Forside

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): 15:00 – 19:00
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.

1 Teorioppgaver (20%)

Velg det svaret du mener er mest riktig av alternativene. For hvert spørsmål gis det poeng på følgende måte:

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

Det er ikke mulig å få under 0 poeng totalt. Maks uttelling gir 20 poeng.

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

Hva står CPU for?

- Central processing unit
- Circuit processing unit
- Control programming unit

Hva er cache?

- Forgjengeren til dagens internett.
- En metode for å printe ledninger på chiper i flere lag.
- En veldig rask minneteknologi.

Hvilke 5 steg er med i “Fetch/Execute Cycle”?

- Instruction Fetch(IF), Instruction Execute(EX), Instruction Decode(ID), Data Decode(DD), Result Return(RR)
- Instruction Fetch(IF), Data Fetch(DF), Instruction Decode(ID), Result Return(RR), Instruction Execute(EX)
- Instruction Fetch(IF), Data Fetch(DF), Instruction Decode(ID), Data Decode(DD), Result Return(RR)

Hva er pipelining?

- En teknikk der en CPU kan utføre flere instruksjoner parallelt.
- En teknikk som fungerer som en sikker tunnel mellom din maskin og en tjener.
- En teknikk der man sender data mellom de forskjellige delene i maskinen i «pipes».

Hvilken av disse lagringsenhetene er IKKE en sekundærlagringsenhet?

- Hurtigbufferet i datamaskinen.
- En minnepinne
- En SSD satt rett i PCI Expressbussen

Hvilket heksadesimalt tall representerer $(100111)_2$?

- 27
- 43
- 39

Navnet “Bob” skrives som “0100 0010 0110 1111 0110 0010” i Extended ASCII. Hvilket alternativ representerer ordet “obo” i Extended ASCII?

- 0110 1111 0100 0010 0110 1111
- 0110 1111 0110 0010 0110 1111
- 0110 0010 0100 0010 0110 0010

Informasjon som beskriver informasjon er kalt:

- Collating data
- Metadata
- Special data

Et bilde har 800*600 piksler, i 16 bit fargeformat. Hvor mye plass trenger det ukomprimert?

- Omtrent 1 MB
- Omtrent 2 MB
- Omtrent 500 kB

Hvilken av følgende komprimeringer er loss-less?

- Run-length coding
- MP3
- JPEG

Hva er phishing?

- At uvedkommende tar kontroll over en brukers datamaskin.
- Å opptre som en kjent nettside (f.eks. nettbank) for å få tak i personlig informasjon som f.eks. aksesskoder, kontonummer, etc.
- Å fiske etter personlig informasjon ved å late som om maskinen er under virusangrep. Angriperen lover brukeren at feilen skal rettes opp dersom brukeren først oppgir kontonummeret sitt.

Hva er scams?

- At uvedkommende tar kontroll over en brukers datamaskin.
- Bevisst blokkering av tilgang til en nettside eller tjeneste
- Å lure brukere av f.eks. en nettside eller en tjeneste til å investere penger eller gjøre noe ulovlig.

Hvordan fungerer replay-angrep?

- Pakker fra tidligere sesjoner fanges opp og sendes. For eksempel passord-pakker fra tidligere påloggінger.
- Bruker blir oppringt fra en tjeneste som ber brukeren ringe tilbake. Hvis brukeren ringer tilbake blir hun eller han belastet for store beløp for en ringetjeneste som koster mye å bruke.
- Chat-bot som svarer brukere på nettsider feilaktig for å innhente sensitiv informasjon.

Hva skjer når en melding krypteres?

- Dataene i meldingen endres, slik at kun riktig mottaker kan rekonstruere den opprinnelige meldingen.
- Dataene i meldingen deles i små pakker slik at mottaker er den eneste som får tak i dem alle.
- Meldingen må gå gjennom en brannmur, slik at det blir vanskeligere for angripere å få tak i dataene.

Hva er riktig om Payload encryption?

- Krypterer både pakkehodet og meldingsinnholdet i pakken.
- Krypterer kun meldingsinnholdet i pakken og ikke selve pakkehodet.
- Krypterer kun pakkehodet og ikke selve meldingsinnholdet i pakken.

Hva er sant om transportlaget?

- Transportlaget inneholder alle spesifikasjoner relatert til radiofrekvenser.
- Transportlaget består blant annet av spesifikasjoner om nettverksadressering og det maksimale antallet pakker som et nettverk kan støtte.
- Transportlaget sørger for at all data blir levert slik den ble sendt; komplett og i riktig rekkefølge.

Hva står forkortelsen WAN for i forbindelse med nettverk?

- World Area Network
- Wired Area Network
- Wide Area Network

Hva vil universal service si i forbindelse med nettverk?

- Å tillate kommunikasjon mellom datamaskiner uavhengig av hvilken type nettverk de sitter på.
- Å tilby streaming-tjenester for alle koplet til internett.
- Å tilby tilgang til skytjenester for alle datamaskiner.

Hva er masken til IPv4-addressen 255.255.128.0 i CIDR-notasjon?

- /36
- /256
- /17

Hva står TCP for i nettverkssammenheng?

- Transmission Control Protocol
- Transmission Channel Protocol
- Transport Channel Protocol

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%)

Velg det svaret du mener er mest riktig av alternativene. For hvert spørsmål gis det poeng på følgende måte:

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

Oppgave 2a

Hva skrives ut fra følgende kode?

```
def myst3(liste,navn):
    dikt = {}
    for i in liste:
        dikt[i[0]] = f'{(sum(i[1:])/(len(i)-1)):.2f}'
    return navn,dikt

liste = [['Per',98,67],['Anne',80,66],['Svein',99,88]]
print(f'{myst3(liste,"Anne")}')
```

Velg ett alternativ

- ('Anne', {'Anne': '73.00'})
- ('Anne', {'Per': '82.50', 'Anne': '73.00', 'Svein': '93.50'})
- {'Per': 82.5, 'Anne': 73.0, 'Svein': 93.5}
- 'Anne' {'Per': '82,50', 'Anne': '73,00', 'Svein': '93,50'}
- ('Anne', {'Per': 82.5, 'Anne': 73.0, 'Svein': 93.5})
- {'Anne': '73.00'}

Maks poeng: 3

3 Oppgave 2b

Hva skrives ut fra denne kodebiten?

```
def myst(a):
    b = []
    while len(a) > 0:
        c = a[0]
        for i in a:
            if i < c:
                c = i
        a.remove(c)
        b.append(c)
    return b


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

Velg ett alternativ

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

Maks poeng: 3

4 Oppgave 2c

Hva skrives ut fra denne koden?

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

print(myst(30,4))
```

Velg ett alternativ

- 3
- 4
- 1
- 2
- 5
- 6

Maks poeng: 3

5 Oppgave 2d

Hva skrives ut fra denne koden?

```
def myst2(a,b):
    while len(a)%b != 0:
        a = '0' + a
    return a


def myst(a,b,c):
    d = ''
    while a > 0:
        d = str(a%b) + d
        a = a//b
    return myst2(d,c)

print(myst(18,2,4))
```

Velg ett alternativ

- 00001001
- 01000010
- 00110010
- 00010010
- 0010010
- 100010

Maks poeng: 3

6 Oppgave 2e

Hva skrives ut av denne koden?

```
def myst(mat):
    r = len(mat)
    c = len(mat[0])
    for i in range(r):
        for j in range(c):
            if mat[i][j] >= 6:
                mat[i][j] = 0
            else:
                mat[i][j] = 1
    return sum(mat[1])

print(myst([[3, 2, 0], [19, 0, 18], [0, 6, 0]]))
```

Velg ett alternativ

- 1
- 6
- 2
- 48
- 37
- 5

Maks poeng: 3

7 Oppgave 2f

Hva skrives ut av denne koden?

```
def myst1(x,y,z):
    t = x
    x = y
    y = t
    for i in range(z):
        z = x + y
        y*= 2
    return z

print(myst1(2,4,6))
```

Velg ett alternativ

- 32
- 100
- 69
- 68
- 132
- 36

Maks poeng: 3

8 Oppgave 2g

Hva skrives ut av denne koden?

```
def myst(a,b,c):
    d = ''
    while a > 0:
        d = str(a%b) + d
        a = a//b
    return d

print(myst(18,2,4))
```

Velg ett alternativ

- 01110
- 11001
- 11100
- 10010
- 0010
- 1100

Maks poeng: 3

9 Oppgave 2h

Hva må a og b være for at programmet skal skrive ut MARTIN?

```
def myst(tekst,a,b):
    return tekst[a::b].upper()

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

Velg ett alternativ

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

Maks poeng: 3

10 Oppgave 2i

Hva skrives ut av denne koden?

```
def myst(a,b):
    b = [[a[0],1]]
    for i in range(1,len(a)):
        funnet = False
        for j in b:
            if a[i] == j[0]:
                j[1] += 1
                funnet = True
        if not funnet:
            b.append([a[i],1])
    return b

tekst = 'DettE er en enkel streng. Eller to'
b = []
b = myst(tekst,b)
print(b[1])
```

Velg ett alternativ

- IndexError: list index out of range
- ['e',8]
- ['e',7]
- ['e','7']
- ['e',9]
- ['e']

Maks poeng: 3

11 Oppgave 2j

Hva skrives ut av denne koden?

```
import numpy as np

a = np.arange(1,5,0.5)
c = a.size
b = (a+c)*2
print(b)
```

Velg ett alternativ

- [18. 19. 20. 21. 22. 23. 24.]
- [18 19 20 21 22 23 24 25]
- [18. 19. 20. 21. 22. 23. 24. 25.]
- [18., 19., 20., 21., 22., 23., 24., 25.]
- [18., 19., 20., 21., 22., 23., 24., 25.,26.]
- [18. 19. 20. 21. 22. 23. 24. 25. 26.]

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*

Under finner du innholdet i filen nameregister.py, som ligger i samme folder som pythonkoden du skal lage:

```python
# module nameregister

'''
days_in_month(month)
The function takes an int between 1 and 12, and returns the number
of days in that month. Leap years are ignored.
The function crashes for numbers outside the range 1-12.
Example: days_in_month(1) returns 31
'''
def days_in_month(month):
    return [31,28,31,30,31,30,31,31,30,31,30,31][month-1]

'''
register_name(people, fnr, name)
Used to add persons to the dictionary 'people'
The function takes three parameters:
people: Dictionary containing personal ID number (key) and name (value)
fnr (string): Personal ID number (fødselsnummer) of the person to be added
name (string): Name of the person to be added

You're not to write this - only use it
'''
def register_name(people, fnr, name):
    people[fnr] = name

'''
find_name(people, fnr)
Used to find the name of a person in 'people', if you have the ID number.
The function takes to parameters:
people: Dictionary containing personal ID number (key) and name (value)
fnr (string): Personal ID number of the person to look up
The function returns the name of the person if it exists. If not, it
returns an empty string, ''

You're not to write this - only use it
'''
def find_name(people, fnr):
    return people.get(fnr,'')
```

Programmering (50%)

Førerkortprikker og botbetaling

I denne eksamenen skal du lage et system for registrering av prikker som norske borgere får når de begår trafikkforseelser. Du skal lage et sett funksjoner, som oppfyller en del krav til systemet, disse blir beskrevet under hver oppgave. Dette er et oversiktsbilde av systemet, så følger en beskrivelse av datastrukturen som skal lages og modulen som allerede eksisterer. Merk for øvrig at dette oppgavesettet kun er inspirert av det faktisk prikkbelastningssystemet som finnes i Norge.

Modulen nameregister.py

Alle personer som registreres i systemet må registreres med fødselsnummer og navn (fornavn etternavn). Begge disse er av type strenger. Modulen *nameregister* inneholder tre funksjoner som hjelper deg med å registrere denne koblingen. Modulen ligger i den samme folderen som koden du skal skrive, du skal altså bruke disse i koden din. De tre funksjonene

er

- *register_name(...)*: Denne brukes til å legge til nye fødselsnummer og knytte dem til et navn.
- *find_name(...)*: Denne funksjonen brukes til å finne et navn gitt at en oppgir et fødselsnummer (11 siffer som streng).
- Begge disse funksjonene kalles med en variabel *people*. Denne kan du forvente å ha etablert i koden du skriver i alle funksjonene på denne eksamenen. For en ytterligere beskrivelse av funksjonene, inkludert parametere, refererer vi til beskrivelsen av modulen som ligger i bunnen av funksjonsbeskrivelsen til venstre for oppgavesettet. Du skal altså ikke gå rett inn og endre på *people*, dette er kun tillatt gjennom funksjonene i modulen.
- *days_in_month(...)*: Denne funksjonen tar inn et heltall mellom 1 og 12, og returnerer hvor mange dager det er i denne måneden. Funksjonen ignorerer skuddår. Funksjonen krasjer for andre innverdier enn tallene 1-12! Eksempel: days_in_month(1) returnerer 31

Oppgavens struktur

En funksjon du skal lage i en oppgave kan med hell benytte seg av funksjoner som allerede er beskrevet i *tidligere* oppgaver. Du står fritt til å bruke tidligere funksjoner slik de står beskrevet selv om du ikke har klart å løse dem - bruk funksjonene slik de er spesifisert å virke. Det vil aldri være tilfelle at du skal bruke funksjoner som blir gitt som oppgaver i *senere* oppgaver. Oppgavene kan løses i nedadgående rekkefølge. Noen oppgaver er vanskeligere enn andre. Det er lurt å bruke litt tid til å lese igjennom alle først, for å danne seg et inntrykk av hvordan en skal angripe oppgaven.

Datastruktur

Datastrukturen som du skal utvikle består denne av en *dictionary* som kalles *register*. Nøkkelen er et *fødselsnummer* (11 siffer lang streng). Verdien knyttet til hver nøkkel er prikkene som hver fører har fått så langt. Prikkene er lagret i form av en *liste*, der nye forseelser legges inn til slutt i listen. Eksempelvis vil et register med to førere som begge har fått noen prikker se slik ut:

register = {'12046712125':[2,2,3],'23099045653':[3]}

Det er ikke nødvendig å tenke på strukturen til det som finnes i modulen - her trenger en bare å bruke modulens funksjoner på korrekt måte. Du kan forvente at det eksisterer en variabel *people* som brukes når en kaller opp funksjonene her.

Alle oppgavene krever at du skal skrive en funksjon. Når dere ser beskrivelsen (...) etter en funksjon betyr det at en skal fylle inn antall parametre og navnet på dem slik det står beskrevet i oppgavene.

Oppgave 3a teller 7 poeng

Oppgave 3b teller 8 poeng

Oppgave 3c teller 7 poeng

Oppgave 3d teller 7 poeng

Oppgave 3e teller 7 poeng

Oppgave 3f teller 7 poeng

Oppgave 3g teller 7 poeng

12 Oppgave 3a - Datosjekk (7%)

Nye førere må registreres (mer om det senere), og da må en legge inn folks fødselsnummer (streng med 11 siffer). Vi ønsker ikke å legge inn verdier som er feil i systemet. Derfor må det kontrolleres at datoer som legges inn i alle fall er reelle.

Skriv funksjonen **check_date**

- Funksjonen skal ta inn en streng som skal verifiseres som en reell dato. Strengen er hele fødselsnummeret til føreren, der de seks første dekker fødselsdatoen. Disse er på formatet ddmmyy - 12 mars 1979 blir da '120379'. Litt informasjon og krav:
- Alder: år personen er født (kun de to siste sifrene). Du kan se bort fra alder på personene, altså ingen sjekk om på om de gamle nok til å ha førerkort.
- Måned: tallet må være mellom 01 og 12, og alltid med to siffer (mars blir '03')
- Dag: dag passer inn med måneden brukt.
- Det finnes allerede en funksjon *days_in_month(month)*. Denne ligger i modulen nameregister.py i den samme mappen som programmet ditt. Denne må du gjerne bruke, og du finner en beskrivelse av den nederst på funksjonsarket.
- Du kan også se bort fra alt som har med skuddår å gjøre.
- Hvis datoen er reell skal funksjonen returnere *True*, ellers skal den returnere *False*.

Eksempel:

```
>>> print(check_date('29127311245'))
True
>>> print(check_date('31117532456'))
False
```

Skriv ditt svar her...

```
1
```

Maks poeng: 7

13 Oppgave 3b - Registrer ny fører (8%)

Når en ny person begår en trafikkforseelse må denne registreres. Her er det to ulike systemer dette skal registreres i - *people* og *register*.

Skriv funksjonen **register_person**

Funksjonen tar fire parametre:

- *people*: En dictionary som gir kobling mellom fødselsnummer og navn.
- *register*: En dictionary som har fødselsnummer (streng) som nøkkel og en liste over prikker som verdi.
- *fnr*: En streng som inneholder fødselsnummeret til den som skal registreres. Du kan forvente at datoen her er korrekt, og at det ikke finnes noen med dette fødselsnummeret fra før.
- *name*: En streng som inneholder navnet til den som skal registreres.

Funksjonen skal gjøre følgende:

- I *people* skal du legge til et nytt innslag med fnr og navn. *register_name* i modulen *nameregister* bør være til hjelp.

- I *register* skal det registreres et nytt innslag med nøkkel *fnr*, verdien skal være en tom liste.
- Variablene *people* og *register* kan du forutsette at du har tilgang på allerede - du trenger altså ikke lage dem. Dette gjelder også seinere oppgaver.

Eksempel:

```
>>> print(people) # Empty dictionary
{}
>>> print(register) # Empty dictionary
{}
>>> register_person(register, people, '11010154321', 'Terje Rydland')
>>> register_person(register, people, '23056644521', 'Børge Haugset')
>>> print(people)
{'11010154321': 'Terje Rydland', '23056644521': 'Børge Haugset'}
>>> print(register)
{'11010154321': [], '23056644521': []}
```

Skriv ditt svar her...

```
1
```

Maks poeng: 8

14 Oppgave 3c - Telle prikker (7%)

Personer får registrert prikker i en dictionary kalt *register*. Denne har som nøkkel et personnummer (streng med 11 siffer). Verdien er en liste som inneholder alle prikkene

personen har fått, den første først.

Skriv funksjonen **count_dots**

- Du kan forvente at du har tilgang til variabelen *register*
- Funksjonen skal ha to parametre:
  - Den første er *register* (dictionary), som er nevnt over.
  - Den andre er *fnr* (streng, 11 siffer), fødselsnummeret til føreren man skal finne antall prikker til.
- Funksjonen skal returnere summen av prikker (som heltall) denne personen har fått.
- Du kan forvente at fødselsnummeret som oppgis finnes i *register*.

Eksempel:

```
>>> # They have respectively 8 and 3 dots
>>> register = {'11010154321':[3,2,3],'23056644521':[3]}
>>> count_dots(register, '11010154321')
8
```

Skriv ditt svar her...

```
1
```

Maks poeng: 7

15 Oppgave 3d - Legg til prikker (7%)

Prikker legges inn som verdi i dictionary *register*, knyttet til nøkkelen fødselsnummer (streng med 11 siffer). Prikkene er lagret i listeform, og nye prikker registreres på slutten av den eksisterende listen. Listen består av heltall, og kan se slik ut: [2, 2, 3]. Når nye prikker har blitt lagt til må en sjekke om totalt antall prikker er 10 eller mer. Hvis de er det skal det skrives ut en beskjed om at føreren har gått over grensen, og førerkortet skal beslaglegges. Her holder det å skrive ut en beskjed til brukeren som legger inn prikkene.

Skriv funksjonen **add_dots**

- Funksjonen skal ha tre parametre
  - *register* - dictionary som inneholder fødselsnummer og prikker for alle, som før.
  - *fnr* - fødselsnummeret til den som har kjørt i grøfta. Du kan forvente at dette fødselsnummeret allerede er registrert.
  - *dots* - antallet prikker som skal legges til i slutten av denne førerens liste.
- Hvis antallet prikker for føreren etter innlegging av nye prikker er 10 eller mer, skal det skrives ut en beskjed. Se eksempelet under. Husk funksjoner som er laget før selv om du ikke har skrevet dem selv.
- Du kan forvente å ha tilgang til variablene *people* og *register.*

Eksempel:

```
>>> print(register)
{'11010154321': [3, 2, 3], '23056644521': [3]}
>>> add_dots(register, '11010154321', 3)
The driver has more than 10 dots, confiscate!
>>> add_dots(register, '23056644521', 3) # Børge previously had 3
>>> # Nothing is printed, as the total is less than ten.
```

Skriv ditt svar her...

```
1
```

Maks poeng: 7

16 Oppgave 3e - Manuell innlegging av prikker (7%)

Funksjonene som er skrevet til nå har alle basert seg på at informasjon som skal legges inn allerede er definert. Nå har vi kommet til delen der brukeren av systemet må skrive inn ting.

Skriv funksjonen **manual_registration**

- Funksjonen har to parametre - *register* og *people*. De er like som i oppgavene før.
- Brukeren skal først skrive inn et fødselsnummer (streng med 11 siffer).
- Det må sjekkes om fødselsdatoen i fødselsnummeret er en reell dato. Hvis ikke - avbryt funksjonen. Fødselsdatoen er de seks første sifrene i fødselsnummeret.
- Hvis dette fødselsnummeret ikke allerede er registrert må denne nye personen registreres i både *people* og *register*, i tråd med funksjonsbeskrivelsene gitt i tidligere oppgaver. Husk å spørre om navn!
- Til slutt skal funksjonen spørre om hvor mange prikker denne føreren har fått, og registrere dem.

Eksempel:

```
>>> # Existing user:
>>> manual_registration(register, people)
Personal ID number: 23056644521
Dots to register: 2
>>> # Invalid date of birth:
>>> manual_registration(register, people)
Personal ID number: 12345678909
The date is invalid.
>>> # New ID number
>>> manual_registration(register, people)
Personal ID number: 30118866154
Not registered before. Name: Alf Inge Wang
Dots to register: 1
```

Skriv ditt svar her...

```
1
```

Maks poeng: 7

17 Oppgave 3f - Betale seg ut av problemer (7%)

Det er alltid mulig å betale seg ut av problemer i dette systemet. En fører har mulighet til å betale for å redusere antallet prikker som er registrert på seg.

Skriv funksjonen **pay**

Funksjonen **pay** har en parameter:

- *register*: Registeret over fødselsnummer og nåværende prikker.

Den fungerer på følgende måte:

- Brukeren skal bli spurt om å skrive inn fødselsnummeret og summen føreren skal betale.
- Funksjonen skal legge inn negative prikker. 1 prikk per 10 000 kroner. Hvis en fører vil betale 35 000 kroner skal tallet -3 legges til på slutten av listen, som i eksempelet under. Husk tidligere funksjoner.
- Hvis føreren ikke er registrert skal meldingen 'Fører ikke registrert' skrives ut og funksjonen avsluttes.

Eksempel:

```
>>> register = {'11010154321': [3, 2, 3, 2], '23056644521': [3]}
>>> pay(register)
Personal ID number: 11010154321
How much will the driver pay? 35000
>>> print(register['11010154321'])
[3, 2, 3, 2, -3]
```

Skriv ditt svar her...

Maks poeng: 7

18 Oppgave 3g - Lagring til fil (7%)

Vi ønsker å lagre informasjon om førerne som er registrert på fil.
Skriv funksjonen **save_to_file**

- Funksjonen tar to parametre, *register* og *people*
- Filen det skal skrives til skal hete '*dots.txt*'.
- For hver av de registrerte førerne skal det lages en linje der følgende streng står (uten ' foran og bak): '11010154321 (Terje Rydland) har 7 prikker.'

Eksempel:

```
>>> register = {'11010154321': [3, 2, 3, 2], '23056644521': [3]}
>>> people = {'11010154321': 'Terje Rydland', '23056644521': 'Børge Haugset'}
>>> save_to_file(register, people)
```

Etter at koden over har kjørt, vil *dots.txt* inneholde følgende:

11010154321 (Terje Rydland) har 10 prikker.
23056644521 (Børge Haugset) har 3 prikker.

Skriv ditt svar her...

```
1
```

Maks poeng: 7