

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. Man får ikke

Ingen deler av eksamen har minuspoeng for feil svar.

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

- ☐ 1A
- ☐ 29
- ☐ B1
- ☐ 1B



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

- ☐ 24
- ☐ 16
- ☐ 32
- ☐ 8



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

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



4) Lesing av data fra RAM til register gjøres i hvilken fase:

- ☐ Command
- ☐ Decode
- ☐ Fetch
- ☐ Execute

**5) Hvilken påstand om transistorer (sammenliknet med radorør) er IKKE korrekt:**

- ☐ Flere operasjoner på samme tid
- ☐ Høyere strømforbruk
- ☐ Mindre i størrelse
- ☐ Ødelegges ikke like lett

**6) Minnetypen som ligger mellom RAM og sekundærminnet kalles**

- ☐ Det finnes ikke noe nivå mellom RAM og sekundærminne.
- ☐ Følge
- ☐ Buffer
- ☐ Flash

**7) Hva er hovedgrunnen til at cache er tatt i bruk?**

- ☐ RAM er treigt, så den raskere cachen sitter som en buffer og lagrer de 'viktigste' dataene. Cachen er raskere enn RAM, men med mindre kapasitet.
- ☐ ALUen kan bare kjøre én beregning av gangen. IO-enhetene (mus, skjerm osv.) sender alle data samtidig. Disse må lagres i tilfelle ALUen slutter å fungere.
- ☐ Cache brukes til å filtrere instruksjonskall som er skadelige før de blir sendt opp til registeret for å utføres.
- ☐ Cachen er en form for 'turbo' som kan brukes i spesielle tilfeller der ekstra regnekapasitet trengs, men kan bare brukes i korte perioder.

8) Hva er IKKE en del av 'pipelining'?

- ☐ Minimaliserer venting av kretser i CPU
- ☐ CPUen tillater at en instruksjon (som fetch) kan leses av multiple senere instruksjoner, så en slipper å hente dem flere ganger. ✓
- ☐ Fetch, decode og execute kan kjøres i parallell.
- ☐ Mer effektiv bruk av CPU slik at flere deler av CPU kan brukes samtidig.


9) Gitt funksjonen `foo(parameter)`: Hvilken påstand om tilstanden til programmet etter at `foo(parameter)` er kjørt er korrekt:

- ☐ Endringer man gjør på parameter lagres kun på utsiden hvis man returnerer parameter eksplisitt.
- ☐ Endringer man gjør på parameter lagres på utsiden hvis parameter er en liste, selv om `foo` ikke returnerer eksplisitt. ✓
- ☐ Endringer gjort inni lagres kun hvis parameter er definert med global i koden som kaller `foo`.
- ☐ Endringer man gjør på parameter lagres på utsiden hvis man bruker 'global parameter' inni funksjonen.


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?

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


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

- ☐ Informasjonen er gjemt fra andre og potensielt uønskede brukere.
- ☐ Bruk av informasjon overvåkes, for å oppdage ureglementert bruk.
- ☐ Informasjonen er komplett og ikke korrupt. 
- ☐ Informasjonen er sann og uten feil.


12) Hvilken tredje type tiltak kan man i følge McCumber gjøre, ut over policy (retningslinjer) og technology (teknologi):

- ☐ Authorization (verifikasjon av tilgjengelighet)
- ☐ Scattering (kryptering av informasjon)
- ☐ Surveillance (overvåking)
- ☐ Education (ryggmargsrefleksen) 

13) Hvor mange forskjellige symboler kan man representere i UNICODE med 4 byte?

- ☐ 16
- ☐ 8
- ☐ 4
- ☐ 2 

14) Hvilken primærfarge ligger fargekoden #f1f1f1 tettest opp mot:

- ☐ Rød 
- ☐ Oransje
- ☐ Blå
- ☐ Magenta

15) Hva slags klokkehastighet ligger moderne datamaskiner på?

- ☐ Noen få MHz
- ☐ Noen få kHz
- ☐ Noen få THz
- ☐ Noen få GHz

**16) Hva skjer IKKE i "fetchfasen"?**

- ☐ Man lagrer neste instruksjon som skal kjøres i RAM.
- ☐ Man henter neste instruksjon fra RAM til instruksjonsregisteret.
- ☐ Man henter adresse for neste instruksjon i instruksjonsadresseregisteret.
- ☐ Man henter inn neste instruksjon som skal kjøres.

**17) Hvordan fungerer en transistor**

- ☐ Fungerer ved at mange metalledninger (brushes) lager elektriske forbindelser med en metallkule.
- ☐ Enheten omformer 220V vekselstrøm til likestrøm som kan brukes til de ulike enhetene (CPU, lydkort, grafikkort, harddisk, RAM osv.) i datamaskinen.
- ☐ Ligger som en bro mellom to plater som leder strøm, og bærer dermed med seg elektroner fra den ene platen til den andre.
- ☐ En bryter som det enten kan gå strøm gjennom eller ikke, og som man kan styre ved hjelp av strøm.

**18) Hva er formålet til ALU?**

- ☐ Det er den delen av CPUen som har ansvar for å peke riktig minneadresse.
- ☐ Det er den delen av CPUen som har ansvar for beregningene.
- ☐ Den er den delen av CPUen som har ansvar for å hente data fra minnet.
- ☐ Det er den delen av hovedkortet som sikrer at data fra IO-enhetene samles inn.



19) Hvilken av disse lagringsenhetene er IKKE en sekundærlagringsenhet?

- ☐ Minnet (RAM) i datamaskinen. ✓
- ☐ M.2 SSD
- ☐ En minnepinne
- ☐ Den VHS videokassetten som du fant på loftet til tante Ingrid som inneholdt halvdårlige opptak av Pompel og Pilt, den barnetvgreia til NRK som ga deg traumer vettu.

20: Hvilken påstand om bildekomprimering er GAL:

- ☐ Du kan oppnå ulik grad av komprimering på bekostning av bildekvalitet.
- ☐ Krymper fysisk bits i minnet (RAM) slik at det blir plass til mer. ✓
- ☐ Du kan lagre flere bilder på samme plass.
- ☐ Du kan overføre et bilde raskere mellom to enheter.

Maks poeng: 33.3

Her kan du kladde:

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 $a \leq N \leq b$.

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 $a \leq N \leq b$ for $a \leq b$ and $b \leq N \leq a$ 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 $e = 2.718281\dots$ 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 $\pi = 3.141592\dots$, to available precision.

math.e

The mathematical constant $e = 2.718281\dots$, to available precision.

2 Operatorer og typer (4%)

Anta at vi har definert tre variable som følger:

`i = 4`

`x = 3.0`

`s = 'Python'`

`t = 'thon'`

I tabellen nedenfor står en del uttrykk, hvorav noen vil kunne beregnes og få en verdi, mens andre vil gi feilmelding.

For hver rad, se på uttrykket til venstre, og kryss av hvilken datatype resultatet vil bli - eller velg ERROR dersom uttrykket vil gi feilmelding. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker.

	float	bool	int	string	ERROR
<code>i * x</code>	<input type="radio"/> <input checked="" type="checkbox"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>s - t</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> <input checked="" type="checkbox"/>
<code>i / i</code>	<input type="radio"/> <input checked="" type="checkbox"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>s[x]</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> <input checked="" type="checkbox"/>
<code>i == x</code>	<input type="radio"/>	<input type="radio"/> <input checked="" type="checkbox"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>i % 0</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> <input checked="" type="checkbox"/>
<code>s + t</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> <input checked="" type="checkbox"/>	<input type="radio"/>
<code>7 // i</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> <input checked="" type="checkbox"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 5.6

3 Fix (5%)

Koden for funksjonen **fix()** er gitt som følger, sammen med fem print-setninger hvor **fix()** kalles (**NB:** Merk #kommentarnummer til høyre for print-setningene, viktig for å svare på riktig spørsmål i tabellen under)

```
def fix(strg, cond):
    new_strg = strg[0]
    for i in range(1, len(strg)):
        if (strg[i] > new_strg[-1]) == cond:
            new_strg += strg[i]
        elif strg[i] == new_strg[-1]:
            new_strg += strg[i]
    return new_strg
```

```
print(fix('riiga', True))    #1
print(fix('rigra', True))    #2
print(fix('agir', True))     #3
print(fix('rigar', False))   #4
print(fix('agir', False))    #5
```

For hver rad i tabellen, kryss av i den kolonna som viser hva som vil bli printa av den korresponderende print-setninga. Merk at radene refererer til kommentarnummer over, og kan ha blitt stokket om i forhold til nummerrekkefølge.

Finn de som passer sammen.

	a	agir	iiga	riiga	riga	r	rr	rigar
#3	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
#4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
#2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
#1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
#5	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 5

4 files and exceptions (5%)

Figuren under viser koden for funksjonen **respons()**, samt definisjon av strengvariable **d** og **t**. Anta at tekstfila **datafil.txt** har innhold som vist i den innfelte boksen nede til høyre, og at dette er *den eneste* .txt-fila på samme katalog som programkoden vår.

```
def respons(fil, r, c):
    try:
        with open(fil) as f:
            line = f.readlines()[r].split(';')
            if c == 0:
                return line[3].strip()[int(line[c])]
            else:
                return float(line[c])
    except FileNotFoundError:
        return -1
    except:
        return -2

d = 'datafil.txt'
t = 'tallfil.txt'
```

3	;	2.5	;	3.5	;	Alta
1	;	0.25	;	1.0	;	Tana
2.0	;	1	;	1.0	;	Hamar
5	;	7	;	3	;	9; Oslo

I tabellen viser hver rad ett kall av funksjonen og kolonnene viser mulige returverdier. Alle rader skal brukes, men ikke nødvendigvis alle kolonner.

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.

	1.0	7.0	-1	0.25	5.0	't'	'a'	'o'
respons(d, 1, 2)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
respons(d, 0, 0)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
respons(d, 3, 0)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
respons(t, 0, 0)	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
respons(d, 3, 1)	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 5

5 Alle linjer teller (2%)

I denne oppgaven bygger vi opp et lite program en linje av gangen.

Hva er verdien til a?

```
a = (True and False) or (True and True)
```

Velg ett alternativ

True



False



Hva er verdien til b?

```
b = (3 == 4)
```

Velg ett alternativ

True



False



Hva er verdien til c?

```
c = not(b or a)
```

Velg ett alternativ

True



False



Hva er verdien til d?

```
d = ('x' in 'xyz')
```

Velg ett alternativ

True



False



Hva skrives ut?

```
if c and d:  
    print(1)
```

```
elif c and not d:
    print(2)
elif not c and d:
    print(3)
else:
    print(4)
```

Velg ett alternativ:

- 3 1 2 4
- ☒ ☐ ☐ ☐

Maks poeng: 2

6 ¿Que? (2%)

Hva kan denne funksjonen brukes til?

```
def f(x):
    try:
        float(x)
        return True
    except:
        return False
```

Velg ett alternativ:

- ☐ Teste om x er et tall
- ☐ Konvertere x til flyttall
- ☐ Teste om x er flyttall
- ☐ Konvertere x til bool



Maks poeng: 2

7 I mean... (2%)

Funksjonen under skal finne forskjell mellom hvert tall i ei liste, og regner så ut snittet. F.eks. om vi sender inn `[0, 1, 3, 10]` burde funksjonen returnere `3.33...` fordi:

- $1 - 0 = 1$
- $3 - 1 = 2$
- $10 - 3 = 7$

Vi tar så gjennomsnittet ved å bruke `sum` og `len`-funksjonene, og får $(1+2+7)/3 = 10/3 = 3.33...$

Men koden funker ikke! D:

Hvilken feilmelding får vi hvis vi kjører denne koden?

```
def finn_gjennomsnittlig_forskjell(liste):
    forskjeller = []
    for i in range(len(liste)):
        a = liste[i]
        b = liste[i + 1]
        forskjell = b - a
        forskjeller.append(forskjell)
    gjennomsnitt = sum(forskjeller) / len(forskjeller)
    return gjennomsnitt
```

```
finn_gjennomsnittlig_forskjell([0, 1, 3, 10])
```

Velg ett alternativ:

- ☐ TypeError: range expected at least 2 argument, got 1
- ☐ ZeroDivisionError: division by zero
- ☒ IndexError: list index out of range ✓
- ☐ NameError: function finn_gjennomsnittlig_forskjell() expected 1 argument, got 4

Maks poeng: 2

Her kan du kladde!

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 $a \leq N \leq b$.

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 $a \leq N \leq b$ for $a \leq b$ and $b \leq N \leq a$ 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 $e = 2.718281\dots$ 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 $\pi = 3.141592\dots$, to available precision.

math.e

The mathematical constant $e = 2.718281\dots$, to available precision.

8 Lottery (4%)

I et lotteri hvor brukerne skraper fram kombinasjoner av bokstaver, er reglene slik:

- 1.premie hvis det er 3 eller flere A'er: 100
- 2.premie hvis det er 3 eller flere B'er: 60
- 3.premie hvis det er 3 eller flere C'er: 20
- Ingen premie hvis det ikke er 3 eller flere av noen bokstav, eller hvis det bare er 3 eller flere av andre bokstaver enn A, B, C.

Man får kun én premie (dvs. hvis det f.eks. er 6 A'er, får man likevel bare 100), og kun den beste mulige premien (hvis det er både 3 A'er og 3 B'er, får man 100, ikke 60).

Eksempel på ønsket resultat ved kjøring av funksjonen:

- `calc_prize('ABABBCCCCA')` skal returnere 100
- `calc_prize('ABACCBCCCCC')` skal returnere 20
- `calc_prize('DDDXABDDDDXX')` skal returnere 0

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. Merk at det allerede står apostrofer inni parentesene i kallene av funksjonen `count()` og i `[]`-parentesene for prizes, slik at du selv IKKE skal skrive apostrofer der.

```
def calc_prize(string):
    prizes = {'A': 100, 'B': 60, 'C': 20}
    reward =  (0)

    if string.count('  (A) ')  (>=) 3:
        reward = prizes['  (A) ']

    elif string.count('B')  (>=) 3:
        reward = prizes['B']

     (elif string.count('  (C) ')  (>=) 3:
        reward = prizes['  (C) ']

    return  (reward)
```

Maks poeng: 4

9 Names (5%)

Gitt lister av navn, f.eks.

```
navn = ['Logan Paul', 'Ada Wong', 'Andy Warhol', 'Paul Anka',
        'Grace Hopper', 'Ada Lovelace', 'Kalle Anka', 'Andy Anka']
```

For enkelhets skyld antar vi at alle navn består av bare to ord, fornavn og etternavn, uten noe mellomnavn eller andre tillegg. Vi ønsker å lage to funksjoner:

- **name_sets(name_list)** skal få inn ei liste av navn som f.eks. ovenstående, og returnere tre mengder (set), nemlig ei mengde som inneholder fornavnene, ei med etternavnene, og ei med initialene som fantes i lista.
- **dual_use_names(first_names, last_names)** skal returnere mengden av navn som er i bruk både som fornavn og etternavn, dvs. som forekommer i begge mengdene som gis inn som parametre

Eksempel på kode for å kalle funksjonene:

```
first, last, inits = name_sets(navn)
print(first)
print(last)
print(inits)
print(dual_use_names(first, last))
```

Ønsket resultat av kjøring, hvis variabelen navn har innhold som gitt øverst:

```
{'Andy', 'Logan', 'Ada', 'Kalle', 'Grace', 'Paul'}
{'Lovelace', 'Wong', 'Paul', 'Hopper', 'Anka', 'Warhol'}
{'GH', 'LP', 'KA', 'AL', 'AA', 'AW', 'PA'}
{'Paul'}
```

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

 [Hjelp](#)

first_names

name.split()

names[1]

names[1][0]

names[0]

fi_n + la_n

intersection(last_names)

names[0][0]

union(last_names)

set(), set(), set()

```

def name_sets(name_list):
    fi_n, la_n, init = set(), set(), set()
    for name in name_list:
        names = name.split( )
        fi_n.add( names[0] )
        la_n.add( names[1] )
        init.add( names[0][0] + names[1][0] )
    return fi_n, la_n, init

```

```

def dual_use_names(first_names, last_names):
    return first_names . intersection(last_names)

```

Maks poeng: 5

10 dictionary, mutering (5%)

Vi ønsker å justere en ordbok (dictionary) hvor verdiene er lister av heltall slik at alle partall fjernes, mens oddetall blir værende igjen. Hvis ei liste som følge av fjerning av partall blir tom, skal hele den tilhørende nøkkelen fjernes fra ordboka. Vi ønsker to funksjoner for dette:

- **remove_even_nos(d)** som muterer ordboka (dictionary) og listene inni
- **even_nos_removed(d)**, som returnerer ny versjon av ordboka og listene inni, uten å endre originaldataene.

For den muterende **remove_even_nos** skal endringene skje i samme ordbok og mutere listene. For **even_nos_removed** skal det lages en ny ordbok med nye lister som verdier.

Eksempel på kjøring av funksjonene:

```
d = {'A': [3, 2, 4, 5], 'B': [5, 4, 1, 1, 8, 3], 'C': [1, 7, 7], 'D': [2, 4, 6]}
print(even_nos_removed(d)) # viser justert ordbok, endrer ikke original
print(d)                  # printer original ordbok
remove_even_nos(d)        # endrer original ordbok ved mutering
print(d)                  # printer original ordbok på nytt
```

Gjør at følgende printes:

```
{'A': [3,5], 'B': [5,1,1,3], 'C': [1, 7, 7]}
{'A': [3,2,4,5], 'B': [5,4,1,1,8,3], 'C': [1, 7, 7], 'D': [2, 4, 6]}
{'A': [3,5], 'B': [5,1,1,3], 'C': [1, 7, 7]}
```

Som vi ser forsvinner nøkkelen 'D' helt fra ordboka ved justering i dette eksemplet fordi alle tallene i lista til den nøkkelen var partall, slik at lista endte opp med å bli tom.

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](#)

for i in range(len(d[key])-1,-1,-1):

del d[key][i]

del d[key]

new_d = { }

for num in d[key]:

new_val.append(num)

new_val = []

if d[key][i] % 2 == 0:

new_d[key] = new_val

return new_d

if len(new_val) > 0:

if num % 2 != 0:

if len(d[key]) == 0:

```
def remove_even_nos(d): # Mutating the dictionary
    for key in list(d): #list(d) avoids iterating d itself while changing
```

```
for i in range(len(d[key])-1,-1,-1) ✓
```

```
if d[key][i] % 2 == 0: ✓
```

```
del d[key][i] ✓
```

```
if len(d[key]) == 0: ✓
```

```
del d[key] ✓
```

```
def even_nos_removed(d): # not changing the original
```

```
    new_d = { } ✓
```

```
    for key in d:
```

```
        new_val = [ ] ✓
```

```
        for num in d[key]: ✓
```

```
            if num % 2 != 0: ✓
```

```
                new_val.append(num) ✓
```

```
        if len(new_val) > 0: ✓
```

```
            new_d[key] = new_val ✓
```

```
    return new_d ✓
```

Maks poeng: 6.5

11 Dice game (6%)

Terningspillet Titusen spilles med seks terninger og har (litt forenklet) følgende poengregler:

- Tre spesialkombinasjoner gir ekstra mye poeng:
 - to ganger tre like, f.eks. [2,2,2,5,5,5], gir 2500 poeng
 - straight [1,2,3,4,5,6] gir 2000 poeng
 - tre par, f.eks. [1,1,4,4,6,6] gir 1500 poeng
- Hvis man ikke har noen spesialkombinasjon, gjelder følgende:
 - tre enere oppnådd i samme kast gir 1000 poeng
 - tre like av annet tall, gir tallet x 100 (f.eks. tre 6'ere, 600, tre 2'ere, 200)
 - fire like gir det dobbelte av tre like, fem like det dobbelte av fire like, og seks like igjen dobbelt av dette (f.eks. 4 enere gir 2000, 5 enere 4000, 6 enere 8000).
 - enkeltstående 1 gir 100 poeng per stykk, og enkeltstående 5 gir 50 poeng per stykk. Andre tall gir ikke noen poeng enkeltvis.

Vi ønsker en funksjon **points(throw)** som kan regne ut poengverdien av ett kast med 6 terninger, hvor inn-parameteren **throw** er ei liste av heltall som var i kastet, og hvor vi kan anta at tallene alltid kommer sortert i stigende rekkefølge. Eksempel på ønsket resultat av kjøring:

points([1,2,3,4,5,6]) gir 2000 poeng (straight)

points([1,1,1,2,2,2]) gir 2500 poeng (to tripletter)

points([1,1,1,2,5,5]) gir 1050 poeng (1100 for 3 enere + 2 x 50 for to femmere)

points([1,2,2,2,4,4]) gir 300 poeng (200 for 3 toere, pluss 100 for den ene eneren)

points([2,3,4,4,4,4]) gir 800 poeng (for 4 firere, dvs. dobbelt av poeng for tre firere)

points([1,2,3,3,4,5]) gir 150 poeng (100 for eneren, 50 for femmeren)

points([2,3,3,4,4,6]) gir 0 poeng

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 points(throw):
```

```
    if throw ==  ([1,2,3,4,5,6], set(throw), sorted(throw)):
```

```
        score = 2000 # straight
```

```
    elif len(set(throw)) == 2 and  (throw[3] < throw[4], throw[1] == throw[2],
```

```
    throw[2] != throw[3]):
```

```
        score = 2500 # two triplets
```

```
    elif len(set(throw)) == 3 and  (throw[0] == throw[1], throw[0::2] ==
```

```
    throw[1::2], throw[0,2,4] == throw[1,3,5]):
```

```
        score = 1500 # three pairs
```

```
    else: # other cases
```


Velg alternativ (score = 0, throw.sort(), result = [])

for n in set(throw):

ct = throw.count(n)

Velg alternativ (if ct >= 3, if ct > 3, if ct == 3):

multi = Velg alternativ (2 ** (ct-3), 4 - ct, (4 - ct) ** 2)

if n == 1:

score += Velg alternativ (1000 * multi + 50 * ct, 1000 * multi, (1000 + 50 *
ct) * multi)

else:

score += Velg alternativ ((n - 1) * multi, n * multi, n*100 * multi)

Velg alternativ (elif ct < 3:, elif ct <= 3:, elif n == 1:)

score += ct * 100

Velg alternativ (else:, elif n > 1:, elif n == 5:)

score += ct * 50

return score

Maks poeng: 9

12 Long division (5%)

Lærer "Leif" ønsker å lage et program som kan illustrere manuelt utført divisjon for elevene. Som del av dette programmet ønskes en funksjon **show_division(n, m)** som skal returnere en streng over flere linjer som viser hvordan man går fram for å dele n på m. Eksempel på kjøring:

show_division(127, 4) skal returnere en streng på 7 linjer som vist under. Linje 2 viser -12 fordi første siffer i løsningen er 3, og 3×4 (divisoren) er 12. Linje 4 viser 7 fordi det er det man har igjen av dividenden når man trekker fra $12 - 12$. Linje 5 viser -4 fordi neste siffer i løsningen er 1, som multiplisert med divisoren gir 4. Til slutt får vi 3 i rest på linja nederst.

```
127 : 4 = 31
-12
----
  7
 -4
 ----
  3 i rest
```

Hvis divisjonen går opp, f.eks. **show_division(124, 4)** skal det stå **0 i rest** på siste linje av resultatstrengen. Merk at det ikke nødvendigvis blir 7 linjer i strengen som returneres. Hvis svaret bare blir ensifret, vil resultatet ha færre linjer, og hvis det blir mer enn tosifret, flere linjer.

OPPGAVE: Velg rett alternativ i hvert tomrom slik at funksjonen show_division() virker som den skal. Ingen minuspoeng for feil valg, så du BØR svare noe også der du er usikker. Funksjonen len_diff() er en hjelpefunksjon som brukes for å finne differansen i lengde mellom to tall (i antall siffer). Tekststrengen '\n' betyr linjeskift.

```
def len_diff(num1, num2):
    return len(str(num1)) - len(str(num2))
```

```
def show_division(n, m):
```

Velg alternativ (result = n / m, result = n // m, **result = str(n // m)**)

```
division_string = f' {n} : {m} = {result}'
```

```
indent = " # to enkle apostrofer helt inntil hverandre, dvs tom streng
```

```
num = n
```

```
for digit in result:
```

Velg alternativ (subtr = n - digit, **subtr = int(digit) * m**, subtr = digit * m)

```
division_string += Velg alternativ (f'\n{indent}-{subtr}', f'{indent}-{subtr}\n', f'{m*indent}-
{subtr}\n')
```

```
division_string += Velg alternativ (f'\n{indent}----', f'{indent}----\n', f'{m*indent}----\n')
```

Velg alternativ (new_num = num - subtr * 10 **, new_num = subtr - num * 10 *,

new_num = num - subtr * 10 *) len_diff(num, subtr)

indent += ' ' * Velg alternativ (len_diff(num, subtr), len_diff(num, new_num),

len_diff(subtr, num)) # en blank mellom apostrofene

Velg alternativ (num = new_num, new_num = num - subtr, num = new_num - subtr)

division_string += Velg alternativ (f'\n {indent}{new_num-subtr}', f' {indent}

{new_num}\n', f'\n {indent}{num}')

division_string += ' i rest\n'

return division_string

Maks poeng: 7.5

Datastruktur:

```

"""Module providing solution to ITGK s2023."""
user_data = {
    'Børge': [
        ['e', 23, 26, 22, 18],
        ['f', 27, 19, 33, 28, 25, 21],
        ['h', 25, 20],
        ['g', 22, 27, 23, 26, 30],
        ['i', 30, 18, 21, 29, 24, 19, 27],
    ],
    'Sara': [
        ['e', 21, 18],
        ['k', 31, 24, 20, 22, 26],
        ['h', 19, 27, 28, 23],
        ['r', 26, 30, 17],
        ['i', 28, 31, 16, 24, 25, 29],
    ],
    'Lars': [
        ['e', 28, 30, 32, 25, 27, 29],
        ['f', 23, 26, 22, 32, 21, 18, 19],
    ],
}

```

Her kan du kladde

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 ($\backslash n$) 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 $a \leq N \leq b$.

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 $a \leq N \leq b$ for $a \leq b$ and $b \leq N \leq a$ 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 $e = 2.718281\dots$ 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 $\pi = 3.141592\dots$, to available precision.

math.e

The mathematical constant $e = 2.718281\dots$, to available precision.

13 Counting words (5%)

Vi ønsker en funksjon **count_start_end_words(word_list, letter)**

Funksjonen har to parametre, **word_list** som er ei liste av tekststrenger, og **letter**, som er en enkelt bokstav. Funksjonen skal returnere antall strenger i **word_list** som er slik at de både begynner og slutter på bokstaven **letter**.

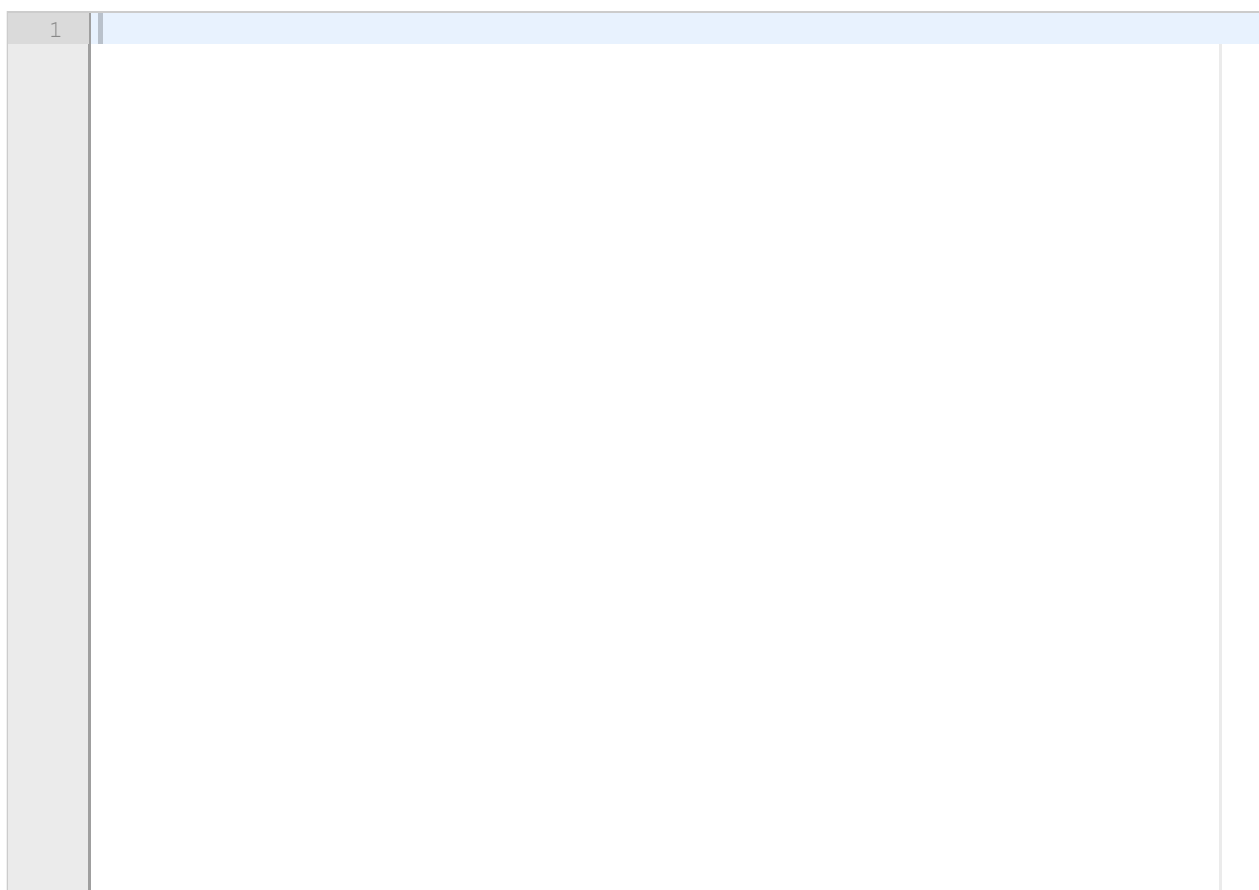
Eksempel på ønsket virkemåte av funksjonen, hvis vi har

word_list = ["ada", "ida", "alta", "ana", "y"]

så skal

- **count_start_end_words(word_list, "a")** returnere 3, fordi 3 strenger, "ada", "alta", "ana" har "a" i begge ender.
- **count_start_end_words(wordlist, "i")** returnere 0 (ingen streng har "i" i begge ender)
- **count_start_end_words(wordlist, "y")** returnere 1 (dvs. en streng på bare 1 bokstav skal telle med, gitt at bokstaven er riktig)

Skriv koden for funksjonen count_start_end_words()



Maks poeng: 5

14 letter_value (5%)

Hver ansatt har sine verdier i en todimensjonal liste, som vist over. Din første funksjon er å lage *letter_value*. Funksjonen tar inn en brukers totale todimensjonale liste *letter_list*, samt en bokstav *letter*. Funksjonen skal returnere verdiene som er lagret for den gitte bokstaven. Hvis bokstaven ikke finnes så skal funksjonen returnere en tom liste.

Eksempel:

```
>>> letter_list = [['e', 28, 30, 32, 25, 27, 29],  
                  ['f', 23, 26, 22, 32, 21, 18, 19]]  
>>> print(letter_value(letter_list, 'e'))  
[28, 30, 32, 25, 27, 29]
```

Skriv ditt svar her

1	
---	--

Maks poeng: 5

15 add_letter_test (5%)

Legg til forsøk

Funksjonen `add_letter_test` brukes til å registrere nye bokstavøvinger for en ansatt - vi bygger altså her opp den todimensjonale listen som ligger som verdi i `user_data`. Funksjonen tar inn en allerede eksisterende liste `letter_list` og utvider den med et nytt innslag for bokstaven `letter` og verdien `score`. Hvis bokstaven er registrert tidligere skal `score` legges til som en ny verdi til slutt i dennes liste. Funksjonen skal returnere den oppdaterte listen.

```
```python
letter_list = []
>>> print(add_letter_test(letter_list, 'a', 23))
[['a', 23]]
>>> print(add_letter_test(letter_list, 'a', 24))
[['a', 23, 24]]
>>> print(add_letter_test(letter_list, 'b', 19))
[['a', 23, 24], ['b', 19]]
```
```

Skriv ditt svar her

| | |
|---|--|
| 1 | |
|---|--|

Maks poeng: 5

16 The winner fakes it all: get_winner (8%)

Det er to måter å være 'best' i denne konkurransen. Man kan ha øvet flest ganger på en gitt bokstav, eller du har oppnådd den korteste tiden (det laveste tallet). Funksjonen `get_winners` tar inn tre parametre: `user_data` inneholder den kjente datastrukturen, så en streng som er enten `"quickest"` eller `"most"` - og aldri noe annet - og til slutt en bokstav.

- Hvis parameter 2 er `"quickest"` skal den returnere en streng for alle som har den raskeste verdien registrert for bokstaven i den siste parameteren.

- Husk at ikke alle personer har øvd på alle bokstaver!

- Hvis parameter 2 er `"most"` skal funksjonen returnere en streng for alle som har øvd flest ganger på bokstaven i den siste parameteren.

Følgende eksempel viser teksten som returneres, basert på verdiene i datastrukturen brukt tidligere:

```
>>> print(get_winners(user_data, "quickest", "e"))
Raskest for e er Børge, Sara med 18.
>>> print(get_winners(user_data, "quickest", "f"))
Raskest for f er Lars med 18.
>>> print(get_winners(user_data, "most", "e"))
Ivrigst for e er Lars med 6 forsøk.
>>> print(get_winners(user_data, "most", "f"))
Ivrigst for f er Lars med 7 forsøk.
```

Skriv ditt svar her

| | |
|---|--|
| 1 | |
|---|--|

Maks poeng: 8

17 Fil me in: store_data (7%)

Skriv funksjonen `store_data(user_data)`. Funksjonen tar inn en *dictionary* med struktur som beskrevet tidligere.

Funksjonen `store_data` lagrer denne til fil som skal hete **touch.txt** i samme katalog.

Funksjonen skal skrive ut dataene til hver enkelt deltaker, samt hvem som er raskest for hver av bokstavene som er øvd på. Bokstavene trenger ikke skrives ut i alfabetisk rekkefølge.

Spesifikasjon og eksempel på resultat av funksjonskall finner du under.

Skriv ditt svar her

Merk at alle seksjoner for en ansatt sine verdier alltid skal avsluttes med en linje med '---':

```

navn1
bokstav1:verdi1,verdi2
bokstav2<...>
---
navn2
<...>

```

Eksempel

Tekstversjonen av `user_data` vil se slik ut:

```

Børge
e:23,26,22,18
f:27,19,33,28,25,21

```

h:25,20

g:22,27,23,26,30

i:30,18,21,29,24,19,27

Sara

e:21,18

k:31,24,20,22,26

h:19,27,28,23

r:26,30,17

i:28,31,16,24,25,29

Lars

e:28,30,32,25,27,29

f:23,26,22,32,21,18,19

Raskest for r er Sara med 17.

Raskest for f er Lars med 18.

Raskest for k er Sara med 20.

Raskest for g er Børge med 22.

Raskest for h er Sara med 19.

Raskest for i er Sara med 16.

Raskest for e er Børge, Sara med 18.

Maks poeng: 7