Forside

Institutt for Datateknologi og Informatikk

Eksamensoppgave i TDT4109 Informasjonsteknologi Grunnkurs

Eksamensdato: 8. desember 2022

Eksamenstid (fra-til): 15:00-19:00

Hjelpemiddelkode/Tillatte hjelpemidler: D
Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Faglig kontakt under eksamen: Børge Haugset: 934 20 190

Faglig kontakt møter i eksamenslokalet: NEI

ANNEN INFORMASJON:

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

Les oppgavene nøye, gjør dine egne antagelser og presiser i besvarelsen (på oppgaver der du skal skrive kode selv) eventuelle 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. Henvend deg til en eksamensvakt hvis du ønsker å kontakte faglærer. Noter gjerne spørsmålet ditt på forhånd.

Vekting av oppgavene: Vekting i prosent er angitt for hver deloppgave.

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. Du kan finne igjen varselet ved å klikke på bjella øverst til høyre.

Trekk fra/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: Etter eksamen finner du besvarelsen din i arkivet i Inspera. Merk at det kan ta én virkedag før eventuelle håndtegninger vil være tilgjengelige i arkivet.

1 Kopi av Oppgave 1 - teori (25%)

Marker det du mener er det mest riktige alternativet. Svaralternativene kommer i tilfeldig rekkefølge.

Det vil ikke gis minuspoeng for feil svar. Ved gjetning vil man i snitt få rett på 5 av 20 oppgaver, gitt 4 alternativ. Poengsummen fra teori (25) vil fordeles i spekteret 5 – 20 rette. Hvis man får mindre enn 5 rette vil man få 0 poeng på hele teorien, man vil altså aldri få minuspoeng.

Ingen deler av eksamen har minuspoeng for feil svar.

NY 1) Hva er heksadesimalrepresentasjonen av binærtallverdien 11011? Heksadesimalsystemet har 16 tall, [0..9,A..F]

- 29
- 1A
- 1B
- B1

NY 2) En telefonprodusent har en skjerm med oppløsning 1920x1080 piksel, og et ukomprimert bilde bruker 8.294.400 byte. Hva er tonedybden (fargedybden) per piksel?

- 24
- 32
- 16
- 8

NY 3) Hvis man komprimerer en lydfil, vil man kunne gjenskape akkurat den samme lydfilen igjen etterpå? Merk at argumentasjonen må være korrekt.

- Ja, all komprimering kan pakkes ut til det en startet med.
- Ja, hvis metoden for komprimering er definert slik.
- Ja, hvis man holder seg til Nyquist-regelen.
- Nei, komprimering innebærer alltid en form for tap.

4) Hva er et eksempel på rekursjon?

- En funksjon som kaller seg selv.
- En funksjon som gradvis jobber seg frem til en løsning.
- En evig løkke
- En løkke uten åpenbar, testbar avslutning (som 'while True:')

5) I hvilken situasjon er en sekvensiell søkealgoritme mest effektiv?

- Når de ønskede dataene er tall i sortert rekkefølge.
- Når de ønskede dataene ligger i starten av listen.
- Når de ønskede dataene ligger enten i slutten eller i starten av listen.
- Når datamengden er stor.

NY 6) Minnetypen som ligger mellom primæreminnet og registeret kalles

- Det finnes ikke noe nivå mellom cache og registeret.
- Cache
- Flash
- Buffer

7) Informasjon om informasjon kalles gjerne:

- Data
- Spesialdata
- Ekstrainformasjon
- Metadata

8) Hvordan kan man overklokke en datamaskin?

- Man endrer klokkefrekvensen direkte. Dette gjør man gjerne i BIOSen.
- Prosessorer jobber binært. Enten virker de i en gitt hastighet, eller ikke i det hele tatt.
- Man endrer klokkefrekvensen på strømmen inn, da en prosessors hastighet er en faktor av strømfrekvensen.
- Man endrer strømstyrken, da en prosessors hastighet avhenger av strømstyrken inn.

NY 9) Vi skal lage et nytt git repository i mappen vi har åpnet i et terminalvindu. Hvilken kommando bruker vi?

- git remote
- git make
- git start
- git init

10) Vi ønsker å dytte lokale endringer på filer (som er lagt til med add) opp til en git-server. Hva slags git-kommando(er) bruker vi i et terminalvindu?

- 'push' inkluderer automatisk 'commit', så kun 'push'.
- 'commit', da denne tar seg av å lagre ting i en versjon lokalt og dytte det til serveren.
- 'commit' for å akseptere endringene i denne versjonen av prosjektet, så 'push' for å lagre på serveren.
- 'commit' hvis du ikke har laget din egen gren (branch), og så 'push'.

11) Hva ligger i begrepet konfidensialitet når vi snakker om informasjon?

- Uautoriserte personer eller tjenester skal ikke ha tilgang til informasjonen.
- Informasjonen er personsensitiv, eller på annen måte konfidensiell.
- Oversikten over autoriserte tjenester og personer som skal ha tilgang til informasjonen.
- Prosessen med å avgjøre om informasjon er følsom eller ikke.

12) Data kan i følge McCumber være i tre tilstander: Storage (lagring), Processing (kjøring) og

- Deletion (sletting av informasjon)
- Transmission (overføring av informasjon)
- Authorization (verifikasjon av tilgjengelighet)
- Scattering (kryptering av informasjon)

13) Hva er riktig om harddisker?

- Volatilt og sequential access.
- Volatilt og random access.
- Permanent og sequential access
- Permanent og random access.

NY 14) Hvilken primærfarge ligger fargekoden #effe1a tettest opp mot:

- Blå
- Magenta
- Gul
- Oransje

15) Hva sier Moores lov?

- Klokkefrekvensen til prosessorer dobles annethvert år.
- Antallet integrerte kretser som selges dobles annethvert åt.
- Antallet transistorer en kan skvise inn i en integrert krets dobles annethvert år.
- Prisen på grafikkort dobles annethvert år.

16) Hvorfor leter CPU først i cachen etter data før den eventuelt går til RAM?

- Cachen har data liggende på et format som raskere kan tolkes av CPUen .
- Cachen bruker mindre strøm enn RAM.
- Cachen er koblet opp mot CPU med en høyere kommunikasjonshastighet.
- Premisset er feil. CPU spør først RAM, deretter cache.

17) I forbindelse med sampling, hva vil en endring av 'bitdybde' resultere i?

- Evnen til å fange opp endringer i frekvens.
- Endring av maksimum volumnivå.
- Endring av antallet målinger per tid.
- Evnen til å fange opp endringer i volum.

18) Hva er IKKE formålet til kontrollenheten (CU)?

- Hente instruksjoner fra cache og legge dem i registeret.
- Holde programtelleren.
- Holde en liste over instruksjoner.
- Holde en liste over enheters tilgangsrettigheter.

19) Hva sier Nyquist-regelen for sampling?

- Nyquist-regelen sier at samplingsfrekvensen må være minst like rask som den raskeste frekvensen. Ettersom menneskelige ører kan høre lyder opp til ca. 20 000Hz, vil samplingsfrekvens på 20 000Hz oppfylle Nyquists regel for digital lydopptak.
- Nyquist-regelen sier at samplingsfrekvensen må være minst fire ganger så rask som den raskeste frekvensen. Ettersom menneskelige ører kan høre lyder opp til ca. 20 000Hz, vil samplingsfrekvens på 80 000Hz oppfylle Nyquists regel for digital lydopptak.
- Nyquist-regelen sier at samplingsfrekvensen må være minst halvparten av den raskeste frekvensen. Ettersom menneskelige ører kan høre lyder opp til ca. 20 000Hz, vil samplingsfrekvens på 10 000Hz oppfylle Nyquists regel for digital lydopptak.
- Nyquist-regelen sier at samplingsfrekvensen må være minst dobbelt så rask som den raskeste frekvensen. Ettersom menneskelige ører kan høre lyder opp til ca. 20 000Hz, vil samplingsfrekvens på 40 000Hz oppfylle Nyquists regel for digital lydopptak.

20: Hvordan representerer man et flyttall i en datamaskin?

- Man avrunder verdien til nærmeste heltallsbrøk og lagrer disse verdiene.
- Heltallsdelen og desimaldelen lagres som hvert sitt 32-bits heltall.
- Det lagres i datamaskinens eget flyttallsminne (CFM)
- Man regner seg frem til en tilnærmet verdi og lagrer denne i stedet.

Maks poeng: 20

2 Useful Functions and Methods

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*

2(a) min_maks (2%)

Skriv funksjonen **min_maks(liste, minimum, maksimum)**.
Funksjonen skal returnere et heltall basert på følgende krav:

- Hvis et element i listen er mer enn maksimum skal det legges til 2.
- Hvis et element i listen er mindre enn minimum skal det legges til 1.

Eksempel:
>>> min_maks([1,2,3,4,5,6], 2, 4)
5

```
def min_maks(liste, minimum, maksimum):
    total = 0

    [Velg alternativ] (while True, for i in range(len(liste)), for total in liste, for elem in liste):

        if elem > [Velg alternativ] (minimum, len(liste), liste[i], maksimum):
            total += 2

        [Velg alternativ] (for, while, if, if (elem > 0) and) elem < [Velg alternativ] (maksimum,
total, liste[i], minimum):
            total += 1
    return total
```

Maks poeng: 2.67

2(b) sjekk_liste (3%)

Skriv funksjonen **sjekk_liste(liste)**
Funksjonen gå igjennom listen og regne summen, med følgende krav:

- Hvis summen av foregående tall i listen er større enn tallet du har kommet til skal funksjonen returnere tallet 0.
- Hvis tallet du kommer til er lik summen av foregående tall skal funksjonen returnere tallet 2.
- Hvis du kommer til tallet 0 skal funksjonen returnere summen av tallene så langt.
- Hvis man kommer til slutten av listen skal funksjonen returnere det siste tallet i listen.

>>> print(sjekk_liste([1,2,2,0,8])) # Skriver ut 0
>>> print(sjekk_liste([1,2,3,4,12,17])) # Skriver ut 2
>>> print(sjekk_liste([1,2,4,0,8])) # Skriver ut 7
>>> print(sjekk_liste([1,2,4,8,16,44])) # Skriver ut 44

Dra og slipp kodelinjer til riktig sted under slik at funksjonen virker som den skal.

Hjelp

liste[i]	tall	sum	2	<	>	1, len	0
<=	len						

```
def sjekk_liste(liste):
    sum = 0
    for i in range(          (liste)):
        tall =
        if tall == 0:
            return
        if tall          sum:
            return 0
        if tall == sum:
            return
        sum += tall
    return
```

Maks poeng: 3

2(c) tall_i_streng (3%)

Lag funksjonen **tall_i_streng(streng, tall)**
Funksjonen skal returnere hvor mange ganger tallet i parameter 2 opptrer i strengen (parameter 1).
Hvis tallet ikke finnes skal funksjonen returnere 0.
Tall kan ikke telles med flere ganger. Som eksempel vil tall_i_streng("11133333111",33) returnere 2 siden fire tretall vil brukes opp og det da kun gjenstår ett tretall.

>>> print(tall_i_streng("1211121211",11)) # skriver ut 2
>>> print(tall_i_streng("1133113333311111",33)) # skriver ut 3
>>> print(tall_i_streng("1211121211",1)) # skriver ut 7
>>> print(tall_i_streng("121112221211",222)) # skriver ut 1
>>> print(tall_i_streng("1211121211",3)) # skriver ut 0

Dra og slipp kodelinjer til riktig posisjon så funksjonen virker som den skal. Hjelp

tall	hvor == -1	len(streng) > 0	1	True
str(tall)	hvor	return antall		

```
def tall_i_streng(streng, tall):
    antall = 0
    while [          ] :
        hvor = streng.find(str(tall))
        if [          ] :
            [          ]
        antall += [          ]
        streng = streng[ [          ] + len( [          ] ) :]
```

Maks poeng: 3

2(d) fnum (5%)

Gitt funksjonen **fnum( )** som vist under, samt opprettelse av strengvariable filx, FILyy og variablene T, P, R, H som inneholder de tilsvarende enkeltbokstavene. Anta også at tekstfila **x.txt** har innhold som vist i den innfelte boksen, og at dette er *den eneste* .txt-fila på samme katalog som programkoden vår.

```
def fnum(filename, data, col):
    try:
        with open(filename) as f:
            for line in f:
                L = line.split()
                if L[0] == data:
                    return float(L[col])
        return 'Error 66'
    except FileNotFoundError:
        return 'Error 67'
    except:
        return 'Error 68'

filx = 'x.txt'
FILyy = 'YY.txt'
T, P, R, H = 'T', 'P', 'R', 'H'
```

```
T 2.0 1.5 3.3
P 1.1 WWW 1.3
R 1.1 1.3
0.5 H 0.7
```

I tabellen viser hver rad ett kall av funksjonen og kolonnene viser mulige returverdier.

For hver rad, sett kryss i kolonnen som viser riktig returverdi for kallet av funksjonen. Ingen minuspoeng for feil så du BØR svare noe også der du er usikker.

	Error 66	0.5	2.0	1.1	Error 67	www	Error 68	3.3	1.3
fnum(filx, R, 3)	○	○	○	○	○	○	○	○	○
fnum(filx, P, -1)	○	○	○	○	○	○	○	○	○
fnum(FILyy, T, 1)	○	○	○	○	○	○	○	○	○
fnum(filx, P, 2.0)	○	○	○	○	○	○	○	○	○
fnum(filx, H, 0)	○	○	○	○	○	○	○	○	○
fnum(filx, T, -1)	○	○	○	○	○	○	○	○	○
fnum(filx, P, 1)	○	○	○	○	○	○	○	○	○

Maks poeng: 5

2(e) adjust (5%)

Vi ønsker å justere en liste av lister med tall til å være av en bestemt lengde, og ønsker å lage to funksjoner for dette:

- **adjust_lists(L, length)** som muterer listen, og listene i listen
- **lists_adjusted(L, length)** , som returnerer ny versjon av listen og listene i listen, uten å endre originalen.

I begge tilfeller er parameteren L den lista som gis inn, og **length** er ønsket lengde på de indre listene. I resulterende liste skal

- lister som er kortere enn length, endres ved å legge til 0'ere bakerst slik at listens lengde blir **length**.
- lister som er for lange, kuttes de bakerste elementene, slik at lengden blir **length**.

For den muterende **adjust_lists** skal endringene skje i samme liste, og mutere listene i listen. For **lists_adjusted** skal det lages en ny liste med nye lister som elementer.

Eksempel på kjøring:

```
L = [ [3,1,4], [9,8,7,6,5], [1,2,3,4,5], [1,1,2,3,5,8,13,21]]
print(lists_adjusted(L, 5))   # viser justerte lister men har ikke endret original liste
print(L)                      # printer original liste
adjust_lists(L, 5)            # endrer original liste ved mutering
print(L)                      # printer original liste på nytt
```

Gjør at følgende printes:

```
[[3,1,4,0,0], [9,8,7,6,5], [1,2,3,4,5], [1,1,2,3,5]]
[ [3,1,4], [9,8,7,6,5], [1,2,3,4,5], [1,1,2,3,5,8,13,21]]
[[3,1,4,0,0], [9,8,7,6,5], [1,2,3,4,5], [1,1,2,3,5]]
```

OPPGAVE: Trekk kodelinjer til rett posisjon slik at de to funksjonene virker som de skal. Ingen minuspoeng for feil valg, så du BØR svare noe også der du er usikker.

Hjelp

del L[i][length:]	if len(L[i]) < length:
for i in range(len(L)):	if len(lst) >= length:
new_L = []	for lst in L:
new_L.append(lst + [0] * (length - len(lst)))	new_L.append(lst[:length])
return new_L	L[i] += [0] * (length - len(L[i]))

```
def adjust_lists(L, length):  # Mutating the lists
```

```
        else:
```

```
def lists_adjusted(L, length):  # Returning new lists
```

```
        else:
```

Maks poeng: 5

2(f) globetrotters (5%)

En globetrotterklubb registrerer for hver medlem hvilke land de har vært i utenom Europa. Her antar vi for enkelhets skyld at hvert medlem sine land fins i set-variable. Vi ønsker å lage fire funksjoner:

- **countries_either(m, m2)** skal returnere mengden av land som det ene (m) og/eller det andre medlemmet (m2) har vært i
- **countries_both(m, m2)** skal returnere mengden av land som har vært besøkt av både m og m2
- **countries_covered(club_members)** får inn ei liste av klubbmedlemmer - hver av dem representert ved ei mengde, og skal returnere mengda av alle land som noen i klubben har vært i
- **countries_exclusive(m, club_members)** skal returnere mengden av land som medlemmet m har besøkt, men som ingen av de andre medlemmene har besøkt

Eksempel på kjøring, hvor testkoden som kjøres er vist i grå boks, og det som returneres fra funksjonene står under boksen:

```
22 joe = {'Mali', 'Laos', 'Peru'}
23 ada = {'Mali', 'Niger', 'Peru'}
24 zac = {'Japan', 'China', 'Mali'}
25 club_members = [joe, ada, zac]
26 # Test running the functions
27 print(countries_either(joe,ada))
28 print(countries_both(joe,ada))
29 print(countries_covered(club_members))
30 print(countries_exclusive(ada, club_members))
```

```
{'Peru', 'Laos', 'Mali', 'Niger'}
{'Peru', 'Mali'}
{'Peru', 'Laos', 'Niger', 'Mali', 'Japan', 'China'}
{'Niger'}
```

OPPGAVE: Trekk kodelinjer til riktig plass så de fire funksjonene virker som de skal. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker.

Hjelp

set()	m.union(m2)	if m2 is not m:
result.difference(m2)	for m in club_members:	m
result.union(m)	m.intersection(m2)	

```
def countries_either(m, m2):
    result = 
    return result

def countries_both(m, m2):
```

```
    result =
    return result

def countries_covered(club_members):
    result =                                    .
                                                
        result =
    return result

def countries_exclusive(m, club_members):
    result =
    for m2 in club_members:
                                                
            result =
    return result
```

Maks poeng: 5

3 Useful Functions and Methods

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*

3(a) streng_beregning (3%)

Hvilket tall skrives ut når følgende kode og funksjonskall utføres?
Du skal skrive tallet, ikke strengen.

```
def streng_beregning(streng):
    starttegn = streng[0]
    antall = -1
    for tegn in streng:
        if tegn == starttegn:
            antall += 1
    return antall

print(streng_beregning("ascbasbasacc"))
```

.

Maks poeng: 3

3(b) Liste (3%)

Hvilke to tall skrives ut når følgende kode og funksjonskall utføres?
Merk at det er to bokser, en for hvert svar.

```
def foo(a,b):
    if len(a) > b:
        return a[b]
    return a.reverse()[b]

print(foo([1,2,3,4,5],4))
print(foo([1,2,3,4,5],6))
```

Velg alternativ (5, 3, 4, TypeError: 'NoneType' object is not subscriptable)

Velg alternativ (IndexError: list index out of range, 4, 5, 3, TypeError: 'NoneType' object is not subscriptable)

Maks poeng: 3

3(c) Liste 2 (3%)

Hvilke to tall skrives ut når følgende kode og funksjonskall utføres?
Merk at det er to bokser, en for hvert svar.
Du skal skrive tallene, ikke strengene.

```
def foo(l1,l2):
    x = 0
    for i in range(len(l1)):
        if l1[i] > l2[i]:
            x += 1
        else:
            x -= 1
    return x

print(foo('abcd','dcba'))
print(foo([1,2,3,4,5],[5,4,3,2,1]))
```

Maks poeng: 3

3(d) Kjør! (2%)

Hvilket tall skrives ut når følgende kode kjøres ut?
Du skal skrive tallet, ikke strengen.

```
liste = [1, 2, 3, 4, 5]
liste.append(6)
liste.reverse()
liste = liste[:3]
liste[2] = 7

print(sum(liste))
```

.

Maks poeng: 2

3(e) eksamen er streng (3%)

Hvilken streng skrives ut når følgende kode kjøres?
Merk at du skal skrive ut **kun strengen**, ikke noe ' eller ".
(tips: enumerate gir både verdi og indeks på en gang.)

```
streng = "e1ks2ame3n"
streng2 = ""

for i, tegn in enumerate(streng):
    if tegn.isnumeric():
        streng2 += streng[i-1]
        streng2 += str(i)

print(streng2)
```

Maks poeng: 3

3(f) card number (4%)

Vi ønsker å lage en funksjon card_no(card) som får inn en streng som representerer et spillkort og returnerer kortets tallverdi. Eksempel på strenger er '♥3', '♠4', '♣10', '♣Q', '♦A', dvs. første tegn i strengen er alltid et symbol for kortets farge (♥/♠/♣/♦), øvrige 1-2 tegn (2 i tilfelle tallet er 10) representerer tallverdien. Bokstavene J, Q, K, A teller som 11, 12, 13, 14.

Eksempel på kjøring:

print(f"{card_no('♣10')} {card_no('♥3')} {card_no('♣Q')} {card_no('♦A')}")

gir utskriften

10 3 12 14

Fyll inn det som mangler slik at funksjonen virker som den skal. Ingen minuspoeng for feil, så du BØR svare noe også der du er usikker.

```
def card_no(card):
    if card[          ]     'JQKA':
        return      + 'JQKA'.find(card[          ])
    else:
        return      (card[      ])
```

Maks poeng: 4

3(g) card games (6%)

I denne oppgaven skal vi lage enda to funksjoner som kan brukes i kortspill, nemlig **sorted_cards(hand**) og **n_plays(cards)**.

Funksjonen **sorted_cards(hand)** får inn ei liste av kort som en spiller har fått utdelt, gjerne hulter til bulter i forhold til farge og tallverdi. Funksjonen skal returnere en dictionary med samme kort gruppert etter farge og sortert etter tallverdi.

Eksempel på kjøring
sorted_cards(['♥3', '♥5', '♠7', '♦9', '♦J', '♦10', '♦A', '♦K', '♥4'])
skal returnere
{'♥': [3, 4, 5], '♠': [7], '♦': [9, 10, 11, 13, 14]}

Konvertering av J, Q, K, A til tall gjøres ved hjelp av funksjonen card_no() som ble laget i forrige deloppgave.

Funksjonen **n_plays(cards)** får inn som parameter en dictionary som nettopp vist over (hvilke tallverdier av kort man har i hver farge). Denne funksjonen skal brukes for å regne ut hvor mange runder man trenger for å bli kvitt alle kortene, gitt at man har lov å legge en hel rekke av kort på en runde, så sant kortene er av samme farge og i løpende nummerrekkefølge.

Eksempel på kjøring:
n_plays({'♥': [3, 4, 5], '♠': [7], '♦': [9, 10, 11, 13, 14]})
skal returnere **4**
Dette fordi man trenger ett utspill på å bli kvitt rekka '♥': [3, 4, 5], ett på den ene '♠': [7], og to runder på å bli kvitt '♦', nemlig 9,10,11 og 13,14 fordi man mangler nr 12.

Fyll inn det som mangler slik at de to funksjonene virker som de skal. Ingen minuspoeng for feil, så du BØR svare noe også der du er usikker.

```
def sorted_cards(hand):

    kinds = [Velg alternativ](tuple(), [ ], { })
    for card in hand:

        if [Velg alternativ](card[1], card[0], card) in [Velg alternativ](hand, kinds, card):

            [Velg alternativ](kinds[card[0]], kinds, hand).append(card_no(card))
        else:

            kinds[card[0]] = [Velg alternativ]([card_no(card)], card_no(card[1:]),
card_no(card))
```

```
    for k in kinds:

        kinds[ Velg alternativ (k, 0, 1:)].sort()
    return kinds

def n_plays(cards):
    n = 0

    for kind in Velg alternativ (len(range(n-1)), range(n), cards):

        Velg alternativ (cards.append(n), n = 1, n += 1)

        for i in range( Velg alternativ (1, 0, -1), len(cards[kind])):

            if Velg alternativ (cards[kind][i], cards[kind][0], cards[kind][n]) >

Velg alternativ (cards[kind][i+1] + 1, cards[kind][n] + 1, cards[kind][i-1] + 1):

                Velg alternativ (n = 0, break, n += 1)
    return n
```

Maks poeng: 9

3(h) rekursjon (3%)

Hva blir resultatet av følgende kode og funksjonskall:
Merk at det er to kall.

```
def foo(bar, x):
    if not bar: return 0
    return int(x) + foo(bar[1+int(x):],bar[0])

print(foo("101","1"))
print(foo("100110101011","0"))
```

Maks poeng: 3

4 Useful Functions and Methods

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*

4(a) les_meteoritter (10%)

Skriv funksjonen **les_meteoritter(filnavn)**

Funksjonen tar inn et filnavn, og returnerer en todimensjonal liste med delvis tolkede verdier basert på linjene i filen.

- csv-filen heter *'meteoritter.csv'*, er på UTF-8-format, separatoren er komma, ',' og starter slik:

name,id,class,mass,year,latitude,longitude
Aachen,1,L5,21.0,1880,50.775,6.08333
Aarhus,2,H6,720.0,1951,56.18333,10.23333
Abee,6,EH4,107000.0,1952,54.21667,-113.0
Acapulco,10,Acapulcoite,1914.0,1976,16.88333,-99.9
...
Vi har noen krav til tolkningen:

- Merk at første linje er overskriftene, og skal ikke med. Rekkefølgen på disse er fast.
- Massen skal tolkes som en flyttall.
- Det finnes nedslag der masse ikke er registrert. Her skal massen settes til 0.
- Nedslagsår skal tolkes som heltall.
- De resterende feltene kan lagres som strenger (alle bortsett fra *year* og *mass*)
- Format kan spesifiseres med *encoding="utf8"*
- Problemet er at enkelte meteorittklasser (*class*) også har komma i klassebetegnelsen, som på følgende linje (merk hvordan feltet for klassen isoleres med "):

Akyumak,433,"Iron, IVA",50000.0,1981,39.91667,42.81667

Alle disse spesialtilfellene er jernmeteoritter. Vi løser dette ved å erstatte klassen for disse variantene (*"Iron, IVA"* og alle andre meteorittklasser med komma) med strengen *'Iron'*.

Resultatet vil bli følgende:
>>> m = les_meteoritter("meteoritter.csv")
>>> print(m)
[['Aachen', '1', 'L5', 21.0, 1880, '50.775', '6.08333'], ['Aarhus', '2', 'H6', 720.0, 1951, '56.18333', '10.23333'], ..., ['Akyumak', '433', 'Iron', 50000.0, 1981, '39.91667', '42.81667'], ...]

Skriv ditt svar her

```
1
```

Maks poeng: 10

4(b) sorter_masse (5 %)

Skriv metoden **sorter_masse(nedslag, kolonne)**

- Funksjonen skal returnere en dictionary der de ulike nedslagenes verdi i den gitte kolonnen (*class*, *year* etc.*)* skal brukes som nøkkel. Bak denne nøkkelen skal så alle massene for disse 'like' nedslagene **summeres** (enten nøkkelen er år, nedslagstype eller noe annet).
- nedslag er den todimensjonale listen som kommer fra funksjonen *les_meteoritter*.
- kolonne er en streng som indikerer kolonnenavnet angitt i forrige oppgave (første linje i filen). Du kan forutsette at kolonnene alltid har denne rekkefølgen.

Eksempel (her er listen forkortet til to elementer, to av ulike typer. Merk at eksempelet ikke viser to nedslag av samme klasse):
>>> m = [['Aachen', '1', 'L5', 21.0, 1880, '50.775', '6.08333'], ['Aarhus', '2', 'H6', 720.0, 1951, '56.18333', '10.23333']]
>>> sortert = sorter_masse(m,'class') # sorter på type (class, altså kolonne 2)
>>> print(sortert)
{'L5': 21.0, 'H6': 720.0}

Skriv ditt svar her

```
1
```

Maks poeng: 5

4(c) lagre/hent (4%)

Skriv funksjonene **lagre_sorterte_nedslag(sortert)** og **hent_sorterte_nedslag()**.

Disse funksjonene skal henholdsvis lagre og hente inn samlingen *sortert* fra forrige oppgave (en dictionary) til/fra en fil. Du velger selv hvordan du gjør det, innenfor pensum av kurset. Forutsett at filen lagres i den samme mappen som 'koden din ligger'.

Skriv ditt svar her

```
1
```

Maks poeng: 4

4(d) Hvilket år... (6%)

Skriv ut hvilke(t) år det ble registrert størst total nedslagsmasse!

- Du skal skrive all koden du trenger, det eneste du har er filen *'meteoritter.csv'* som er beskrevet i oppgave 4a.
- Hvis flere år har samme høyeste masse registrert skal du skrive ut alle, ett årstall på hver linje.
- Du trenger ikke skrive denne koden som en ny funksjon, men det kan lurt å bruke noen av funksjonene fra tidligere oppgaver. Du kan forvente at de fungerer slik som spesifisert, selv om du ikke har klart å løse dem.

Skriv ditt svar her

```
1
```

Maks poeng: 6