Forside

Institutt for datateknologi og informatikk

Eksamensoppgave i TDT4109 - Informasjonsteknologi, grunnkurs

Eksamensdato: 1. desember 2021
Eksamenstid (fra-til): 09:00 – 13:00
Hjelpemiddelkode/Tillatte hjelpemidler: A / Alle hjelpemidler tillatt
Faglige kontakter under eksamen:

- Børge Haugset (tlf.: 934 20 190, kan også kontaktes via Teams)
- Dag Olav Kjellemo (tlf: 476 81 639, kan også kontaktes via Teams)
- Magnus Tvilde (tlf: 480 92 241, kan også kontaktes via Teams)

Teknisk hjelp under eksamen: NTNU Orakel **Tlf:** 73 59 16 00

Får du tekniske problemer underveis i eksamen, må du ta kontakt for teknisk hjelp snarest mulig, og senest innen eksamenstida løper ut/prøven stenger. Kommer du ikke gjennom umiddelbart, hold linja til du får svar.

ANNEN INFORMASJON:
Ikke ha Inspera åpen i flere faner, eller vær pålogget på flere enheter, samtidig, da dette kan medføre feil med lagring/levering av besvarelsen din.

Skaff deg overblikk over oppgavesettet før du begynner på besvarelsen din.

Les oppgavene nøye, gjør dine egne antagelser og presiser i besvarelsen hvilke forutsetninger du har lagt til grunn i tolkning/avgrensing av oppgaven. Faglig kontaktperson kan kontaktes dersom du mener det er feil eller mangler i oppgavesettet.

Det er angitt i prosent hvor mye hver deloppgave i eksamenssettet teller ved sensur. Hele oppgavesettet gir totalt **XX** poeng som tilsvarer 100%. Les gjennom hele oppgavesettet før du begynner å løse oppgaven. Disponer tiden godt!

Gjør dine egne antagelser og presiser i besvarelsen hvilke forutsetninger du har lagt til grunn i tolkning/avgrensing av oppgaven. Faglig kontaktperson skal kun kontaktes dersom det er direkte feil eller mangler i oppgavesettet.

Annen viktig informasjon

Lagring: Besvarelsen din i Inspera Assessment lagres automatisk. Jobber du i andre programmer – husk å lagre underveis.

Juks/plagiat: Eksamen skal være et individuelt, selvstendig arbeid. Det er tillatt å bruke hjelpemidler, men vær obs på at du må følge eventuelle anvisningen om kildehenvisninger under.

Under eksamen er det ikke tillatt å kommunisere med andre personer om oppgaven eller å distribuere utkast til svar. Slik kommunikasjon er å anse som juks.

Alle besvarelser blir kontrollert for plagiat. *Du kan lese mer om juks og plagiering på eksamen her*.

Kildehenvisninger: Det skal henvises til kilder som brukes *utover* lærebok og slides brukt i faget. Kilder oppgis ved referanse til en bok, eller nettsiden der du fant informasjonen., som "https://no.wikipedia.org/wiki/Binært_tallsystem". Hvis du bruker kode du finner på en nettside, lim inn addressen til denne nettsiden.

Varslinger: Hvis det oppstår behov for å gi beskjeder til kandidatene underveis i eksamen (f.eks. ved feil i oppgavesettet), vil dette bli gjort via varslinger i Inspera. Et varsel vil dukke opp som en dialogboks på skjermen i Inspera. Du kan finne igjen varselet ved å klikke på bjella øverst i høyre hjørne på skjermen. Det vil i tillegg bli sendt SMS til alle kandidater for å sikre at ingen går glipp av viktig informasjon. Ha mobiltelefonen din tilgjengelig.

OM LEVERING:

Automatisk innlevering: Besvarelsen din leveres automatisk når eksamenstida er ute og prøven stenger, forutsatt at minst én oppgave er besvart. Dette skjer selv om du ikke har klikket «Lever og gå tilbake til Dashboard» på siste side i oppgavesettet. Du kan gjenåpne og redigere besvarelsen din så lenge prøven er åpen. Dersom ingen oppgaver er besvart ved prøveslutt, blir ikke besvarelsen din levert. Dette vil anses som “ikke møtt” til eksamen.

Trekk/avbrutt eksamen: Blir du syk under eksamen, eller av andre grunner ønsker å levere blankt/avbryte eksamen, gå til “hamburgermenyen” i øvre høyre hjørne og velg «Lever blankt». Dette kan ikke angres selv om prøven fremdeles er åpen.

Tilgang til besvarelse: Du finner besvarelsen din i Arkiv etter at sluttida for eksamen er passert.

Eksamensstruktur

Eksamen er delt i tre:

- Del 1: teoridel som teller **25%** av den totale vurderingen.
- Del 2: ulike små programmeringsoppgaver, 2a - 2i. Disse er helt uavhengige av hverandre, og teller til sammen **40**%.
- Del 3: denne består av et sett med oppgaver som hører sammen. Denne teller **35**%.
- Programmeringsoppgavene i del 2, og de første to oppgavene i del 3, vil kunne testes i Inspera. Det er viktig at du ikke fjerner kildekoden som står der allerede, og skriver din kode der det tår markert, ellers vil du ikke kunne teste koden. Hvis dette fjernes vil sensor måtte manuelt gå igjennom koden din, så skriv løsningen din uansett.

Hvordan skal du best programmere på ITGK-eksamen?

1. Du kan godt skrive kode rett inn i oppgavene, hvis du ønsker. Du vil derimot ikke alltid kunne få kjørt koden din. Vi anbefaler ikke dette - når du tross alt har mulighet bør man bruke en ordentlig editor.
2. Du kan se på oppgaven i Inspera, og så løse oppgavene i en ekstern editor. Du vil da kunne kjøre koden din, for å teste om den er korrekt etc. Du må da skrive testene dine selv.
3. Hvis du jobber utenfor Inspera, så kan det være lurt å kopiere inn løsningene etter hvert. Skulle noe skje med datamaskinen din, da har du levert så mye som mulig.
4. Når du kommer til oppgave 3 kan du hente ned én fil: ***h21_oppgave3.py***. Du finner den i denne zipfilen h21_oppgave3. Pakk ut filen, og åpne **h21_oppgave3.py** i en valgfri editor som Thonny. Du slipper å skrive en god del selv, og kan konsentrere deg om å løse oppgavene. **h21_oppgave3.py** inneholder testkode til oppgave 3. **Du skal IKKE laste opp noen filer - alle funksjonene skal skrives inn i sine respektive deloppgaver i Inspera. Det er heller ikke utvidet eksamenstid for å legge kode inn i Inspera.**

- Husk at du har alle hjelpemidler til rådighet. Dette betyr alle som ikke involverer samarbeid med andre, uansett om de tar eksamen eller ikke. Hvis du finner kode som passer på en nettside, **skal du** legge nettsiden inn som en kommentar i koden. Alle som programmerer søker opp masse ting på nettet.
- Vi anbefaler sterkt at dere ikke forsøker dere på samarbeid. Vi kommer til å kjøre en solid sjekk av likheter i både teori og programmeringsoppgaver. Det hjelper ikke å endre variabelnavn og slikt - vi kan finne det. Vi kommer til å lete, men dette er en jobb vi håper vi gjør forgjeves. Derfor: hvis dere kopierer noe fra nettet, og noen andre også har gjort det: det er helt ok, men hvis vi har en referanse slipper vi å anklage dere unødig.

Lykke til!

1 Regler og samtykker

REGLER OG SAMTYKKER

Dette er en **individuell** eksamen. Du har ikke lov til å kommunisere (gjennom web-forum, chat, telefon, hverken i skriftlig, muntlig eller annen form), ei heller samarbeide med noen andre under eksamen.

Før du kan fortsette til selve eksamen må du forstå og SAMTYKKE i følgende:

NTNU-policy for juks ved eksamen:
(**Norsk**) https://innsida.ntnu.no/wiki/-/wiki/Norsk/Juks+på eksamen

Under Eksamen:

Jeg skal IKKE motta hjelp fra andre.

- Aksepter

Jeg skal IKKE hjelpe andre eller dele løsningen min med noen.

- Aksepter

Jeg skal IKKE copy-paste noe kode fra noen eksisterende online/offline kilder. Du kan se, og deretter skrive din EGEN versjon av koden, eller legge kilden til i en kommentar.

- Aksepter

Jeg er klar over at jeg kan stryke uavhengig av hvor korrekt svarene mine er, hvis jeg ikke følger reglene og/eller IKKE aksepterer disse utsagnene.

- Aksepter

Jeg er også klar over at juks kan ha alvorlige konsekvenser, som å bli utestengt fra universiteter og få annullert eksamensresultater.

- Aksepter

I tillegg: Jeg gir samtykke til at mine eksamensdetaljer (anonymisert) kan brukes til forskningsformål av ansatte ved NTNU for å forbedre undervisningen i faget.

- ○ Ikke bruk resultatene mine til forskning
- ○ Aksepter

Maks poeng: 0

Oppgave 1 (25%)

Denne oppgaven teller 25% av karakteren.
Svar på disse teorioppgavene på en kort, komplett og konsis måte.

2 1a - komprimering (5%)

Denne oppgaven har to deler:

1. Forklar forskjellen på *Lossy* og *Lossless* komprimering.
2. Gi et eksempel på hvordan "run-length encoding" kan brukes til å komprimere noe.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

3 1b - minne (5%)

Forklar kort de ulike minnene vi benytter oss av i en datamaskin, og når vi benytter oss av forskjellige minnetypene.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

4 TCP/UDP (5%)

Forklar forskjellen på TCP og UDP, og når vi bør benytte oss av den ene eller andre.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

5 1d - sikkerhet (5%)

Forklar de ulike begrepene Konfidensialitet, Integritet og Tilgjengelighet i et sikkerhets-perspektiv, og hva en ondsinnet angriper kan gjøre for å bryte med disse prinsippene.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

6 1e - IP (5%)

Forklar hvorfor overgangen fra IPV4 til IPV6 er så vanskelig, og hvorfor vi må gå over til ny versjon:

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

Funksjonsliste

Useful Functions and Methods

Du kan også finne en slik oversikt på denne nettsiden: functions.pdf.

Built-in:

format(numeric_value, format_specifier)

Formats a numeric value into a string according to the format specifier, which is a string that contains special characters specifying how the numeric value should be formatted. Examples of various formatting characters are “f=floating-point, e=scientific notation, %=percentage, d=integer”. A number before the formatting character will specify the field width. A number after the character “.” will format the number of decimals.

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.

Random methods:

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.

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.

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.

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.

numpy Library:
numpy.array(list)
Generates an array. If the list or tuple is a 2D-list it generates a matrix. With a 1D-list it generates a vector
ndarray.ndim
the number of axes (dimensions) of the array.
ndarray.shape
the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, shape will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.
ndarray.size
the total number of elements of the array. This is equal to the product of the elements of shape.
ndarray.dtype
an object describing the type of the elements in the array. One can create or specify dtype's using standard Python types. Additionally, NumPy provides types of its own. numpy.int32, numpy.int16, and numpy.float64 are some examples.
numpy.zeros(tuple)
creates a matrix with 0 using the tuple to define the number of lines and rows in the matrix
numpy.ones(tuple)
creates a matrix with 1 using the tuple to define the number of lines and rows in the matrix
numpy.arange(start,stop,step)
creates a vector starting at *start*, ending at *stop* using *step* between the values
numpy.ndarray.reshape(lines,rows)
reshapes a ndarray according to the values in *lines* and *rows*

Oppgave 2 (40%)

Oppgave 2 består av 9 ulike oppgaver (2a - 2i) med programmering og dra-og-slipp / innfylling i programkode. Til sammen utgjør disse oppgavene **40**% av karakteren.
På oppgaver hvor du kan teste om koden din virker, vil du normalt score mer poeng ved å ha et program som passerer iallfall noen av testene, enn ved å ha et program som prøver å få til alt, men får kjørefeil og ikke passerer noen av testene. Det er likevel bedre å levere noe kode selv om den ikke virker, enn å levere blankt, da sensor med et levert svar kan ha mulighet til å se over koden manuelt og vurdere om det kan gis litt poeng for delvis riktig tankegang.

7 2a - Max Manus (3%)

Pythons standardfunksjon **max()** returnerer den største av verdiene i en sekvens. For eksempel vil **max("manus")** gi verdien **"u"** fordi u er lengst ut i alfabetet av bokstavene i manus. Tilsvarende vil **min("manus")** gi verdien **"a"**.

Skriv en funksjon **alfa(streng, tall)** som får inn en tekststreng pluss et tall. Hvis tallet er 0 skal funksjonen returnere det minste tegnet i strengen (dvs. fremst i alfabetet), hvis tallet er noe annet enn 0, skal den returnere det største tegnet i strengen.

Skriv koden din på anvist sted. IKKE endre forhåndsoppgitt kode.

Test case #	Input	Forventet output
1	manus 0	a
2	manus 1	u
3	manus 0.5	u

```
streng = input()
tall = float(input())
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
print(alfa(streng, tall))
```

Test kode

Kode som du kan lime inn hvis du er uheldig og sletter
den forhåndsdefinerte koden i svarvinduet:

```
streng = input()
tall = float(input())
# IKKE endre koden over her
```

```
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
print(alfa(streng, tall))
```

Maks poeng: 3

8 2b - Byer i Belgia (3%)

Gitt ei ordbok kalt **geografi** som har land som nøkkelverdi, og for hver nøkkel ei liste av byer i dette landet. Et eksempel på mulig innhold er:

```
geografi = {'Irland': ['Dublin', 'Cork'],
            'Polen': ['Lodz', 'Krakow', 'Gdansk'],
            'Belgia': ['Bryssel', 'Gent', 'Liege', 'Namur']
           }
```

Funksjonen **ant_land(geografi)** skal returnere antallet land som fins i ordboka. Med eksemplet over skal dette kallet returnere tallet 3 fordi det fins info om tre land i ordboka.

Hint: funksjonen **len()** gir antall element i f.eks. strenger, lister, tupler, mengder, ordbøker.

NB: Funksjonen din skal også virke for ordbøker med annet antall land enn eksemplet gitt her, men strukturen kan alltid antas å være den samme (land som nøkkelverdi, så liste med byer for hvert land).

Skriv koden din på anvist sted. IKKE endre forhåndsoppgitt kode.

Test case #	Input	Forventet output
1	{'Belgia': ['Bryssel', 'Gent', 'Liege', 'Namur'], 'Polen': ['Lodz']}	2
2	{'Irland': ['Dublin'], 'Belgia': ['Gent', 'Liege', 'Namur'], 'Japan': [] }	3

```
geografi = eval(input())
# IKKE endre koden som står over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden nedenfor her
print(ant_land(geografi))
```

Test kode

```
# Kode som du kan lime inn hvis du er uheldig og sletter
# den forhåndsdefinerte koden i svarvinduet:
geografi = eval(input())
# IKKE endre koden som står over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden nedenfor her
print(ant_land(geografi))
```

Maks poeng: 3

9 2c - Sjekk tall (4%)

Skriv en funksjon **sjekk(tall)** som får inn et tall. Hvis tallet er et oddetall, skal funksjonen returnere strengen "odde", hvis partall returnere strengen "par", og hvis tallet har en desimaldel > 0, skal funksjonen returnere strengen "des". (Dvs. f.eks., 3 -> "odde", 3.0 -> "odde", 3.1 -> "des", 4.0 -> "par", 4.2 -> "des") NB: Funksjonen **skal returnere** tekststrengen, **IKKE** printe den.

Skriv koden din på anvist sted. IKKE endre på forhåndsgitt kode.

Test case #	Input	Forventet output
1	3	odde
2	3.0	odde
3	3.1	des
4	4	par
5	4.2	des

```
tall = float(input())
# IKKE endre koden over
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under
print(sjekk(tall))
```

Test kode

Kode som du kan lime inn hvis du er uheldig og sletter
den forhåndsdefinerte koden i svarvinduet:

```
tall = float(input())
# IKKE endre koden over
# Skriv koden din mellom her...
# ...og her
```

```
# IKKE endre koden under
print(sjekk(tall))
```

Maks poeng: 4

10 2d - Sum av lengder (5%)

Funksjonen **sum_len(L)** får inn ei liste L som inneholder heltall, og skal returnere **summen av lengdene** til tallene i lista.

F.eks. sum_len ([34, 502, 4]) skal returnere 6 fordi det første tallet er 2 siffer langt, det andre 3 siffer langt, og det tredje 1 siffer, og 3+2+1 er 6. Trekk kodefragmenter til riktig sted så funksjonen virker som den skal. (NB: Det fins flere kodefragment enn hull i koden, dvs. en del av fragmentene skal ikke brukes)

Det blir ikke gitt minuspoeng for feilplasserte kodefragment, så du bør fylle inn hull du ikke er sikker på, heller enn å sette blankt.

Hjelp

(tall)	str	summen =	summen	for tall
in liste:	sum	for liste	summen = 0	(L)
len	sum (L)	print	int	summen +=
return				

```
def sum_len(liste):
    [ ]
    [ ]  [ ]
    [ ]  [ ]  (  [ ]  [ ]  )
    [ ]  [ ]
```

Maks poeng: 5

11 2e - Største tverrsum (5%)

Tverrsummen til et tall er summen av sifrene, f.eks. tverrsummen til 125 er 1+2+5, altså 8. Anta at det allerede fins en funksjon **tverrsum(tall)**, du trenger ikke lage denne. Vi skal nå lage funksjonen max_tverrsum(L), som får inn som parameter ei liste med heltall. Funksjonen skal returnere **den største tverrsummen** blant tallene i lista, samt indeksposisjonen denne ble funnet på. For eksempel, max_tverrsum ([8, 12, 77, 1111]) skal returnere tuppelet (14, 2) fordi den største tverrsummen er 14, og denne får vi for tallet 77 som står på indeks 2 i lista. Lista som kommer inn som parameter antas å alltid inneholde minst ett tall, dvs. er aldri tom.

Oppgave: Velg riktige alternativ nedenfor, slik at koden for max_tverrsum() vil funke. Det blir IKKE gitt minuspoeng for feil svar, så du BØR fylle inn alle hull heller enn å sette noe blankt.

```
def max_tverrsum(L):

    Velg alternativ (mpos = 0, tverrsum = 0, L = [ ])

    msum = Velg alternativ (L[0], 0, tverrsum(L[0]))

    for i in range(1, Velg alternativ (len(L), len(L) - 1, len(L - 1))):

    if Velg alternativ (tverrsum(msum), tverrsum(i), tverrsum(L[i])) Velg alternativ (>, ==, <) msum:

        Velg alternativ (mpos = i, mpos = L[i], mpos += L[i])

        msum Velg alternativ (+= tverrsum(L[i]), += L[i], = tverrsum(L[i]))

    Velg alternativ (return, L.append, print) (msum, mpos)
```

Maks poeng: 5

12 2f - Ordbok fra ordbok (5%)

Gitt ei ordbok **tallbok** hvor nøklene er strenger som er forkortelser for språk (norsk, svensk, ...), og hvor verdien på hver nøkkel er ei liste med ord for tallene 0-6 på dette språket.

tallbok = {'NO': ['null', 'en', 'to', 'tre', 'fire', 'fem', 'seks'],
'SE': ['noll', 'ett', 'två', 'tre', 'fyra', 'fem', 'sex'],
'DK': ['null', 'en', 'to', 'tre', 'fire', 'fem', 'seks'],
'IT': ['zero', 'uno', 'due', 'tre', 'cuattro', 'cinque', 'sei'],
'ES': ['cero', 'uno', 'dos', 'tres', 'cuatro', 'cinco', 'seis'],
'PT': ['zero' 'um', 'dois', 'tres', 'cuatro', 'cinco', 'seis']
}

Vi skal lage en funksjon **same_word(num, tallbok)** som får inn som parameter et heltall (0-6) og ei ordbok med struktur som vist over. Den skal returnere ei ny ordbok hvor nøklene er ord for dette tallet, og hvor verdifeltet er lista over alle språk som har dette ordet for tallet - *såframt mer enn ett språk har samme ord*. Eksempel på resultat ved kjøring:

- same_word(3, tallbok) returnerer {'tre': ['NO', 'SE', 'DK', 'IT'], 'tres': ['ES', 'PT']}
- same_word(6, tallbok) returnerer {'seks': ['NO', 'DK'], 'seis': ['ES', 'PT']} - merk at her er nøklene SE og IT ikke med i returnert ordbok fordi hver av disse var alene om sin skrivemåte for tallet 6.

Trekk kodefragmentene til riktig sted så funksjonen virker som den skal. (3 av fragmentene skal ikke brukes). Det blir ikke gitt minuspoeng for feilplasserte kodefragment, så du bør fylle inn alle hull heller enn å sette noe blankt.

Hjelp

for landkode in tallbok:	result[landkode].append(tallord)
result[tallord] = [landkode]	result += tallbok[num]
if tallord in tallbok:	tallord = tallbok[landkode][num]
return result	temp[tallord].append(landkode)
if tallord in temp:	result[tallord] = temp[tallord]

```
def same_word ( num, tallbok ) :
    temp, result = { } , { }
    [                                   ]
        [                               ]
        [                               ]
            [                           ]
            [                           ]
    else:
            [                           ]
    [                                   ]
```

Maks poeng: 5

13

2g - Fjerne serieduplikat (5%)

Vi ønsker en funksjon **fjern_dup(liste)** som får inn ei liste med tall som parameter, og som skal fjerne serieduplikater i lista, dvs. tilfeller hvor samme tall kommer flere ganger rett etter hverandre. Som man kan se av gitte test cases:

1. Her kommer ingen tall flere ganger på rad. Dermed endres lista ikke.
2. Tallet 4 kommer to ganger på rad, derfor fjernes en av disse firerne
3. Lista har flere på rad både av 2'ere, 3'ere, 1'ere, i alle disse tilfellene fjernes tall fra lista slik at det ikke kommer identiske tall på rad

NB: Funksjonen skal gjøre **muterende** endring i lista som kommer inn som parameter, den trenger derfor ikke returnere noe. Den skal heller ikke printe noe.

Skriv koden din på anvist sted. IKKE endre forhåndsoppgitt kode.

Test case #	Input	Forventet output
1	[1, 4, 3, 2]	[1, 4, 3, 2]
2	[1, 4, 4, 3]	[1, 4, 3]
3	[2, 2, 3, 3, 3, 1, 4, 1, 1, 1, 1]	[2, 3, 1, 4, 1]

```
tl = input()[1:-1].split(',')
liste = [int(n) for n in tl]
# IKKE endre koden over
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under
fjern_dup(liste)
print(liste)
```

Test kode

tl = input()[1:-1].split(',')
liste = [int(n) for n in tl]
# IKKE endre koden over

```
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under
fjern_dup(liste)
print(liste)
```

Maks poeng: 5

14 2h - Liste fra streng (5%)

Funksjonen **list_str(s)** skal få inn en streng s som parameter. Den skal returnere ei liste med bokstaver, nærmere bestemt de bokstavene i strengen s som har tegnet 'f' 1 tegn foran seg, **og** tegnet 'f' 6 tegn bak seg. Eksempel:

- **list_str**('abcdefX') skal returnere ei tom liste [ ]. Bokstaven X har riktig nok 'f' 1 tegn foran seg, men har ikke 'f' 6 tegn bak seg.
- **list_str**('fedcabYabcdef') skal returnere ei tom liste [ ]. Bokstaven Y har riktig nok 'f' 6 tegn bak seg, men bokstaven 'f' står ikke 1 tegn foran Y.
- **list_str**('xyzzy') skal returnere ei tom liste [ ]. Ingen av bokstavene i strengen har påkrevd tegn foran eller bak.
- **list_str**('abcdefSabcdefOabcdefSabcdef') skal returnere ['S', 'O', 'S'] fordi akkurat disse tre tegnene i strengen har 'f' 1 tegn foran seg og 'f' 6 tegn bak seg.

Skriv koden din på anvist sted. IKKE endre forhåndsdefinert kode!

Test case #	Input	Forventet output
1	abcdefX	[]
2	fedcabYabcdef	[]
3	xyzzy	[]
4	abcdefSabcdefOabcdefSabcdef	['S', 'O', 'S']
5	abcdef*abcdef	['*']
6	abcdefPabcdefYabcdef	['P', 'Y']

```
s = input()
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
print(list_str(s))
```

Test kode

```
# Kode som du kan lime inn hvis du er uheldig og sletter
# den forhåndsdefinerte koden i svarvinduet:
s = input()
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
print(list_str(s))
```

Maks poeng: 5

15

2i - Sjekk to og to (5%)

Funksjonen **sjekk_to(streng)** får inn en streng som alltid er 6 tegn lang, og hvor hvert tegn er et siffer, f.eks. '051033'. Følgende regler gjelder for disse sifrene: Tallet som dannes av de to første sifrene, kan maks være 27. Tallet som dannes av de to midterste sifrene, kan maks være 14. Tallet som dannes av de to bakerste sifrene, kan maks være 58. Funksjonen skal returnere:

- 1 hvis strengen er ok, jfr. test #1 - alle verdier er innenfor grensene.
- 2 hvis to første siffer gir for høyt tall, jfr. test #2 med 42 mens maks tillatt var 27.
- 3 hvis to midterste siffer gir for høyt tall, jfr. test #3 med 20 mens maks tillatt var 14.
- 5 hvis to bakerste siffer gir for høyt tall, jfr. test #4 med 99 mens maks tillatt var 58.
- 6 hvis både fremste to og midterste to er for høye, jfr. test #5.
- 10 hvis både fremste to og bakerste to er for høye, jfr. test #6.
- 30 hvis både fremste, midterste, bakerste er for høye, jfr. test #7.

Merk: Funksjonen må fungere generelt for strenger med 6 siffer, ikke bare for de strengene som er gitt som test case her.

Skriv koden din på anvist sted. IKKE endre forhåndsoppgitt kode!

Test case #	Input	Forventet output
1	051033	1
2	421033	2
3	052033	3
4	051099	5
5	422033	6
6	421099	10
7	422099	30

```
streng = input()
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
print(sjekk_to(streng))
```

Test kode

```
# Kode som du kan lime inn hvis du er uheldig og sletter
# den forhåndsdefinerte koden i svarvinduet:
streng = input()
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
print(sjekk_to(streng))
```

Maks poeng: 5

Funksjonsliste

Useful Functions and Methods

Du kan også finne en slik oversikt på denne nettsiden: functions.pdf.

Built-in:

format(numeric_value, format_specifier)

Formats a numeric value into a string according to the format specifier, which is a string that contains special characters specifying how the numeric value should be formatted. Examples of various formatting characters are “f=floating-point, e=scientific notation, %=percentage, d=integer”. A number before the formatting character will specify the field width. A number after the character “.” will format the number of decimals.

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.

Random methods:

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.

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.

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.

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.

numpy Library:

numpy.array(list)

Generates an array. If the list or tuple is a 2D-list it generates a matrix. With a 1D-list it generates a vector

ndarray.ndim

the number of axes (dimensions) of the array.

ndarray.shape

the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, shape will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.

ndarray.size

the total number of elements of the array. This is equal to the product of the elements of shape.

ndarray.dtype

an object describing the type of the elements in the array. One can create or specify dtype's using standard Python types. Additionally, NumPy provides types of its own. numpy.int32, numpy.int16, and numpy.float64 are some examples.

numpy.zeros(tuple)

creates a matrix with 0 using the tuple to define the number of lines and rows in the matrix

numpy.ones(tuple)

creates a matrix with 1 using the tuple to define the number of lines and rows in the matrix

numpy.arange(start,stop,step)

creates a vector starting at *start*, ending at *stop* using *step* between the values

numpy.ndarray.reshape(lines,rows)

reshapes a ndarray according to the values in *lines* and *rows*

Oppgave 3 (35%)

Løsing av oppgave 3:
Du kan velge å skrive kode direkte inn i Inspera hvis du ønsker det. Vi anbefaler derimot at du gjør følgende:

- Last ned filen h21_oppgave3.zip til din egen datamaskin. Du finner denne filen på første side av eksamen, på venstre side.
- Du kan alternativt åpne filene, og kopiere innholder inn i filer på din datamaskin
- Løs koden i en egnet editor, eksempelvis Thonny.
- **NB: Lim de ferdige funksjonene inn i Inspera.** Gjør det gjerne underveis, i tilfelle noe skulle skje med datamaskinen din.

Grunnen til dette er at vi har laget en del testkode som er kommentert ut. Fjern kommentarene, og dere vil kunne se om koden deres virker!

Først: Husk på at selv om du ikke har fått til en oppgave, så kan du bare late som og gå videre til neste. Det er ikke alltid en kan teste koden sin da, men du kan fremdeles skrive fullkommen kode etterpå!

Beskrivelse av oppgave 3:
Oppgave 3 består av et sett med deloppgaver innenfor samme tema. I enkelte oppgaver vil man kunne bruke funksjoner skrevet i tidligere oppgaver for å hjelpe seg. Man kan velge å kalle disse funksjonene selv om man ikke har klart å løse dem - det er bare å 'late som' i besvarelsen.

Du skal nå lage et program for en dagligvarebutikk. Programmet holder styr på lager, håndterer oppdatering av lager og salg. Dataen lagres i fil når butikken stenger, og leses fra fil når butikken åpnes. Butikken kan også generere middagsforslag til en gitt pris. Oppgaven baserer seg på strukturene matvarer og beholdning.

'matvarer' er en todimensjonal liste med følgende struktur:
[[navn, pris, rett],[navn2, pris2, rett2], ...]
Hver innerliste inneholder en string 'navn', et heltall 'pris' og en string 'rett' for en matvare.

matvarer = [['laks', 59, 'middag'], ['kjøttdeig', 25, 'middag'],
['ris', 15, 'middag'], ['ost', 99, 'frokost'], ['bønner', 7, 'middag'],
['soyasaus', 33, 'middag'],['banan', 4, 'mellommåltid']]

'beholdning' er en dictionary med følgende struktur:
{navn:antall, navn2:antall2, ...}
Hver nøkkel er en string 'navn', og hver verdi er et heltall 'antall' som beskriver hvor mange av den spesifikke varen som er på lager. Du kan anta at alle varer som finnes i beholdning også finnes som en vare i matvarer. Andre eventuelle antagelser må skrives i din besvarelse.

beholdning = {'laks': 6, 'kjøttdeig': 14, 'ris': 15, 'ost': 9, 'bønner': 6, 'soyasaus': 0, 'banan': 1}

Tallene i eksemplene til oppgavene forutsetter at man har fått til tidligere oppgaver.

16

3.1 - finn_pris(matvarer, let_etter) (4%)

Vi trenger å finne prisen på matvarene i butikken.

Skriv funksjonen finn_pris(matvarer, let_etter):

- Du skal finne prisen til en bestemt matvare, gitt av parameteret 'let_etter' i den todimensjonale listen 'matvarer'.
- Hvis let_etter ikke finnes i matvarer skal funksjonen returnere 0, ellers skal prisen på matvaren returneres.

Parametre:

- 'matvarer' er den todimensjonale listen som beskrevet i starten av oppgave 3.
- 'let_etter' er en streng som inneholder matvaren som det letes etter.

Eksempel:

>>> print(finn_pris(matvarer, 'ost'))

99

>>> print(finn_pris(matvarer, 'finnesikke'))

0

Skriv koden din på anvist sted. IKKE endre forhåndsoppgitt kode.

Test case #	Input	Forventet output
1	ost	99
2	finnesikke	0
3	laks	59
4	lakks	0
5	middag	0

```
matvarer = [['laks', 59, 'middag'], ['kjøttdeig', 25, 'middag'],
            ['ris', 15, 'middag'], ['ost', 99, 'frokost'], ['bønner', 7, 'middag'],
            ['soyasaus', 33, 'middag'],['banan', 4, 'mellommåltid']]
vare = input()
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
print(finn_pris(matvarer, vare))
```

Test kode

```
# Kode som du kan legge inn dersom du er uheldig og sletter
# den forhåndsdefinerte koden i svarvinduet:
matvarer = [['laks', 59, 'middag'], ['kjøttdeig', 25, 'middag'],
            ['ris', 15, 'middag'], ['ost', 99, 'frokost'], ['bønner', 7, 'middag'],
            ['soyasaus', 33, 'middag'],['banan', 4, 'mellommåltid']]
vare = input()
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
print(finn_pris(matvarer, vare))
```

Maks poeng: 4

17

3.2 - oppdater_matvare(beholdning, matvare, antall) (4%)

Butikken har behov for å kunne oppdatere en og en vare i beholdningen.
Skriv funksjonen oppdater_matvare(beholdning, matvare, antall):

- hvis 'antall' har en negativ verdi skal beholdningen gå med med tilsvarende mengde.
- du kan anta at alle matvarer man får inn som parametre allerede eksisterer.
- du kan anta at du ikke får en oppdatering som gir en negativ beholdning av en matvare.
- returner den oppdaterte beholdningen.

Parametre:

- dictionaryen 'beholdning' som beskrevet i starten av oppgave 3.
- en streng 'matvare'.
- heltallet 'antall'.

Eksempel på kjøring:
>>> print(beholdning['kjøttdeig'])
14
>>> oppdater_matvare(beholdning, 'kjøttdeig', -1)
>>> print(beholdning['kjøttdeig'])
13
Skriv koden din på anvist sted. IKKE endre forhåndsoppgitt kode.

Test case #	Input	Forventet output
1	kjøttdeig -1	13
2	kjøttdeig -3	11
3	laks 2	8
4	bønner 4	10

```
beholdning = {'laks': 6, 'kjøttdeig': 14, 'ris': 15, 'ost': 9, 'bønner': 6, 'soyasaus': 0
vare = input()
tall = int(input())
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
oppdater_matvare(beholdning, vare, tall)
print(beholdning[vare])
```

Test kode

```
# Kode som du kan legge inn dersom du er uheldig og sletter
# den forhåndsdefinerte koden i svarvinduet:
beholdning = {'laks': 6, 'kjøttdeig': 14, 'ris': 15, 'ost': 9, 'bønner': 6, 'soyasaus': 0, 'banan': 1}
vare = input()
tall = int(input())
# IKKE endre koden over her
# Skriv koden din mellom her...
# ...og her
# IKKE endre koden under her
oppdater_matvare(beholdning, vare, tall)
print(beholdning[vare])
```

Maks poeng: 4

18 3.3 - oppdater_beholdning(beholdning, endringer) (4%)

Butikken får påfyll av ulike matvarer når det skjer en leveranse. Da må beholdningen oppdateres. På samme måte må beholdningen reduseres når noen kjøper noe i butikken.

Skriv funksjonen oppdater_beholdning(beholdning, endringer):

- legg til/trekk fra endringer til beholdningen.
- returner den oppdaterte beholdningen.

Parametre:

- 'beholdning' er som tidligere.
- 'endringer' er en todimensjonal liste med matvarer og antall på format: [[navn,antall], [navn2,antall2], ...]

Eksempel:

>>> print(beholdning)

{'laks': 6, 'kjøttdeig': 13, 'ris': 15, 'ost': 9, 'bønner': 6, 'soyasaus': 0, 'banan': 1}

>>> beholdning = oppdater_beholdning(beholdning, [['ost', 2], ['kjøttdeig', -4], ['bønner', -1]])

>>> print(beholdning)

{'laks': 6, 'kjøttdeig': 9, 'ris': 15, 'ost': 11, 'bønner': 5, 'soyasaus': 0, 'banan': 1}

Skriv ditt svar her

```
1
```

Maks poeng: 4

19

3.4 - vis_priser(beholdning, matvarer, tekst) (5%)

Butikken trenger å vite hvilke varer som er i sortimentet!

Skriv funksjonen vis_priser(beholdning, matvarer, tekst):

- finn hvilke matvarer som er i 'tekst', fjern unødvendige tegn og ord som ikke er matvarer.
- returner en liste av tupler der hvert tuppel inneholder navn på vare og pris på den varen.
- du kan anta at 'tekst' er i lower-case.

Parametre:

- 'beholdning' og 'matvarer' er lik tidligere.
- 'tekst' er en string med matvarer og eventuelle andre ord, der ordene er skilt ved mellomrom. Teksten kan inneholde '.' og ','. F.eks: 'bønner og kjøttdeig, med banan.'

Eksempel:

```
>>> print(vis_priser(beholdning, matvarer, 'bønner med soyasaus, og ris'))
[('bønner', 7), ('soyasaus', 33), ('ris', 15)]
```

Skriv ditt svar her

```
1
```

Maks poeng: 5

20 3.5 - salg(matvarer, beholdning, handleliste) (5%)

Butikken er opptatt av glade kunder! Derfor ønsker butikken å gi kundene mulighet til å levere en handleliste. Dette er en liste med alle matvarene kunden ønsker.

Hvis kunden ønsker å handle flere av samme vare så vil matvaren gjentas (som ['ost', 'ost', 'banan']). Butikken skal selge matvarer helt til det er tomt for dem. Hvis kunden ønsker to ost, men det finnes kun en, så skal én ost selges.

Funksjonen salg(matvarer, beholdning, handleliste) skal gjøre følgende ting:

- Skrive ut alle matvarer som selges, og hvilken pris de selges for.
- Oppdatere beholdning for å reflektere salget som er gjennomført.
- Skrive ut sluttsummen for alle matvarer som ble solgt.
- Returner et tuppel bestående av alle matvarer som ble solgt.

Parametre:

- 'matvarer' og 'beholdning' er som tidligere.
- 'handleliste' er en liste med matvarer. Du kan anta at alle matvarene finnes i både 'matvarer' og 'beholdning'.

Eksempel:

```
>>> print(salg(matvarer, beholdning,['ost','banan','banan', 'kjøttdeig', 'kjøttdeig']))
ost 99
banan 4
kjøttdeig 25
kjøttdeig 25
Totalsum: 153
('ost', 'banan', 'kjøttdeig', 'kjøttdeig')
```

Skriv ditt svar her

```
1
```

Maks poeng: 5

21 3.6 - skriv_beholdning(beholdning), les_beholdning() (5%)

Når butikken stenger må varebeholdningen lagres til fil, og så må varebeholdningen leses inn fra fil igjen når butikken åpner. Skriv funksjonene skriv_beholdning(beholdning) og les_beholdning().

skriv_beholdning(beholdning) skal:

- lagre innholdet i dictionaryen 'beholdning' på et passende format.
- hvis filen finnes fra før, skal denne skrives over.

Parametre skriv_beholdning(beholdning):

- 'beholdning' er som tidligere.

les_beholdning() skal:

- lese den samme filen, opprette den samme dictionaryen, og returnere den.

Du velger selv format og struktur på filen, men det kan jo være enkelt å ta utgangspunkt i den samme folderen som programmet kjøres fra. NB! Husk at eventuelle moduler må importeres.

Filen trenger ikke være lesbar for mennesker, det du måles på er at det fungerer.

Skriv ditt svar her

```
1
```

Maks poeng: 5

22 3.7 - tilfeldig_middag(matvarer, budsjett) (8%)

Noen ganger trenger kunder litt inspirasjon for å lage seg en middag. Lag en funksjon tilfeldig_middag(matvarer, budsjett) som skal generere en tilfeldig middag med forskjellige matvarer av retten 'middag' innenfor gitt budsjett. Funksjonen trenger ikke ta hensyn til om matvaren er på lager, den skal kun gi inspirasjon til middagsretter.

Parametre:

- 'matvarer' er som tidligere.
- 'budsjett' er et heltall.

tilfeldig_middag(matvarer, budsjett) skal:

- velge tilfeldige matvarer i kategorien 'middag' som man har nok penger til å kjøpe, duplikat av matvarer er ikke lov.
- holde styr på hvor mye de tilfeldige varene koster.
- returnere en liste med tilfeldige middagsvarer som koster tilsammen maks 'budsjett' når det ikke lenger finnes varer som kan legges til.
- løsningen skal implementeres ved hjelp av en REKURSIV FUNKSJON på en vesentlig måte, enten funksjonen tilfeldig_middag selv, eller en hjelpefunksjon som gjør hoveddelen av jobben. Hvis ikke, så vil det maks gi halv poengsum for deloppgaven.

Eksempel på kjøring:

```
>>> tilfeldig_middag(matvarer, 70)
['laks', 'bønner']
>>> tilfeldig_middag(matvarer, 50)
['ris', 'bønner', 'kjøttdeig']
>>> tilfeldig_middag(matvarer, 20)
['ris']
```

Skriv ditt svar her

```
1
```

Maks poeng: 8