Eksamen1 TDT4110

Institutt for datateknologi og informatikk

Eksamensoppgave i TDT4110 - Informasjonsteknologi, grunnkurs

Faglige kontakter under eksamen:

- Børge Haugset (tlf.: 934 20 190)
- Terje Rydland (tlf.: 957 73 463)

Eksamensdato: 10. desember 2019
Eksamenstid (fra-til): 09:00 – 13:00
Hjelpemiddelkode/Tillatte hjelpemidler: D

Annen informasjon:
Det er angitt i prosent hvor mye hver deloppgave i eksamenssettet teller ved sensur. 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.

i

Forside

Institutt for datateknologi og informatikk

Eksamensoppgåve i TDT4110 - Informasjonsteknologi, grunnkurs

Faglege kontaktar under eksamen:

- Børge Haugset (tlf.: 934 20 190)
- Terje Rydland (tlf.: 957 73 463)

Eksamensdato: 10. desember 2019
Eksamenstid (frå-til): 09:00 – 13:00
Hjelpemiddelkode/Tillatte hjelpemiddel: D

Annan informasjon:
Der er gitt i prosent kor mykje kvar deloppgåve i eksamenssettet tel ved sensur. Les gjennom hele oppgåvesettet før du startar å gjere oppgåven. Disponer tida godt!

Merk! Studentar finn sensur på StudentWeb. Har du spørsmål om din sensur må du kontakte intituttet ditt. Eksamenskontoret vil ikkje kunne svare på slike spørsmål.

1

Teorispørsmål (20%)

Vel det svaret du meiner er mest riktig av alternativa. For kvart spørsmål blir det gitt poeng på følgjande måte:

- Korrekt avkrysning 1 poeng
- Feil avkrysning -1/2 poeng
- Ingen avkrysning -1/2 poeng

Det er ikkje mogleg å få under 0 poeng totalt. Maks utteljing er 20 poeng.

Der det er spesielle uttrykk står den engelske omsetjinga i parentes.

Kva er Alan Turing mest kjend for?

- Han forma det matematiske grunnlaget for dagens datamaskiner
- Han var med på å forme Turing-arkitekturen
- Han laga den første digitale maskina i verda

I eit moderne minne består kvar lokasjon av...

- 1 bit
- 1 byte
- 10 byte

Kontrolleininga har to registre. Det ene er programteljaren. Kva heiter det andre registeret?

- Dataregister
- Kontrollregister
- Instruksjonsregister

Kva utsegn omkring kompilering og tolking er korrekt?

- Ved kompilering vert kodelinene omsetje ein for ein og utført med en gang
- Ved tolking vert kodelinene omsetje ein for ein og utført med en gang
- Utføring av tolket kode er alltid raskare enn utføring av kompilert kode

Kva er ein device driver?

- Ein spesialdatamaskin for køyretøy, som er god til visuell analyse.
- Spesialisert programvare for input/output, slik at utstyr kan kommunisera med resten av systemet.
- Einina som held reie på neste instruksjon som skal utførast av en prosessor.

Kva seier Nyquist-regelen for sampling?

- Nyquist-regelen seier at samplingsfrekvensen må vere minst dobbelt så rask som den raskaste frekvensen. Sidan menneskelege øyre kan høyra lydar opp til ca. 20 000Hz, vil samplingsfrekvens på 40 000Hz oppfylle Nyquists regel for digital lydopptak
- Nyquist-regelen seier at samplingsfrekvensen må vere minst halvparten av den raskaste frekvensen. Sidan menneskelege øyre kan høyra lydar opp til ca. 20 000Hz, vil samplingsfrekvens på 10 000Hz oppfylle Nyquists regel for digital lydopptak
- Nyquist-regelen seier at samplingsfrekvensen må vere minst like rask som den raskaste frekvensen. Sidan menneskelege øyre kan høyra lydar opp til ca. 20 000Hz, vil samplingsfrekvens på 20 000Hz oppfylle Nyquists regel for digital lydopptak.

Kva binært tal vert representert av det hexadesimale talet 39A?

- 0011 1001 1010
- 0011 1001 1011
- 0101 1011 1100

Kor mange symboler kan representerast med 6 bit?

- 64
- 32
- 36

Digital informasjon:

- er utsett for forvrenging og elektronisk støy
- er kontinuerleg
- er diskret

Kva vert -14 i 2ers komplement representert med 8 bit?

- 11110010
- 01110010
- 11111110

Kva er misrepresentasjon?

- Å oppgje feilaktige opplysningar om eit produkt eller teneste eller levera produkt/tenester som er falsk eller av dårlig kvalitet.
- Å lure brukarar til å investere pengar eller gjere noko ulovleg på ein nettside.
- At uvedkomande tar kontroll over datamaskina til ein brukar.

Kva går buffer overflow ut på?

- Ved DoS-angrep mot ein server, vil ein annan server overta for å halde tenesta oppe for dei tiltenkte brukarane.
- Eit nettverksbuffer vert overbelasta ved at ein angripar sender svært mange datapakker som gjer tenesta utilgjengeleg for dei tiltenkte brukarane.
- At det vert oversendt meir data enn mottakar forventar. Overskrid grensene til ein databuffer og skriv til nabolokasjonar i minnet. Kan føre til minneaksessproblem, feil resultat og krasj.

Kva er password breaking?

- Automatiserte system laga for å knekke passord og dekrypteringsnøklar for å få uautorisert aksess til ein nettressurs.
- Eit angrep som endrar passordet til ein eller fleire brukarar av eit system, og dermed nektar dei åtgang til systemet.
- Eit system som sørger for at brukaren lagar eit passord som ikkje enkelt kan brytast av eit system eller kan gjettast av angriparar.

Kva er IKKJE sant om brannmurar (firewalls)?

- Dei er oftast innbygd i switchar og ruterar.
- Overvaker og kontrollerer trafikk inn og ut av eit nettverk
- Ein tryggleiksteknologi som tener dataintegritet.

Kvifor oppstår det ofte forseinkninger ved bruk av VPN?

- VPN resulterer gjerne i at datapakker ofte må gå fram og tilbake over internett fleire gonger mellom ein brukar og det lokale nettverket han er kopla opp mot.
- VPN resulterer gjerne i at datapakker må krypterast og dekrypterast fleire gonger på vegen mellom ein brukar og det lokale nettverket han er kopla opp mot.
- VPN resulterer gjerne i at datapakker går veldig sakte mellom ein brukar og det lokale nettet han er kopla opp mot ettersom det er mange brannmurar som pakkene må gjennom.

Kva er IKKJE sant om Cloud computing?

- Ein tenesteleverandør (cloud provider) bygger eit stort cloud data center med mange datamaskiner og diskar, der eit individ eller selskap kan betale for å få lagra informasjon.
- Cloud services er elastiske, noko som betyr at kundar bare treng å betale for dei ressursane dei faktisk bruker.
- Cloud services gjer som regel betre yting på nettenester for ein bedrift, men er som regel dyrare i drift enn å ha egne tenarar (servere).

Kva er IKKJE sant om kanalkoding (channel coding)?

- Kanalkoding er ei samlenemning på matematiske teknikkar som har som hensikt å avdekke og rette datafeil, og auke pålitelegheit.
- SPC (Single Parity Checking) er ein form for kanalkoding der sendaren legg til ein ekstra bit til kvar byte slik at talet på bit med verdi 1 vert anten eit partal eller oddetal.
- Kanalkoding er å kryptere meldingar frå avsendar til mottaker slik at andre ikkje kan lese innhaldet i meldinga.

Kva er det som gjer illusjonen av internett som eit enkeltståande, saumlaust kommunikasjonssystem bestående av mange datamaskiner mogleg?

- Bruk av syklisk sjekksum (Cyclic Redundancy Codes).
- TCP/IP
- Formatet på pakkene som sendes over nettet.

Korleis er ein IPv6-adresse bygd opp?

- Global prefiks, subnet og vert.
- Global prefiks, lokal suffiks og vert.
- Lokal prefiks, suffiks og subnet.

Kva er replay error i nettverkssamanheng?

- At det vert sendt duplikat til mottakar, som då må handtere dette.
- At ei stadfesting (ACK) vert sendt dobbelt, noko som kan føre til at avsender trur ei pakke har kome fram sjølv om han ikkje har det.
- At ei forsinka pakke frå tidlegare sesjon vert akseptert i seinere sesjon, og at korrekt pakke dermed vert avvist som duplikat.

Maks poeng: 20

Useful Functions and Methods

Built-in:

format(numeric_value, format_specifier)

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

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

try:

Code to test

except:

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

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

else:

Runs if no exception occurs

finally:

Runs regardless of prior code having succeeded or failed.

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

Random methods:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

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

s.reverse()

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

s.sort()

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

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

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

s.issuperset(t)

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

s.union(t)

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

s.intersection(t)

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

s.difference(t)

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

s.symmetric_difference(t)

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

s.copy()

New set with a shallow copy of *s*

s.update(t)

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

s.add(x)

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

s.remove(x)

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

s.clear()

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

Dictionary operations:

d.clear()

Clears the contents of a dictionary

d.get(key, default)

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

d.items()

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

d.keys()

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

d.pop(key, default)

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

d.popitem()

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

d.values()

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

del d[k]

Deletes element k in d.

d.copy()

Makes a copy of d.

Files:

open()

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

f.read(size)

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

f.readline()

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

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

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

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

math.e

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

numpy Library:

numpy.array(list)

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

ndarray.ndim

the number of axes (dimensions) of the array.

ndarray.shape

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

ndarray.size

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

ndarray.dtype

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

numpy.zeros(tuple)

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

numpy.ones(tuple)

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

numpy.arange(start,stop,step)

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

numpy.ndarray.reshape(lines,rows)

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

Kodeforståelse (30%)

Vel det svaret du meiner er mest riktig av alternativa. For kvart spørsmål blir det gitt poeng på følgjande måte:

- Korrekt avkrysning 3 poeng
- Feil avkrysning 0 poeng
- Ingen avkrysning 0 poeng

2

2a

Kva vert skrive ut av denne koden?

```
def myst2(a):
    b = [] + ['0']*len(a)
    for i in range(len(a)):
        if a[i]:
            b[i] = '1'
    return b

def myst(streng):
    a = list(streng)
    for i in range(len(a)):
        a[i] = int(a[i])
        a[i] = not a[i]
    return myst2(a)

print(''.join(myst('11011010')))
```

Vel eitt alternativ

- 00100101
- 10101010
- ['0', '0', '1', '0', '0', '1', '1', '1']
- 11011011
- 11011010
- IndexError: list index out of range

Maks poeng: 3

3

2b

Kva vert skrive ut av denne koden?

```
def myst9(minListe):
    nyListe = []
    while len(minListe) > 0:
        ind = 0
        minste = minListe[ind]
        for i in range(len(minListe)):
            if minListe[i] < minste:
                ind = i
        nyListe.append(minListe.pop(ind))
    return nyListe

print(myst9([99,17,65,19,82]))
```

Vel eitt alternativ

- [99,82,65,19,17]
- [82,19,65,17,99]
- [82,17,65,19,99]
- [99,17,65,19,82]
- [17,19,65,82,99]
- [19,82,65,99,17]

Maks poeng: 3

4

2c

Kva må a og b vere for at at programmet skal skrive ut:
marion?

```
def myst(s,x,y):
    s1 = ''
    for i in range(x,len(s),y):
        s1+= s[i]
    return s1

print(myst('mgfmiyaieabarbnramisdtnaooeiehnnrnza',a,b))
```

Vel eitt alternativ

- a = 1, b = 5
- a = 3, b = 6
- a = 3, b = 7
- a = 3, b = 5
- a = 1, b = 6
- a = 0, b = 6

Maks poeng: 3

5

2d

Kva vert skrive ut når denne koden vert køyrd?

```
def myst(noe):
    noe2 = set(noe[0])
    for i in range(1,len(noe)):
        noe2 = noe2.intersection(set(noe[i]))
    return list(noe2)

print(myst([[1,2,2,3,3,4,5],[1,2,2,2,4,3,3],[4,5,2,2,3,3,3,8],[2,2,3,3,4,5,7]]))
```

Vel eitt alternativ

- [2,3,4,5]
- [1,2,3]
- [2,2,3,3,4]
- [2,3,4]
- [2,2,2,3,3,3,4]
- [7,8]

Maks poeng: 3

6

2e

Kva vert skrive ut når denne koden vert køyrd?

```
def myst(noe):
    a = '01'
    verdi = noe[-1]
    f = noe[-1] == a[1]
    for i in noe[-2::-1]:
        if not f:
            verdi = i+verdi
            if i == a[1]:
                f = True
        else:
            if i == a[1]:
                verdi = a[0]+verdi
            else:
                verdi = a[1] + verdi
    return verdi


print(myst('01011010'))
```

Vel eitt alternativ

- 11011010
- 11110000
- 10100110
- 10100101
- 10101010
- 00110011

Maks poeng: 3

7

2f

Kva vert skrive ut når denne koden vert køyrd?

```
def myst(a,b):
    c,d = 0,0
    for i in a[::-1]:
        d += int(i)*b**c
        c += 1
    return d

print(myst('11110',3))
```

Vel eitt alternativ

- 1200
- 210
- 140
- 180
- 120
- 160

Maks poeng: 3

8

2g

Kva vert skrive ut når denne koden vert køyrd?

```
def myst(x):
    a,b,c = 0,0,0
    while a <= x:
        c += 1
        b = a
        if c % 2 == 0:
            a -= c**2
        else:
            a += c**2
    return b

print(myst(5))
```

Vel eitt alternativ

- -4
- -5
- -3
- 4
- 5
- 3

Maks poeng: 3

9

2h

Kva vert skrive ut når denne koden vert køyrd?

```
def myst(a,b):
    if a > b:
        c = a
    else:
        c = b
    for i in range(2,c+1):
        if a%i == 0 and b%i == 0:
            return i

print(myst(30,7))
```

Vel eitt alternativ

- 7
- None
- 2
- 3
- 5
- 1

Maks poeng: 3

10

2i

Kva vert skrive ut når denne koden vert køyrd?

```
def myst2(a,b):
    while b != 0:
        c = b
        b = a % b
        a = c
    return a

def myst(a,b):
    c = myst2(a,b)
    a = a//c
    b = b//c
    return a,b

a,b = myst(42,12)
print(f'{a}/{b}',sep='')
```

Vel eitt alternativ

- 12/4
- 7/2
- 8/2
- 9/3
- 14/4
- 6/2

Maks poeng: 3

11

2j

Kva vert b etter at denne kodebiten har køyrt?

```
import numpy as np

a = np.array([[2,3,4,5],[6,7,8,9]])
c = a.size
b = (a+c)*2
b=b.reshape((4,2))
print(b)
```

Vel eitt alternativ

- [[20 22], [24 26], [28 30], [32 34]]
- [20, 22, 24, 26, 28, 30, 32, 34]
- [[20 22] [24 26] [28 30] [32 34]]
- [20 22 24 26 28 30 32 34]
- [[20 22 24 26] [28 30 32 34]]
- [[20 22 24 26], [28 30 32 34]]

Maks poeng: 3

Useful Functions and Methods

Built-in:

format(numeric_value, format_specifier)

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

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

try:

Code to test

except:

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

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

else:

Runs if no exception occurs

finally:

Runs regardless of prior code having succeeded or failed.

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

Random methods:

random.random()

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

random.randint(a,b)

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

random.choice(seq)

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

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

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

random.uniform(a, b)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

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

s.reverse()

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

s.sort()

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

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

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

s.issuperset(t)

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

s.union(t)

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

s.intersection(t)

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

s.difference(t)

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

s.symmetric_difference(t)

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

s.copy()

New set with a shallow copy of *s*

s.update(t)

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

s.add(x)

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

s.remove(x)

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

s.clear()

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

Dictionary operations:

d.clear()

Clears the contents of a dictionary

d.get(key, default)

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

d.items()

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

d.keys()

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

d.pop(key, default)

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

d.popitem()

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

d.values()

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

del d[k]

Deletes element k in d.

d.copy()

Makes a copy of d.

Files:

open()

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

f.read(size)

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

f.readline()

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

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

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

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

math Library:

math.ceil(x)

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

math.floor(x)

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

math.exp(x)

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

math.cos(x)

Return the cosine of x radians.

math.sin(x)

Return the sine of x radians.

math.tan(x)

Return the tangent of x radians.

math.pi

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

math.e

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

numpy Library:

numpy.array(list)

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

ndarray.ndim

the number of axes (dimensions) of the array.

ndarray.shape

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

ndarray.size

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

ndarray.dtype

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

numpy.zeros(tuple)

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

numpy.ones(tuple)

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

numpy.arange(start,stop,step)

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

numpy.ndarray.reshape(lines,rows)

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

Programmering (50%)

Domenebeskriving

Oppgåvene dreiar seg om å hjelpe fiskebåtar med registrering av fangst, og vidareformidling av sal til kundar. Firmaet Avkok A/S tek inn fangst frå mange ulike båtar. Fangsten varierer i fiskeslag og mengde kilo kjøpt. Når butikkar og restaurantar kjøper fisk skal båten som fanga fisken få betalt etter gjeldane kilopris. Vi kan føresetje at når ein handel vert gjord i systemet, så vert også den fysiske flyttinga av fisken gjennomført samstundes. Du kan òg føresetje at du ikkje treng å tenkje på kvar fisk fysisk vert lagra, og at dette vert avgjort av båt og kunde. Du er berre mellomleddet her, og tek ein viss provisjon per handel. Du treng heller ikkje tenke på at fisk rotnar.

Beskriving av datastruktur

Registeret over båtar og lagra fangst er ein dictionary kalla *register*. Nøkkelen er registreringsnummeret til båten. Dette er unikt for alle båtar. Knytta til kvar nøkkel ligg verdien: *lagra fangst*. Denne er representert ved ei todimensjonal liste der kvart indre element representerer kvart sitt fiskeslag og talet på kilo som er tilgjengeleg for sal.

Døme

Båten med registreringsnummer N64Ø har 511 kilo torsk og 342 kilo hyse til sals. Båt Z4F har 233 kilo torsk og 122 kilo hyse. Då ser strukturen slik ut:

```
>>> print(register)
{'N64Ø': [['laks', 511], ['hyse', 342]], 'Z4F': [['torsk', 233], ['hyse', 122]]}
```

Gjennom heile oppgåva kan du bruke funksjonar frå tidlegare deloppgåver. Om du ikkje har løyst den tidlegare oppgåva kan du anta at funksjonen eksisterer.

Oppgåve 3a tel 7 poeng
Oppgåve 3b tel 8 poeng
Oppgåve 3c tel 7 poeng
Oppgåve 3d tel 7 poeng
Oppgåve 3e tel 7 poeng
Oppgåve 3f tel 7 poeng
Oppgåve 3g tel 7 poeng

12

Oppgave 3a - Registrering av fiskebåt (7%)

Du tek imot fisk frå kjente og nye fiskebåtar. Fiskaren registrerer ein ny båt gjennom å taste inn nummeret til båten i samsvar med fiskeriregisteret. I bakgrunnen kallar datasystemet opp ein funksjon kalla **check_registration** med *regnummer* som parameter. Funksjonen returnerer True eller False alt etter om registreringsnummeret følgjer retningslinjene i registeret. Reglane er som følgjer: *Nummer i fiskeriregisteret har forma bokstav-tal-bokstav, til dømes N64Ø («Solbris»). Første bokstav svarar til fylkesbokstaven som vart brukt på bilskilt fram til omkring 1970. Tala vert fortlaupande gitte til fiskarar i kvar enkelt kommune, og kan derfor ha anten eit eller to siffer. Til slutt står ein eller to bokstavar som angir kommune. Dømet N64Ø som er oppgitt viser at dette var fiskefartøy nummer 64 i Øksnes kommune i Nordland. Vi gjentar: <1 bokstav><1-2 siffer><1-2 bokstavar>*.

Skriv funksjonen **check_registration** som skal ta imot eit *registreringsnummer* og returnere om det er skrive i samsvar med retningslinjene i fiskeriregisteret.

Døme:

```
>>> print(check_registration('N64Ø'))
True
>>> print(check_registration('NN4Ø'))
False
```

Døma over er berre døme - ein skal ikkje 'hardkode' sjekk mot akkurat desse.

Følgjande registreringsnummer skal til dømes òg returnere True: N4A, B46MC, P53V, Z4HB.

Følgjande registreringsnummer skal returnere False: NR5H, P557J, 672NH.

Skriv svaret ditt her...

```
1
```

Maks poeng: 7

13

Oppgave 3b - Hvor mye av en viss fisk (8%)

Kvar båt har lagra fisken i ei todimensjonal liste. Det kan vere fint å finne ut kor mykje av ein enkelt fiskeart som er lagra. Her kan vi ta for gitt at lagra fisk for ein gitt fiskar er tilgjengeleg gjennom parameteren *store*, mens fisketypen er ein streng gitt i parameteren *kind*. Viss det ikkje finst noko fisk av denne typen skal funksjonen returnere talet 0.

Skriv funksjonen **fish_amount** som tek parameterane *store* og *kind*.

Døme:

```
>>> store = [['torsk', 200],['sei',100]]
>>> print(fish_amount(store, 'torsk'))
200
>>> print(fish_amount(store, 'månefisk'))
0
```

Skriv svaret ditt her...

```
1
```

Maks poeng: 8

14

Oppgave 3c - Legg til fisk (7%)

Lageret skal oppdaterast når det kjem inn meir fisk. Dette vert oppgitt som ei liste med fiskeslag og talet på kilo: ['torsk', 200]. Lag eit nytt element viss fisketypen ikkje allereie finst i lageret.

Skriv funksjonen **add_fish**, som tar *store* og ei *liste* som inputparameterar og returnerer det oppdaterte lageret.

Døme:

```
>>> print(store)
[['torsk', 200], ['sei', 100]]
>>> add_fish(store, ['torsk', 100]) # Already in store
>>> add_fish(store, ['hyse', 70]) # Not in store
>>> print(store)
[['torsk', 300], ['sei', 100], ['hyse', 70]]
```

Skriv svaret ditt her...

```
1
```

Maks poeng: 7

15

Oppgave 3d - Lagre masse fisk (7%)

Ein fiskar kan fange mange typar fisk på ein gong - og ynskjer dermed å rapportere inn dette. Fleire typer fisk vert rapportert inn som ei todimensjonal liste, på same format som *store*: [['type', mengde], ['type', mengde], ['type', mengde]]

Skriv funksjonen **add_much_fish** som tek *store* og ei *2D-liste* som inputparameterar.

Døme:

```
>>> print(store)
[['torsk', 300], ['sei', 100], ['hyse', 70]]
>>> add_much_fish(store, [['kveite', 120], ['torsk', 200]])
>>> print(store)
[['torsk', 500], ['sei', 100], ['hyse', 70], ['kveite', 120]]
```

Skriv svaret ditt her...

```
1
```

Maks poeng: 7

16

Oppgave 3e - Fjerne fisk (7%)

Fisk vert fjerna frå lista av to grunner - anten har han vorte for gammal, eller så har han vorte seld (meir om det seinare). Kva og kor mykje som skal fjernast vert sende med som parameter, på formatet ['type', mengde].

Lag funksjonen **remove_fish** som har parameterane *store* og *remove*.

Funksjonen skal fjerne den gitte mengda fisk frå lageret (*store*). Dersom det skulle gjerast eit forsøk på å fjerne meir frå lageret enn det faktisk finst, så skal det givast beskjed om dette. I så fall skal ingen fisk fjernast når funksjonen vert kalla. Du finn eit eksempel på dette ved spørjing etter kveite under.

Døme:

```
>>> print(store)
[['torsk', 500], ['sei', 100], ['hyse', 70], ['kveite', 120]]
>>> remove_fish(store, ['torsk', 300])
>>> print(store)
[['torsk', 200], ['sei', 100], ['hyse', 70], ['kveite', 120]]
>>> remove_fish(store, ['kveite', 200])
There's not enough kveite left
>>> print(store)
[['torsk', 200], ['sei', 100], ['hyse', 70], ['kveite', 120]]
```

Skriv svaret ditt her...

```
1
```

Maks poeng: 7

17

Oppgave 3f - Lese inn fisk fra fil (7%)

Fiskebåtane er aktive på natta. Under oppteljing rapporterar dei inn fangstmengda av ulike fiskeslag gjennom nettsida di, dette vert lagra rett i ei fil. Når du kjem på kontoret køyrer du funksjonen **read_fish_from_file**. Denne funksjonen les innhaldet i fila *fishy.txt* og oppdaterer eit register med båtar og kva fisketyper og mengder dei har tilgjengelig. Fila består av eit sett med liner på forma båtnamn:fisketype:mengde. (Andre og sjette innslag her er feil).

N64Ø:torsk:343
N434:torsk:200
Z4F:torsk:120
N64Ø:torsk:200
N64Ø:hyse:110
NB4A:kveite:100
Z4F:hyse:120
N64Ø:kveite:300

Skriv funksjonen **read_fish_from_file**. Funksjonen tek inn ein dictionary *register* - denne er beskriven i første del av oppgåveteksta. Dersom det ikkje eksisterer ein båt med båtnamnet i registeret frå før, og båtnamnet stemmer med namnesjekken i oppgåve a skal det opprettast ein ny nøkkel på denne og fangsten som er rapportert skal leggjast inn. Dersom båtnavnet ikkje er korrekt bygd opp skal det ikkje leggjast til i registeret, men funksjonen skal i staden skrive ut:
'Båtkode' var ugyldig - hopper over fangstrapportering.

Døme:

```
>>> register = {}
>>> read_fish_from_file(register)
N434 was illegal - skipping catch reporting
NB4A was illegal - skipping catch reporting
>>> print(register)
{'N64Ø': [['torsk', 543], ['hyse', 110], ['kveite', 300]], 'Z4F': [['torsk', 120], ['hyse', 120]]}
```

Skriv svaret ditt her...

```
1
```

Maks poeng: 7

18 Oppgave 3g - Send kjøpere til fiskerne som har rett type (7%)

Gjennom nettsida di sender kjøparar deg førespurnadar om dei kan kjøpe fisk av ulik type og mengde, men berre ein type av gongen.

Funksjonen **handle_customer** med parameterane *register* og *need* handterer dette. Kundane skal få vite kva for båtar dei må kontakta, og kor mykje fisk av ulikt slag dei kan kjøpe frå kvar av båtane. Dei skal i tillegg få veskjed om kor mykje dette kostar, og få vite det dersom det ikkje er nok fisk av typen til å dekke behovet. Han skal òg beregne *din inntekt* av videreformidlinga.

Skriv funksjonn **handle_customer** som beskrive.

Variabelen *register* inneheld den fisken som er tilgjengeleg hos alle fiskeseljarane akkurat no, medan *need* er fiskearten og mengda som kunden vil ha. Formatet på *need* er ei liste ['type', mengde]. Dette behovet skal så fyllast av ein eller fleire fiskebåtar, viss mogleg. Du treng ikkje tenke på rettferdig fordeling av sal mellom fiskebåtane. Viss det ikkje er mogleg å samle nok fisk til å dekke behovet til kunden skal det *skrivast ut* ein beskjed , se dømet under. Dersom behovet er dekt skal tilsvarande mengder av fisk som kunden skal ha frå kvar båt *fjernast* frå registeret til båtane.

Viss eit behov ikkje kan dekkast skal ingen fisk fjernast. Viss eit sal lykkast skal du ha *betalt frå kunden*. Du tar ein fast pris på ti kroner per kilo for all fisk som vert omsett, og femti kroner for kvar båt som det skal handlast frå. Det skal skrivast ut til kunden kor mykje handelen kostar samstundes som dei får beskjed om kva for båtar dei kan handle frå.

Døme:

```
>>> register = {'N64Ø': [['torsk', 543], ['hyse', 110], ['kveite', 300]], 'Z4F': [['torsk', 120], ['hyse', 120]]}
>>> handle_customer(register, ['torsk', 300]) # This is available
The following boats have torsk: [['N64Ø', 300]]. The total cost is 3050 kroner.
>>> handle_customer(register, ['torsk', 300]) # This is available
The following boats have torsk: [['N64Ø', 243], ['Z4F', 57]]. The total cost is 3100 kroner.
>>> handle_customer(register, ['kveite', 310]) # This is too much
There's not enough kveite left to cover your requirement
>>> print(register)
{'N64Ø': [['torsk', 0], ['hyse', 110], ['kveite', 300]], 'Z4F': [['torsk', 63], ['hyse', 120]]}
```

Skriv svaret ditt her...

```
1
```

Maks poeng: 7