TDT4109 - s2024 - Forside

Institutt for datateknologi og informatikk

Eksamensoppgave i TDT4109 - Informasjonsteknologi, grunnkurs

Eksamensdato: 6. august 2024

Eksamenstid (fra-til): 09:00-13:00

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

Faglig kontakt under eksamen: Børge Haugset
Tlf.: 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 hvilke forutsetninger du har lagt til grunn i tolkning/avgrensing av oppgaven. Faglig kontaktperson kontaktes kun dersom det er direkte feil eller mangler i oppgavesettet. Henvend deg til en eksamensvakt hvis du mistenker feil og mangler. Noter spørsmålet ditt på forhånd.

Ingen håndtegninger: Denne eksamenen tillater ikke bruk av håndtegninger. Har du likevel fått utdelt skanne-ark, er dette en feil. **Arkene vil ikke bli akseptert for innlevering, og de vil derfor heller ikke sendes til sensur.**

Vekting av oppgavene: Oppgavenes vekting står i eksamen. Poeng er ca. lik prosent, med forbehold om at dette kan endres under sensur.

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.

Eksamensstruktur og råd

Vekting av oppgavene: Det er angitt i prosent hvor mye hver deloppgave i eksamenssettet teller ved sensur. Poengandelen er ca. lik prosent. Altså 5 poeng ~= 5% av eksamen. Vi forbeholder oss retten til å kunne avvike fra disse andelene i sensurarbeidet hvis dette skulle bli nødvendig. Les gjennom hele oppgavesettet før du begynner å løse oppgaven. Disponer tiden godt!

Eksamen er delt i tre seksjoner:

- Del 1: Teoridel som teller ca. **25%** av den totale vurderingen. I denne seksjonen fordeles 25 poeng mellom 5 og 20 rette, siden man med gjetting vil kunne få 25% korrekt. Man får derimot uansett aldri mindre enn 0 poeng på del 1.
- Del 2 (oppgaver 2.1 - 4.6): Ulike automatisk rettede programmeringsoppgaver med multiple choice, drag and drop og innfylling av resultat. Disse teller til sammen ca. **45**%.
- Del 3: Programmering. Disse teller til sammen ca. **30**%. Du kan ikke kjøre koden.

Struktur, info og tips:

- Det er **ikke** minuspoeng for å svare feil i del 2, så gjett heller enn å hoppe over oppgaven.
- Les oppgavene nøye (som det står på forrige side), så du ikke misforstår dem. Noen oppgaver spør bl.a. hvilket alternativ som er feil, ikke riktig.
- Svar det du får til på programmeringsoppgavene. Husker du ikke hvordan å løse hele oppgaven, men deler av den? Gjør det. Skriver du ingenting får du 0 poeng. Skriver du litt, kan du få litt poeng. Og inkluder gjerne korte kommentarer i koden du skriver i programmeringsseksjonen, så det er lettere å skjønne hva du gjør eller prøver å gjøre. **NB!** Hvis du løser oppgavene på flere måter, vil vi ta gjennomsnittet av utdelte poeng, heller enn å velge den beste. Det er altså *ikke* nødvendigvis en fordel å vise alle måtene en kan løse den på.
- Det er mange oppgaver på eksamen. De siste tar sannsynligvis lengst tid. Fordel tiden smart.
- Inspera liker ikke kode i oppgaveteksten, og kodeblokkene har noen ganger hoppet opp til toppen av oppgaven. Derfor har jeg lagt til referense i tekst og i kodeblokken. Dette kan f.eks. se sånn ut: *"I koden under (#2) ser vi blabla..."*. Dette vil si kodensnutten som starter med *"#2"*, som sikkert vises i toppen av oppgaven.

PC-tips:

- På vanlig **Windows**-tastatur kan vi skrive følgende tegn på følgende måte:
  - (): shift + 8/9
  - []: alt.gr. + 8/9
  - {}: alt.gr + 7/0
  - /: shift + 7
  - \: tasten til venstre for backspace
  - ': tasten til venstre for enter
- På **Macbook** kan vi skrive det samme på følgende måte:
  - (): shift + 8/9
  - []: option + 8/9
  - {}: option + shift + 8/9
  - /: shift + 7
  - \: option + shift + 7
  - ': tasten til venstre for 1
- Alt.gr. (Windows) finnes rett til høyre for space/mellomrom
- option (Mac) er nederst, litt til høyre og venstre for space/mellomrom
- Shift er pilen opp, nesten nede i venstre hjørne

Lykke til!!

1

Her kan du kladde

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

Useful Functions and Methods

These are listed in the following order:

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

Built-in:

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

```
try:
# Code to test
except:
# If code fails. E.g exception types: IOError, ValueError, ZeroDivisionError.
   # Variant: except Exception as exc # Let's you print the exception.
else:
   # Runs if no exception occurs
finally:
   # Runs regardless of prior code having succeeded or failed.
```

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

Removes the first element containing the item. Works in_place. Returns None

s.reverse()

Reverses the order of the items in a list. Works in_place. Returns None

s.sort()

Rearranges the elements of a list so they appear in ascending order. Works in_place. Returns None

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

Test whether every element in *s* is in *t*

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

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

Files:
open()
Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be ‘r’ (read only), ‘w’ (writing only), ‘a’ (appending), ‘r+’ (both reading and writing).
f.read(size)
Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.
f.readline()
Reads a single line from the file (reads until newline character (\n) is found), and returns it as a string.

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

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

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

Random Library:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

math.e

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

1 Teori (25%)

Marker det du mener er det mest riktige alternativet. Svaralternativene kommer i tilfeldig rekkefølge.
Ved gjetting vil man i snitt få rett på 5 av 20 oppgaver, gitt 4 alternativ. Poengene distribueres dermed mellom 5 og 20 rette. Under 5 rette gir ikke minuspoeng på eksamen, så gjett ivei!

1. Sekundærminne...

- er raskere enn primærminne.
- er dyrere per byte enn primærminne.
- må ha strøm for å lagre informasjon.
- lagrer informasjonen også etter at strømmen er skrudd av.

2. Hva er binærrepresentasjonen til HEX-tallet '1E'?

- 00011110
- 00011100
- 00011111
- 00111110

3. Hvor mange byte trenger man for å representere et 4K-bilde (3840x2160) hvis bildet er ren svart-hvitt?

- 16588800
- 8294400
- 1036800
- 4147200

4. Hvilken påstand er korrekt for transistorer?

- De brukes til å lagre informasjon permanent, for eksempel i en SSD.
- De kan gjøre om digitale signaler om til analoge signaler, men ikke motsatt.
- De erstattet radiorør i gamle dager, men er nå erstattet av ny teknologi.
- De er en slags brytere som styres ved hjelp av strøm.

5. Hva er hovedoppgaven til en ALU?

- Styre programtelleren (Program Counter).
- Hente og utføre instruksjoner.
- Styre dataflyten mellom prosessoren og co-prosessoren.
- Utføre regneoperasjoner.

6. Hvis ascii-strengen 'bare' representeres på binærformen 0110 0010 0110 0001 0111 0010 0110 0101 - hvordan representeres ascii-strengen 'bra'?

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

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

- Kompilering betyr at én og en kodelinje oversettes til datamaskinspråk, og kjøres.
- Tolket kode er raskere å kjøre enn kompilert kode.
- Tolking betyr at man oversetter et helt program til et annet og mer datamaskinnært språk som datamaskinen så kjører.
- Tolking betyr at én og én kodelinje oversettes til datamaskinspråk, og kjøres.

8. Hvis jeg gir deg den heksadesimale koden "#ffff00" - hva burde du kalt fargen?

- rød
- blå
- grønn
- gul

9. Hva er pipelining?

- En teknikk som gjør at en CPU kan utføre flere prosesser i parallel.
- En teknikk som brukes for å oppnå en trygg forbindelse mellom to datamaskiner (for eksempel i bruk i VPN).
- En teknikk som gjør at man kan sende data mellom ulike deler av datamaskinen (for eksempel fra nettverkskort til CPU til SSD).
- En teknikk der lagringsmedier kan brukes i parallel for å bli raskere (for eksempel i RAID).

10. Gitt UNICODE - hvor mange byte er det minste man kan bruke for å representere et tegn?

- 4
- 3
- 1
- 2

11. Hvis samplingsfrekvensen i en sampling settes ned, hva slags endring medfører det?

- Man kan minste lydinformasjon.
- Spørsmålet er feil stilt - samplingsfrekvensen er definert ved hjelp av Nyquists teorem, og er 44.1kHZ.
- Man kan introdusere mer støy.
- Volumet på signalet blir lavere.

12. Hvilke informasjonssikkerhetsbehov har vi for data, i henhold til McCumbers kube?

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

13 - Hva sier Moores lov?

- Klokkehastigheten til en prosessor øker proporsjonalt med antallet transistorer prosessoren har.
- Samplingsfrekvensen er direkte knyttet til volumet på signalet.
- Jo mer diskplass man har, jo mer plass tar spillene og programmene man installerer.
- Antallet transistorer i en integrert krets dobles omtrent hvert annet år, på grunn av teknologiske fremskritt.

14. Organiser etter hastighet, raskest til tregest:

- Transistor, relé, radiorør
- Transistor, radiorør, relé
- Radiorør, relé, transistor
- Relé, transistor, radiorør

15. Kontrollenheten har to registre, den ene er instruksjonsregisteret. Hva heter det andre?

- Dataregisteret
- Minnekontrollen
- Seleksjonsregisteret
- Programtelleren

16. Vi skal lage oss et nytt git repository, og står inni et terminalvindu. Hva er kommandoen vi skriver?

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

17. Hva ligger i begrepet konfidensialitet når vi snakker om informasjon?

- Informasjonen er av en art som gjør at man ikke ønsker at hvem som helst skal få endre på.
- Informasjon er av en art som man ikke ønsker at hvem som helst skal få se.
- Når man har brutt seg inn i et datasystem så har man oppnådd konfidensialitet.
- Det er ikke noen andre enn spesielt definerte entiteter som får se informasjon.

18. Hva er et register? (I konteksten datamaskinarkitektur)

- En rask og liten datalagringsenhet inne i CPU som brukes til å lagre og hente data for raske operasjoner.
- En enhet for langsiktig lagring av data i datamaskinens hovedminne.
- Et spesielt område i RAM som holder oversikt over alle filene på datamaskinen.
- En tabell over hvordan man utfører regneoperasjoner som AND, XOR og liknende som benyttes av prosessoren.

19. Innhenting av data fra RAM til registeret gjøres i en spesiell fase, men hvilken...

- Command
- Fetch
- Execute
- Decode

20. Hvis en funksjon i Python tar inn en liste som parameter, og man gjør endringer på listen inni funksjonen, hvordan kan endringene beholdes etter at funksjonen avsluttes?

- Hvis man i en funksjon endrer på en liste man får inn som parameter så vil endringene i den originale listen beholdes selv uten at listen eksplisitt returneres.
- Siden dette er aaaaller siste gang jeg skriver noen form for teorispørsmål til ITGK-eksamen så skal dere som en premie få ett alternativ mindre på den siste oppgaven. Til gjengjeld kommer noen av dere til å lese gjennom hele denne teksten, og dermed tape bittelitt tid. Det tror jeg går fint. Lykke til videre! - Børge
- Hvis man skriver 'global liste' inni funksjonen så vil tolkeren skjønne at endringene skal beholdes.
- Eneste mulighet er å returnere den oppdaterte listen.

Maks poeng: 33.3

2

Her kan du kladde

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

Useful Functions and Methods

These are listed in the following order:

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

Built-in:

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

```
try:
# Code to test
except:
# If code fails. E.g exception types: IOError, ValueError, ZeroDivisionError.
   # Variant: except Exception as exc # Let's you print the exception.
else:
   # Runs if no exception occurs
finally:
   # Runs regardless of prior code having succeeded or failed.
```

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

Removes the first element containing the item. Works in_place. Returns None

s.reverse()

Reverses the order of the items in a list. Works in_place. Returns None

s.sort()

Rearranges the elements of a list so they appear in ascending order. Works in_place. Returns None

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

Test whether every element in *s* is in *t*

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

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

Files:
open()
Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be ‘r’ (read only), ‘w’ (writing only), ‘a’ (appending), ‘r+’ (both reading and writing).
f.read(size)
Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.
f.readline()
Reads a single line from the file (reads until newline character (\n) is found), and returns it as a string.

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

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

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

Random Library:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

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

2.1 Kodeforståelse 1

```
# 1
CF_0 = 150_000
n = 10
r = 0.05

def PV(sum, år, rente):
    return sum * (1 + r)**år

print(PV(CF_0, n, 0.03))
print(PV(CF_0, n, 0.04))
```

Det er flere ting som er dårlig eller feil med koden under (#1). Hvilket av alternativene er **ikke** en av disse?

Velg ett alternativ:

- Vi kaller et av `PV` sine parametere `sum`, og overskriver dermed den innebyggede funksjonen med samme navn
- Vi lager `r=0.05` i starten, men bruker heller 0.03 og 0.04
- `PV` har `rente` som parameter, men bruker heller den globale variablen `r`
- Vi lagrer ikke resultatet fra `PV` i en variabel før vi prøver å skrive ut

Maks poeng: 2

2.2 Kodeforståelse 2

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

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

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

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

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

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

Vi har funksjonene `read_file` (#A) og `read_savegame` (#B).

Det er en mulighet at filen (file path/navn/lokasjon) vi sender inn i `read_file` ikke finnes. Da vil programmet krasje. Her er fire forslag på metoder å løse dette på. Men én av de vil **ikke** fungere. Hvilken fungerer **ikke**?

Hint: Bruk se Python-dokumentasjonen for mer om try-except og modulene os/os.path og sys.

Velg ett alternativ:

- # 3
- # 1
- # 2
- # 4

Maks poeng: 3

2.3 Kodeforståelse 3

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

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

Hvilken påstand er sann?

Velg ett alternativ:

- Hvis brukeren skriver inn noe annet enn en av kategoriene, vil programmet krasje.
- Hvis brukeren skriver inn noe annet enn en av kategoriene, vil `None` returneres. Det er dårlig kodepraksis.
- Vi lager en string (med '') i funksjonen, men den lagres ikke. Det er dårlig kodepraksis.
- Vi lager en string (med '') i funksjonen, men den lagres ikke. Det gjør at programmet vil krasje.

Maks poeng: 2

2.4 Kodeforståelse 4

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

Vi har følgende mappestruktur (#1) i prosjektet vårt.

Altså heter prosjektmappen vår `prosjektmappe`. Inni den har vi programmet vårt `program.py`. Vi har også en undermappe kalt `undermappe` i prosjektmappen. Den har filene `init.py` og `modul.py` inni seg. Vi vil importere koden fra `modul.py` inn i `program.py`, ved bruk av relativ import. (`prosjektmappe` er ikke en installert modul.) Hva skriver vi i `program.py`?

Hvilken importering funker fra `program.py`?

Velg ett alternativ:

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

Maks poeng: 2

2.5 Kodeforståelse 5

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

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

Hvilke(n) datatype(r) blir returnert? Eller krasjer den?

Velg ett alternativ:

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

Maks poeng: 2

2.6 Kodeforståelse 6

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

I denne oppgaven skal vi finne ut hvilke kodelinjer som gir `True` eller `False`.

Hva er verdien til `x`? (#1)
Velg ett alternativ

True False

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

Hva er verdien til `y`? (#2)
Velg ett alternativ

False True

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

Hva er verdien til `z`? (#3)
Velg ett alternativ

True False

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

Hva er verdien til `r`? (#4)
Velg ett alternativ

False True

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

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

Velg ett alternativ:

False	True
○	○

Maks poeng: 3

2.7 Kodeforståelse 7

Me har dei følgjande setta (#1).

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

Loyale følgere av Mark Zuckerberg er på alle Facebook/Meta sine plattformer (Facebook og Instagram, i dette tilfellet) og ingen andre (LinkedIn eller Twitter/X, i dette tilfellet). Hvilken linje med kode skriver ut de loyale følgerne?

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

Velg ett alternativ:

- ○ 3
- ○ 4
- ○ 1
- ○ 2

Maks poeng: 3

3

Her kan du kladde

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

Useful Functions and Methods

These are listed in the following order:

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

Built-in:

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

```
try:
# Code to test
except:
# If code fails. E.g exception types: IOError, ValueError, ZeroDivisionError.
   # Variant: except Exception as exc # Let's you print the exception.
else:
   # Runs if no exception occurs
finally:
   # Runs regardless of prior code having succeeded or failed.
```

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

Removes the first element containing the item. Works in_place. Returns None

s.reverse()

Reverses the order of the items in a list. Works in_place. Returns None

s.sort()

Rearranges the elements of a list so they appear in ascending order. Works in_place. Returns None

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

Test whether every element in *s* is in *t*

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

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

Files:
open()
Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be ‘r’ (read only), ‘w’ (writing only), ‘a’ (appending), ‘r+’ (both reading and writing).
f.read(size)
Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.
f.readline()
Reads a single line from the file (reads until newline character (\n) is found), and returns it as a string.

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

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

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

Random Library:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

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

3.1 Fyll inn 1

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

Her skal du skrive inn svaret. **BARE AKKURAT NØYAKTIG SVARET!** Ingenting mer! Ikke "Svaret er ..." Nei! Inspera vil ikke ha noe annet enn bare svaret. Husk å bruke . i stedet for , i tall (pi = 3.14, ikke 3,14) og sånt.

Følgende (#1) kode kjøres. Hva skrives ut?

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

Svar:

Maks poeng: 3

3.2 Fyll inn 2

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

Følgende (#1) kode kjøres. Hva skrives ut?

Svar:

Maks poeng: 3

3.3 Fyll inn 3

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

c = 2
z = 0

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

print(z)
```

Følgende (#1) kode kjøres. Hva skrives ut?

Svar:

Maks poeng: 3

3.4 Fyll inn 4

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

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

Følgende (#1) kode kjøres. Hva skrives ut?

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

Maks poeng: 2

4

Her kan du kladde

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

Useful Functions and Methods

These are listed in the following order:

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

Built-in:

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

```
try:
# Code to test
except:
# If code fails. E.g exception types: IOError, ValueError, ZeroDivisionError.
   # Variant: except Exception as exc # Let's you print the exception.
else:
   # Runs if no exception occurs
finally:
   # Runs regardless of prior code having succeeded or failed.
```

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

Removes the first element containing the item. Works in_place. Returns None

s.reverse()

Reverses the order of the items in a list. Works in_place. Returns None

s.sort()

Rearranges the elements of a list so they appear in ascending order. Works in_place. Returns None

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

Test whether every element in *s* is in *t*

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

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

Files:
open()
Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be ‘r’ (read only), ‘w’ (writing only), ‘a’ (appending), ‘r+’ (both reading and writing).
f.read(size)
Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.
f.readline()
Reads a single line from the file (reads until newline character (\n) is found), and returns it as a string.

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

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

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

Random Library:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

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

4.1 Drag and drop 1

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

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

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

Vi er gitt følgende string og ordbok, og skal lage funksjonen `oversetter`.

Den skal gjøre en dum oversetting, ved å bytte ut ord for ord. 'hei' skal byttes ut med 'hey' hvis vi oversetter norsk-til-engelsk (nor2eng). Ellers skal 'hey' byttes ut med 'hei'.

Bygg opp funksjonen `oversetter`.

Hjelp

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

```
def oversetter(ordbok, string, nor2eng):
    for [          ] in [          ] :
        if [          ] :
            nor, eng = eng, nor
        string = [          ]
    return string
```

Maks poeng: 2.4

4.2 Drag and drop 2

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

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

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

Vi er amerikanere, så vi vil ikke bruke kommunisttid (24h). Funksjonen under konverterer til frihetstid (12h). Altså kan vi forvente at tiden 15:35 blir konvertert til 3:35. I tillegg skal den returnere om tiden er am (før 12:00 dagtid). Se eksempler over.

Hjelp

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

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

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

Maks poeng: 3

4.3 Drag and drop 3

Vi skal lage en funksjon som returnerer `True` eller `False` basert på om det er skuddår eller ikke. Reglene er som følger:

- Det er skudd år hvis årstallet (`år`) er delelig med 4 ...
- ... med mindre det også er delelig med 100 ...
- ... med mindre det også er delelig med 400
- Ellers er det ikke skuddår.

For eksempel:

- 2004 er delelig med 4: Skuddår
- 2100 er delelig med 4, men også med 100: Ikke skuddår
- 2400 er delelig med 4, og 100, men også med 400: Skuddår

Bygg opp funksjonen under.

Hjelp

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

```
def er_skuddår(år):
    if          :
        return True
    elif        :
        return
    elif        :
        return
    else:
        return False
```

Maks poeng: 3

4.4 Drag and drop 4

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

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

Hjelp

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

```
def finn_gjennomsnittslengden(tabell):
    summen = [          ]
    antall = 0
    for [          ] in [          ]
        for [          ] in rad:
            summen += [          ]
            antall += [          ]
    return [          ]
```

Maks poeng: 2.8

4.5 Drag and drop 5

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

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

Vi vil gjøre en statistisk analyse på terningkast.

Hvis vi kaster 60 terninger forventer vi å få ca. 10 av hvert tall. 10/60 er 16.67%. Jo flere ganger vi kaster terninger, jo nærmere vil vi komme dette tallet. #1 viser et rettferdig eksempel, mens #2 viser ett som kan være suspekt. Noen tall kommer for sjelden, andre for ofte.

Koden under skal kaste 10 000 tilfeldige kast, og så finne ut om 'terningen' er tilfeldig. Vi bestemmer at den er det hvis hvert av tallene kastes mellom 10% og 20% av det totale antall kast.

Koden skriver ut prosentandelen for hver terning. I tillegg skal den skrive ut en feilmelding for hvert av beregnede prosenter som IKKE er mellom 10 og 20%.

Hjelp

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

```
from [          ] import randint

antall_kast = 10_000
maks_prosent = 20
min_prosent = 10

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

resultater = [          ]

for i in [          ]:
    kast = [          ]
    resultater[ [          ] ] += 1

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

```
    if not (                        ):
        print(f'Feil! Kravet vår:                    oppfylles ikk
```

Maks poeng: 2.8

4.6 Drag and drop 6

```
# 1
AA11111

# 2
AAA111
```

Funksjonen `check_regnummer` skal ta i mot et regnummer (string) og returnere True eller False basert på om regnummeret følger norske standarder. To bokstaver etterfulgt av 5 tall. Altså bokstav-bokstav-tall-tall-tall-tall-tall. Skilt #1 skilt godkjennes (returnere True) mens skilt #2 skal returnere False.

Hjelp

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

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

Maks poeng: 3

5

Her kan du kladde

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

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

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

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

Useful Functions and Methods

These are listed in the following order:

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

Built-in:

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

try:

Code to test

except:

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

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

else:

  # Runs if no exception occurs

finally:

  # Runs regardless of prior code having succeeded or failed.

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

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

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

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

Deletes element k in d.

d.copy()

Makes a copy of d.

Files:

open()

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

f.read(size)

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

f.readline()

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

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

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

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

Random Library:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

math.e

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

Del 3 - Turforberedelser

Oppgavene dreier seg om forberedelser til en tur. Det er jo mange ting som må pakkes til en tur, og da er det greit å ha en oversikt over hva som er pakket. I alle fall i prinsippet!
Denne delen består av flere oppgaver som vil kunne fungere sammen.

- Husk at du gjerne kan bruke funksjoner fra tidligere oppgaver selv om du skulle ha hoppet over dem!
- Husk også at du har 'fuskelappen' tilgjengelig.

Alle 'ting' som skal pakkes er knyttet til én eller flere kategorier:

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

Personene som skal være med på turen pakker hver sine ting, som skal lagres i følgende struktur. Hver person er en subliste, med navnet først og så de ulike tingene denne personen har pakket:

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

Eksemplene i de påfølgende oppgavene bruker nettopp disse verdiene som utgangspunkt.

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

Det første vi trenger er å vite hvem som er med på turen. Skriv funksjonen *participants*.

- Funksjonen skal ta inn listen *packed*.
- Funksjonen skal returnere en liste med navnene til turens deltakere.

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

Skriv ditt svar her

```
1
```

Maks poeng: 3

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

Vi må også vite hva én av deltakerne har pakket. Skriv funksjonen has_brought.

- Funksjonen skal ta inn en *streng* med et navn, samt listen *packed*.
- Funksjonen skal returnere en *liste* med tingene som denne personen har pakket.

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

Skriv ditt svar her

```
1
```

Maks poeng: 3

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

Nå må vi vite om alle de ulike tingene som skal bli med. Skriv funksjonen unique_items.

- Funksjonen skal ta inn dictionarien *things*.
- Funksjonen skal returnere en *liste* over alle de unike tingene man kan ha med.
- Man skal kun ha tingene, ikke typen hver ting er knyttet til.
- Én ting kan være del av flere ulike typer, eksempelvis kan en tannbørste være både toalettsaker og verktøy. Alle ting skal bare være med én gang.
- Rekkefølgen på tingene som returneres er ikke viktig.

Eksempel:

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

Skriv ditt svar her

```
1
```

Maks poeng: 3

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

Det er bare lov til å pakke ting som er på listen, selvfølgelig! Derfor må vi ha en sjekk på om en ting er tillatt eller ikke. Skriv funksjonen *item_allowed*.

- Funksjonen skal ta inn en *streng* (tingen) og dictionarien *things*.
- Funksjonen skal returnere en *boolean*, *True* dersom tingen er tillatt og *Talse* hvis den ikke er det.

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

Skriv ditt svar her

```
1
```

Maks poeng: 4

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

Nå skal en person få pakket en ting. Skriv funksjonen *pack_item*.

- Funksjonen har tre parametre:
  - *Streng* (navnet på personen)
  - *Streng* (tingen som skal legges inn)
  - *Liste*: (packed)
- Funksjonen skal legge til tingen i sublisten til den riktige personen.
- Alle andre personers ting skal være uendret.
- Funksjonen skal returnere en *liste* med oppdaterte verdier.
- Dersom tingen ikke er lov til å ta med så skal tingen ikke legges til.
- NB: Funksjonen skal *ikke* endre på listen den henter inn.
- Det er lov til å ta med flere ting av samme type, man kan jo pakke to bukser.
- *things* er en global variabel.

Eksempel:

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

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

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

Skriv ditt svar her

```
1
```

Maks poeng: 5

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

For å kunne dra på tur må det være pakket minst én av hver ting. Skriv funksjonen *what_is_missing*.

- Funksjonen skal ta inn dictionarien *things* og listen *packed*.
- Funksjonen skal returnere en *liste* over tingene som mangler, altså alle ting som ikke er pakket av noen.

Eksempel:
>>> print("Mangler:",what_is_missing(things, packed))
Mangler: ['ulltrøye']

Skriv ditt svar her

```
1
```

Maks poeng: 6

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

Til slutt skal det leses inn en fil med innhold som skal legges til. Skriv funksjonen *add_all*.

- Funksjonen skal lese innholdet i filen *pakkeliste.txt*.
- Funksjonen har én parameter: *liste* (already_packed) med struktur som *packed.*
- Filen har to linjer, det er den *andre linjen* som har informasjonen.
  - Linjen er strukturet slik: *navn:ting,ting:navn:ting:navn:ting,ting,ting...*
  - Det kan jo være ulike ting og navn med, men alle navnene finnes.
  - Et eksempel på teksten på linje 2: ***Børge:jakke,ost,Ole:ost,Anna:ost,paprika***
- Ting som ikke er tillatt å ta med skal ikke legges til.
- Funksjonen skal returnere en *oppdatert liste* over alt som nå er pakket, inkludert det som var pakket fra før.
- Du kan forutsette at det ikke er noen navn som også er ting eller motsatt.

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

Skriv ditt svar her

```
1
```

Maks poeng: 7