Eksamensforside

Institutt for datateknologi og informatikk
Eksamensoppgave i TDT4110 Informasjonsteknologi, grunnkurs

Dato: 25. november 2024 (09:00-13:00)
Faglig kontakt: Atle Nes (tlf. 98852760) og Dag Olav Kjellemo (tlf. 92050951)
Møter i eksamenslokalet: Nei

Hjelpemiddelkode/Tillatte hjelpemidler:
D / Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

ANNEN INFORMASJON

Les oppgavene nøye og gjør dine egne antagelser. Presiser i besvarelsen hvilke forutsetninger du har lagt til grunn i tolkning/avgrensing av oppgaven.

Faglig kontaktperson skal kun kontaktes dersom det er direkte feil eller mangler i oppgavesettet. Henvend deg til en eksamensvakt hvis du mistenker feil og mangler. Noter spørsmålet ditt på forhånd.

FAGSPESIFIKK INFORMASJON

Ingen håndtegninger:
Denne eksamenen tillater ikke bruk av håndtegninger. Har du likevel fått utdelt skanne-ark, er dette en feil. **Arkene vil ikke bli akseptert for innlevering, og de vil derfor heller ikke sendes til sensur.**

Vekting av oppgavene:
Vekting i prosent er angitt for hver deloppgave.

Varslinger:
Eventuelle beskjeder under eksamen (f.eks. ved feil i oppgavesettet), sendes ut via varslinger i Inspera. Et varsel vil dukke opp som en dialogboks på skjermen. Du kan finne igjen varselet ved å klikke på bjella øverst til høyre.

Trekk fra/avbrutt eksamen:
Dersom du ønsker å levere blankt/avbryte eksamen, gå til “hamburgermenyen” i øvre høyre hjørne og velg «Lever blankt». Dette kan ikke angres selv om prøven fremdeles er åpen.

Tilgang til besvarelse:
Etter eksamen finner du besvarelsen din under tidligere prøver i Inspera. Merk at det kan ta én virkedag før eventuelle håndtegninger vil være tilgjengelige i arkivet.

1

Her kan du kladde

Useful Functions and Methods

These are listed in the following order:

- built-in (standard library)
  - often used functions and operators
  - exceptions
  - string methods
  - list operations
  - set operations
  - dictionary operations
  - files
  - pickle library (binary files)
- random library
- math library
- numpy library
- matplotlib.pyplot

Built-in:

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

try:

Code to test

except:

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

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

else:

Runs if no exception occurs

finally:

Runs regardless of prior code having succeeded or failed.

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

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

s.reverse()

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

s.sort()

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

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

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

s.issuperset(t)

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

s.union(t)

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

s.intersection(t)

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

s.difference(t)

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

s.symmetric_difference(t)

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

s.copy()

New set with a shallow copy of *s*

s.update(t)

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

s.add(x)

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

s.remove(x)

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

s.clear()

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

Dictionary operations:

d.clear()

Clears the contents of a dictionary

d.get(key, default)

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

d.items()

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

d.keys()

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

d.pop(key, default)

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

d.popitem()

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

d.values()

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

del d[k]

Deletes element k in d.

d.copy()

Makes a copy of d.

Files:

open()

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

f.read(size)

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

f.readline()

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

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

Close the file and free up any system resources taken up by the open file. If using **with open(filename) as ...** when you open the file, close() is not necessary, since it is implied by the ending of the with-block

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

Random Library:

random.random()

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

random.randint(a,b)

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

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.

indexing and slicing in arrays

for 1d arrays (vectors), indexing and slicing works the same way as for lists. For multi-dimensional arrays, there is a more compact index notation for arrays, e.g , if A is an array, A[i, j] is synonymous to A[i][j]. This more compact notation also allows more powerful slicing, for instance A[:,j] making a slice containing just column j of A (since colon : for the first index indicates all rows)

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.abs(x)

returns the absolute value of x. Unlike the abs() function in the standard library, which only works on single values, numpy.abs() can process an entire array, returning an array with the corresponding absolute values

numpy.sum(x)

calculates the sum of the values in the array x

numpy.prod(x)

calculates the product of the values in the array x

numpy.sin(x), numpy.cos(x), numpy.sqrt(x)

calculates the sinus, cosinus and the square root, similar to the same functions in the math library. However, while the math functions work just on single values, the numpy functions can also work on entire arrays, then returning arrays of the corresponding sine, cosine or square root values.

numpy.arange(start,stop,step)

creates a vector starting at *start*, ending at (but not including) *stop* using *step* between the values

numpy.linspace(start, stop, num)

creates a vector starting at *start*, ending at (and including) *stop*, and with *num* evenly spaced values in total

numpy.ndarray.reshape(lines,rows)

reshapes an ndarray according to the values in *lines* and *rows.* The reshape method does not change the original array, instead returning a reshaped copy

numpy.ndarray.resize(lines,rows)

reshapes an ndarray (similar to reshape method above), but resize changes the original array in place, rather than making a copy. Hence, this method returns None.

numpy.ndarray.tolist()

returns a list (of lists) with the same dimensions and content as the numpy array. Unlike list(A) which will only convert the first dimension of a multi-dimensional array to a list, A.tolist() makes a deep conversion, making all dimensions into lists

numpy.loadtxt(filename)

Loads data from text file, returning them as a numpy array.

numpy.savetxt(filename, A)

Saves array A to text file.

matplotlib.pyplot:

subplots()

Creates a *figure* containing a single *axes* object for plotting.

plot(x, y)

Plots a graph of y-values against x-values

bar(x, height)

Plots bars in positions given by vector *x*, with heights given by vector *height*

show()

Shows the plot on the screen

1.1 Eksakt 3%

Hvilke av disse tallene kan representeres eksakt som enten heltall eller flyttall i Python?

Hvis man skriver uttrykket i venstre kolonne, så velg "Heltall" hvis tallet lagres som heltall, "Eksakt flyttall" hvis det lagres som flyttall med den eksakte verdien til uttrykket, og "Ikke eksakt" ellers.

For hvert tall, sett kryss i rett kolonne. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker.

	Eksakt flyttall	Ikke eksakt	Heltall
11**30 / 1	○	○ ✔	○
1 + 1/2**20	○ ✔	○	○
1.1	○	○ ✔	○
1+10**20	○	○	○ ✔
42	○	○	○ ✔
1 + 1/4	○ ✔	○	○

Maks poeng: 3

1.2 Regnerekkefølge 2%

Skriv svaret i boksen. Skriv svaret så enkelt som mulig
Det er ingen minusupoeng for feil svar.

a) **(7 + 3) * 5 - 2**2** (46)

b) **4 < 3 and 2 > 2** (False)

c) **False or 1 + 1 == 6 % 4** (True)

d) **5 + 2 == 7 and 1** (1)

Maks poeng: 2

1.3 Kodesnutt 2%

Hva blir resultatet av følgende kodesnutt?

```
1   flag = ""
2   a = 0
3   i = 1
4   while(a < 3):
5       j = 1
6       if flag:
7           i = j * i + 5
8       else:
9           i = j * i + 1
10      a = a + 1
11  print(i)
```

Velg ett alternativ:

- 11
- 16
- 12
- 4 ✔

Maks poeng: 2

1.4 Frukt 2%

Hva blir resultatet av følgende kodesnutt?

```
1   frukt = ["eple", "banan", "pære"]
2   frukt[1:2] = ["mango", "kirsebær"]
3   print(frukt)
```

Velg ett alternativ:

- ['eple', 'mango', 'kirsebær', 'pære'] ✔
- ['eple', 'banan', 'pære']
- ['eple', 'mango', 'kirsebær']
- ['mango', 'kirsebær', 'pære']

Maks poeng: 2

1.5 Python programmering 2%

Hva blir resultatet av følgende kodesnutt?

```
1   ord = "Python programmering"
2   n = len(ord)
3   ord1 = ord.upper()
4   ord2 = ord.lower()
5   nyttord = ""
6   for i in range(n):
7       if i % 2 == 0:
8           nyttord += ord2[i]
9       else:
10          nyttord += ord1[i]
11  print(nyttord)
```

Velg ett alternativ:

- PYTHON PROGRAMMERING
- python programmering
- PyThOn pRoGrAmMeRiNg
- pYtHoN PrOgRaMmErInG ✔

Maks poeng: 2

1.6 Matrise 2%

Hvilket av følgende alternativer produserer resultatet 6 når A er definert som

```
1    A = [[13, -9, -15],
2         [-5, 2, 6],
3         [8, 0, 12]]
4
```

Velg ett alternativ:

- A[2][3]
- A[3][2]
- A[1][2] ✔
- A[2][1]

Maks poeng: 2

1.7 Synlighet 2%

Hva blir resultatet av følgende kodesnutt?

```
1   verdi = "herlig"
2
3   def minfunksjon():
4       global verdi
5       verdi = "fantastisk"
6
7   minfunksjon()
8   print("Python er " + verdi)
```

Velg ett alternativ:

- Python er herlig
- undefined
- error
- Python er fantastisk ✔

Maks poeng: 2

1.8 Muterbarhet 2%

Hvilken av følgende datatyper er muterbar i Python?

Velg ett alternativ:

- int
- float
- list ✔
- tuple

Maks poeng: 2

1.9 Nøstet ordbok 2%

Hva blir korrekt syntaks for å skrive ut navnet "Håkon" gitt at du har følgende datastruktur?

```
1   a = {'navn' : 'Kristin', 'alder' : '20'}
2   b = {'navn' : 'Håkon', 'alder' : '23'}
3   kunder = {'c1' : a, 'c2' : b}
```

Velg ett alternativ:

- print(kunder.c2.b['navn'])
- print(kunder['c2']['navn']) ✔
- print(kunder['c2'].navn)
- print(kunder.c2.navn)

Maks poeng: 2

1.10 Import 2%

Hva skriver du om du bare ønsker å importere funksjonen sqrt fra math-modulen?

Velg ett alternativ:

- import sqrt from math
- from math import sqrt ✔
- from sqrt import math
- import math.sqrt

Maks poeng: 2

1.11 Funksjonen regnut() 2%

```
def regnut(a):
    a = a + 2
        a = a * 2
    return a
print(regnut(2))
```

Hva er resultatet av følgende program?

Velg ett alternativ:

- Runtime Error
- 8
- 4
- Indentation Error ✔

Maks poeng: 2

1.12 Tallsekvens 2%

Nedenfor har vi en for-løkke som skriver ut en sekvens med tall.
Hvilke tall vil bli skrevet ut av løkkene?

```
1    for i in range(1,7,3):
2        print(i)
```

OBS! Gale svar på denne oppgaven gir minuspoeng.

Velg ett eller flere alternativer

- [ ] 0
- [ ] 8
- [ ] 1 ✔
- [ ] 7
- [ ] 5
- [ ] 3
- [ ] 6
- [ ] 2
- [ ] 4 ✔

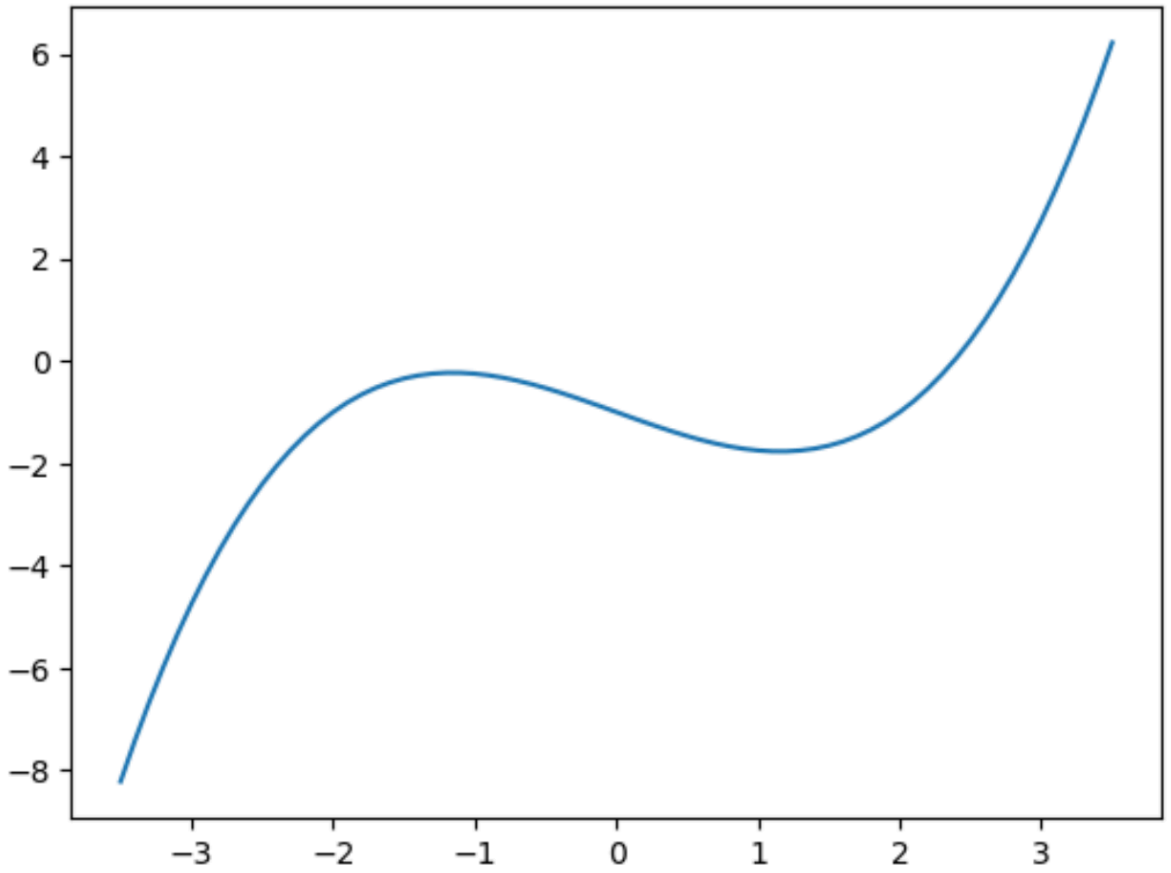
Maks poeng: 2

1.13 Plotte graf 4%

Gitt følgende kode:

```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return 0.25*x**3 - x - 1
```

Vi ønsker nå å plotte funksjonen f over intervallet -3.5 .. 3.5, som vist på figuren, dvs. startpunktet på den blå kurven er x = -3.5, y = -8.22 og sluttpunktet er x = 3.5, y = 6.22.



Trekk kodelinjer til riktig plass så vi får det ønskede plottet. Kun 4 av kodelinjene skal brukes.

Hjelp

plt.subplots()	ax.plot(x, f(x))	np.show()
x = np.arange(-3.5, 3.6, 0.1)	x = np.arange(-3.5, 3.5, 0.1)	plt.show()

fig, ax = plt.subplots() ✔

x = np.arange(-3.5, 3.6, 0 ✔

ax.plot(x, f(x)) ✔

plt.show() ✔

Maks poeng: 4

1.14 Iterere 2d array 7%

Funksjonen **check_array(arr, c, d)** skal få inn tre parametre arr, c, d, hvor arr er et numpy array med tall, mens c og d er heltall. Funksjonen skal returnere **True** hvis inntil **c** av radene i arrayet arr inneholder minst ett partall som er < d, og **False** hvis mer enn c rader har partall < d. Eksempel:

Hvis arrayet arr er
[[5 -1 1]
[5 6 7]
[2 9 1]]

så skal

- check_array(arr, 1, 4) gi **True** fordi bare 1 rad (nederste) har et partall < 4 (nemlig 2).
- check_array(arr, 1, 7) gi **False** fordi det er mer enn én rad (de to nederste) som har partall < 7.

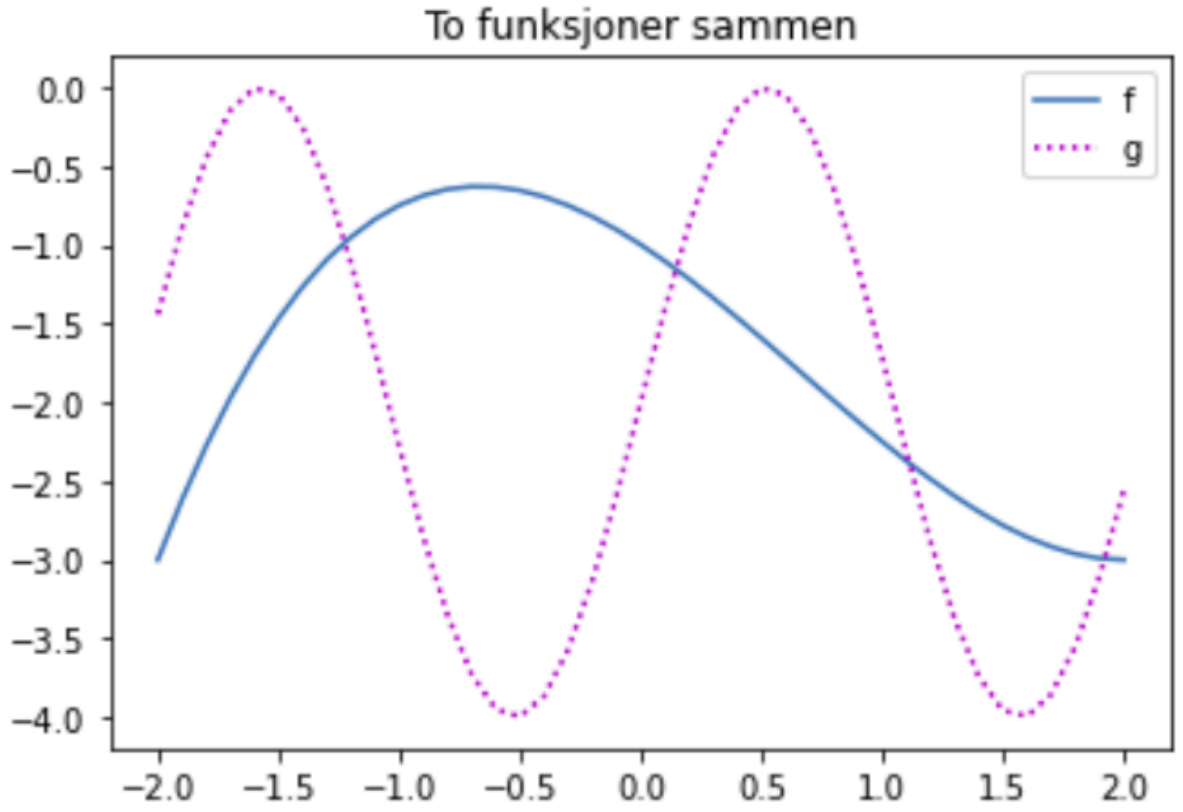
Velg riktig alternativ i hullene slik at funksjonen virker som den skal:

```
def check_array(arr, c, d):
    count = 0

    Velg alternativ (for row in range(len(arr)):, for row in range(count):, for row in arr:)
        for number in row:

            Velg alternativ (if number < d and number % 2 == 0:, if number < d or number % 2 == 0:, if number // 2 == 0:)

                Velg alternativ (return False, return True, count += 1)

                Velg alternativ (return True, break, return False)

        Velg alternativ (if count == 0:, if count > c:, if count < c:)

            Velg alternativ (break, return True, return False)

    Velg alternativ (return True, return False, return ok)
```

Maks poeng: 7

1.15 Plotte to funksjoner 6%

Koden under skal plotte to funksjoner i samme bilde, som vist her. Merk spesielt den lille boksen i hjørnet som viser hvilken graf som tilhører hvilken funksjon. For å skille kurvene er ulike farger brukt, samt at den ene har heltrukken linje og den andre prikkete. Det reguleres med et tredje argument til plot-funksjonen. I et kall som **ax.plot(x, y, ":m")** sier tegnet ":" at vi skal ha prikkete linje (mens f.eks. "--" ville gi streket linje, og "o" store rundinger), og "m" angir farge magenta (mens f.eks. "g" - grønn, "r" - rød, "b" - blå, "k" - svart). Merk deg også hvorvidt kurvene går over samme x-intervall eller ikke, samt om de er glatte eller hakkete, som indikerer om det er mange eller få verdier i x-vektoren som ble plottet.



Velg rett alternativ i hvert tomrom i koden under, slik at den vil resultere i eksakt samme figur som vist.

Velg alternativ (import numpy as np, import numpy, from numpy import *)

```
import matplotlib.pyplot as plt

def f(x):
    return 0.25*x**3 - 0.5*x**2 - x - 1

def g(x):
    return 2 * np.sin(3*x) - 2

fig, ax = plt.subplots()
```

x = np. Velg alternativ (arange(-2.0, 2.1, 0.5), arange(-2.0, 2.1, 0.1), linspace(-2.0, 2.0, 9))

ax. Velg alternativ (plot(x, f(x), ":m"), plot(x, f(x)), plot(f(x), x))

Velg alternativ (# ingen kode trengs her, v = np.arange(-3-0, 2.0, 0.001), v = np.arange(-2.0, 3.0, 0.1))

ax. Velg alternativ (plot(x, g(x)), plot(x, g(x), ":m"), plot(v, g(v), ":m"))

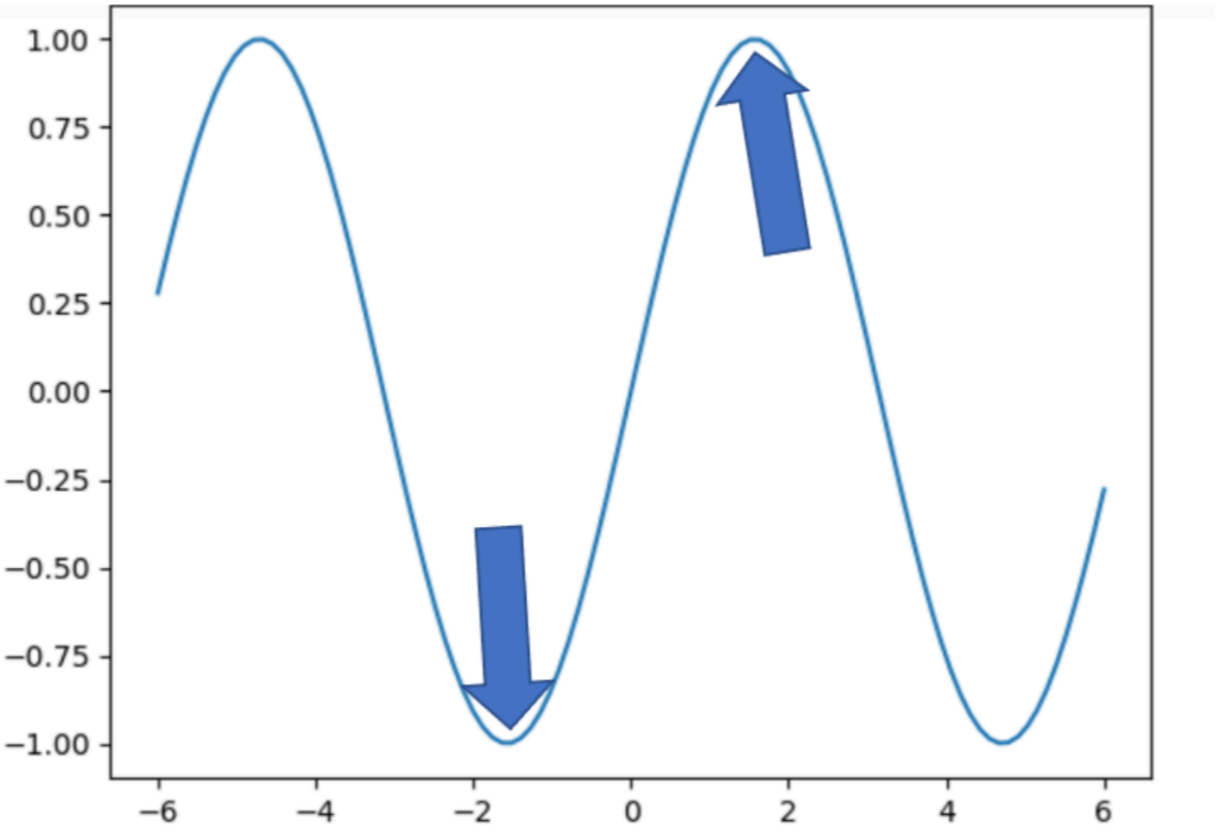
ax.legend(Velg alternativ (["f , g"], ["f(x)", "g(x)"], ["f", "g"]))
plt.title("To funksjoner sammen")
plt.show()

Maks poeng: 6

1.16 Funksjoner som parametre 8%

Vi ønsker en funksjon **longest_increase(f, x_values)** som får inn som parametre en funksjon **f** og et array med flyttall **x_values**. Funksjonen skal returnere et tuppel som angir første og siste x-verdi i det lengste intervallet innenfor x_values hvor **f** er strengt stigende. Som et eksempel, anta at funksjonen som gis inn for parameter **f** er **np.sin(x)** som vist i plottet under, og at x_values er np.arange(-6.0, 6.0, 0.01). Funksjonen skal da finne intervallet fra x = -1.57 til x = 1.57 siden dette er det lengste strekket hvor funksjonsverdien stiger over arrayet x_values, fra bunnpunktet til toppunktet vist med blå piler (bunnpunktet for sinuskurve er på cirka x = -1.571 og toppunktet på x = 1.571, men stegverdien 0.01 på arange begrenser hvor nær vi kan komme de eksakte verdiene). Altså,
longest_increase(np.sin(x), np.arange(-6.0,6.0,0.01)) skal returnere **(-1.57, 1.57)**.

Hvis det ikke fins noen strekning hvor f er stigende i det gitte intervallet, eller x_values er tom eller bare inneholder ett element, skal funksjonen longest_increase returnere et **tomt tuppel**.



Velg riktig i feltene under slik at funksjonen virker som den skal.

```
import numpy as np

def longest_increase(f, x_values):
    result = ( )
    current_start = None
    for i in range(1, len(x_values)):
        if f(x_values[i]) > f(x_values[i-1]):
```

Velg alternativ (if current_start is None:, if result == ( ):, if f(x_values[i]) > f(x_values[i-1]):)

Velg alternativ (current_start = result[0], result[0] = current_start, current_start = x_values[i-1])

Velg alternativ (elif i == len(x_values) - 1:, elif result = ( ):, elif current_start is None:)

Velg alternativ (if result == () or (x_values[i] - current_start > result[1] - result[0]):, if result == () and not (x_values[i] - current_start > result[1] - result[0]):, if x_values[i] - current_start > result[1] - result[0]:)

Velg alternativ (result = (f(x_values[i]), f(x_values[i-1])), result = (current_start, x_values[i]), result = (result[0], x_values[i] - current_start))
else:

Velg alternativ (if result = ( ):, if current_start is None:, if current_start is not None:)

Velg alternativ (if result == ():, if x_values[i - 1] - current_start > result[1] - result[0]:, if result == () or (x_values[i - 1] - current_start > result[1] - result[0]):)
result = (current_start, x_values[i - 1])

Velg alternativ (current_start = x_values[i], current_start = None, current_start = f(x_values[i]))
return result

Maks poeng: 8

1.17 Dyretelling 4%

Nå skal du løse funksjonen *les_registreringer()*, av dyreliv i Norge. Først litt ekstra informasjon, så kommer kravene bedre beskrevet nær der koden skal spesifiseres:

Innholdet i eksempelfilen 'dyr.txt'.

elg 10
ku 13 16 2
elg 5
svin 4 12
elg 13 1
marsvin 3
kråke 2
maur 103463

Eksempel på bruk med denne filen:

>>> print(les_registreringer('dyr.txt', ['rev', 'elg', 'bjorn']))
{'elg': [10, 5, 13, 1]}
>>> print(les_registreringer('dyr.txt', ['maur', 'elg', 'bjorn']))
{'elg': [10, 5, 13, 1], 'maur': [103463]}

Slep og slipp elementene under for å kode funksjonen *les_registreringer(...).* Bruk tekstfilen og eksempelet over til å hjelpe deg.

- Funksjonen har to parametre:
  - *filnavn*: filnavnet som skal leses. Du kan forutsette at filen ligger i samme mappe.
  - *dyr*: En liste med strenger som beskriver hvilke dyr som skal registreres.
- En registrering består av et dyrenavn, et mellomrom, og så ett eller flere tall separert med mellomrom. (elg 12 eller ku 12 14)
- Funksjonen skal ignorere eventuelle tomme linjer, eller om registreringene omhandler dyr som ikke er omtalt i listen 'dyr'.
- Funksjonen skal returnere en dictionary med følgende struktur:
  - Nøklene skal være dyrene som vi ønsket å lese registreringene til.
  - Verdiene skal være en liste over de ulike registreringene til det spesifikke dyret, i rekkefølgen det ble lest inn.

Hjelp

parts[1:]	parts[:1]	line	result[dyrenavn]
line.isEmpty():	parts[0]	line = line.strip()	parts.keys()
line.split()	parts	read()	readline()
extend	append	add	

```
def les_registreringer(filnavn, dyr):
    result = {}
    with open(filnavn, 'r') as file:
        for line in file:
```

```
        line = line.stri
    if  line                :
        parts =  line.split()
        dyrenavn =  parts[0]
        registreringer = []
        for x in  parts[1:]     :
            registreringer.  append     (int(x))
        if dyrenavn in dyr:
            if dyrenavn not in result:
                result[dyrena      = []
            result[dyrenavn].  extend     (registreringer)
return result
```

Maks poeng: 4

1.18 De fleste skal bli de første 6%

I denne oppgaven skal du bruke resultatet av funksjonskallet til forrige oppgave. Slep og slipp elementene for å løse funksjonen *sort_by_registrations* i henhold til spesifikasjonen under:

- Funksjonen har én parameter, *registreringer* (dictionary med nøkkel dyreart (string), verdi en liste med heltallsregistreringer) som for eksempel *{'elg': [10, 5, 13, 1], 'maur': [103463]}*
- Funksjonen skal skrive ut summen av hver enkelt dyrearts registreringer.
- Rekkefølgen på utskriften avhenger av antall dyr registrert, høyest antall registreringer først.
- Utskriften skal følge følgende mal (utelat <>):
  - <dyreart>: <totalt antall registreringer> (<hvor mange færre enn den over, på øverste>)

Eksempel merk at det fløy inn en ekstra undulat:
>>> r = {'elg': [10, 5, 13, 1], 'maur': [103463], 'undulat' : [1]}
>>> sort_by_registrations(r)
maur: 103463 (0)
elg: 29 (103434)
undulat: 1 (28)

def sort_by_registrations(registreringer):

Velg alternativ (also_important, important, something, stuffed) = Velg alternativ (set(), [], {}, registreringer.keys())

for Velg alternativ (variable,, Alternativ 4, variable, variable, something) in

Velg alternativ (registreringer, registreringer.keys(), registreringer.values(), registreringer.items()):

important[variable] = Velg alternativ (sum(something), something[-1], len(something), something)

stuffed = []

Velg alternativ (while, if, Alternativ 4, for) variable in Velg alternativ (important, Alternativ 4, important.items(), important[1:]):

Velg alternativ (stuffed.extend(, print(, stuffed.append(, something.contains()((variable, important[variable]))

for i in range(len(stuffed)):

```
    for j in range(i + 1, len(stuffed)):
        if stuffed[i][1] < stuffed[j][1]:
            stuffed[i], stuffed[j] = stuffed[j], stuffed[i]

also_important = None

for variable, total in Velg alternativ (important, stuffed.items(), registreringer, stuffed):

    if also_important is Velg alternativ (stuffed[1:], stuffed.items(), None, stuffed):
        print(f"{variable}: {total} (0)")
    else:

        print(f"{variable}: {total} ({ Velg alternativ (len(also_important[variable]), variable,
len(variable), also_important) - total})")

    Velg alternativ (also_important, total, variable, also_important +)= total
```

Maks poeng: 6

1.19 Les kamper 10%

I denne oppgaven skal vi lese resultater fra kamper i fotball eller liknende idrett.

Hver linje en tekstfil har resultatet fra en kamp skrevet på formen

hjemmelag;bortelag:resultat

Resultat angir antall mål som henholdsvis hjemmelaget og bortelaget har scoret. Se eksempel lenger ned.

Skriv funksjonen **les_kamper(filnavn, logger)** som leser alle radene i filen og lagrer resultatene i en ordbok, som returneres fra funksjonen.

Elementene i ordboka skal ha hjemmelag som nøkkel. Verdien er en ny ordbok. Her er nøklene bortelagene som hjemmelaget har spilt mot. Verdiene til denne er en liste med resultatene til alle kampene mellom de to lagene, med hjemmelaget først. Kamper hvor hjemmelaget spilte borte, vil da ligge under bortelaget.

Eksempel: Gitt en fil med følgende innhold:

Lag1;Lag2:2-1
Lag2;Lag3:0-0
Lag1;Lag3:1-2
Lag3;Lag1:0-1
Lag2;Lag1:1-1
Lag2;Lag1:4-2

så skal funksjonen returnere en dictionary som ser slik ut:

{'Lag1': {'Lag2': ['2-1'], 'Lag3': ['1-2']}, 'Lag2': {'Lag3': ['0-0'], 'Lag1': ['1-1', '4-2']}, 'Lag3': {'Lag1': ['0-1']}}

Resultatene er lagret som en streng, hvor det første tallet er antall mål hjemmelaget har scoret, og det andre tallet er antall mål bortelaget har scoret.

Input-parameteren logger er en funksjon, som tar inn en streng, og returnerer ingenting (None). Den skal brukes til å logge feil i filen. Hvis den er None, så skal vi skrive til terminal med print.

Hvis det oppstår en feil ved lesning i en linje, så skal feilmeldingen logges, med kall til logger-funksjonen eller skrives til terminalen.

Hvis det oppstår en feil ved åpning av filen, så skal skal dette logges, og det returneres en tom ordbok. I motsatt fall logges det at filen er lest når alle linjene er lest.

OPPGAVE: Trekk kodelinjer til rett sted slik at de fire funksjonene virker som de skal. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker. Alle kodelinjene skal brukes.

Hjelp

{ }

else:

hjemmelag, bortelag = lagene.split(";")

logger(f"Ferdig å lese filen {filnavn}")

logger(e)

except Exception as e:

return hjkamper

lagene, resultat = linje.strip().split(":")

hjkamper[hjemmelag] = {}

if hjemmelag not in hjkamper:

try :

hjkamper[hjemmelag][bortelag] = []

hjkamper[hjemmelag][bortelag].append(resultat)

logger = print

for linje in f:

with open(filnavn, "r", encoding="utf-8") as f:

if bortelag not in hjkamper[hjemmelag]:

except IOError as e:

def les_kamper(filnavn, logger):

if not logger:

logger = print ✔

hjkamper = { } ✔

try : ✔

with open(filnavn, "r", encoding="utf-8") as f: ✔

for linje in f: ✔

try:

```
lagene, resultat = linje.strip().split(":")
hjemmelag, bortelag = lagene.split(";")
if hjemmelag not in hjkamper:
    hjkamper[hjemmelag] = {}
if bortelag not in hjkamper[hjemmelag]:
    hjkamper[hjemmelag][bortelag] = []
hjkamper[hjemmelag][bortelag].append(result
except Exception as e:
    logger(e)
except IOError as e:
    logger(e)
else:
    logger(f"Ferdig å lese filen {filnavn}")
return hjkamper
```

Maks poeng: 10

2

Her kan du kladde:

Useful Functions and Methods

These are listed in the following order:

- built-in (standard library)
  - often used functions and operators
  - exceptions
  - string methods
  - list operations
  - set operations
  - dictionary operations
  - files
  - pickle library (binary files)
- random library
- math library
- numpy library
- matplotlib.pyplot

Built-in:

f-string

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

%

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

//

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

len(s)

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

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

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

range(stop)

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

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

start: Starting number of the sequence.

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

step: Difference between each number in the sequence.

chr(i)

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

ord()

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

tuple(iterable)

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

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

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

Exceptions:

try:

Code to test

except:

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

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

else:

Runs if no exception occurs

finally:

Runs regardless of prior code having succeeded or failed.

String methods:

s.isalnum()

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

s.isalpha()

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

s.isdigit()

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

s.center(width)

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

s.ljust(width)

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

s.rjust(width)

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

s.lower()

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

s.upper()

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

s.strip()

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

s.strip(char)

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

s.split(str)

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

s.splitlines([keepends])

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

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

s.endswith(substring)

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

s.startswith(substring)

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

s.find(substring)

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

s.replace(old, new)

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

str.format(*args, **kwargs)

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

List operations:

s[i:j:k]

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

***item* in s**

Determine whether a specified item is contained in a list.

min(list)

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

max(list)

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

s.append(x)

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

s.insert(index,item)

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

s.index(*item*)

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

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

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

s.reverse()

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

s.sort()

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

Sets operations:

len(s)

Number of elements in set *s*

s.issubset(t)

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

s.issuperset(t)

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

s.union(t)

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

s.intersection(t)

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

s.difference(t)

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

s.symmetric_difference(t)

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

s.copy()

New set with a shallow copy of *s*

s.update(t)

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

s.add(x)

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

s.remove(x)

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

s.clear()

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

Dictionary operations:

d.clear()

Clears the contents of a dictionary

d.get(key, default)

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

d.items()

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

d.keys()

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

d.pop(key, default)

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

d.popitem()

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

d.values()

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

del d[k]

Deletes element k in d.

d.copy()

Makes a copy of d.

Files:

open()

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

f.read(size)

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

f.readline()

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

f.readlines()

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

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes the contents of a list to file

f.seek(offset, from_what)

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

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

f.tell()

Return the position of the file pointer.

f.close()

Close the file and free up any system resources taken up by the open file. If using **with open(filename) as ...** when you open the file, close() is not necessary, since it is implied by the ending of the with-block

Pickle Library:

pickle.dump(obj,file)

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

pickle.load(file_object)

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

Random Library:

random.random()

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

random.randint(a,b)

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

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.

indexing and slicing in arrays

for 1d arrays (vectors), indexing and slicing works the same way as for lists. For multi-dimensional arrays, there is a more compact index notation for arrays, e.g , if A is an array, A[i, j] is synonymous to A[i][j]. This more compact notation also allows more powerful slicing, for instance A[:,j] making a slice containing just column j of A (since colon : for the first index indicates all rows)

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.abs(x)

returns the absolute value of x. Unlike the abs() function in the standard library, which only works on single values, numpy.abs() can process an entire array, returning an array with the corresponding absolute values

numpy.sum(x)

calculates the sum of the values in the array x

numpy.prod(x)

calculates the product of the values in the array x

numpy.sin(x), numpy.cos(x), numpy.sqrt(x)

calculates the sinus, cosinus and the square root, similar to the same functions in the math library. However, while the math functions work just on single values, the numpy functions can also work on entire arrays, then returning arrays of the corresponding sine, cosine or square root values.

numpy.arange(start,stop,step)

creates a vector starting at *start*, ending at (but not including) *stop* using *step* between the values

numpy.linspace(start, stop, num)

creates a vector starting at *start*, ending at (and including) *stop*, and with *num* evenly spaced values in total

numpy.ndarray.reshape(lines,rows)

reshapes an ndarray according to the values in *lines* and *rows.* The reshape method does not change the original array, instead returning a reshaped copy

numpy.ndarray.resize(lines,rows)

reshapes an ndarray (similar to reshape method above), but resize changes the original array in place, rather than making a copy. Hence, this method returns None.

numpy.ndarray.tolist()

returns a list (of lists) with the same dimensions and content as the numpy array. Unlike list(A) which will only convert the first dimension of a multi-dimensional array to a list, A.tolist() makes a deep conversion, making all dimensions into lists

numpy.loadtxt(filename)

Loads data from text file, returning them as a numpy array.

numpy.savetxt(filename, A)

Saves array A to text file.

matplotlib.pyplot:

subplots()

Creates a *figure* containing a single *axes* object for plotting.

plot(x, y)

Plots a graph of y-values against x-values

bar(x, height)

Plots bars in positions given by vector *x*, with heights given by vector *height*

show()

Shows the plot on the screen

2.1 Skihopp 6%

I skihopp får man poeng basert på lengde og stil,

- **Lengdepoeng**: Hoppere får 60 poeng for å nå K-punktet (standard lengde i bakken). For hvert meter over K-punkt, legges det til 1.8 poeng, og for hver meter under trekkes det 1.8 poeng.
- **Stilpoeng**: Fem dommere gir stilpoeng på en skala fra 0 til 20. Det høyeste og laveste poenget fjernes, og de tre midterste poengene legges sammen.

Lag funksjonen **beregnPoeng()** som returnerer antall poeng for et gitt skihopp. Inputparametere til funksjonen er **lengde, stilpoeng og kpunkt.**

Stilpoengene kommer på formen **stilpoeng = [p1,p2,p3,p4,p5]** der elementene i listen er stilpoeng fra hver dommer.

Skriv ditt svar her

```
1
```

Maks poeng: 6

2.2 Generere passord 6%

Vi skal lage en enkel funksjon som genererer passord.

Følgende kriterier må være oppfylt:

- Passordet skal være minst 12 tegn.
- Passordet må inneholde både store og små bokstaver.
- Passordet må inneholde tall.
- Passordet må inneholde spesialtegn.

Lag kode for funksjonen **lagPassord(lengde)** som returnerer en tekststreng med et nytt tilfeldig passord av en bestemt lengde i henhold til reglene ovenfor. Om lengden som gis som input er under 12 skal den settes til 12 som er minimum.

Skriv ditt svar her

```
1
```

Maks poeng: 6

2.3 Kjennemerke for bil 6%

Lag en funksjon **nyttRegNr()** som genererer og returnerer et tilfeldig kjennemerke for bil. Type kjøretøy gis inn som parameter i funksjonen og kan være enten "gass", "elektrisk", "hydrogen" eller "fossil", gitt som en streng.

Kjennemerket, også kalt bilnummeret, skal bestå av to bokstaver etterfulgt et tall bestående av fem sifre. Bokstavene I, M, O, Q, Æ, Ø og Å brukes ikke på grunn av likheten med andre bokstaver eller sifre. 0 kan ikke være det første av de 5 sifrene i tallet.

Noen bokstavkoder er reservert. Dersom kjøretøyet drives med gass skal funksjonen sette bokstavkoden til GA, hydrogen til HY og elektrisk til EJ. Disse er ikke tilgjengelig om bitypen er fossil. Bokstavkoden CD er reservert ambassadebiler og skal ikke kunne genereres.

Skriv ditt svar her

```
1
```

Maks poeng: 6

2.4 Cæsarchiffer 6%

Innen kryptografi er cæsarchiffer, også kjent som Caesars chiffer, skiftchiffer, Cæsars kode og cæsarskift, en av de enkleste og mest kjente krypteringsteknikker. Det er en type substitusjonschiffer der hver bokstav i klarteksten erstattes med en annen bokstav et gitt antall steg lenger ut i alfabetet. Med et skift på tre steg erstattes D med A, E med B, F med C og så videre. Metoden har navn etter Julius Cæsar som benyttet denne koden til å kommunisere med sine generaler.

Lag funksjonen **krypter(tekst,skift)** som krypterer en tekst med cæsarchiffer. Inputvariabelen skift forteller hvor mange steg en bokstav forflytter seg i alfabetet.

Vår funksjon er noe forenklet og skal kun ta hensyn til bokstaver i det norske alfabetet. Funksjonen skal returnere den krypterte teksten med store bokstaver, og alle mellomrom skal være fjernet.

Eksempelvis vil **krypter("Kryptering er gøy",1)** returnere den krypterte teksten **LSZQUFSJOHFSHÅZ**

Skriv ditt svar her

```
1
```

Maks poeng: 6

2.5 Sjakkbrett 6%

Tenk deg et sjakkbrett med 8x8 ruter kodet som en 2-dimensjonal matrise.

De forskjellige brikkene er representert ved forskjellige bokstaver. For eksempel er kongene kodet med bokstaven "K". Vi bryr oss i denne oppgaven ikke om andre brikker, ei heller skiller vi mellom hvit eller sort konge.

Figuren under viser en visualisering av denne matrisen med 2 konger plassert i hver sin rute.

	A	B	C	D	E	F	G	H
8								
7								
6					K			
5								
4								
3								
2		K						
1								

Skriv en funksjon **opprettSjakkbrett()** som oppretter en 2-dimensjonal matrise og legger inn to konger på to tilfeldige plasser i matrisen. Figuren over er kun et eksempel og vi vet ikke i hvilke av rutene som kongene havner i. La funksjonen returnere matrisen.

Skriv en ny funksjon **finnKonger()** som løper igjennom den samme matrisen og finner alle rutene med konger. Det vil si ruter i den 2-dimensjonale matrisen som inneholder bokstaven "K".

La denne funksjonen returnere koordinatene til rutene hvor det befinner seg konger på følgende format:

Konge E6
Konge B2

Skriv ditt svar her

```
1
```

Maks poeng: 6