **BERRE AKKURAT NØYAKTIG SVARET!** Ingenting meir! Ikkje "Svaret er ..." Nei! Inspera vil ikkje ha noko anna enn berre svaret. Hugs å bruka . i staden for , i tal (pi = 3.14, ikkje 3,14) og sånt.

Følgjande (#1) kode blir køyrd. Kva blir skrive ut?

Legg merke til at det står 'TD41' og ikkje 'TDT4109'.

Svar: (eksamen.3134-.td52, eksamen.3134-.td52, eksamen.3134-.td52, `eksamen.3134-.td52`, ´eksamen.3134-.td52´, «eksamen.3134-.td52»)

Maks poeng: 3

3.2 Fyll inn 2

```
# 1
# indeks 0  1  2  3  4  5  6  7  8
liste = [1, 2, 3, 4, 5, 6, 7, 8, 9]
liste[3] -= 1
liste[7] -= liste[3]*2
tall = liste.pop(1)
liste.append(2)
print(liste.count(2) + liste.count(3) + tall)
```

Følgjande (#1) kode blir køyrd. Kva blir skrive ut?

Svar: (6, 6, 6, `6`, ´6´, «6»)

Maks poeng: 3

3.3 Fyll inn 3

```
# 1
def f(z, c):
    return z**2 + c

c = 2
z = 0

while z < 10:
    z = f(z, c)

print(z)
```

Følgjande (#1) kode blir køyrd. Kva blir skrive ut?

Svar: (38, 38, 38, `38`, ´38´, «38»)

Maks poeng: 3

3.4 Fyll inn 4

```
# 1
tekst = 'hvorfor'
bokstaver = list(tekst)
sett = set(bokstaver)
liste = list(sett)
liste.sort()
tekst = ''.join(liste)
print(tekst)
```

```
# 2
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z  Æ  Ø  Å
```

Følgjande (#1) kode blir køyrd. Kva blir skrive ut?

(fhorv, fhorv, fhorv, `fhorv`, ´fhorv´, «fhorv»)

Her (#2) er alfabetet (i alfabetisk rekkjefølgje) med indeks.

Maks poeng: 2

math.e
The mathematical constant e = 2.718281…, to available precision.

4.1 Drag and drop 1

```
# 1
norsk = 'hei på deg'
engelsk = 'hello there'
ordbok = {
    'hei': 'hey',
    'på': 'on',
    'deg': 'you',
    'hallo': 'hello',
    'der': 'there'
    # ...
}

# Eksempel 1
>>> oversetter(ordbok, norsk, True)
'hey on you'

# Eksempel 2
>>> oversetter(ordbok, engelsk, False)
'hallo der'
```

Me er gitt følgjande string og ordbok, og skal laga funksjonen `oversetter`.

Den skal gjera ei dum omsetjing, ved å byta ut ord for ord. 'hei' skal bytast ut med 'hey' viss me omset norsk-til-engelsk (nor2eng). Elles skal 'hey' bytast ut med 'hei'.

Bygg opp funksjonen `oversetter`.

Hjelp

eng, nor	nor == eng	string.replace(nor, eng)
ordbok.items()	nor2eng is "True"	ordbok
nor, eng	nor != eng	nor2eng == False

```
def oversetter(ordbok, string, nor2eng):
    for [ nor, eng ✔ ] in [ ordbok.items() ✔ ] :
        if [ nor2eng == Fals ✔ ] :
            nor, eng = eng, nor
        string = [ string.replace( ✔ , eng) ]
    return string
```

Maks poeng: 2.4

4.2 Drag and drop 2

```
# Eksempel 1
>>> to_freedom(time(15, 35))
(3, 35, False)

Eksempel 2
>>> to_freedom(time(3, 35))
(3, 35, True)

Eksempel 3
>>> to_freedom(time(23, 00))
(11, 00, False)
```

Me er amerikanarar, så me vil ikkje bruka kommunisttid (24h). Funksjonen under konverterar til fridomstid (12h). Altså kan me forventa at tida 15:35 blir konverterte til 3:35. I tillegg skal ho returnera om tida er am (før 12:00 dagtid). Sjå døme over.

Hjelp

h % 12 | True | h > 12 | False | am | t | h -= 12

```
def to_freedom_time(t):
    '''(15, 45) -> (3, 45, False)'''
    h, m = t
    am = True
    if h > 12 :
        h -= 12
        am = False
    return (h, m, am )

to_freedom_time((15, 45))
# Expected result: (3, 45, True)
```

Maks poeng: 3

4.3 Drag and drop 3

Me skal laga ein funksjon som returnerer `True` eller `False` basert på om det er skotår eller ikkje. Reglane er som følgjer:

- Det er skotår viss årstalet (`år`) er deleleg med 4 ...
- ... med mindre det også er deleleg med 100 ...
- ... med mindre det også er deleleg med 400
- Elles er det ikkje skotår.

Til dømes:

- 2004 er deleleg med 4: Skotår
- 2100 er deleleg med 4, men også med 100: Ikkje skotår
- 2400 er deleleg med 4, og 100, men også med 400: Skotår

Bygg opp funksjonen under.

Hjelp

år % 4	False	år % 1	år % 400	True	år % 100

```
def er_skuddår(år):
    if [år % 4] :
        return True
    elif [år % 400] :
        return [False]
    elif [år % 100] :
        return [True]
    else:
        return False
```

Maks poeng: 3

4.4 Drag and drop 4

```
# Eksempel 1
>>> finn_gjennomsnittslengden([['Paul', 'Erna'], ['Obama'], ['Støre', '1234567']])
5
```

Bygg opp funksjonen `finn_gjennomsnittslengden`. Dømet under gir resulatet 5.

Hjelp

1	len(navn)
finn_gjennomsnittslengden(summen, antall)	tall
navn	rad
tabell	summen / antall
0	len(tabell)
sum(tabell)	

```
def finn_gjennomsnittslengden(tabell):
    summen = [0] ✔
    antall = 0
    for [rad] ✔ in [tabell]
        for [navn] ✔ in rad:
            summen += [len(navn)] ✔
            antall += [1] ✔
    return [summen / antall] ✔
```

Maks poeng: 2.8

4.5 Drag and drop 5

```
# 1
Side    1  2  3  4  5  5
Antall  8 13 11  9 10  9

# 2
Side    1 2 3 4  5  5
Antall  3 2 5 8 18 24
```

Me vil gjera ein statistisk analyse på terningkast.

Viss me kastar 60 terningar forventar me å få ca. 10 av kvart tal. 10/60 er 16.67%. Jo fleire gonger me kastar terningar, jo nærare vil me komma dette talet. #1 viser eit rettferdig døme, medan #2 viser eitt som kan vera suspekt. Nokre tal kjem for sjeldan, andre for ofte.

Koden under skal kasta 10 000 tilfeldige kast, og så finna ut om 'terningen' er tilfeldig. Me bestemmer at den er det viss kvart av tala blir kasta mellom 10% og 20% av alle kast.

Koden skriv ut prosentdelen for kvar terning. I tillegg skal han skriva ut ei feilmelding for kvart av berekna prosent som IKKJE er mellom 10 og 20%.

Hjelp

range(1, antall_kast)	kast	minn < resultat < maks
1	randint(1, 6)	len(6)
[0 * 6]	antall_kast	resultater
[0] * 6	kast - 1	math
range(len(antall_kast))	range(antall_kast)	[].count(6)
{minn} < {resultat} < {maks}	random	

```
from [ random ✔ ] import randint

antall_kast = 10_000
maks_prosent = 20
min_prosent = 10

maks = antall_kast * maks_prosent / 100
minn = antall_kast * min_prosent / 100

resultater = [ [0] * 6 ✔ ]

for i in [ range(antall_kast) ✔ ] :
    kast = [ randint(1, 6) ✔ ]
    resultater[ [ kast - 1 ✔ ] ] += 1

print(resultater)
for resultat in [ resultater ✔ ] :
```

```
    if not ( minn < resultat < s ):
```

Maks poeng: 2.8

4.6 Drag and drop 6

```
# 1
AA11111

# 2
AAA111
```

Funksjonen `check_regnummer` skal ta i mot eit regnummer (string) og returnera True eller False basert på om regnummeret følgjer norske standardar. To bokstavar etterfølgd av 5 tal. Altså bokstav-bokstav-tall-tall-tall-tall-tall. T.d. skal skilt #1 over blir godkjent (returnera True), medan skilt #2 ikkje blir godkjent (returnera False).

Hjelp

7	isalpha	False
len(check_regnummer)	True	regnummer[2:]
isdigit	len(regnummer)	len(regnummer[:2])
regnummer[:2]		

```
def check_regnummer(regnummer):
    if len(regnummer) != 7:
        return False
    if not regnummer[:2] . isalpha ():
        return False
    if not regnummer[2:]. isdigit ():
        return False
    return True
```

Maks poeng: 3

5

Her kan du kladde

Husk at fullt cheat-sheet og dok. for Python, Numpy og Matplotlib finnes som ressurser.

Oppgavens datastrukturer (beklager at den er et bilde, men Inspera nektet å lagre teksten...):

```
things = {
    'toalettsaker' : ['tannbørste', 'tannkrem'],
    'ytterklær': ['jakke',"bukse"],
    'isolasjon' : ['stillongs', 'ulltrøye'],
    'mat': ['brød', 'melk', 'ost'],
    'verktøy': ['hammer', 'skrutrekker', 'tannbørste']
}

packed = [['Børge', 'tannbørste', 'ost', 'bukse'],
          ['Ole', 'tannkrem', 'brød', 'jakke'],
          ['Anna', 'stillongs', 'melk', 'hammer'],
          ['Emma', 'skrutrekker', 'tang', 'jakke']]
```

Useful Functions and Methods

These are listed in the following order:

- built-in (standard library)
  - often used functions and operators
  - exceptions
  - string methods
  - list operations
  - set operations
  - dictionary operations
  - files
  - pickle library (binary files)
- random library
- math library
- numpy library
- matplotlib.pyplot

Built-in:

f-string

Syntax: f'…..{expression}…' where expression can be variable or any expression. Can also use formatting characters such as : d=integer, f=floating-point, e=scientific notation, %=percentage, s=string. A number before the formatting character will specify field width. A number after the character “.” will format the number of decimals. E.g. print PI with a field width of 5 with two decimals will be: print(f'{math.pi:5.2f}')

%

Remainder (modulo operator): Divides one number by another and gives the remainder.

//

Floor/integer division: Returns the integral part of the quotient.

len(s)

Return the length (the number of items) of a string, tuple, list, dictionary or other data structure.

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

Return a string containing a nicely printable representation of an object.

range(stop)

Returns a list with integers from 0, up to but not including stop. range(3) = [0, 1, 2]. Often used in combination with for loops: for i in range(10)

range([start], stop[, step])

start: Starting number of the sequence.

stop: Generate numbers up to, but not including, this number.

step: Difference between each number in the sequence.

chr(i)

Return a string of one character whose ASCII code is the integer i. For example, chr(97) returns the string 'a'. This is the inverse of ord()

ord()

Given a string of length one, return an integer representing the Unicode code point of the character when the argument is a unicode object, or the value of the byte when the argument is an 8-bit string. For example, ord('a') returns the integer 97.

tuple(iterable)

Accepts something iterable (list, range etc.) and returns the corresponding tuple. A tuple is immutable.

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

Enters a loop with x being one and one item from iterable. idx is an integer containing the loop number.

Exceptions:

try:

Code to test

except:

If code fails. E.g exception types: IOError, ValueError, ZeroDivisionError.

  # Variant: except Exception as exc # Let's you print the exception.

else:

  # Runs if no exception occurs

finally:

  # Runs regardless of prior code having succeeded or failed.

String methods:

s.isalnum()

Returns true if the string contains only alphabetic letters or digits and is at least one character of length. Returns false otherwise.

s.isalpha()

Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.

s.isdigit()

Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.

s.center(width)

Return the string center justified in a string of length width.

s.ljust(width)

Return the string left justified in a string of length width.

s.rjust(width)

Return the string right justified in a string of length width.

s.lower()

Returns a copy of the string with all alphabetic letters converted to lowercase.

s.upper()

Returns a copy of the string with all alphabetic letters converted to uppercase.

s.strip()

Returns a copy of the string with all leading and trailing white space characters removed.

s.strip(char)

Returns a copy of the string with all instances of *char* that appear at the beginning and the end of the string removed.

s.split(str)

Returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified).

s.splitlines([keepends])

Return a list of the lines in the string, breaking at line boundaries. This method uses the universal newlines approach to splitting lines. Line breaks are not included in the resulting list unless keepends is given and true.

Python recognizes "\r", "\n", and "\r\n" as line boundaries for 8-bit strings.

s.endswith(substring)

The substring argument is a string. The method returns true if the string ends with substring.

s.startswith(substring)

The substring argument is a string. The method returns true if the string starts with substring.

s.find(substring)

The substring argument is a string. The method returns the lowest index in the string where substring is found. If substring is not found the method returns -1.

s.replace(old, new)

The old and new arguments are both strings. The method returns a copy of the string with all instances of old replaced by new.

str.format(*args, **kwargs)

Perform a string formatting operation. The string on which this method is called can contain literal text or replacement fields delimited by braces {}. Each replacement field contains either the numeric index of a positional argument, or the name of a keyword argument. Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument.

List operations:

s[i:j:k]

Return slice starting at position i extending to position j in k steps. Can also be used for strings.

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

Returns the item that has the lowest value in the sequence.

max(list)

Returns the item that has the highest value in the sequence.

s.append(x)

Append new element x to end of s. Works in_place. Returns None

s.insert(index,item)

Insert an item into a list at a specified position given by an index.

s.index(*item*)

Return the index of the first element in the list containing the specified item.

s.pop()

Return last element and remove it from the list.

s.pop(i)
Return element i and remove it from the list.
s.remove(item)
Removes the first element containing the item. Works in_place. Returns None
s.reverse()
Reverses the order of the items in a list. Works in_place. Returns None
s.sort()
Rearranges the elements of a list so they appear in ascending order. Works in_place. Returns None

Sets operations:
len(s)
Number of elements in set *s*
s.issubset(t)
Test whether every element in *s* is in *t*
s.issuperset(t)
Test whether every element in *t* is in *s*
s.union(t)
New set with elements from both *s* and *t*
s.intersection(t)
New set with elements common to *s* and *t*
s.difference(t)
New set with elements in *s* but not in *t*
s.symmetric_difference(t)
New set with elements in either *s* or *t* but not both
s.copy()
New set with a shallow copy of *s*
s.update(t)
Return set *s* with elements added from *t*
s.add(x)
Add element *x* to set *s*. Works in_place. Returns None
s.remove(x)
Remove x from set s; raises *KeyError* if not present. Works in_place. Returns None
s.clear()
Remove all elements from set *s*. Works in_place. Returns None

Dictionary operations:
d.clear()
Clears the contents of a dictionary
d.get(key, default)
Gets the value associated with a specific key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.
d.items()
Returns all the keys in a dictionary and their associated values as a sequence of tuples.
d.keys()
Returns all the keys in a dictionary as a sequence of tuples.
d.pop(key, default)
Returns the value associated with a specific key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.
d.popitem()
Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.
d.values()
Returns all the values in dictionary as a sequence of tuples.
del d[k]

Deletes element k in d.

d.copy()

Makes a copy of d.

Files:

open()

Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be 'r' (read only), 'w' (writing only), 'a' (appending), 'r+' (both reading and writing).

f.read(size)

Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.

f.readline()

Reads a single line from the file (reads until newline character (\n) is found), and returns it as a string.

f.readlines()

Reads data from the file and returns it as a list of strings.

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

Move the file pointer in the file and return the new position of the file pointer. The parameter

from_what can have three values: 0 – beginning of file, 1 – current position in file, and 2 – end of file. The offset parameter defines how much you will move from the from_what position.

f.tell()

Return the position of the file pointer.

f.close()

Close the file and free up any system resources taken up by the open file. If using **with open(filename) as ...** when you open the file, close() is not necessary, since it is implied by the ending of the with-block

Pickle Library:

pickle.dump(obj,file)

Write a pickled (serialized) representation of an obj to the open file object file.

pickle.load(file_object)

Read a string from the open file object file and interpret it as a pickle data stream, reconstructing and returning the original object hierarchy.

Random Library:

random.random()

Return the next random floating-point number in the range [0.0, 1.0).

random.randint(a,b)

Return a random integer *N* such that a <= N <= b.

random.choice(seq)

Return a random element from the non-empty sequence seq. If seq is empty, raises IndexError.

random.randrange(start, stop [, step])

Return a randomly selected element from range(start, stop, step).

random.uniform(a, b)

Return a random floating-point number N such that a <= N <= b for a <= b and b <= N <= a for b < a.

math Library:

math.ceil(x)

Return the ceiling of x, the smallest integer greater than or equal to x.

math.floor(x)

Return the floor of x, the largest integer less than or equal to x.

math.exp(x)

Return e raised to the power x, where e = 2.718281… is the base of natural logarithms.

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

The mathematical constant π = 3.141592…, to available precision.

math.e

The mathematical constant e = 2.718281…, to available precision.

Del 3 - Turforberedelser

Oppgåvene dreier seg om førebuingar til ein tur. Det er jo mange ting som må pakkast til ein tur, og då er det greitt å ha ei oversikt over kva som er pakka. I alle fall i prinsippet!
Denne delen består av fleire oppgåver som kan fungera saman.

- Hugs at du gjerne kan bruka funksjonar frå tidlegare oppgåver sjølv om du skulle ha hoppa over dei!
- Hugs også at du har 'fuskelappen' tilgjengeleg.

Alle 'ting' som skal pakkast er knytt til éin eller fleire kategoriar:

things *= {*
'toalettsaker' : ['tannbørste', 'tannkrem'],
'ytterklær': ['jakke',"bukse"],
'isolasjon' : ['stillongs', 'ulltrøye'],
'mat': ['brød', 'melk', 'ost'],
'verktøy': ['hammer', 'skrutrekker', 'tannbørste']
}

Personane som skal vera med på turen pakkar kvar tinga sine, som skal lagrast i følgjande struktur. Kvar person er ein subliste, med namnet først og så dei ulike tinga denne personen har pakka:

packed *= [['Børge', 'tannbørste', 'ost', 'bukse'],*
['Anna', 'tannkrem', 'brød', 'jakke'],
['Ole', 'stillongs', 'melk', 'hammer'],
['Emma', 'skrutrekker', 'tang', 'jakke']]

Døma i dei følgjande oppgåvene bruker nettopp desse verdiane som utgangspunkt.

5.1 Oppgave 1 - Hvem er med på tur! (participants) - 3%

Det første me treng er å vita kven som er med på turen. Skriv funksjonen *participants*.

- Funksjonen skal ta inn lista *packed*.
- Funksjonen skal returnera ei liste med namna til deltakarane til turen.

Døme:
>>> print(participants(packed))
['Børge', 'Ole', 'Anna', 'Emma']

Skriv svaret ditt her

```
1
```

Maks poeng: 3

5.2 Oppgave 2 - hva har én person pakket (has_brought) - 3%

Me må også vita kva éin av deltakarane har pakka. Skriv funksjonen blir hatt_brought.

- Funksjonen skal ta inn ein *streng* med eit namn, og dessutan lista *packed*.
- Funksjonen skal returnera ei *liste* med tinga som denne personen har pakka.

Døme:
>>> print(has_brought("Børge", packed))
['tannbørste', 'ost', 'bukse']

Skriv svaret ditt her

```
1
```

Maks poeng: 3

5.3 Oppgave 3 - Unike ting! (unique_items) - 3%

No må me vita om alle dei ulike tinga som skal bli med. Skriv funksjonen unique_items.

- Funksjonen skal ta inn dictionarien *things*.
- Funksjonen skal returnera ei *liste* over alle dei unike tinga ein kan ha med.
- Ein skal berre ha tinga, ikkje typen kvar ting er knytt til.
- Éin ting kan vera del av fleire ulike typar, eksempelvis kan ein tannkost vera både toalettsaker og verktøy. Alle ting skal berre vera med éin gong.
- Rekkefølgen på tinga som blir returnerte er ikkje viktig.

Døme:

>>> print(unique_items(things))
[brød, tannbørste, skrutrekker, ulltrøye, ost, hammer, bukse, tannkrem, stillongs, melk, jakke]

Skriv svaret ditt her

```
1
```

Maks poeng: 3

5.4 Oppgave 4 - Tillatte ting (item_allowed) - 3%

Det er berre lov til å pakka ting som er på lista, sjølvsagt! Derfor må me ha ein sjekk på om ein ting er tillate eller ikkje. Skriv funksjonen *item_allowed*.

- Funksjonen skal ta inn ein *streng* (tingen) og dictionarien *things*.
- Funksjonen skal returnera ein *boolean*, *True* dersom tingen er tillaten og *False* viss han ikkje er det.

Døme:
>>> print(item_allowed("bok", things))
False
>>> print(item_allowed("ost", things))
True

Skriv svaret ditt her

```
1
```

Maks poeng: 4

5.5 Oppgave 5 - pakking! (pack_item) - 5%

No skal ein person få pakka ein ting. Skriv funksjonen *pack_item*.

- Funksjonen har tre parametrar:
  - *Streng* (namnet på personen)
  - *Streng* (tingen som skal leggjast inn)
  - *Liste*: (packed)
- Funksjonen skal leggja til tingen i sublisten til den rette personen.
- Alle andre personars ting skal vera uendra.
- Funksjonen skal returnera ei *liste* med oppdaterte verdiar.
- Dersom tingen ikkje er lov til å ta med så skal tingen ikkje leggjast til.
- NB: Funksjonen skal *ikkje* endra på lista ho hentar inn.
- Det er lov til å ta med fleire ting av same type, ein kan jo pakka to bukser.
- *things* er ein global variabel.

Døme:

>>> packed2 = pack_item("Børge", "bukse", packed)

>>> print("Børge har pakket:",has_brought("Børge", packed2))

Børge har pakket: ['tannbørste', 'ost', 'bukse', 'bukse']

Skriv svaret ditt her

```
1
```

Maks poeng: 5

5.6 Oppgave 6 - Hva mangler (what_is_missing) - 6%

For å kunna dra på tur må det vera pakka minst éin av kvar ting. Skriv funksjonen *what_is_missing*.

- Funksjonen skal ta inn dictionarien *things* og lista *packed*.
- Funksjonen skal returnera ei *liste* over tinga som manglar, altså alle ting som ikkje er pakka av nokon.

Døme:

>>> print("Manglar:",what_is_missing(things, packed))

Manglar: ['ulltrøye']

Skriv svaret ditt her

```
1
```

Maks poeng: 6

5.7 Oppgave 7 - innlesing fra fil (add_all) - 7%

Til slutt skal det lesast inn ei fil med innhald som skal leggjast til. Skriv funksjonen *add_all*.

- Funksjonen skal lesa innhaldet i fila *pakkeliste.txt*.
- Funksjonen har éin parameter: *liste* (already_packed) med struktur som *packed*.
- Fila har to linjer, det er den *andre linja* som har informasjonen.
  - Linja er strukturet slik: namn:ting,ting:namn:ting:namn:ting,ting,ting...
  - Det kan jo vere ulike ting og namn med, men alle namna finst.
  - Eit døme på teksten på linje 2: ***Børge:jakke,ost,Ole:ost,Anna:ost,paprika***
- Ting som ikkje er tillate å ta med skal ikkje leggjast til.
- Funksjonen skal returnera ei oppdatert liste over alt som no er pakka, inkludert det som var pakka frå før.
- Du kan forutsette at det ikkje er nokre namn som også er ting eller motsatt.

Døme:
>>> updated = add_all(packed)
>>> print("Børge pakkar etter oppdatering:", has_brought("Børge", updated))
Børge pakker etter oppdatering: ['tannbørste', 'ost', 'bukse', 'jakke', 'ost'] # jakke og ost er lagt til!

Skriv svaret ditt her

```
1
```

Maks poeng: 7