Eksamen er delt i tre:

- Del 1: teoridel som teller 25% av den totale vurderingen
- Del 2: ulike små programmeringsoppgaver. Disse er helt uavhengige av hverandre, og teller til sammen 25%
- Del 3: denne består av et sett med oppgaver som hører sammen i en oppgave om valg. Denne teller 50% .

Hvordan skal du best programmere på ITGK-eksamen?

1. Du kan godt skrive kode rett inn i oppgavene, hvis du ønsker. Du vil derimot ikke kunne få kjørt koden din. Vi anbefaler ikke dette - når du tross alt har mulighet bør man bruke en ordentlig editor.
2. Du kan se på oppgaven i Inspera, og så løse oppgavene i en ekstern editor. Du vil da kunne kjøre koden din, for å teste om den er korrekt etc. Du må da skrive testene dine selv.
3. Hvis du jobber utenfor Inspera, så kan det være lurt å kopiere inn løsningene etterhvert. Skulle noe skje med datamaskinen din, da har du levert så mye som mulig.
4. Når du kommer til oppgave 3, da kan du hente ned to filer, *votes.txt* og *valg2020.py*. Du finner dem i denne zipfilen *valg2020.zip*. Pakk dei ut i same foldar, og opne valg2020.py i ein valfri editor som Thonny. Du slepp å skriva ein god del sjølv, og kan konsentrera deg om å løysa oppgåvene. valg2020.py inneheld testkode til oppgåve 3, votes.txt er fila som skal lesast frå. **Du skal IKKE laste opp noen filer - alle funksjonene skal skrives inn i sine respektive deloppgaver. Det er heller ikke utvidet eksamenstid for å legge kode inn i Inspera.**

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

Fra oss som har undervist i TDT4110 - takk for at dere har tatt en merkelig høst på strak arm - og gjør oss ordentlig stolte nå. Tvi-tvi!

- Özlem, Terje, Alf Inge og Børge

Regler og samtykker

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

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

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

Under Eksamen:

Jeg skal IKKE motta hjelp fra andre.

- Aksepter

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

- Aksepter

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

- Aksepter

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

- Aksepter

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

- Aksepter

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

- Aksepter
- Ikke bruk resultatene mine til forskning.

Forside

Institutt for datateknologi og informatikk

Eksamensoppgave i TDT4110 -

Informasjonsteknologi, grunnkurs

Faglige kontakter under eksamen:

- Terje Rydland (tlf.: 957 73 463)
- Børge Haugset (tlf.: 934 20 190, kan også kontaktes via Teams)
- Özlem Özgöbek (tlf: 735 94 427, kan også kontaktes via Teams)

Eksamensdato: 1.desember 2020
Eksamenstid (fra-til): 09:00 – 13:00
Hjelpemiddelkode/Tillatte hjelpemidler: A / Alle hjelpemidler tillatt
Teknisk hjelp under eksamen: NTNU Orakel **Tlf:** 73 59 16 00

Annen informasjon:
Det er angitt i prosent hvor mye hver deloppgave i eksamenssettet teller ved sensur. Hele oppgavesettet gir totalt 240 poeng som tilsvarer 100%. Les gjennom hele oppgavesettet før du begynner å løse oppgaven. Disponer tiden godt!

Merk! Studenter finner sensur i Studentweb. Har du spørsmål om din sensur må du kontakte instituttet ditt. Eksamenskontoret vil ikke kunne svare på slike spørsmål.

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

Annen viktig informasjon

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

Juks/plagiat: Eksamen skal være et individuelt, selvstendig arbeid. Det er tillatt å bruke hjelpemidler, men vær obs på at du må følge eventuelle anvisningen om kildehenvisninger under. Under eksamen er det ikke tillatt å kommunisere med andre personer om oppgaven eller å distribuere utkast til svar. Slik kommunikasjon er å anse som juks.
Juks på eksamen kan få følgende konsekvenser:

- annullering av gjeldende eksamen eller prøve eller godkjenning av kurs, jf. uhl §4-7
- utestenging fra NTNU og fratakelse av retten til å gå opp til eksamen ved andre institusjoner under universitets og høyskoleloven inntil ett år, jf. uhl §4-8

Når det fattes vedtak om utestenging, er dette en reaksjon som kommer i tillegg til annullering.
Du kan lese mer om juks og plagiering på eksamen her.

Kildehenvisninger: Det skal henvises til kilder som brukes utover lærebok og slides bruk i faget. Kilder oppgis med på formen Forfatter/Institusjon, "Tittel på side eller dokument", Lenke til dokument, oppdatert dato, årstall. Eks:
Wikipeda, "Binært tallsystem", https://no.wikipedia.org/wiki/Binært_tallsystem, Oppdatert 2.august 2018.

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

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

Trekk fra eksamen: Ønsker du å levere blankt/trekke deg, gå til hamburgermenyen i øvre høyre hjørne og

velg «Lever blankt». Dette kan ikke angres selv om prøven fremdeles er åpen.

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

i Oppgave 1 (25%)

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

1 Oppgave 1.1 (5%)

Forklar kort, komplett og konsist sammenhengen mellom et høynivå programmeringsspråk som f.eks. Python og maskinkode.

Skriv ditt svar her

Format

Words: 0/250

Maks poeng: 10

2

Oppgave 1.2 (5%)

Forklar kort, komplett og konsist hvilke begrensninger antall bit et tall er representert med medfører (forklar med bruk av eksempler):

Skriv ditt svar her

Format

Words: 0/250

Maks poeng: 10

3

Oppgave 1.3 (5%)

Sammenlign kort, komplett og konsist de fire lokalnettverktopologiene buss, ring, mesh og stjerne:

Skriv ditt svar her

Format

Words: 0/250

Maks poeng: 10

4

Oppgave 1.4 (5%)

Forklar kort, komplett og konsist hvilke mekanismer i TCP som sikrer overføring av data over internett:

Skriv ditt svar her

Format

Words: 0/250

Maks poeng: 10

5

Oppgave 1.5 (5%)

Sammenlign kort, komplett og konsist sikkerhetsangrepene wiretapping, replay, buffer overflow og portskanning:

Skriv ditt svar her

Format

Words: 0/250

Maks poeng: 10

Useful Functions and Methods

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

Built-in:

format(numeric_value, format_specifier)

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

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

try:

Code to test

except:

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

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

else:

    # Runs if no exception occurs

finally:

    # Runs regardless of prior code having succeeded or failed.

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

Random methods:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

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

s.reverse()

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

s.sort()

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

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

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

s.issuperset(t)

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

s.union(t)

New set with elements from both *s* and *t*

s.intersection(t)

New set with elements common to *s* and *t*

s.difference(t)

New set with elements in *s* but not in *t*

s.symmetric_difference(t)

New set with elements in either *s* or *t* but not both

s.copy()

New set with a shallow copy of *s*

s.update(t)

Return set *s* with elements added from *t*

s.add(x)

Add element *x* to set *s.* Works in_place. Returns None

s.remove(x)

Remove x from set s; raises *KeyError* if not present. Works in_place. Returns None

s.clear()

Remove all elements from set *s.* Works in_place. Returns None

Dictionary operations:

d.clear()

Clears the contents of a dictionary

d.get(key, default)

Gets the value associated with a specific key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.

d.items()

Returns all the keys in a dictionary and their associated values as a sequence of tuples.

d.keys()

Returns all the keys in a dictionary as a sequence of tuples.

d.pop(key, default)

Returns the value associated with a specific key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.

d.popitem()

Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.

d.values()

Returns all the values in dictionary as a sequence of tuples.

del d[k]

Deletes element k in d.

d.copy()

Makes a copy of d.

Files:

open()

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

f.read(size)

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

f.readline()

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

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

f.tell()

Return the position of the file pointer.

f.close()

Close the file and free up any system resources taken up by the open file.

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

math.e

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

numpy Library:

numpy.array(list)

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

ndarray.ndim

the number of axes (dimensions) of the array.

ndarray.shape

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

ndarray.size

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

ndarray.dtype

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

numpy.zeros(tuple)

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

numpy.ones(tuple)

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

numpy.arange(start,stop,step)

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

numpy.ndarray.reshape(lines,rows)

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

Oppgave 2 (25%)

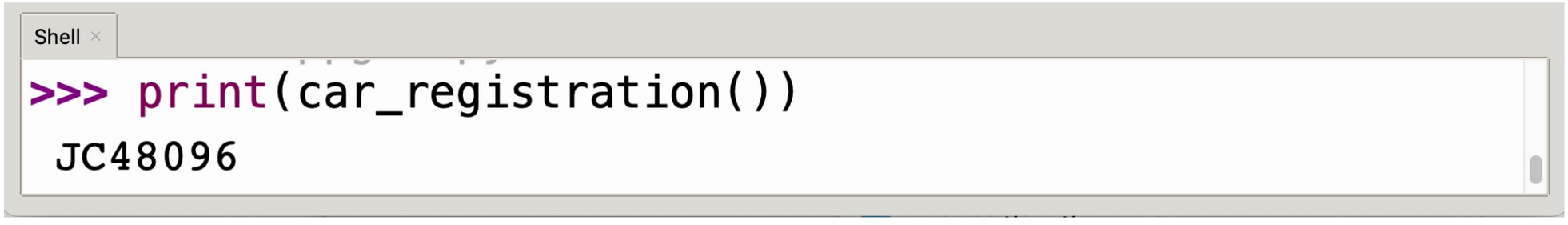
Denne oppgaven består av ulike programmeringsoppgaver du skal løse. Oppgaven tilsvarer 25% av karakteren.

Husk at du også kan få poeng på løsninger selv om de ikke fungerer ved kjøring.

6

Oppgave 2.1 (5%)

Lag en funksjon **car_registration** som genererer og returnerer et tilfeldig bilnummer.
Bilnummeret skal bestå av 2 store bokstaver og 5 siffer.
Bokstavene Æ,Ø og Å skal ikke inngå. 0 kan ikke være det første av de 5 sifrene
Eksempel på kjøring:



Skriv ditt svar her

```
1
```

Maks poeng: 10

7

Oppgave 2.2 (5%)

Lag en funksjon **prepare** som tar inn et binært tall og forbereder det til å bli konvertert til oktalt. Funksjonen skal sjekke at det som skrives på tastaturet faktisk representerer et binært tall.

Deretter skal funksjonen legge til 0-er i begynnelsen av tallet slik at det består av nok siffer til å kunne konverteres til oktalt (der hvert siffer representeres av 3 binære siffer - eks 1010011 skal bli til 001010011).

Eksempel på kjøring:

Shell

```
>>> print(prepare('10201'))
 Det er ikke et binært tall.
>>> print(prepare('10101'))
 010101
>>> print(prepare('1101'))
 001101
```

Skriv ditt svar her

```
1
```

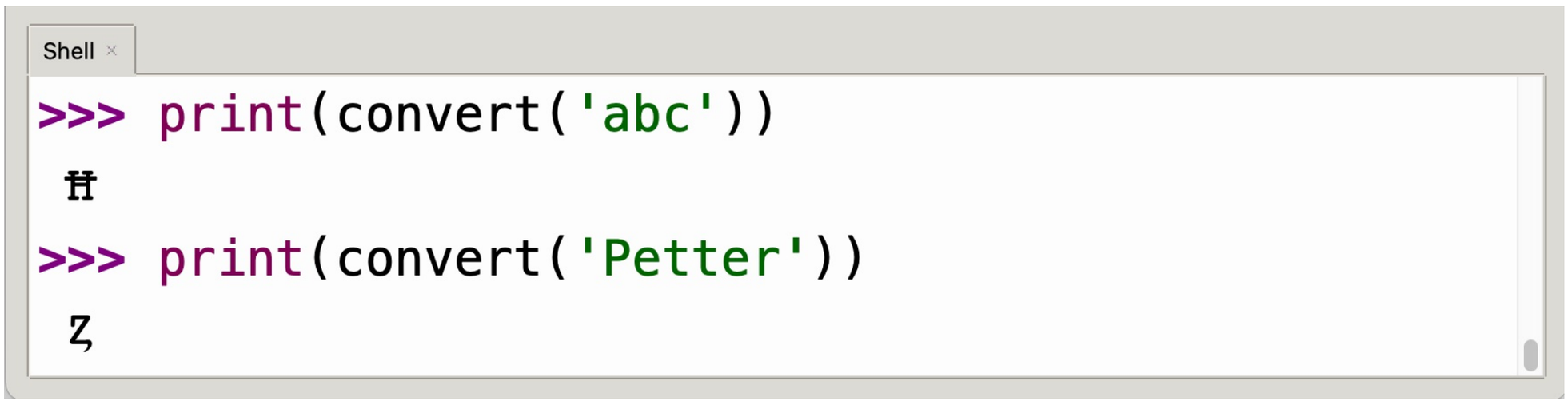
Maks poeng: 10

8

Oppgave 2.3 (5%)

Lag funksjonen **convert** som tar inn en tekststreng, konverterer tegnene til sine tilsvarende tallverdier *(vi har gjerne kalt dette å finne ASCII-verdien, men det virker for alle tegn i UNICODE)*, summerer disse sammen og så returnerer UNICODE-tegnet som tilsvarer tallet du er kommet fram til. Hvis man skriver inn mer enn 5 tegn skal funksjonen bare bruke de 5 siste.

Eksempel på kjøring:



Skriv ditt svar her

```
1
```

Maks poeng: 10

9

Oppgave 2.4 (5%)

Skriv funksjonen **add_matrix** som legger sammen 2 matriser/2D-lister og returnerer resultatet. Du kan forutsette at alle 'underlister' er like lange, a og b har samme dimensjoner.

Eksempel på kjøring:

```
Shell
>>> a = [[1,2,3,4,5],[3,4,5,6,7]]
>>> b = [[2,4,6,8,9],[6,8,10,12,13]]
>>> print(add_matrix(a,b))
 [[3, 6, 9, 12, 14], [9, 12, 15, 18, 20]]
```

eller:

```
Shell
>>> a = [[1,2,3,4,5],[3,4,5,6,7]]
>>> b = [[2,4,6,8,9],[6,8,10,12,13]]
>>> print(add_matrix(a,b))
 [[ 3  6  9 12 14]
  [ 9 12 15 18 20]]
```

Skriv ditt svar her

```
1
```

Maks poeng: 10

10

Oppgave 2.5 (5%)

Du har data lagret som en 2D-liste der hvert element er [band,tittel]

Disse dataene skal lagres til disk som en tekstfil. Skriv funksjonen list_to_disk() som tar 2D-listen og et filnavn som input og lagrer dataene på den oppgitte filen på følgende format:
band;låttittel

Et eksempel:

Etter at 2D-listen [['Beatles','And I love her'],['Beatles','All my loving'],
['Traffic','John Barleycorn must die'],['Cream','Sunshine of my love'],['Cream','SLWABR']]
er lagret på disk skal filen ha følgende innhold:

Beatles;And I love her
Beatles;All my loving
Traffic;John Barleycorn must die
Cream;Sunshine of my love
Cream;SLWABR

Skriv ditt svar her

```
1
```

Maks poeng: 10

Useful Functions and Methods

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

Built-in:

format(numeric_value, format_specifier)

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

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

try:

Code to test

except:

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

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

else:

    # Runs if no exception occurs

finally:

    # Runs regardless of prior code having succeeded or failed.

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

Random methods:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

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

s.reverse()

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

s.sort()

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

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

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

s.issuperset(t)

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

s.union(t)

New set with elements from both *s* and *t*

s.intersection(t)

New set with elements common to *s* and *t*

s.difference(t)

New set with elements in *s* but not in *t*

s.symmetric_difference(t)

New set with elements in either *s* or *t* but not both

s.copy()

New set with a shallow copy of *s*

s.update(t)

Return set *s* with elements added from *t*

s.add(x)

Add element *x* to set *s.* Works in_place. Returns None

s.remove(x)

Remove x from set s; raises *KeyError* if not present. Works in_place. Returns None

s.clear()

Remove all elements from set *s.* Works in_place. Returns None

Dictionary operations:

d.clear()

Clears the contents of a dictionary

d.get(key, default)

Gets the value associated with a specific key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.

d.items()

Returns all the keys in a dictionary and their associated values as a sequence of tuples.

d.keys()

Returns all the keys in a dictionary as a sequence of tuples.

d.pop(key, default)

Returns the value associated with a specific key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.

d.popitem()

Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.

d.values()

Returns all the values in dictionary as a sequence of tuples.

del d[k]

Deletes element k in d.

d.copy()

Makes a copy of d.

Files:

open()

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

f.read(size)

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

f.readline()

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

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

f.tell()

Return the position of the file pointer.

f.close()

Close the file and free up any system resources taken up by the open file.

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

math.e

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

numpy Library:

numpy.array(list)

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

ndarray.ndim

the number of axes (dimensions) of the array.

ndarray.shape

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

ndarray.size

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

ndarray.dtype

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

numpy.zeros(tuple)

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

numpy.ones(tuple)

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

numpy.arange(start,stop,step)

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

numpy.ndarray.reshape(lines,rows)

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

Oppgave 3 - valgberegning

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

- Last ned filene *votes.txt* og *valg2020.py* til din egen datamaskin. Du finner disse lenkene og en oppskrift på første side av eksamen, på venstre side.
- Du kan alternativt åpne filene, og kopiere innholder inn i filer på din datamaskin.
- Pass på at de to filene ligger i samme folder
- Løs koden i en egnet editor, eksempelvis Thonny
- Lim de ferdige funksjonene inn i Inspera. Gjør det gjerne underveis, i tilfelle noe skulle skje med datamaskinen din.

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

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

Beskrivelse av oppgave 3:
Du har blitt bedt av en mediebedrift om å lage et dataprogram som kan tydeliggjøre ulike effekter av det amerikanske valgsystemet. Systemet ditt skal lese inn valgresultater i fra et sett med stater i USA, de såkalte vippestatene. De vil at programmet ditt skal kunne: (Disse vil forklares i mer detalj senere, dette er kun ment å være en introduksjon).

- Lese inn valgresultater fra et sett med stater fra en fil.
- Lagring av disse resultatene i en passende struktur
- Beregning av det totale antall stemmer hvert av partiene har i opptellingen.
- Beregning av prosentandel stemmer for hvert parti
- Beregning av valgmenn, med de to metodene 'winner-takes-all' eller basert på rettferdig prosentfordeling. Forklaring kommer.

Vi bruker i denne oppgaven helt fiktive stemmeverdier.

Den endelige datastrukturen er en dictionary kalt '*votes*', der nøkkelen er statenes navn. Verdien er en liste. Denne inneholder et sett med *tupler*, der den første verdien er demokratstemmer, mens den andre er republikanerstemmer.

Eksempel:
votes = {'Pennsylvania': [(55, 106), (68, 103)], 'Nevada': [(16, 60), (87, 127), (36, 89)]}

I dette eksempelet har vi to stater. I Pennsylvania har det blitt talt opp stemmer to ganger, i Nevada tre. Ved første telling i Nevada fikk demokratene 16 stemmer, republikanerne fikk 60. Totalt fikk demokratene i Nevada 139 stemmer, som en får ved å legge sammen tallene 16, 87 og 36.

I hver stat blir det fordelt valgmenn. Antallet valgmenn varierer mellom statene. Dette forklares i oppgaven det gjelder.

11

Oppgave 3.1 (5%)

Fordelingen mellom stemmer til demokrater og republikanere er gjerne oppgitt i antall stemmer de har fått hver. Denne vil vi kunne uttrykke som prosenttandel for begge partier.

Skriv funksjonen *calculate_percentage()*. Funksjonens parameter er et *tuppel*, som inneholder antallet stemmer demokratene har fått, etterfulgt av antallet stemmer republikanerne har fått. Funksjonen skal returnere et *tuppel* som inneholder antallet prosent stemmer som henholdsvis demokratene og republikanerne har fått. Prosentandelene skal returneres med maksimalt *to desimalers nøyaktighet*.

Eksempel:

```
> print(calculate_percentage(139, 276))
(33.49, 66.51)
```

Skriv svaret ditt her...

```
1
```

Maks poeng: 10

12

Oppgave 3.2 (5%)

I hver stat må en kunne telle opp hvor mange stemmer hvert enkelt parti har fått. Disse tallene er lagret som en *todimensjonal liste* (snarere en liste som inneholder tupler). Vi tar i beskrivelsen utgangspunkt i tallene for staten Nevada, og listen til Nevada er: [(16, 60), (87, 127), (36, 89)]

Lag funksjonen return_total_votes(). Den skal summere antallet stemmer demokratene fikk (det første tallet i hvert tuppel) og republikanerne (det siste tallet i hvert tuppel). Disse tallene skal returneres som et tuppel: (demokratstemmer, republikanerstemmer).

Eksempel:

> nevada_votes = [(16, 60), (87, 127), (36, 89)]

> print(return_total_votes(nevada_votes))

(139, 276)

Skriv ditt svar her

```
1
```

Maks poeng: 10

13 Oppgave 3.3 (5%)

Opptelling av stemmer tar lang tid. Først telles stemmene fra de som møter opp i valglokalene på valgdagen. Deretter telles stemmene for de som forhåndsstemte på ymse måter. Det rapporteres med jevne mellomrom hvor mange nye stemmer hvert parti har fått i hver stat. Når disse resultatene ankommer må *votes - din dictionary* - oppdateres for hver nye telling.

Skriv funksjonen update_state. Denne skal ha fire parametre:

- *dict: (din votes-dictionary)*
- *state: navnet på staten, som 'Nevada'*
- *demvotes: antall demokratstemmer (et heltall)*
- *repvotes: antall republikanerstemmer (et heltall)*

Måten votes-dictionaryen skal oppdateres på er at et tuppel (demvotes, repvotes) skal legges til på slutten av listen for dict sin verdi for "state". Funksjonen skal så returnere denne oppdaterte votes-dictionaryen.

NB: Hvis state ikke finnes fra før (første gang denne staten leses inn) så skal det opprettes et nytt innslag i votes-dictionaryen for denne. Den skal altså håndteres.

Eksempel:

> votes = {'Pennsylvania': [(55, 106), (68, 103)], 'Nevada': [(16, 60), (87, 127)]}

> votes = update_state(votes, 'Nevada', 36, 89)

> print(votes)

{'Pennsylvania': [(55, 106), (68, 103)], 'Nevada': [(16, 60), (87, 127), (36, 89)]}

Skriv ditt svar her

```
1
```

Maks poeng: 10

14 Oppgave 3.4 (10%)

Alle valgresultater programmet ditt skal bruke befinner seg i en fil som heter "*votes.txt*", som ligger i samme folder som programmet ditt. Denne kan lastes ned ifra Inspera, legg den i samme folder som du legger det kjørende programmet ditt. Du kan forutsette at alt av data ligger i denne filen, det er kun snakk om å lese igjennom hele filen én gang. Du kan også forutsette at filen eksisterer.

Tekstfilen leses linje for linje. Linjene har følgende struktur:
Arizona,66,103
Nevada,86,66
<stat>,<demokratstemmer>,<republikanerstemmer>

Hver stat rapporterer stemmer flere ganger. Det betyr at en altså kan få flere linjer med samme stat. Etter at filen er ferdig lest skal funksjonen returnere en dictionary med statsnavnet er nøkkel, og hver avlesing av data fra denne staten blir et tuppel i en todimensjonal liste. (slik den brukes som parameter i oppgave b)

Hvis funksjonen under lesing av linjene ikke klarer å tolke linjen korrekt, så skal den skrive ut en *feilmelding* som også bør inkludere innholdet av linjen. Funksjonen skal derimot *fortsette å lese* igjennom de restende linjene, og legge disse inn i samlingen. Til slutt skal funksjonen *returnere dictionaryen* med innholdet fra alle filene.

Skriv funksjonen read_from_file(). Den skal oppføre seg som beskrevet over.

Example (with one line where republican numbers are lacking):
> votes = read_from_file()
Linje kunne ikke tolkes: Nevada, 57
> print(votes)
{'Arizona': [(66, 103), (26, 145), ... # and more data follows, from each state.
Skriv ditt svar her

```
1
```

Maks poeng: 20

15

Oppgave 3.5 (5%)

I USA er det ikke antallet stemmer som bestemmer hvem som blir den neste presidenten, men antallet valgmenn som stemmer for en demokratisk eller republikansk president. Det velges et visst antall valgmenn fra hver enkelt stat. Antallet valgmenn varierer mellom statene.

Antallet valgmenn per stat finner du i den *todimensjonale listen electoral_votes*. Hvert element i listen inneholder først navnet på staten, og så antallet valgmenn denne staten har.

Skriv funksjonen get_ev_for_state(state). Denne skal ta inn navnet på en stat, og *returnere* hvor mange valgmenn det er i denne staten.

Du kan forutsette at listen electoral_votes finnes i systemet, så denne kan du bare referere til. Du kan også forutsette at det ikke skjer noen feil, som at navnet på staten ikke eksisterer.

Eksempel, som inkluderer definisjon av electoral_votes:

```
> electoral_votes = [["Arizona", 11], ["Nevada", 6], ["Pennsylvania", 20], ["Georgia", 16]]
> print(get_ev_for_state('Nevada'))
6
> print(get_ev_for_state('Georgia'))
16
```

Skriv ditt svar her

```
1
```

Maks poeng: 10

16

Oppgave 3.6 (5%)

I de fleste statene så er det det partiet som får flest stemmer som stikker av med alle valgmennene for denne staten. Det betyr at selv om republikanerne får 50.1% av stemmene for Nevada, så får de alle 6 valgmennene.

Skriv funksjonen get_actual_ev(state, dempercent, reppercent).
Denne tar inn *state* (navnet på en stat), og så *prosentandelene* til henholdsvis demokratene og republikanerene. Den skal så *returnere* hvor mange valgmenn hvert av partiene har fått (tuppel med begge partier sine valgmenn, som tidligere). Du kan forutsette at ikke begge partiene har fått akkurat like mange prosent stemmer.

Eksempel:
>print(get_actual_ev('Nevada', 25, 75))
(0, 6)
Skriv ditt svar her

```
1
```

Maks poeng: 10

17

Oppgave 3.7 (5%)

I et par stater bruker de et annet system, der partiene får valgmenn basert på hvor mange prosent stemmer de har fått. Dette likner på slik det er i Norge.

Skriv funksjonen get_actual_ev_fair. Den skal ta inn de samme *parametrene* som i forrige oppgave. Forskjellen er at de to partiene skal få fordelt valgmenn *etter prosent* stemmer i staten. Funksjonen skal *returnere* et tuppel med antallet valgmenn til henholdsvis demokratene og republikanerne.

PS:

Vi kommer *ikke* til å se på hvor perfekt løsningen er, som i om en skal runde av den ene eller andre veien. Det viktige er at du får vist at du forstår grunnprinsippet.

Example:

```
> print get_actual_ev_fair("Nevada", 50.1, 49.9)
(3, 3)
> print get_actual_ev_fair("Georgia", 80, 20)
(13, 3)
```

Skriv ditt svar her

```
1
```

Maks poeng: 10

18

Oppgave 3.8 (10%)

Skriv funksjonen print_nation_results(votes). Funksjonen skal:

- skrive ut hvor mange stemmer begge partiene fikk.
- deretter skrive ut hvor mange valgmenn hvert parti fikk med den normale beregningsmetoden (get_actual_ev).
- til slutt skrive ut hvor mange valgmenn de to ulike partiene hadde fått hvis alle statene hadde brukt prosentfordeling (get_actual_ev_fair).

Eksempel:
> print_nation_results(votes)
Dems got 7614 votes, reps got 7764.
With wta, dems got 36 while reps got 17 electoral votes
With fair, dems got 26 while reps got 27 electoral votes

Skriv ditt svar her

```
1
```

Maks poeng: 20