

i **Kopi av Forside**

Institutt for Datateknologi og Informatikk

Eksamensoppgave i TDT4110 Informasjonsteknologi Grunnkurs

Eksamensdato: 8. desember 2022

Eksamenstid (fra-til): 09:00-13:00

Hjelpemiddelkode/Tillatte hjelpemidler: D

Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Faglig kontakt under eksamen: Dag Olav Kjellemo, Guttorm Sindre

Tlf.: 47681639

Tlf.: 94430245

Faglig kontakt møter i eksamenslokalet: NEI

ANNEN INFORMASJON:

Skaff deg overblikk over oppgavesettet før du begynner på besvarelsen din.

Les oppgavene nøye, gjør dine egne antagelser og 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 ønsker å kontakte faglærere. Noter gjerne spørsmålet ditt på forhånd.

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

Varslinger: Hvis det oppstår behov for å gi beskjeder til kandidatene underveis i eksamen (f.eks. ved feil i oppgavesettet), vil dette bli gjort via varslinger i Inspira. 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: Blir du syk under eksamen, eller av andre grunner 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 i arkivet i Inspira. Merk at det kan ta én virkedag før eventuelle håndtegninger vil være tilgjengelige i arkivet.

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 $a \leq N \leq b$.

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 $a \leq N \leq b$ for $a \leq b$ and $b \leq N \leq a$ 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 $e = 2.718281\dots$ 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 $\pi = 3.141592\dots$, to available precision.

math.e

The mathematical constant $e = 2.718281\dots$, 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 Kopi av Ada Python (4%)

Anta at vi har koden

```
s = 'Ada'
t = 'Python'
```

I tabellen nedenfor står en del uttrykk, hvorav noen vil kunne beregnes og få en verdi, mens andre vil gi feilmelding.

For hver rad, se på uttrykket til venstre, og kryss av hvilken datatype resultatet vil bli - eller velg ERROR dersom uttrykket vil gi feilmelding. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker.

	float	string	ERROR	bool	int
4 / 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7 // 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3 % 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2 * 5.0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
s * 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
s * t	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3 == 3.0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
t[2.0]	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 5.6

2 Kopi av exact (3%)

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

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

	Eksakt	IKKE eksakt
-1.6	<input type="radio"/>	<input type="radio"/>
29824587928897	<input type="radio"/>	<input type="radio"/>
0.4	<input type="radio"/>	<input type="radio"/>
24	<input type="radio"/>	<input type="radio"/>
2.25e-5	<input type="radio"/>	<input type="radio"/>
0.75	<input type="radio"/>	<input type="radio"/>

Maks poeng: 6

3 Kopi av myst (5%)

Gitt kode for funksjonen **myst()** samt fem print-setninger hvor det gjøres kall til myst, nummerert #1-#5 med kommentarer nede til høyre (**NB!** merk #-nummerne, så du ikke tar feil av hvilken du svarer på)

```
def myst(strg):
    grg = 'X'
    for i in range(len(strg)):
        if strg[i].isdigit():
            grg += strg[i-1] * (int(strg[i])-1)
        else:
            grg += strg[i]
    return grg

print(myst(''))           #1
print(myst('Python'))    #2
print(myst('Pyt2on'))    #3
print(myst('Py3on'))     #4
print(myst('Py-1on'))    #5
```

Radene #... i tabellen viser til hver sin av de nummererte print-setningene over. For hver print-setning, kryss av i riktig kolonne for hva den kommer til å printe. (**NB!** rekkefølgen på radene i tabellen kan være omstokket i forhold til rekkefølge i koden, se nøye på nummerne bak #-tegnet). Ingen minuspoeng for feil svar, du **BØR** svare noe også der du er usikker.

	X	XPy-on	XX	XPython	XPy3on	XPy-1on	XPytton	XPyyyon
#3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
#2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
#1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
#4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
#5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 5

4 Kopi av data.txt (5%)

Gitt funksjonen `get_data()` som vist under, samt opprettelse av to strengvariable `fil` og `f2`. Anta at tekstfila `data.txt` har innhold som vist i tekstboksen nede til høyre i bildet, og at `data.txt` er *den eneste* .txt-fila som ligger på samme katalog som programkoden vår.

```
def get_data(filename, row, col):
    try:
        with open(filename) as f:
            L = f.readlines()
            if row < len(L):
                return float(L[row].split()[col])
            else:
                return -11
    except FileNotFoundError:
        return -22
    except (IndexError, ValueError, TypeError):
        return -33
```

```
fil = 'data.txt'
f2 = 'more_data.txt'
```

```
mandag  2.3 4.5 3.1
tirsdag 1.1 1.0
onsdag   0.7 xxx 3.2
```

I tabellen viser hver rad ett kall av funksjonen og kolonnene viser mulige returverdier.

For hver rad, sett kryss i kolonnen som viser riktig returverdi for kallet. Ingen minuspoeng for feil svar, du BØR svare noe også der du er usikker.

	3.2	-22	2.3	4.5	-11	-33	1.0	0.7	xxx
<code>get_data(fil, 1, -1)</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>get_data(fil, 3, 0)</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>get_data(fil, 0, 1)</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>get_data(fil, 2, 2)</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>get_data(f2, 0, 1)</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>get_data(fil, 1, 3)</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<code>get_data(fil, 2, -3)</code>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 5

5 Kopi av rounding errors (5%)

Bildet viser noe Python-kode i den grå boksen, samt resultat av kjøring under boksen. Anta at variablene a og b er vanlig 64-bits float, som printes med 19 desimaler i kodelinje 3 og 4.

$a^2 - b^2$ er matematisk ekvivalent med $(a+b)(a-b)$

De to printene linje 8 og 9 skulle derfor ha gitt identisk resultat - men på grunn av avrundingsfeil blir de ulike etter de 6 første desimalene. Printen i linje 11 regner ut på en annen måte det som skal være det eksakt riktige svaret. Som man kan se, indikerer dette at uttrykket i linje 9, $(a+b)(a-b)$ gir nøyaktig svar, mens uttrykket $(a^2 - b^2)$ gir et unøyaktig svar.

```

1 a = 1 + 1/2**27
2 b = 1 + 1/2**28
3 print(f'{a:.19f}')
4 print(f'{b:.19f}')
5 print(f'{a**2:.19f}')
6 print(f'{b**2:.19f}')
7 print('Disse to skulle vært like:')
8 print(a**2 - b**2)
9 print((a+b) * (a-b))
10 print('Riktig svar:')
11 print(f'{0.5**27 * (1 + 0.5**28 + 0.5**29)}')
```

1.0000000074505805969

1.0000000037252902985

1.0000000149011611938

1.0000000074505805969

Disse to skulle vært like:

7.450580596923828e-09

7.450580638557192e-09

Riktig svar:

7.450580638557192e-09

Hva er riktig og feil om det vi observerer med denne koden? Ingen minuspoeng for feil svar, så du **BØR** svare noe også der du er usikker.

	RIKTIG	FEIL
Avviket mellom resultatene skyldes underflyt	<input type="radio"/>	<input type="radio"/>
Avviket mellom de to resultatene skyldes addisjon av tall av svært ulik størrelsesorden	<input type="radio"/>	<input type="radio"/>
Avviket mellom de to resultatene skyldes subtraksjon av to nesten like tall som begge er unøyaktige pga avrundingsfeil	<input type="radio"/>	<input type="radio"/>
Avviket mellom resultatene skyldes overflyt	<input type="radio"/>	<input type="radio"/>
Subtraksjonen $a^2 - b^2$ (linje 8) er sannsynlig årsak til problemet, selv om differansen $a - b$ (linje 9) er enda mindre.	<input type="radio"/>	<input type="radio"/>
Verdiene for a og b som sådan er IKKE kilde til avrundingsfeil, siden de kan representeres eksakt i binært tallsystem	<input type="radio"/>	<input type="radio"/>

Maks poeng: 10

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 $a \leq N \leq b$.

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 $a \leq N \leq b$ for $a \leq b$ and $b \leq N \leq a$ 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 $e = 2.718281\dots$ 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 $\pi = 3.141592\dots$, to available precision.

math.e

The mathematical constant $e = 2.718281\dots$, 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

6 Kopi av sum_except (6%)

Funksjonen **sum_except(numlist, n)** skal få inn ei liste med heltall (numlist) og et heltall n. Den skal returnere summen av tallene i lista, **unntatt** eventuelle forekomster av tallet n.

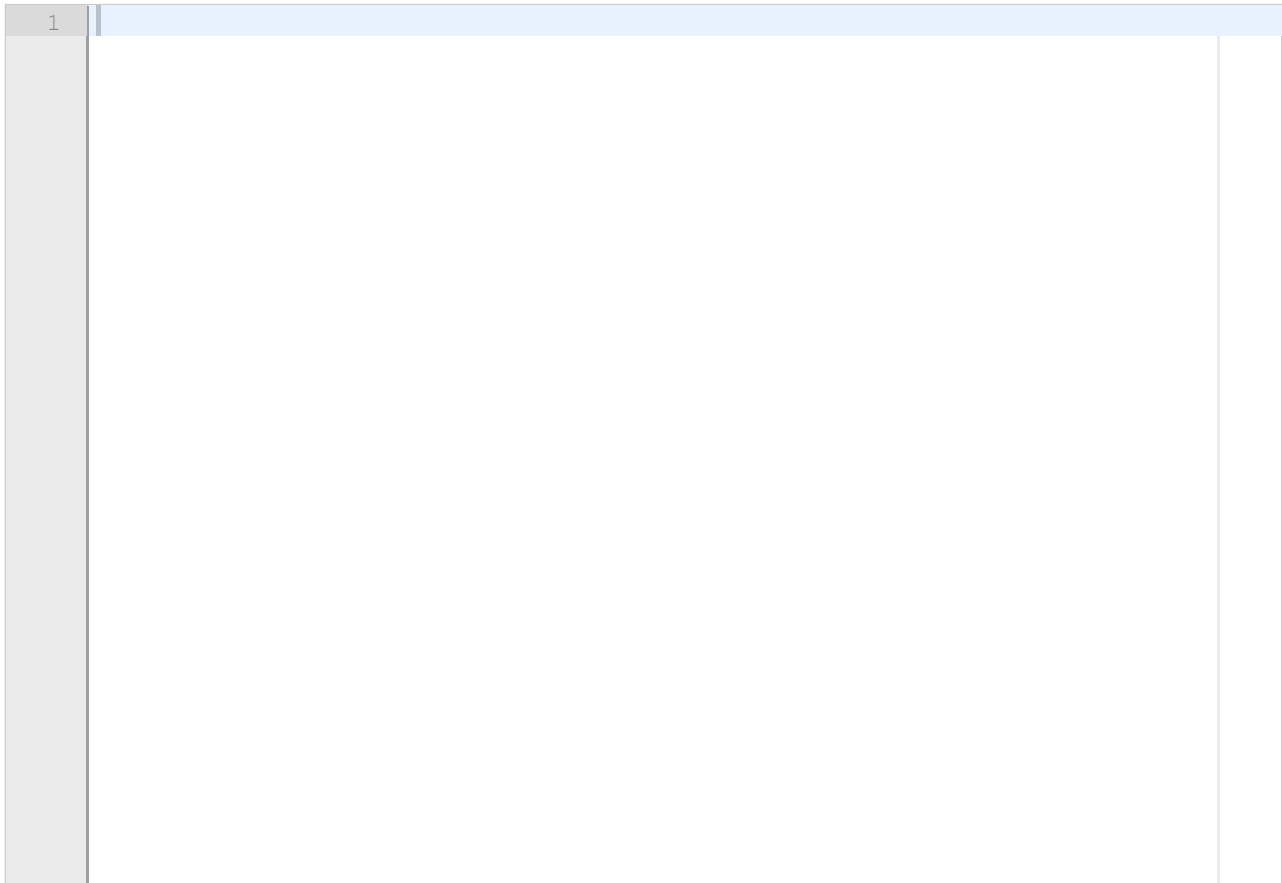
Eksempel på kjøring:

```
>>> sum_except([3, 4, 3, 7 ], 3)
```

```
11
```

Dette kallet returnerer 11 fordi de to 3-tallene i lista ikke blir med i summen, slik at resultatet blir 4+7.

Skriv koden for funksjonen sum_except()



Maks poeng: 6

7 Kopi av football (7%)

Regler for størrelse på fotballbaner er som følger:

- lengde må være mellom 90 og 120 meter
- bredde må være mellom 45 og 90 meter

Hvis banen skal brukes i internasjonale kamper, er kravene mer restriktive:

- lengde 100-110 meter
- bredde 64-75 meter

Lag funksjonen **ok_size(length, width, intl)** hvor de to første parametrene er henholdsvis lengde og bredde (flyttall), og den siste parameteren er True hvis kampen er internasjonal, ellers False. Funksjonen skal returnere True hvis banens størrelse er akseptabel, ellers False. En størrelse som er akkurat på grensa vil være ok (f.eks. lengde 90.0 meter) men såvidt utenfor grensa er **ikke ok** (f.eks. lengde 89.99 meter er uakseptabelt).

Skriv koden for funksjonen ok_size()

1	
---	--

Maks poeng: 7

8 Kopi av car_club(5%)

En klubb med bilentusiaster holder spesielt rede på hvilke sjeldne bilmerker medlemmene har kjørt. Medlemmene sine bilmerker er registrert i hver sin mengde-variabel (set). Vi ønsker å lage fire funksjoner:

- **a_and_b(A, B)** skal returnere mengden av bilmerker som både medlem A og medlem B har kjørt
- **a_butnot_B(A, B)** skal derimot returnere mengden av bilmerker som A har kjørt, men B ikke har kjørt
- **common_cars(car_geeks)** skal returnere mengden av bilmerker som har det felles at hvert eneste medlem i klubben har kjørt dem
- **all_cars(car_geeks)** skal returnere mengden av bilmerker som har vært kjørt av minst ett klubbmedlem

Eksempel på kjøring er vist med testkode i den grå boksen og mengdene som blir returnert like under den grå boksen.

```
21 dan = {'Effa', 'Ultima', 'Piech'}
22 bo = {'Zenvo', 'Ultima', 'Piech'}
23 leah = {'Effa', 'Piech', 'Denza'}
24 car_geeks = [dan, bo, leah]
25 print(a_and_b(dan, leah))
26 print(a_butnot_b(dan, leah))
27 print(common_cars(car_geeks))
28 print(all_cars(car_geeks))
```

```
{'Effa', 'Piech'}
{'Ultima'}
{'Piech'}
{'Ultima', 'Effa', 'Denza', 'Zenvo', 'Piech'}
```

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.

 [Hjelp](#)

car_geeks[0]

set()

for B in car_geeks:

result.union(B)

for B in car_geeks[1:]:

A.intersection(B)

A.difference(B)

result.intersection(B)

def a_and_b(A, B):

result =

return result

def a_butnot_b(A, B):

result =

return result

def common_cars(car_geeks):

result =

result =

return result

def all_cars(car_geeks):

result =

result =

return result

Maks poeng: 5

9 Kopi av flip (5 %)

Vi er gitt en liste med lister (dvs. 2d liste). De indre listene består av heltall. Vi ønsker å lage en tilsvarende 2d liste hvor en del av verdiene er endret fra x til $m-x$.

Vi skal lage to funksjoner for dette:

flip_values(L, m) og **values_flipped(L, m)**

I begge tilfeller er input en liste L bestående av lister av heltall, og en verdi m.

Listene i L med partalls-indeks (dvs. L[0], L[2], osv.) skal **ikke** endres.

Listene i L med oddetalls-indeks (dvs. L[1], L[3] osv.) skal endres slik at hver verdi x erstattes med $m - x$.

flip_values skal *mutere* L og listene i L, dvs. endre selve den originale 2d-lista.

values_flipped skal lage ei ny 2d-liste hvor tallene er endret, men uten å endre den originale.

Eksempel på kjøring:

```
L = [ [1,2,3,4,5,6,6], [1,2,3,3,5,4,7], [7,1,3]]
```

```
print(values_flipped(L,8))    # prints a modified copy of the list
```

```
print(L)                     # prints the original list
```

```
flip_values(L,8)              # mutates the original list
```

```
print(L)                     # prints mutated original list
```

Gjør at følgende printes:

```
[[1, 2, 3, 4, 5, 6, 6], [7, 6, 5, 5, 3, 4, 1], [7, 1, 3]]
```

```
[[1, 2, 3, 4, 5, 6, 6], [1, 2, 3, 3, 5, 4, 7], [7, 1, 3]]
```

```
[[1, 2, 3, 4, 5, 6, 6], [7, 6, 5, 5, 3, 4, 1], [7, 1, 3]]
```

Som vi ser er midterste indre liste (som har oddetallsindeks [1]) blitt endret i første og tredje print, men ikke i andre print, hvor det er den originale lista vi printer før den muteres.

OPPGAVE: Trekk kodelinjer til rett plass slik at de to funksjonene virker som de skal. Ingen minuspoeng for feil, så du BØR plassere noe også der du er usikker.

 [Hjelp](#)

L[i][j] = m - L[i][j]

else:

for j in range(len(L[i])):

return new_L

new_L = []

new_L.append(L[i][:])

for i in range(len(L)):

for i in range(1,len(L),2):

if i % 2 == 0:

new_L.append([m - x for x in L[i]])

```
def flip_values(L, m):
```

```
def values_flipped(L, m):
```

Maks poeng: 5

10 Kopi av chg_file (5%)

Vi har ei tekstfil med tall over flere linjer, innad i hver linje adskilt med mellomrom. Fila har like mange tall på hver linje, og kan inneholde både positive og negative tall. Vi ønsker å lage en funksjon som får inn filnavn som parameter og skal endre fila på følgende måte:

- alle tall som er mindre enn 100 i absoluttverdi skal være positive
- alle tall større eller lik 100 i absoluttverdi, skal være negative

For øvrig skal filas innhold være som før. Eksempel på kjøring, hvis fila tall.txt har innhold:

```
101.2 93.4 213.9 -23.5 -2.0
```

```
22.5 -100.7 -3.2 55.5 -101.5
```

så skal kjøring av **change_file('tall.txt')** gjøre at fila etter kjøring har innhold:

```
-101.2 93.4 -213.9 23.5 2.0
```

```
22.5 -100.7 3.2 55.5 - 101.5
```

OPPGAVE: Trekk kodelinjer til riktig sted så funksjonen virker som den skal. 5 av kodelinjene skal ikke brukes. Ingen minuspoeng for feil svar, så du **BØR** svare noe også der du er usikker.

 [Hjelp](#)

<code>A[i,j] = -A[i,j]</code>	<code>A[i,j] = abs(A[i,j])</code>
<code>np.savetxt(filename, A)</code>	<code>for i in range(len(A)):</code>
<code>np.loadtxt(A)</code>	<code>for j in range(len(A[i])):</code>
<code>if A[i,j] > 100 or -100 < A[i,j] < 0:</code>	<code>A = np.abs(A)</code>
<code>A = np.loadtxt(filename)</code>	<code>A = np.savetxt(filename)</code>
<code>if abs(A[i,j]) >= 100:</code>	

```
import numpy as np
```

```
def change_file(filename):
```

Maks poeng: 5

11 Kopi av same_numbers (6%)

Funksjonen `same_numbers_row_col(M, order)` får inn to parametre:

- et numpy array **M**
- **order**, som er True eller False

Funksjonen skal returnere indekspar [rad, kolonne] for tilfeller hvor det er slik at ei rad og ei kolonne i arrayet inneholder de samme tallene. Hvis order er True, skal det bare returneres indekspar hvor tallene står i eksakt samme rekkefølge. Med order False kan tallene også stå i annen rekkefølge.

Eksempel på kjøring:

```
11 import numpy as np
12 T = np.array([[1.0, 2.0, 3.5, 1.0],
13              [2.0, 1.0, 2.5, 2.0],
14              [2.0, 1.0, 2.5, 3.5],
15              [3.5, 3.5, 2.5, 1.0]])
16
17 print(same_numbers_row_col(T, True))
18 print(same_numbers_row_col(T, False))
```

```
[[0, 3]]
[[0, 1], [0, 3]]
```

Som eksemplet viser:

- Hvis **order** er **True** (første linje i resultat), returneres kun indeksparet [0,3] siden både rad [0] og kolonne [3] i arrayet T har akkurat tallsekvensen 1.0, 2.0, 3.5, 1.0
- Hvis **order** er **False** (andre linje i resultat) returnes dessuten [0,1] siden kolonne [1] også har de fire samme tallene, men i en annen rekkefølge 2.0, 1.0, 1.0, 3.5

OPPGAVE: Fyll inn det som mangler i funksjonen så den virker som den skal. Ingen minuspoeng for feil svar, du BØR svare noe også der du er usikker.

Merk at if-setninga (linje 5) går over tre linjer, som ikke gir syntaksfeil fordi det står () rundt hele betingelsen.

```
def same_numbers_row_col(M, order):
```

```
    result = [ ]
```

```
    for i in range(len(  )):
```

```
        for j in range(len(  )):
```

```
            if (list(M[  ]) == list(M[  ,j])  ):
```

```
                not  
```

```
                sorted(list(M[i]))   (list(M[  ,j]))):
```

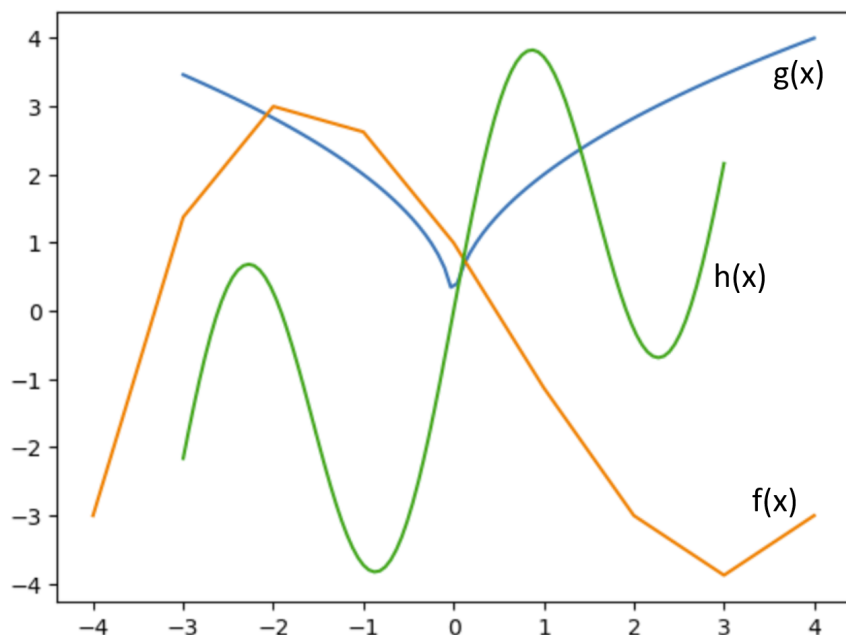
result. ([])

return result

Maks poeng: 6

12 Kopi av plot (5%)

Koden vår skal plote de tre grafene som er vist her. For at oppgaven ikke skal bli en matematikk-test, har vi satt på navn som viser hvilken graf som tilhører hvilken funksjon, selv om generering av disse navnene ikke er med i koden.



OPPGAVE: Trekk kodelinjer til riktig plass så programmet virker som det skal. 6 av kodelinjene skal ikke brukes. Ingen minuspoeng for feil, så du BØR svare noe også der du er usikker.

 [Hjelp](#)

<code>plt.show()</code>	<code>np.arange(-4,5,0.01)</code>	<code>np.arange(-4,5)</code>
<code>numpy</code>	<code>as fig, ax</code>	<code>as plt</code>
<code>np.linspace(-4,5,100)</code>	<code>ax.bar(h(x))</code>	<code>np.linspace(-3,3,100)</code>
<code>plt.subplots()</code>	<code>numpy as np</code>	<code>np.linspace(-3, 4, 100)</code>
<code>ax.plot(x, h(x))</code>	<code>np.arange(-3,4)</code>	

```
import 
import matplotlib.pyplot as 
def f(x):
    return 0.125*x**3 - 0.25*x**2 - 2*x + 1

def g(x):
    return 2*np.sqrt(np.abs(x))

def h(x):
    return 3*np.sin(2*x)+ x
```

```
fig, ax =  .
x = 
ax.plot(x, g(x))
x = 
ax.plot(x, f(x))
x = 


```

Maks poeng: 5

13 Kopi av card value (4%)

Vi ønsker å lage en funksjon `card_val(card)` som får inn en streng som representerer et spillkort og returnerer kortets tallverdi. Eksempel på strenger er '♥3', '♠4', '♣10', '♣Q', '♦A', dvs. første tegn i strengen er alltid et symbol for kortets farge (♥/♠/♣/♦), øvrige 1-2 tegn (2 tegn i tilfelle tallet er 10) representerer tallverdien. Bokstavene J, Q, K, A teller som 11, 12, 13, 14.

Eksempel på kjøring:

```
print(f"{card_val('♣10')} {card_val('♥3')} {card_val('♣Q')} {card_val('♦A')}")
gir utskriften
10 3 12 14
```

OPPGAVE: Fyll inn det som mangler slik at funksjonen virker som den skal. Ingen trekk for feil svar, du BØR svare noe også der du er usikker.

```
def card_val(card):
    special = {'J': 11, 'Q': 12, 'K': , 'A': 14}
    if card[]  special:
        return special[card[]]
    else:
        return  (card[])
```

Maks poeng: 4

14 Kopi av straight flush (6%)

I en del kortspill betyr **"straight"** at man har fem kort i løpende tallrekkefølge, f.eks. 2, 3, 4, 5, 6 eller 5, 6, 7, 8, 9 eller 10, J, K, Q, A hvor de fire sistnevnte teller som 11, 12, 13, 14. A kan også telle som 1, så A, 2, 3, 4, 5 vil også være en straight. Kortene trenger ikke å være i samme farge, så f.eks. vil de fem kortene i lista ['♥3', '♥5', '♠4', '♠7', '♦6'] utgjøre en straight. ['♥3', '♥5', '♠4', '♠7', '♦8'] vil derimot ikke være en straight, fordi man hopper over tallet 6.

En **"flush"** er derimot at alle kortene er av samme farge, f.eks. at man har fem ruter eller fem spar. ['♠K', '♠J', '♠4', '♠7', '♠2'] vil dermed være en flush (kun ♠), mens ['♠K', '♠J', '♠4', '♠7', '♦2'] ikke er det (fire ♠, men en ♦).

I denne oppgaven skal vi lage to funksjoner, straight() og flush(). Hver funksjon skal kunne få inn ei liste med fem kort som parameter, og returnere **True** (hvis man har hhv. straight eller flush) eller **False** (hvis man ikke har det). Funksjonen **card_val()** fra forrige deloppgave benyttes i funksjonen straight() for å finne kortets tallverdi.

OPPGAVE: Velg riktig i hver rute så de to funksjonene virker som de skal. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker.

```
def straight(cards):
```

```
    numbers =  (len(cards), [ ], int(cards))
```

```
    for c in  (range(len(cards)-1), cards, range(len(cards))):
        numbers.append(card_val(c))
```

```
    m =  (min, len, max)(numbers)
```

```
    s =  (set, tuple, sorted)(numbers)
```

```
    return ( s ==  (len(range(m,m+5)), list(range(m,m+5)), set(range(m,m+5)))
```

```
         (and, or, !=) s == {2,3,4,5,  (6, 1, 14)} )
```

```
def flush(cards):
    kinds = []
    for c in cards:
```

```
        kinds.append( (c[1], c[0], c))
```

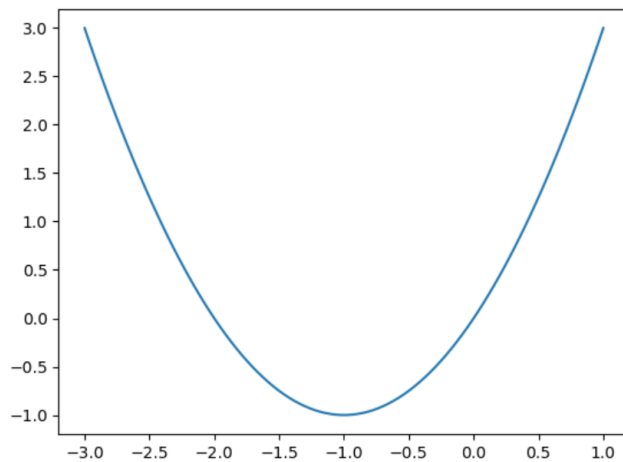
```
    return  (min, set, len)(  (list, len, set)(kinds)) == 1
```

Maks poeng: 9

15 Kopi av func as parameter, p1 (6%)

Vi ønsker å lage en funksjon **strict_incr(f, x_values)** for å teste om en annen funksjon **f** (som kommer inn som parameter) er strengt stigende over et array av x-verdier **x_values**. Hvis **f** er strengt stigende, skal det returneres **True**, ellers **False**.

Eksempel: Anta at funksjonen som gis inn som parameter er $f(x) = x^2 + 2x + 1$ som vist i plottet. Hvis den andre parameteren **x_values** er et array med tallene -2.0, -1.9, ..., 0.0, skal **strict_incr(f, x_values)** returnere **False**, siden funksjonen først synker fra -2.0 til -1.0 og deretter begynner å stige. Hvis derimot **x_values** har tallserien -0.9, -0.8, ..., 0.5, skal funksjonen returnere **True** fordi funksjonen hele tiden stiger over dette intervallet, dvs. $f(x)$ for hver nye x-verdi er større enn den forrige $f(x)$ -verdien. Merk at strengt stigende betyr at hver nye verdi må være større, det er ikke nok at den er lik.



OPPGAVE: Plasser kodelinjer slik at funksjonen **strict_incr()** virker som den skal. To av kodelinjene skal ikke brukes. Ingen minuspoeng for feil svar, så du **BØR** svare noe også der du er usikker.

 [Hjelp](#)

for i in range(len(x_values)):

break

for i in range(1, len(x_values)):

return result

result = True

if f(x_values[i]) > f(x_values[i-1]):

if f(x_values[i]) <= f(x_values[i-1]):

result = False

def strict_incr(f, x_values):

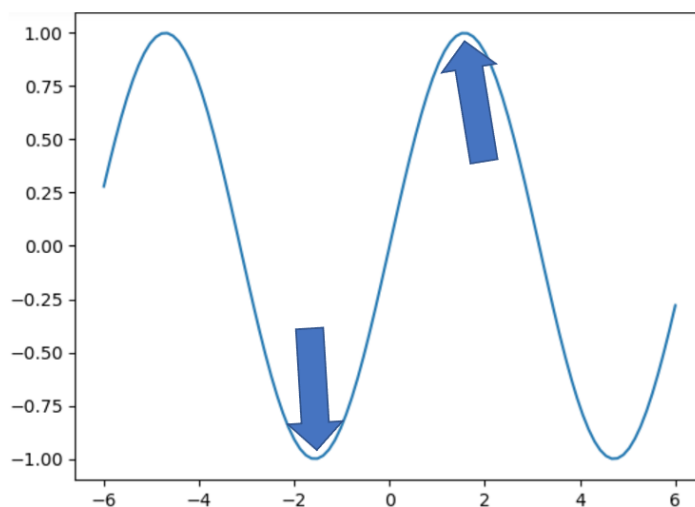
Maks poeng: 6

16 Kopi av func as parameter, p2 (6%)

Vi ønsker nå en funksjon **longest_incr(f, x_values)** som skal finne det lengste intervallet hvor en funksjon **f** er strengt stigende innenfor serien av x-verdier i arrayet **x_values**. 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 funksjonen er stigende her, fra bunnpunktet til toppunktet vist med blå piler. Altså,

longest_incr(np.sin(x), np.arange(-6.0,6.0,0.1)) skal returnere **(-1.57, 1.57)**.

Hvis det ikke fins noen strekning hvor **f** er stigende i det gitte intervallet, f.eks. hvis vi har samme funksjon men **x-values** er **np.arange(2.0, 4.0, 0.1)**, der funksjonen bare synker, skal **longest_incr()** returnere et tuppel med to like tall, f.eks. **(2.0, 2.0)**, siden programmet som skal bruke funksjonen vår, vil kunne skjønne at med to like tall i retur, fins det egentlig ikke noe intervall med stigning.



OPPGAVE: Trekk kodelinjer til riktig posisjon slik at funksjonen virker som den skal. To av kodelinjene skal ikke brukes. Ingen minuspoeng for feil svar, så du **BØR** svare noe også der du er usikker.

 [Hjelp](#)

if end - start > long_end - long_start:

for i in range(1, len(x_values)):

start = end = x_values[i]

if f(x_values[i]) > f(x_values[i-1]):

for i in range(len(x_values)-1):

else:

end = x_values[i]

end = x_values[-1]

return (long_start, long_end)

long_start, long_end = start, end

```
def longest_incr(f, x_values):
    start = long_start = end = long_end = x_values[0]
```



Maks poeng: 6

17 Kopi av export (5%)

Landet Y har følgende avgiftssystem for eksport av diamanter:

- inntil 10 gram diamanter: ingen avgift
- fra 10 inntil 100 gram: 2000 (i lokal valuta) per gram
- fra 100 inntil 500 gram: 3000 per gram
- fra 500 inntil 1000 gram: 4000 per gram
- fra 1000 inntil 5000 gram: 6000 per gram
- over 5000 gram: 8000 per gram

Merk at hver sats gjelder kun for grammene i det aktuelle intervallet, dvs. om man eksporterer 150 gram, er de første 10 grammene fortsatt avgiftsfrie, deretter 2000 per gram for de neste 90 grammene, og 3000 per gram for de siste 50 grammene. Lag funksjonen **export_fee(wt)** som får inn én parameter (gramvekt diamanter) og returnerer avgiften som skal betales.

OPPGAVE: Velg rett alternativ i hvert tomrom slik at funksjonen virker som den skal. Ingen minuspoeng for feil valg, så du BØR svare noe også der du er usikker.

```
import numpy as np
def export_fee(wt):
```

```
    wt_limits = np.array( Velg alternativ ((10, 100, 500, 1000, 5000), np.arange(5000, 5),
np.linspace(10, 5000, 5)) )
    fees = np.array((2000, 3000, 4000, 6000, 8000))
```

```
    wt_fees = Velg alternativ (np.zeros(len(fees)), wt_limits * fees, (wt_limits[1:] - wt_limits[:])
* fees)
```

```
    for i in Velg alternativ (range(len(wt_fees)), range(-1,-len(wt_fees)-1,-1), range(1,
len(wt_fees))):
```

```
        if Velg alternativ (wt > wt_limits[i], wt_fees[i] == 0, wt[i] in wt_limits):
```

```
            Velg alternativ (fees[i], wt_fees[i], wt_limits[i]) = Velg alternativ ((wt -
wt_limits[i]) * fees[i], wt * fees[i] - wt_limits[i], wt * fees[i])
```

```
            wt = Velg alternativ (wt - wt_limits[i], wt % wt_limits[i], wt_limits[i])
```

```
    return Velg alternativ (np.sum(fees), np.sum(wt_fees), wt_fees * fees)
```

Maks poeng: 7.5

18 Kopi av local_min (12%)

Funksjonen **count_local_min(A)** skal få inn et todimensjonalt numpy array med flyttall som parameter, og skal returnere et heltall som er antall lokale minima i dette arrayet. Et lokalt minimum er i denne sammenhengen definert som et element i arrayet som er mindre eller lik alle umiddelbart nærliggende element, både over, under, sidelengs og diagonalt. Som eksempel, hvis arrayet som gis inn til funksjonen er A som vist i figuren, så skal funksjonen returnere tallet **3** fordi det er tre lokale minima:

- 1.4 : \leq de umiddelbart nærliggende tallene 1.7, 2.6, 3.8, 3.4, 1.8
- 0.9 : \leq 3.8, 3.4, 3.8, 4.6, 5.4, 6.2, 6.6, 7.3
- 5.2 : \leq 6.2, 6.6, 7.3, 7.8, 8.6

```
import numpy as np
A = np.array([[1.7, 1.4, 1.8, 2.2],
              [2.6, 3.8, 3.4, 3.8],
              [4.2, 4.6, 0.9, 5.4],
              [5.8, 6.2, 6.6, 7.3],
              [9.9, 7.8, 5.2, 8.6]])
```

Skriv koden for funksjonen **count_local_min(A)**.

1

Maks poeng: 12