Eksamensforside

Institutt for datateknologi og informatikk
Eksamensoppgave i TDT4110 Informasjonsteknologi, grunnkurs

Dato: 25. november 2024 (15:00-19:00)
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.

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 Eksakt v2 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.

	Ikke eksakt	Heltall	Eksakt flyttall
42/1	○	○	○ ✔
13**20/1	○ ✔	○	○
0.2	○ ✔	○	○
1/8 + 10	○	○	○ ✔
5 * 10**20	○	○ ✔	○
1234/2**20	○	○	○ ✔

Maks poeng: 3

2 Regnerekkefølge v2 2%

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

a) **(6 + 2) * 4 - 3**3** (5)

b) **3 < 4 and 1 > 1** (False)

c) **False or 2 - 2 == 8 % 4** (True)

d) **6 - 4 == 2 and 3** (3)

Maks poeng: 2

3 Kodesnutt v2 2%

Hva blir resultatet av følgende kodesnutt?

```
arr = [3 , 2 , 5 , 6 , 0 , 7, 9]
add1 = 0
add2 = 0
for elem in arr:
    if (elem % 1 == 0):
        add1 = add1 + elem
        continue
    if (elem % 3 == 0):
        add2 = add2 + elem
print(add1 , end=" ")
print(add2)
```

Velg ett alternativ:

- 0 18
- 32 0 ✔
- 18 0
- 0 32

Maks poeng: 2

4 Frukt v2 2%

Hva blir resultatet av følgende kodesnutt?

```
1  i = -1
2  frukt = ['eple', 'banan', 'pære']
3  print(frukt[i])
```

Velg ett alternativ:

- IndexError
- pære ✔
- eple
- banan

Maks poeng: 2

5 Python programmering v2 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 += ord1[i]
9       else:
10          nyttord += ord2[i]
11  print(nyttord)
```

Velg ett alternativ:

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

Maks poeng: 2

6 Matrise v2 2%

Hvilket av følgende alternativer produserer resultatet 2 når A er definert som

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

```

Velg ett alternativ:

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

Maks poeng: 2

7 Synlighet v2 2%

Hva blir resultatet av følgende kodesnutt?

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

Velg ett alternativ:

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

Maks poeng: 2

8 Muterbarhet v2 2%

Hvilken av følgende er en ikke-muterbar datatype?

Velg ett alternativ:

- dict
- list
- set
- tuple ✔

Maks poeng: 2

9 Nøstet ordbok v2 2%

Hva blir korrekt syntaks for å skrive ut alderen til "Kristin" 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.c1.a['alder'])
- print(kunder['c1']['alder']) ✔
- print(kunder['c1'].alder)
- print(kunder.c1.alder)

Maks poeng: 2

10 Import v2 2%

Hva skriver du om du ønsker importere NumPy-modulen og gi det aliaset np ?

Velg ett alternativ:

- import numpy as np ✔
- import np as numpy
- import np from numpy
- from numpy import np

Maks poeng: 2

11 Funksjonen regnut() v2 2%

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

Hva blir resultatet av dette programmet?

Velg ett alternativ:

- 8
- 4
- Runtime Error
- Name Error ✔

Maks poeng: 2

12 Tallsekvens v2 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(2,7,4):
2        print(i)
```

OBS! Gale svar på denne oppgaven gir minuspoeng.

Velg ett eller flere alternativer

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

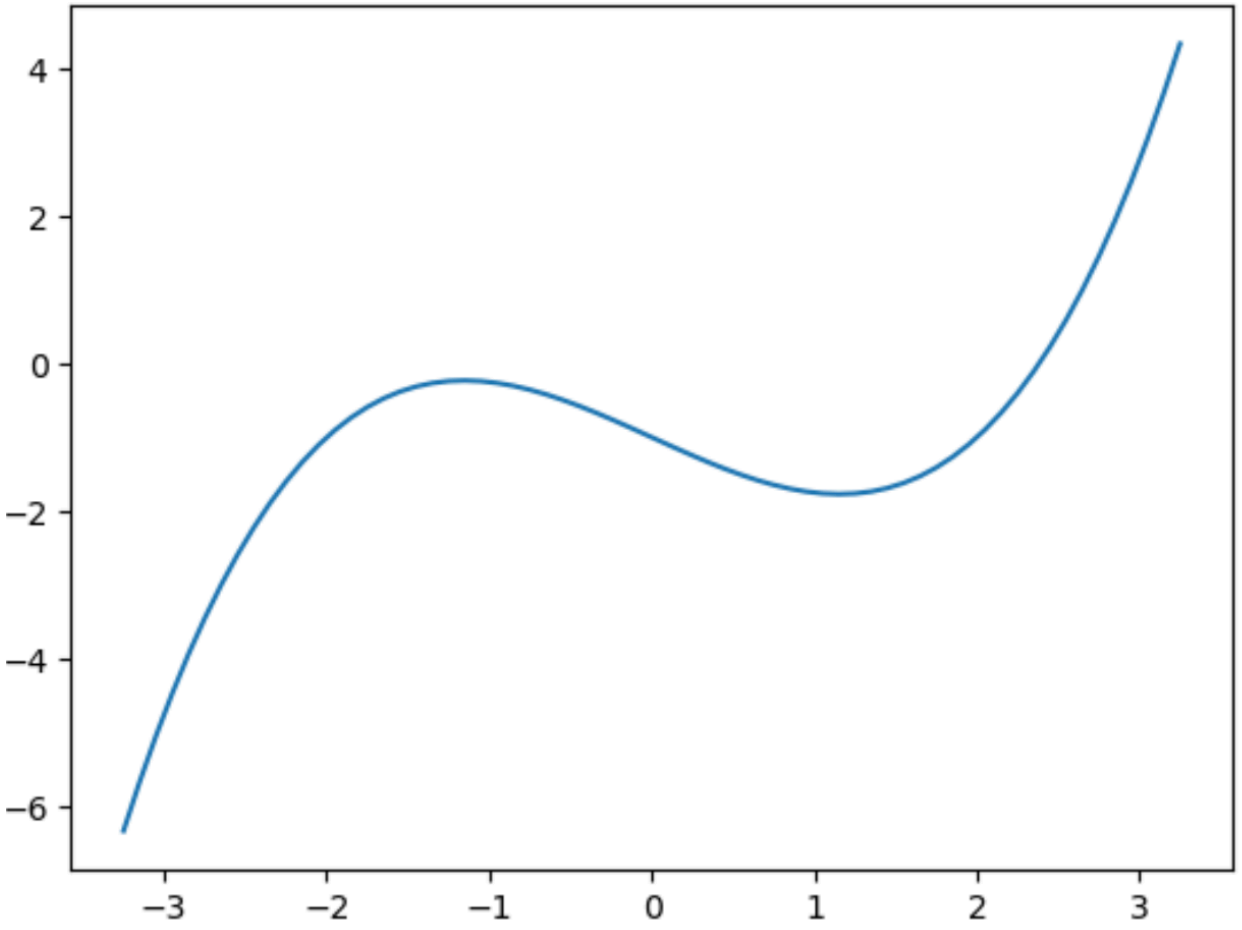
Maks poeng: 2

13 Plotte graf v2 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 -3.25 .. 3.25, som vist på figuren, dvs. startpunktet på den blå kurven er x = -3.25, y = -6.33 og sluttpunktet er x = 3.25, y = 4.33.



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

Hjelp

ax.plot(x, f(x))	x = np.arange(-3.25, 3.3, 0.1)
plt.show()	x = np.linspace(-3.25, 3.3, 100)
plt.subplots()	ax.plot(x, y)

fig, ax = plt.subplots() ✔

x = np.arange(-3.25, 3.3, 0.1) ✔

ax.plot(x, f(x)) ✔

plt.show() ✔

Maks poeng: 4

14 Iterere 2d array v2 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 **ingen rad** i arr inneholder både **c** og **d** samtidig, ellers **False.** Eksempel:

Hvis arrayet arr er
[[5 2 1]
[5 1 7]
[3 5 1]]

så skal

- check_array(arr, 2, 3) gi **True** - det fins riktignok en rad med tallet 2 og en med tallet 3, men ingen rad hvor begge tallene befinner seg sammen
- check_array(arr, 1, 2) gi **False** siden begge disse tallene 1 og 2 forekommer i samme rad (den øverste)

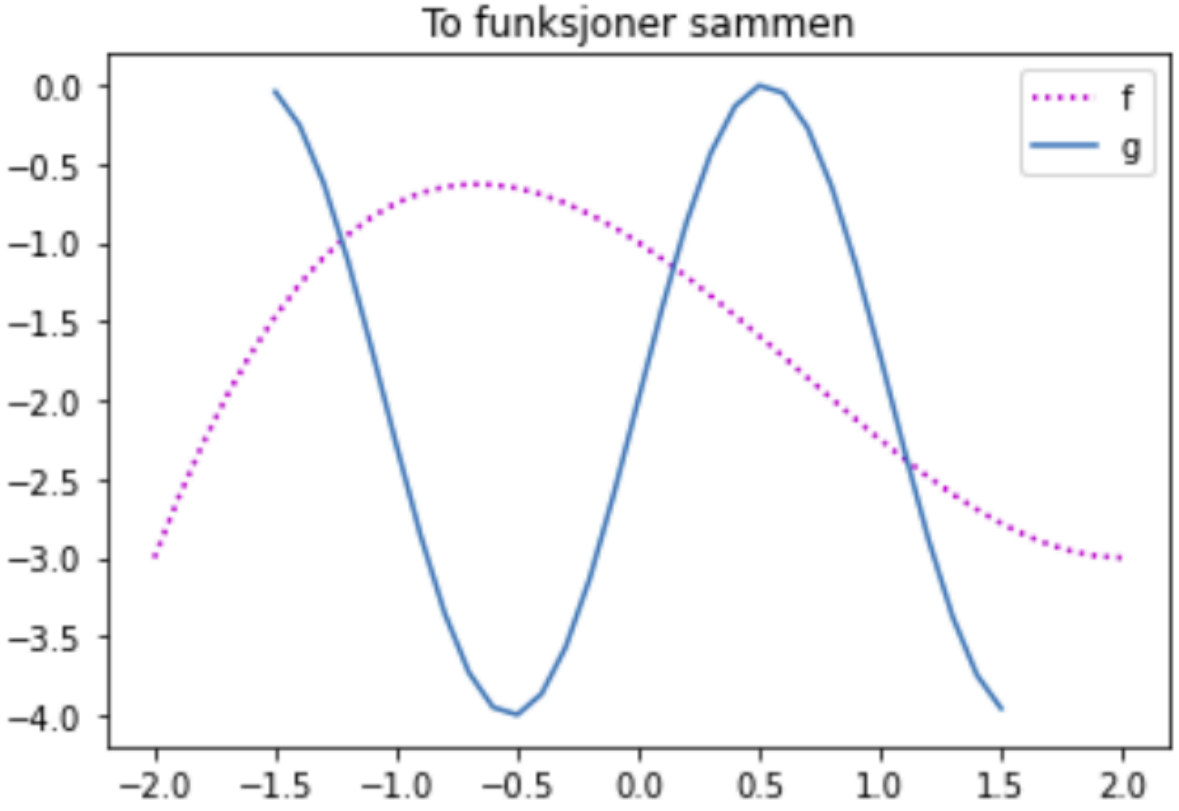
Velg riktig alternativ i hullene slik at funksjonen virker som den skal:

```
def check_array(arr, c, d):

    for i in range( Velg alternativ (a, d, len(arr))):
        has_c = has_d = False

        for j in range( Velg alternativ (arr.shape, len(arr[i]), len(arr))):
            if  arr[i,j] == c:

                Velg alternativ (has_c = False, has_c = True, has_c = has_d)
            if arr[i,j] == d:

                Velg alternativ (has_d = True, return True, return False)

        Velg alternativ (if has_c and has_d:, ok = input("Fortsette?"), if has_c or has_d:)

            Velg alternativ (break, return True, return False)

Velg alternativ (return True, print("OK"), return False)
```

Maks poeng: 7

15 Plotte to funksjoner v2 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, from numpy import *, import numpy)

```
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 (linspace(-2.0, 2.0, 31), arange(-2.0, 2.1, 0.5), linspace(-2.0, 2.0, 7))

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

Velg alternativ (# ingen kode trengs her, v = np.arange(-2.0, 2.05, 0.1), v = np.arange(-1.5, 1.55, 0.1))

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

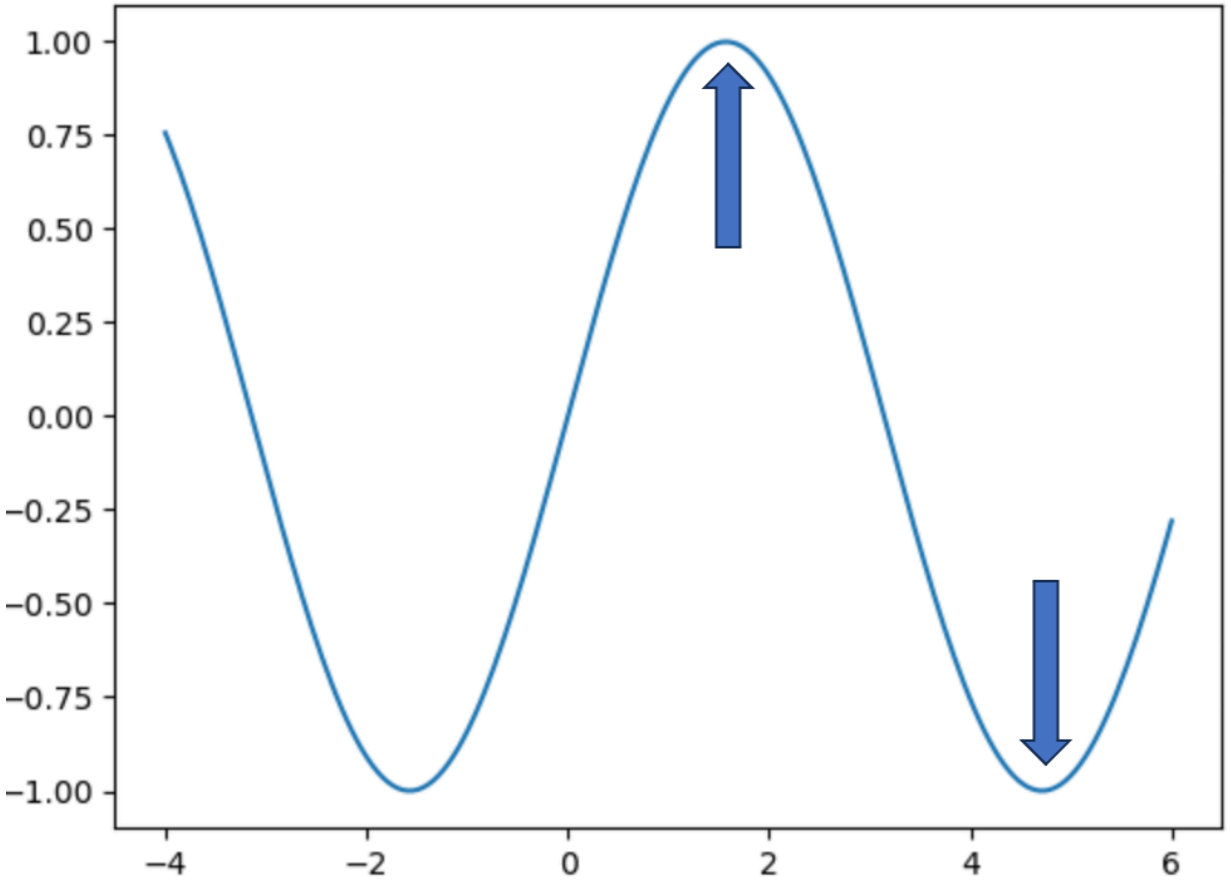
ax.legend(Velg alternativ (["f(x)", "g(x)"], ["f(x) , g(x)"], ["f", "g"]))
plt.title("To funksjoner sammen")
plt.show()

Maks poeng: 6

16 Funksjoner som parametre v2 8%

Vi ønsker en funksjon **longest_decrease(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 avtagende. 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(-4.0, 6.0, 0.01). Funksjonen skal da finne intervallet fra x = 1.58 til x = 4.71 siden dette er det lengste strekket hvor funksjonsverdien synker over arrayet x_values, fra toppunktet til bunnpunktet vist med blå piler (toppunktet for sinuskurve er på cirka x = 1.571 og bunnpunktet på x = 4.712, men stegverdien 0.01 på arange begrenser hvor nær vi kan komme de eksakte verdiene). Altså,
longest_decrease(np.sin(x), np.arange(-4.0,6.0,0.01)) skal returnere **(1.58, 4.71)**.

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



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

```
import numpy as np

def longest_decrease(f, x_values):
    result = ()
    current_start = None
    for i in range(1, len(x_values)):
```

Velg alternativ (if current_start is not None:, if f(x_values[i]) < f(x_values[i-1]):, if current start is None:)

Velg alternativ (if current_start is None:, if f(x_values[i]) >= f(x_values[i-1]):, if current_start is not None:)

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

Velg alternativ (current_start = None, current_start = x_values[i-1], current_start = x_values[i])

Velg alternativ (elif f(x_values[i]) > g(x_values[i]):, elif current_start is None:, elif i == len(x_values) - 1:)

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

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

Velg alternativ (elif current_start is not None:, elif f(x_values[i]) >= f(x_values[i-1]):, elif f(x_values[i]) < f(x_values[i-1]):)
current_start = x_values[i-1]
return result

Maks poeng: 8

17 Dessa dyra, dessa dyra 4%

Først et par detaljer du kan se på senere, forklaringen kommer under:

Dette er innholdet av tekstfilen 'dyreregistreringer.txt':

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

Denne er resultatet av å kalle metoden les_registreringer2 med et par ulike argumenter:

>>> print(les_registreringer2('dyreregistreringer.txt', ['ku', 'marsvin', 'svane']))
({'elg': [10, 5, 13, 1], 'maur': [103463]}, 4)

>>> print(les_registreringer2('dyreregistreringer.txt', ['maur', 'elg', 'bjørn']))
({'ku': [13, 16, 2], 'marsvin': [4, 12, 3], 'svane': [2]}, 4)

Nå skal du lese inn registreringer av dyrearter i Norge.
Slep og slipp elementene under for å kode funksjonen *les_registreringer2(...).*

- Funksjonen har to parametre:
  - filnavn: filnavnet som skal leses. Du kan forutsette at filen ligger i samme mappe.
  - dyr: En liste med strenger, som inneholder dyrearter vi IKKE vil ha med i registeret.
- 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 registreringene som omhandler dyr som er omtalt i listen 'dyr'.
- Du kan forvente at alle linjene i filen inneholder valide registreringer.
- Funksjonen skal returnere to ting: (se over for eksempel)
  - en dictionary med følgende struktur:
    - Nøklene skal være dyrene som vi ønsket å lese registreringene til. Se eksempel over.
    - Verdiene skal være en liste over de ulike registreringene til det spesifikke dyret, i rekkefølgen det ble lest inn.
  - Et heltall som markerer hvor mange linjer som ble ignorert

Hjelp

readlines()	readline()	[1::-1]	read()	extend
toList()	line	line	line.split()	ignored_lines
line.strip()	[1:]	[:1]	{}	append

```
def les_registreringer2(filnavn, dyr):
    result = {}
    ignored_lin = 0
    with open(filnavn, 'r') as file:
```

```
line = file. readline() ✔
while line ✔ :
    if line:
        parts = line.split() ✔
        dyrenavn = parts[0]
        registreringer = []
        for x in parts [1:] ✔ :
            registreringer. append ✔ (int(x))
        if dyrenavn not in dyr:
            if dyrenavn not in result:
                result[dyrenavn] = []
            result[dyrenavn]. extend ✔ (registreringer)
        else:
            ignored_lines += 1
    line = file.readline()

return result, ignored_lines
```

Maks poeng: 4

18 The spammer takes it all 6%

Slep og slipp feltene under for å løse funksjonen *print_more_first()* i henhold til spesifikasjonen gitt under:

- Funksjonen har én parameter, dictionarien fra forrige oppgave. Nøkkel er dyreart (streng), verdi er en liste med heltall. Eksempel: *({'ku': [13, 16, 2], 'marsvin': [4, 12, 3], 'svane': [2]}, 4)*.
- Funksjonen skal sortere observasjonene etter følgende struktur:
  - Først lavest snitt
  - Hvis samme snitt, først lavest antall observasjoner.

- Funksjonen skal skrive ut følgende linjer for alle dyrearter etter følgende mal:
  - <Dyreart>: <x> observasjoner med et snitt på <y> individer per observasjon (<z> flere individer enn <dyreslaget over>.) Se eksempel under
  - Siste punkt i punktet over skal bare med hvis dette ikke er første linje, så det ER et dyreslag over.

Eksempel:
>>> r = {'ku': [13, 16, 2], 'marsvin': [4, 12, 3], 'svane': [2]}
>>> print_more_first(r)
svane: 1 obs, 2.00/obs.
marsvin: 3 obs, 6.33/obs (4.33 flere individer/obs enn svane).
ku: 3 obs, 10.33/obs (4.00 flere individer/obs enn marsvin).

def print_more_first(registrations):

another_variable = Velg alternativ (list, {}, len(registrations), set())

for Velg alternativ (feels_important, feels_important, yeah, yeah, feels_important, yeah) in

registrations Velg alternativ ([::-1], .items(), .keys(), .values()):

Velg alternativ (dunno +, yeah, dunno, recollection)= Velg alternativ (len(yeah), sum(yeah), sum(feels_important), len(feels_important)) / len(yeah)

Velg alternativ (another_variable[feels_important], yeah, another_variable[-1], another_variable[dunno]) = (dunno, len(yeah))

whatsthis = list(Velg alternativ (another_variable.keys(), another_variable.unique(), another_variable.values(), another_variable.items()))

for i in range(Velg alternativ (len(whatsthis), len(another_variable), len(set(whatsthis)),

```
len(feels_important[whatsthis])):
        for j in range(i + 1, len(whatsthis)):
            if another_variable[whatsthis[i]][0] > another_variable[whatsthis[j]][0] or \
               (another_variable[whatsthis[i]][0] == another_variable[whatsthis[j]][0] \
                and another_variable[whatsthis[i]][1] > another_variable[whatsthis[j]][1]):
                whatsthis[i], whatsthis[j] = whatsthis[j], whatsthis[i]


    recollection = Velg alternativ (whatsthis.keys(), None, True, list(dunno))
    for feels_important in whatsthis:


        Velg alternativ (dunno, count, dunno, count, dunno +)=
another_variable[feels_important]


        if recollection Velg alternativ (>= count[-1], not in whatsthis, in whatsthis, is None):
            print(f"{feels_important}: {count} obs,  {dunno:.2f}/obs.")
        else:
            print(f"{feels_important}: {count} obs, {dunno:.2f}/obs ({dunno - recollection:.2f} flere
individer/obs enn {previous_animal}).")


        recollection = Velg alternativ (previous_animal, dunno, count, min(feels_important))
        previous_animal = feels_important
```

Maks poeng: 6

19 Les anmeldelser 10%

Vi skal lese anmeldelser av spisesteder, skrevet som linjer i en fil, og lagre dem i en strukturert form i en ordbok.

Hver linje i filen inneholder informasjon om en anmeldelse, skrevet som:
`spisested;anmelder:dato;vurdering`

- `spisested` er navnet på stedet som anmeldes.
- `anmelder` er navnet på personen som har skrevet anmeldelsen.
- `dato` er datoen for besøket, skrevet som `YYYY-MM-DD`.
- `vurdering` er en score fra 1 til 5 som anmelderen har gitt stedet.

Eksempel på innhold i en fil:

Pastahuset;Ola:2023-05-14;5
Bareburger;Kari:2023-06-01;4
Kyllingen;Kari:2023-06-10;3
Fat Goose;Ola:2023-07-01;4
Pølseboden;Per:2023-07-03;3
Sim Salat;Ola:2023-06-15;5

Du skal skrive en funksjon `les_anmeldelser(filnavn, prosessor)` som leser alle radene i filen og lagrer anmeldelsene i en ordbok, som returneres fra funksjonen.
Elementene i ordboken skal ha spisestedene som nøkkel. Verdien er en ny ordbok hvor nøklene er navnene på anmelderne som har anmeldt stedet. Verdiene til denne er en liste med tupler som inneholder datoen for besøket og vurderingen anmelderen har gitt til stedet.

For eksempel-filen over, så skal funksjonen returnere en ordbok som ser slik ut:
{'Pastahuset': {'Ola': [('2023-05-14', '5')]}, 'Bareburger': {'Kari': [('2023-06-01', '4')]}, 'Kyllingen': {'Kari': [('2023-06-10', '3')]}, 'Fat Goose': {'Ola': [('2023-07-01', '4')]}, 'Pølseboden': {'Per': [('2023-07-03', '3')]}, 'Sim Salat': {'Ola': [('2023-06-15', '5')]}}

Input-parameteren `prosessor` er en funksjon som tar inn en streng og returnerer en streng. Denne skal brukes til å prosessere anmeldelser, for eksempel ved å gjøre tekstjusteringer eller filtrering. Hvis `prosessor` er `None`, skal vi bruke en identitetsfunksjon som returnerer teksten uendret. Du kan anta det finnes en slik funksjon som heter **identitet.**
Hvis det oppstår en feil ved lesning av en linje, skal feilmeldingen logges til terminalen med `print()` og så fortsette å lese og prosessere filen.

Hvis det oppstår en feil ved åpning av filen, skal dette logges, og det returneres en tom ordbok. Hvis filen leses uten feil, skal funksjonen også skrive ut at filen er ferdig 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

anmeldelser[spisested][anmelder].append((dato, vurdering))

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

linje_del1, linje_del2 = linje.split(":")

dato, vurdering = linje_del2.split(";")

anmeldelser[spisested] = {}

{ }

return {}

linje_del1.split(";")

linje = prosessor(linje.strip())

print(f"Feil ved lesning av linje: {linje.strip()} - {e}")

except ValueError as e:

prosessor = identitet

anmeldelser[spisested][anmelder] = []

if anmelder not in anmeldelser[spisested]:

for linje in f:

try

if spisested not in anmeldelser:

return anmeldelser

except IOError as e:

def les_anmeldelser(filnavn, prosessor):

if prosessor == None:

prosessor = identitet ✔

anmeldelser = { } ✔

try ✔

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

for linje in f: ✔

try ✔

linje = prosessor(linje.strip()) ✔

linje_del1, linje_del2 = linje.split(":") ✔

spisested, anmelder =

linje_del1.split(";") ✔

dato, vurdering = linje_del2.split(";") ✔

if spisested not in anmeldelser: ✔

anmeldelser[spisested] = {} ✔

if anmelder not in anmeldelser[spisested]: ✔

anmeldelser[spisested][anmelder] = [] ✔

anmeldelser[spisested][anmelder].append((dato, vurderin ✔

except ValueError as e: ✔

print(f"Feil ved lesning av linje: {linje.strip()} - {e}") ✔

except IOError as e: ✔

print(f"Feil ved åpning av filen: {filnavn} - {e}")

return {} ✔

else:

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

return anmeldelser ✔

Maks poeng: 10

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

20 Terningkast 6%

Lag en funksjon **kastTerninger()** som starter med å trille 5 terninger.

Funksjonen skal simulere flere omganger der vi triller flere terninger. Etter hver omgang tar vi vare på alle de terningene som viser 6. Når alle terningene har fått verdien 6 returnerer funksjonen antall omganger som måtte til.

Dersom vi har veldig flaks og får 5 6-ere i første omgang vil funksjonen returnere 1.

Normalt sett vil det i hver omgang være enten ingen eller noen seksere å ta vare på og dermed blir det etterhvert færre og færre terninger å kaste til alle har blitt seksere.

Skriv ditt svar her

```
1
```

Maks poeng: 6

21 Komprimering 6%

Komprimering av tekststrenger er en vanlig oppgave hvor målet er å redusere lengden på tekststrengen ved å erstatte repeterende tegn i teksten med tegnet etterfulgt av antall gjentagelser.

For eksempel vil tekststrengen **'aabccccccaaa'** kunne komprimeres til **'a2b1c5a3'**.

Lag funksjonen **komprimer()** som komprimerer en tekststreng i henhold til beskrivelsen over. Input til funksjonen skal være en klartekst (ukomprimert tekst) og funksjonen skal returnere den komprimerte tekststrengen.

Skriv ditt svar her

```
1
```

Maks poeng: 6

22 Sjakkbrett v2 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 bøndene kodet med bokstaven "B". Vi bryr oss i denne oppgaven ikke om andre brikker, ei heller skiller vi mellom hvite eller sorte brikker.

Figuren under viser en visualisering av denne matrisen med 16 bønder plassert i sine startposisjoner i hver sin rute.

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

Skriv en funksjon **opprettSjakkbrett()** som oppretter en 2-dimensjonal matrise og legger inn bøndene i matrisen. La funksjonen returnere matrisen.

Skriv en ny funksjon **flyttBonde()** som utfører det første trekket for en tilfeldig bonde. Når man flytter en bonde første gang kan den flytte seg enten en eller to plasser forover på brettet. Det vil si at en bonde på rad 2 kan flytte enten til rad 3 eller 4 og en bonde på rad 7 kan flytte enten til rad 5 eller rad 6.

La denne funksjonen returnere trekket som utføres. Når en bonde flyttes skriver man bare feltet den går til. Hvis en bonde står på e2 og går til e4 skriver man bare ut e4.

Skriv ditt svar her

```
1
```

Maks poeng: 6

23 Kinobestilling 6%

Du skal lage en funksjon **velgSeter()** for å velge seter i en kinosal som har 9 rader og 15 seter på hver rad.

Setene i salen skal lagres i en matrise i programmet. Før hver forestilling nullstilles matrisen så vi kan si at utgangspunktet for denne funksjonen er at matrisen er fyllt med tegnet 'L' for å indikere at setet er ledig.

Funksjonen skal kunne ta hensyn til at man bestiller kinobilletter som gruppe og tar derfor inn antallet i gruppen som variabel.

Ved en bestilling søker funksjonen **velgSeter()** etter ledige seter og finner de beste ledige plassene for gruppen slik at de kan sitte samlet. De beste plassene er nærmest midten av salen. Når plassene er funnet, settes disse til 'R' for reservert i matrisen.

Retur fra funksjonen skal være en liste med seter, eller en tom liste dersom det ikke er plass til at gruppen kan sitte samlet på en rad.

Eksempel på kjøring:
print(velgSeter(4))
['Rad5 Sete6', 'Rad5 Sete7', 'Rad5 Sete8', 'Rad5 Sete9']

Skriv ditt svar her

```
1
```

Maks poeng: 6

24 Tverrsum 6%

Tverrsum er et begrep innen matematikken der man summerer alle sifrene i tallet og får en sum. For eksempel har tallet 1952 tverrsummen 1+9+5+2 = 17.

Lag en funksjon **tverrsum(a,b)** som finner tversummen av alle tall i intervallet a til b. Alle tverrsummene skal skrives til skjerm.

Dette er utskriften for tallene fra 1 til 100:
1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 3, 4, 5,
6, 7, 8, 9, 10, 11, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1

Programmet skal også telle antall av hver tverrsum og returnere en liste av tupler slik at man ser både tversummen og dets frekvens. Dette blir et lite histogram som sier noe om hvor ofte hvert enkelt tversum inntreffer.

Tuplene for samme intervall som over blir:
[(1, 3), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9), (9, 10), (10, 9), (11, 8), (12, 7), (13, 6), (14, 5), (15, 4), (16, 3), (17, 2), (18, 1)]

Skriv ditt svar her

```
1
```

Maks poeng: 6