

1 **Teori**

Hvilken av de følgende påstandene er korrekte?

- ☐ Tupler er ikke-muterbare
- ☐ Lister er ikke-muterbare
- ☐ {2, 3, 4} er en tuppel
- ☐ Både lister og tupler er muterbare

Hva er hovedoppgaven til en ALU, og hva står forkortelsen for?

- ☐ Hente data fra minnet. ALU står for Arithmetic Logic Unit.
- ☐ Utføre regneoperasjoner. ALU står for Amass Logical Units.
- ☐ Utføre regneoperasjoner. ALU står for Arithmetic Logic Unit.
- ☐ Hente data fra minnet. ALU står for Amass Logical Units.

Hvilket tall på desimalform representerer det heksadesimale tallet «DDE»?

- ☐ 3277
- ☐ 3550
- ☐ 4242
- ☐ 3823

Hva står RGB for?

- ☐ Razzmatazz, Gamboge og B'dazzled blue
- ☐ Rosa, Gul og Blå
- ☐ Rød, Gul og Blå
- ☐ Rød, Grønn og Blå

Hvor mange symboler kan representeres med 5 bit?

- ☐ 64
- ☐ 5
- ☐ 16
- ☐ 32

2

Oppgave 1a)

```
def mystery(a):  
    b = ''  
    for i in range(0, len(a), 2):  
        b += a[i]  
    return b  
  
print(mystery('pbajnødran'))
```

Hva vil skrives til skjerm når kodesnutten over kjøres?

- ☐ pbajnødran
- ☐ pandabjørn
- ☐ bjørn
- ☐ panda

I kodesnutten med funksjonen mystery over, hvilken variabeltype vil bli returnert?

- ☐ Boolean
- ☐ None
- ☐ String
- ☐ Integer

---

Maks poeng: 2

3

Oppgave 1b)

```
def nested_loop(a, b):
    for x in range(1, a+1):
        for y in range(1, b+1):
            print(x*y, end=' ')
        print()
    return a*b

nested_loop(3, 4)
```

Alternativ 1:

1 2 3  
2 4 6  
3 6 9  
4 8 12

Alternativ 2:

0 0 0 0  
0 1 2 3  
0 2 4 6

Alternativ 3:

1 2 3 4  
2 4 6 8  
3 6 9 12

Alternativ 4:

1 2 3 4  
2 4 6 8  
3 6 9 12  
4 8 12 16

Hva skrives til skjerm når nested\_loop(3, 4) kjøres? (Alternativene er vist over mellom kodesnutten med funksjonen nested\_loop og denne teksten.)

- ☐ Alternativ 1
- ☐ Alternativ 2
- ☐ Alternativ 3
- ☐ Alternativ 4

Hvilken variabeltype returneres når nested\_loop(3, 4) kjøres, og hvilken verdi vil denne ha?

- ☐ Boolean, True
- ☐ Float, 12.0
- ☐ String, (korrekt svaralternativ på oppgaven over)
- ☐ Integer, 12

Maks poeng: 2

4 Oppgave 1c)

```
def secret(g, k):
    res =  ('', {}, (), [])
    for i in range(0,  (max(len(g), len(k)), min(len(g), len(k)), min(len(g), len(k))+1, max(len(g), len(k)))):
        if  (len(g) % 2 == 0, len(k) % 2 == 0, k % 2 == 0, i % 2 == 0, g % 2 == 0):
             (res.append(g[i]), res.append(i), res.append(k[i]))
        else:
             (res.append(i), res.append(k[i]), res.append(g[i]))
    return res

my_res = secret([1, 0, 0, 1, 1, 0], [0, 1, 1, 0, 1, 0, 0, 0, 1])
print(my_res)
```

I denne oppgaven skal du velge korrekt alternativ i nedtrekken, slik at funksjonen *secret* har følgende funksjonalitet:

- Funksjonen *secret(g, k)* skal ta inn to lister og returnere en ny liste *res*. Denne listen skal bestå av elementer fra listene *g* og *k* som tas inn som argumenter. Listen som returneres skal inneholde annethvert element fra liste *g* og liste *k*, slik at alle partalls-indekser i *res* er lik tilsvarende partalls-indeks i *g*, og alle oddetalls-indekser i *res* er lik tilsvarende oddetalls-indeks i *k*. Dvs. at om *n* er et positivt heltall, vil *res[2n]=g[2n]* og *res[2n+1]=k[2n+1]*.
- Listen *res* som funksjonen returnerer skal inneholde akkurat like mange elementer som den listen funksjonen tar inn med færrest elementer. Dvs. at om *len(g)<len(k)* er *len(res)=len(g)*, om *len(k)<len(g)* er *len(res)=len(k)*, og om *len(g)=len(k)* er *len(res)=len(g)=len(k)*.
- Ved kjøring av kodesnutten over, skal *[1, 1, 0, 0, 1, 0]* skrives til skjerm. Dvs. at *my\_res=[1, 1, 0, 0, 1, 0]* når *g=[1, 0, 0, 1, 1, 0]* og *k=[0, 1, 1, 0, 1, 0, 0, 0, 1]*.

Maks poeng: 5

5 Oppgave 1d)

```
def another_secret(g):
    return str(g).strip('[]').replace(' ', '')

print(another_secret([1, 1, 0, 0, 1, 0]))
```

Hva vil skrives til skjerm når kodesnutten over kjøres?

- ☐ 010011
- ☐ [110010]
- ☐ 110010
- ☐ [010011]

Både 010011 og 110010 er eksempler på binære tall. Hva er verdiene til disse tallene i det desimale tallsystemet?

- ☐ 38 og 100
- ☐ 19 og 100
- ☐ 19 og 50
- ☐ 38 og 50

Maks poeng: 2

6

Oppgave 1e)

I denne oppgaven skal du fullføre kodesnutten under ved å velge korrekt alternativ i nedtrekket, slik at funksjonen *reverse(lst)* tar inn en liste og reverserer denne.

Eksempel på kjøring:

```
>> words = ['Is', 'that', 'your', 'final', 'answer?']
>> print(reverse(words))
['answer?', 'final', 'your', 'that', 'Is']
```

```
def reverse(lst):
    res = []
    for i in range((len(res)+1, len(res), len(lst)+1, len(lst))):
        res.append((i, lst[i], lst[-i], lst[-(i+1)]))
    return res
```

Maks poeng: 2

7 **Oppgave 1f)**

```
def doing_something(lst):
    res = []
    copy = lst.copy()
    while len(copy):
        el = copy[0]
        for ele in copy:
            if ele.lower() < el.lower():
                el = ele
        copy.remove(el)
        res.append(el)
    return res

words = ['Is', 'that', 'your', 'final', 'answer?']
print(doing_something(words))
```

**Ved kjøring av kodesnutten over, hva skrives til skjerm?**

- ☐ ['answer?', 'final', 'Is', 'that', 'your']
- ☐ ['answer?', 'final', 'your', 'that', 'Is']
- ☐ ['Is', 'that', 'your', 'final', 'answer?']
- ☐ ['your', 'that', 'Is', 'final', 'answer?']

**Hva gjør funksjonen doing\_something(lst)?**

- ☐ Tar inn listen lst og returnerer en kopi av listen. Selve funksjonen gjør mye unødvendig arbeid, og kunne bare ha returnert lst.copy() fra starten av.
- ☐ Tar inn listen lst og returnerer en ny liste med elementene i lst reversert.
- ☐ Tar inn listen lst og returnerer en ny liste med elementene i lst sortert i alfabetisk rekkefølge (A-Å).
- ☐ Tar inn listen lst og returnerer en ny liste med elementene i lst sortert i omvendt alfabetisk rekkefølge (Å-A).

**I funksjonen doing\_something(lst) blir en kopi av listen lst lagret i variabelen copy i kodelinjen copy = lst.copy(). Om man hadde byttet denne linjen med copy = lst, hva ville vært forskjellen på de to funksjonene?**

- ☐ Den opprinnelige funksjonen endrer på listen lst (etter kjøring av doing\_something(words), ville man endt opp med words = []). Derimot, ville den foreslåtte endringen resultert i at funksjonen ikke lenger endrer på listen lst (dvs. words blir ikke endret etter kjøring av doing\_something(words)).
- ☐ Ingen av de andre alternativene stemmer.
- ☐ Det ville ikke vært noen forskjell. Ingen av versjonene av funksjonen doing\_something(lst) ville ha resultert i at listen lst blir endret.
- ☐ Den opprinnelige funksjonen endrer ikke på listen lst (dvs. words blir ikke endret etter kjøring av doing\_something(words)). Derimot, ville den foreslåtte endringen resultert i at funksjonen endrer på listen lst, siden både variabelen lst og copy da ville ha pekt på samme objekt i minnet (etter kjøring av doing\_something(words), ville man endt opp med words = []).

Maks poeng: 3

**Useful Python functions and commands**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.

*%*

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

*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.

*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.

*try: except: else: finally:*

try:

# Code to test

except:

# If code fails. Many exception types, like IOError for file operations.

# 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 operations:

#### *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.isspace()*

Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (`\n`), and tabs (`\t`)).

#### *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.join(list)*

Returns a string listing all items in the list, separated by s.

#### *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 (spaces, newlines and tabs) 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).

#### *str.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 every k items. 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.

*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.

*s.reverse()*

Reverses the order of the items in a list.

*s.sort()*

Rearranges the elements of a list so they appear in ascending order.

#### 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.

#### File operations:

*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). Adding 'b' to the read or write attribute lets you use binary mode to save the value of variables to file and load from them. For binary to work you need to import the pickle library.

*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 and including a 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 a sequence of strings (typically a list of strings) to file.

*f.close()*

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

#### Library operations:

*pickle.dump(obj, file)*

Write a pickled representation of 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.

8

## Oppgave 2a)

I denne oppgaven skal du skrive funksjonen *is\_vocal* som tar inn en karakter (f.eks. 'a'), for så å returnere *True* om karakteren er en vokal og *False* ellers.

Eksempel på kjøring:

```
>> print(is_vocal('c'))
False

>> print(is_vocal('e'))
True
```

Skriv ditt svar her...

1

Maks poeng: 10

9 **Oppgave 2b)**

I denne oppgaven skal du lage en funksjon *number\_of\_vocals* som tar inn en streng og returnerer hvor mange vokaler denne strengen inneholder.

Her kan du bruke funksjonen fra oppgave 2a) om du ønsker, og denne kan brukes selv om du ikke har løst forrige oppgave.

Eksempel på kjøring:

```
>>print(number_of_vocals('Winter is coming.'))  
5
```

**Skriv ditt svar her...**

1

Maks poeng: 10

10

Oppgave 2c)

I denne oppgaven skal du lage en funksjon *number\_of\_consonants* som tar inn en streng og returnerer hvor mange konsonanter det er i denne.

Her kan du bruke funksjonene fra oppgave 2a) og 2b) om du ønsker, og disse kan brukes selv om du ikke har løst de forrige oppgavene.

Du kan gå ut ifra at strengen bare inneholder bokstav-karakterer, punktum, mellomrom og komma.

Eksempel på kjøring:

```
>> print(number_of_consonants('Winter is coming.'))
9
```

Skriv ditt svar her...

1	
---	--

Maks poeng: 10

11

Oppgave 3a)

En automatisk bomstasjon tar bilde av hvert eneste kjøretøy som passerer og bruker OCR (Optical character recognition) for å lagre registreringsnummerene i en database.

Anta denne databasen som en liste med strenger, eksempelvis:

reg\_num = ["FP53023", "VD18392", "EL02113", "VR12589", "EV00473"]

I norge består alle registreringsnumre av to bokstaver pluss 5 tall, der bokstavene er avhengig av type kjøretøy eller opphavssted. I dag vil alle elbiler starte med bokstavene "EL", "EV" eller "EK".

Fullfør funksjonen *GetElectricVehicles(num\_lst)* som tar inn en liste med registreringsnumre og returnerer en liste med alle registreringsnumre som tilhører elbiler!

```
def GetElectricVehicles(num_lst):
    electric_cars = ["EL","EK","EV"]
    result =  (int(input("Skriv inn registreringsnummer")), False, [], ["F
for  (car in electric_cars, car in num_lst, i in range(len(num_lst) + 1)):
    if  (num_lst[i][0:2] in electric_cars, car[:2] in electric_cars, num_lst[i-1
         (result = result[i-1], result.append(num_lst[i][i+1]), result.split(num_ls
    return result
```

Maks poeng: 4

12

Oppgave 3b)

Bomstasjonen utvider funksjonaliteten og hvert element i databasen inneholder nå et timestamp på når bilen passerte bomstasjonen.

Vi kan anta databasen som en to-dimensjonal liste (liste av lister). Hvert element i listen er altså en liste på formen [år, måned, dag, time, minutt, regnr], slik at databasen kan se slik ut:

```
reg_num = [ [2019,10,14,10,15,"VG73488"], [2019,10,14,11,1,"AB88372"], [2019,10,14,13,9,"EL41284"],
[2019,10,14,14,1,"DB72331"] ]
```

I denne oppgaven skal du fullføre funksjonen `getPassedCars(num_lst, year, month, day, hour=-1)`, som tar inn et timestamp og en liste med registreringsnummer, og returnerer en liste med alle biler som har passert bomstasjonen på en gitt tid.

Legg merke til at `hour` har *default-verdi* -1. Dette betyr at hvis funksjonen kalles uten at `hour` er spesifisert vil variabelen ta verdien -1. Funksjonen skal fungere slik at hvis `hour` er spesifisert returnerer funksjonen alle biler passert den gitte timen. Ellers returnerer den alle biler passert på angitt dag.

*Les mer om default-verdier under oppgaven.*

Fullfør funksjonen `getPassedCars`

```
def getPassedCars(num_lst, year, month, day, hour=-1):
    res =  (, num_lst, False, [])
    for  (car in range(year - day), i in range(num_lst), i in range(num_st - 1), car in num_l
        if car[0] == year and car[1] == month  (and car[2] == day, and car[2] == hour, and hour ==
            if  (hour % 5 == 0, hour < -1, hour == -1, hour != -1, hour > -1):
                res.append(car[5])
            else:
                if  (car[4] == minute:, car[3] == hour:, car[5] == hour:, car[0] == year:)
                    res.append(car[5])
    return  (num_lst, res, True, False)
```

Jeg skjønner ikke hva default-verdi betyr!  
Eksempelfunksjon med default-verdi:

```
def printNumber(num=42): #42 er default-verdien til num
    print(num)
```

Hvis *num* blir spesifisert, det vil si vi gir den en verdi når vi kaller funksjonen, tar *num* den verdien vi har spesifisert. Hvis ikke tar *num* default-verdien vi har satt i funksjonsdefinisjonen.

Med andre ord, man kan kalle funksjonen *printNumber()* på to måter: Med eller uten argument.  
Kalles den med argument, printes argumentet, for eksempel vil kallet `printNumber(100)` skrive ut 100 til konsollen.  
Kalles den uten argument, `printNumber()`, vil 42 bli printet ut.

Maks poeng: 6

**Useful Python functions and commands**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.

*%*

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

*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.

*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.

*try: except: else: finally:*

try:

# Code to test

except:

# If code fails. Many exception types, like IOError for file operations.

# 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 operations:

#### *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.isspace()*

Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (\n), and tabs (\t) ).

#### *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.join(list)*

Returns a string listing all items in the list, separated by s.

#### *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 (spaces, newlines and tabs) 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).

#### *str.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 every k items. 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.

*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.

*s.reverse()*

Reverses the order of the items in a list.

*s.sort()*

Rearranges the elements of a list so they appear in ascending order.

#### 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.

#### File operations:

*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). Adding 'b' to the read or write attribute lets you use binary mode to save the value of variables to file and load from them. For binary to work you need to import the pickle library.

*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 and including a 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 a sequence of strings (typically a list of strings) to file.

*f.close()*

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

#### Library operations:

*pickle.dump(obj, file)*

Write a pickled representation of 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.

13

Oppgave 4a)

I denne oppgaven skal du skrive funksjonen *converting* som tar inn et binært tall som en streng og returnerer det tilsvarende desimale tallet som et heltall.

Eksempel på kjøring:  
    >> *print(converting("101010"))*  
    42

Skriv ditt svar her...

1	
---	--

Maks poeng: 10

14

Oppgave 4b)

Plasser alternativene på korrekt plass i koden slik at funksjonen *products* tar inn en matrise *matrix* og en vektor *vector*, og returnerer den resulterende matrisen man får ved å multiplisere *matrix* med *vector*.

The general formula for a matrix-vector product is

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \end{bmatrix}.$$

Altså må hver kolonne i *matrix* multipliseres med hver rad i *vector*.

Eksempel på kjøring:

```
# Kjøring av de følgende kodelinjene:
matrix = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]
vector = [1, 2, 4]
print(products(matrix, vector))

# vil gi følgende output:
[[1, 4, 12], [4, 10, 24], [7, 16, 36]]
```

 [Hjelp](#)

res.append(row)

row.append(lst[i]\*vector[i])

i in range(len(lst)):

[]

lst in matrix:

```
def products(matrix, vector):
    res = []

    for 

        row = 

        for 

            

        

    return res
```

Maks poeng: 5

**Useful Python functions and commands**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.

*%*

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

*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.

*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.

*try: except: else: finally:*

try:

# Code to test

except:

# If code fails. Many exception types, like IOError for file operations.

# 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 operations:

#### *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.isspace()*

Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (`\n`), and tabs (`\t`)).

#### *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.join(list)*

Returns a string listing all items in the list, separated by s.

#### *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 (spaces, newlines and tabs) 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).

#### *str.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 every k items. 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.

*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.

*s.reverse()*

Reverses the order of the items in a list.

*s.sort()*

Rearranges the elements of a list so they appear in ascending order.

#### 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.

#### File operations:

*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). Adding `'b'` to the read or write attribute lets you use binary mode to save the value of variables to file and load from them. For binary to work you need to import the pickle library.

*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 and including a 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 a sequence of strings (typically a list of strings) to file.

*f.close()*

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

#### Library operations:

*pickle.dump(obj, file)*

Write a pickled representation of 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.

15

Bonus 1 - Anagram

Anagram er ord som du får ved å stokke om på bokstaver i andre ord. Eksempel:

- Kama Sutra = Austmarka
- Rivaliserende = Avleirende ris
- Somlepave = Av eplemos

I denne oppgaven skal du lage funksjonen `isAnagram(s1, s2)` som tar inn to strenger og returnerer om ordene er hverandres anagram.

**Tips 1:** Du kan få bruk for strengfunksjonen `.lower()`, som returnerer strengen i kun små bokstaver, og `.isalpha()`, som returnerer `True` hvis strengen kun inneholder alfabetiske tegn og `False` ellers.

**Tips 2:** `"".join(sorted(s))` blir en alfabetisk sortert utgave av strengen `s`!

Skriv ditt svar her...

1

Maks poeng: 10