NTNU

KANDIDAT

Henry Græsberg

PRØVE

TDT4109 Insperaøving 2 H24

Emnekode	--
Vurderingsform	--
Starttid	25.10.2024 06:00
Sluttid	30.10.2024 15:00
Sensurfrist	--
PDF opprettet	30.10.2024 09:37

Kodeforståelse

Oppgave	Tittel	Status	Poeng	Oppgavetype
1	Kodeforståelse 1	Riktig	1/1	Flervalg
2	Kodeforståelse 2	Riktig	1/1	Flervalg
3	Kodeforståelse 3	Riktig	1/1	Flervalg
4	Kodeforståelse 4	Riktig	1/1	Flervalg
5	Kodeforståelse 5	Riktig	1/1	Flervalg

Lister og strenger

Oppgave	Tittel	Status	Poeng	Oppgavetype
6	Ryggsekk	Besvart	Rettes manuelt	Programmering
7	Handleliste	Riktig	5/5	Plasser i tekst

Lister og logikk

Oppgave	Tittel	Status	Poeng	Oppgavetype
8	sorterOgGrupper	Besvart	Rettes manuelt	Programmering
9	Divisor	Besvart	Rettes manuelt	Programmering
10	minSum	Besvart	Rettes manuelt	Programmering
11	Massekvadrering	Besvart	Rettes manuelt	Programmering

Dictionaries og mengder

Oppgave	Tittel	Status	Poeng	Oppgavetype
12	Frekvens	Besvart	Rettes manuelt	Programmering
13	Felles elementer	Besvart	Rettes manuelt	Programmering

Her er kladdeark, samt en "jukselapp" med innebygde funksjoner du kan bruke. Merk at det er flere funksjoner enn det som er pensum, bruk de gjerne, men ikke vær redd om de er nye. Lykke til!

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.

1 Kodeforståelse 1

Hva skrives ut av programmet?

```
def myst1(liste, element):
    liste.append(element)
    return liste

x = [2, 3, 4]
y = myst1(x, 1)
print(y)
```

Velg ett alternativ:

- [2, 3, 4, 1, 1]
- [2, 3, 4]
- [2, 3, 4, 1] ✓
- [1, 2, 3, 4]

2 Kodeforståelse 2

Hva skrives ut av programmet?

```
a = [7, 4, 6, 9]
a.pop(2)
a = a[1:]
a[1] = sum(a)
print(a)
```

- [4, 21, 2]
- [4, 21, 9, 2]
- [26, 9]
- [4, 13] ✓

3 Kodeforståelse 3

Hva skrives ut av programmet?

```
def myst2(liste):
    index = len(liste) // 2
    if len(liste) % 2 == 0:
        liste[index] = sum(liste)
    else:
        liste[index] = len(liste)
    return liste

x = [2, 3, 4, 5]
y = myst2(x)
print(y)
```

Velg ett alternativ:

- [2, 3, 4, 5]
- [2, 4, 4, 5]
- [2, 3, 14, 5] ✓
- [2, 3, 16, 5]

4 Kodeforståelse 4

Hva skrives ut av programmet?

```
d = {'a': 10, 'b': 20, 'c': 30}
d['d'] = 40
d['b'] = d['a'] + d['d']
del d['c']
print(d)
```

Velg ett alternativ:

- {'a': 10, 'b': 60, 'd': 40}
- {'a': 10, 'c': 30, 'd': 40}
- {'a': 10, 'b': 50, 'd': 40} ✓
- {'a': 50, 'b': 50, 'd': 40}

5 Kodeforståelse 5

Hva gjør funksjonen under?

```
def m(l):
    s = 0
    for e in l[2:]:
        s += e
    return s
```

Velg ett alternativ:

- (•) Summerer alle elementene i en liste, unntatt de to første. ✓
- ( ) Summerer elementene i en liste
- ( ) Summerer de to første elementene i en liste
- ( ) Returnerer hvert element i en liste

Her er kladdeark, samt en "jukselapp" med innebygde funksjoner du kan bruke. Merk at det er flere funksjoner enn det som er pensum, bruk de gjerne, men ikke vær redd om de er nye. Lykke til!

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.

6 Ryggsekk

Lag en funksjon **printRyggsekk(liste)** som tar inn en liste med strenger og printer ut en oppsummering ved å skrive ut alle ordene med komma mellom. Det skal være "og" mellom de to siste tingene.

Du kan anta at det er minst to strenger i listen

Eksempel på kjøring:
>>>printRyggsekk(["PC", "Blyant", "Matpakke"])
Ryggsekken din inneholder PC, Blyant og Matpakke

Skriv ditt svar her

```
def printRyggsekk(liste):
    return_string = "Ryggsekken din inneholder"

    for i in range(len(liste) - 1):
        return_string += f' {liste[i]}'

    return_string += f' og {liste[-1]}'

    return return_string
```

7 Handleliste

Her er et program for å skrive opp handlelisten sin. Brukeren skal legge til varer på listen, og programmet skal kunne oppdage duplikater, og forkaste dem. Programmet skal fortsette å spørre om ny vare frem til brukeren skriver "ferdig". Etter dette skal den endelige handlelisten skrives ut.

Hjelp

```
handleliste = []
while [True] :
    handleliste = input("Skriv en vare til handlelisten").lower()
    if vare == ["ferdig"] :
        [break]
    elif vare in handleliste:
        print("Denne varen er allerede på listen!")

    else:
        [handleliste.append(va]
print("Din endelige handleliste:")
for [vare] in handleliste:
    print(vare)
```

Her er kladdeark, samt en "jukselapp" med innebygde funksjoner du kan bruke. Merk at det er flere funksjoner enn det som er pensum, bruk de gjerne, men ikke vær redd om de er nye. Lykke til!

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.

8 sorterOgGrupper

Lag en funksjon **sorterOgGrupper(lst)** som tar inn en liste med heltall. Funksjonen skal sortere tallene i stigende rekkefølge, og gruppere dem i to lister, en for partall og en for oddetall.

Funksjonen skal returnere en 2D liste, hvor den sorterte listen med partall har indeks 0 og den sorterte listen med oddetall har indeks 1

Eksempel:

>>>sorterOgGrupper([3, 6, 1, 4, 2, 5])

[[2,4,6], [1,2,3]]

Skriv ditt svar her

```
def sorterOgGrupper(lst):
    return_list = [[], []]

    lst.sort()

    for num in lst:
        if num % 2 == 0:
            return_list[0].append(num)
        else:
            return_list[1].append(num)

    return return_list
```

9 Divisor

Lag en funksjon **divisor(lst, int)** som tar inn en liste med heltall og en divisor. Funksjonen skal returnere summen av tallene i listen som er delelig med divisor.

Eksempel:

>>>divisor([10, -15, 7, 20, 33], 5)

15

Skriv ditt svar her

```
def divisor(lst, int):
    sum = 0

    for num in lst:
        if num % int == 0:
            sum += num

    return sum
```

10 minSum

Lag funksjonen **minSum(lst)**, som tar inn en liste med minimum to tall, og returnerer summen av de to minste elementene i listen.
Dere må selv kontrollere at listen har to tall i seg, hvis listen ikke har det skal en beskjed om dette printes ut.

Eksempel på kjøring:
>>>liste = [1]
>>>minSum(liste)
"Listen er for kort!"

>>>liste2 = [3, 7, 9, 99, 24]
>>>minSum(liste2)
"10"
Skriv ditt svar her

```
1  def minSum(lst):
2      if len(lst) < 2:
3          return "listen er for kort"
4
5      lst.sort()
6
7      return sum(lst[:2])
```

11 Massekvadrering

Lag en funksjon **kvadrer(lst, thresh)** som tar inn en sortert liste lst med heltall (sortert fra minst til størst) og en grenseverdi thresh.

Funksjonen skal kvadrere tallene i listen ett etter ett, og summere de kvadrerte tallene. Når summen av de kvadrerte tallene overstiger thresh, skal prosessen stoppe.

Funksjonen skal returnere en liste med de kvadrerte tallene som gir en sum som er lavere enn eller lik thresh

Eksempel:

>>>kvadrer([1, 2, 3], 5)

[1, 4]

>>>kvadrer([2, 5, 6], 33)

[4, 25]

Skriv ditt svar her

```
def kvadrer(lst, thresh):
    squared_list = []

    i = 0
    while sum(squared_list) < thresh:
        squared_list.append(lst[i]**2)
        i += 1

    return squared_list
```

Her er kladdeark, samt en "jukselapp" med innebygde funksjoner du kan bruke. Merk at det er flere funksjoner enn det som er pensum, bruk de gjerne, men ikke vær redd om de er nye. Lykke til!

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.

12 Frekvens

Lag en funksjon **frekvens(str)** som tar inn en streng, og returnerer en dictionary hvor hvert unike ord er nøkkel, og antall ganger dette ordet forekommer i strengen er value. Funksjonen skal ignorere store/små bokstaver, og skilletegn som ",.!&-" skal ikke telles med i ordene.

Eksempel:

>>>frekvens("Hei hei, verden! Verden er stor.")

{'hei': 2, 'verden': 2, 'er': 1, 'stor': 1}

Skriv ditt svar her

```
def remove_specials(str):
    return_string = ""

    for char in str:
        if char.isalpha() or char == " ":
            return_string += char

    return return_string

def frekvens(str):
    dict = {}

    str_list = str.remove_specials().split()

    for string in str_list:
        if string not in dict.keys():
            dict[string] = 1
        else:
            dict[string] += 1

    return dict
```

13 Felles elementer

Lag funksjonen **felles_elementer(lst, lst)** som tar inn to lister, og returnerer et sett som inneholder de elementene som er felles for begge lister, uten duplikater.

Eksempel:
>>>felles_elementer([1, 2, 2, 3], [2, 3, 4])
{2, 3}
Skriv ditt svar her

```
def felles_elementer(lst1, lst2):
    return_set = set()

    for num in lst1:
        if num in lst2 and num not in return_set:
            return_set.add(num)

    return return_set
```