i

Informasjon

Exam in IN1000 and IN1001 autumn 2018

Time

30th November, 14:30 (4 hours)

The teachers will visit you sometime between 15:00 and 16:00.

The problem set

Problems **1a-f** require a short answers that will be checked automatically. Be careful when giving your answer that every character is correct and that your answer is of the correct type (i.e., no decimal point in an integer number). Text strings may be given without quotes (like this) or with double quotes ("like this"), but never with single quotes ('like this').

Problems **2a-d** are multiple choice questions in which you obtain the stated number of points for a correct answer; for a wrong answer or no answer, you will get 0 points.

Whenever you think a programming problem is unclear, you my make your own assumptions. State these precisely. You may also add your own functions or methods when required, but explain why. Furthermore, you may use solutions to previous questions even if you did not complete them.

Permitted aids

Any written or printed material. No electronic aids.

1

1a) 1 poeng

What is the value of tall after executing the following code?

```
tall = (3+1) * 2
tall = tall - 5
```

Maximum marks: 1

2

1b) 1 poeng

What is the value of variable tekst after executing this code?

```
tall = 7
tekst = "a"
if tall>10:
  tekst = tekst + "b"
elif tall<5:
  tekst = tekst + "c"
else:
  tekst = tekst + "d"
```

Maximum marks: 1

3

1c) 2 poeng

What is the value of variable a after executing the following code?

```
a = 0
for b in [2,4,1]:
   a = 2*a + b
```

Maximum marks: 2

4

1d) 2 poeng

What is printed on the screen when the following code is executed?

```
tallene = [ ]
a = 0
b = 1
while a<4:
   tallene.append(b)
   b = b*2
   a = a+1
print(tallene[0] + tallene[3])
```

Maximum marks: 2

5

1e) 2 poeng

We have a function kalkuler shown below:

```
def kalkuler(tall):
    tall = tall + 1
    return tall*2
```

What is printed on the screen when the following code is executed?

```
print( kalkuler(2) + kalkuler(4) )
```

Maximum marks: 2

6

1f) 3 poeng

What is printed on the screen when the following code is executed?

```
class Tall:
    def __init__(self, a):
        self._a = a
    def m1(self, c):
        self._a = self._a + c
    def m2(self):
        self._a = self._a * 2
    def m3(self):
        return self._a + 10

t1 = Tall(5)
t2 = Tall(2)
t1. m2()
t2.m1( t1.m3() )
print( t2.m3() )
```

Maximum marks: 3

7

2a) 2 poeng

#Assume that the class Person is defined thus:

```
class Person:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder
    def bursdag(self):
        self._alder += 1
    def hentAlder(self):
        return self._alder
    def settAlder(self, nyAlder):
        self._alder = nyAlder
```

What is printed when the following code is executed?

```
far = Person("Gjert", 48)
trener = far
trener.bursdag()
print( far.hentAlder() )
```

Select one alternative:

- 60
- 48
- 49

Maximum marks: 2

8

2b) 1 poeng

#Assume the class Person defined as in problem 2a

What is printed when the following cocde is run?

```
far = Person("Gjert", 48)
trener = far
trener.settAlder(60)
print( far.hentAlder() )
```

Select one alternative:

- 60
- 49
- 48

Maximum marks: 1

9

2c) 1 poeng

#Assume the class Person defined as in problem 2a

What is printed when the following code is executed?

```
far = Person("Gjert", 48)
trener = far
trener.bursdag()
trener = Person("Tone", 60)
print( far.hentAlder() )
```

Select one alternative:

- 48
- 49
- 60

Maximum marks: 1

10

2d) 1 poeng

#Assume the class Person defined as in problem 2a

What is printed when the following code is executed?

```
def feiring(p):
    p.bursdag()

far = Person("Gjert", 48)
feiring(far)
print( far.hentAlder() )
```

Select one alternative:

- 48
- 49
- 60

Maximum marks: 1

11

3a) 5 poeng

Write a function **vinnerlag(hjemmelag, bortelag, hjemmemaal, bortemaal)** which receives as argument the name and number of goals scored for the home and away teams, and returns the name of the winning team (i.e., the one that scores most goals). If the two teams have scored an equal number of goals, the function shall return the text string "uavgjort". You may assume that the arguments giving the names of the teams are text strings (type str) and that the arguments giving the number of goals scored are integers (type int). As an example, the call *vinnerlag("Brann", "Molde", 2,3) shall return* "Molde", while the call *vinnerlag("Brann", "Molde", 2,2)* shall return "uavgjort".

Fill in your answer here

```
1
```

Maximum marks: 5

12

3b) 4 poeng

Write a function **forkort_lagliste(lagliste)** som receives as argument a list of text strings containing team names and returns a new list in which no team name occurs more than once. In other words, if the same text string occurs several times in the argument list, it shall only occur once in the list returned. For example, the call f*orkort_lagliste(["Brann", "Molde", "Brann"])* shall return a list *["Brann", "Molde"]*.

Fill in your answer here

```
1
```

Maximum marks: 4

13

3c) 3 poeng

Write a function **legg_inn_null_maal(lagliste)** which receives as argument a list of text string (team names) and returns a dictionary in which each text string in the list (i.e., each team name) is a key and the corresponding data is the integer value 0. For example, the call *legg_inn_null_maal(["Brann", "Molde"])* shall return a dictionary *{"Brann" : 0, "Molde" : 0}*.

Fill in your answer here

```
1
```

Maximum marks: 3

14

3d) 6 poeng

Write a function **ekstraher_lagliste(fn)** which receives a file name as argument and returns a list of all team names found in the file. The function shall read the file (whose name is given as argument) in which each line consists of the name of a home team, the name of an way team, the number of goals scored by the home team and the number of goals scored by the away team; all items are separated by spaces. The function shall return a list containing all team names found in the file (either as a home team or as an away eam in the stated format) . Whether a team names occurs more than once in the returned list is insignificant. Also, the list order does not matter.

A file to be read may, for instance, look like this:

brann molde 2 0

sarpsborg molde 1 1

In this example, the function shall return a list containing the team names brann, molde and sarpsborg. As mentioned, repetitions may occur, so both ["brann", "molde", "sarpsborg"] and ["brann", "molde", "sarpsborg", "molde"] are acceptable as return values.

Fill in your answer here

```
1
```

Maximum marks: 6

15

3e) 7 poeng

Write a function **regn_poengsum(fn)** which receives a file name as argument and returns a dictionary with the number of points obtained by each team based on the match results found in the file. The file has the same format as in problem 3d, i.e., each line consists of a home team name, an away team name, number of home team goals and number of away team goals, all separated by spaces. Preferably, the function regn_poengsum shall initially call the functions ekstraher_lagliste, forkort_lagliste and legg_inn_null_maal (from problems 3d, 3b and 3c, respectively) to build a dictionary with zero points for all teams in the file. Then, it shall consider all match results (lines) in the file and give 3 points to the winning team, 0 points to the losing team, or 1 point to both teams in case of a draw. You may call the function vinnerlag from problem 3a to compute the points, but this will not influence your score.

(You may call functions/methods from other problems based on their described action, irrespective of whether you solved that problem or not.)

Fill in your answer here

```
1
```

Maximum marks: 7

16

3f) 5 poeng

Write a function **gull(lagoversikt)** which receives as argument a dictionary with team names and number of goals scored (having strings as keys and integer as values) and returns the name of the team with highest number of points (i.e., the dictionary key with highest corresponding data value). You may assume that only one team has the highest number of points (no two teams have the same number of points) and that at least one team has a non-zero number of points. For example, the call *gull({"Brann" : 2, "Molde" : 3})* shall return *"Molde"*.

Fill in your answer here

```
1
```

Maximum marks: 5

17

3g) 3 poeng

Write a procedure **finn_gull(fn)** which receives a file name as argument and prints the name of the team with most points to the screen. The file has the same format as in problem 3d. Points are computed based on the winning and losing team in each match, as described in problem 3e. As in problem 3f, you may assume that exactly one team has the highest number of points (never two teams with the same number of points) and that there is at least one team with a non-zero number of points. You may call any method from problems 3a-3f to create the simplest possible solution for procedure finn_gull. (You may call funktions/procedures from other problems based on their description, irrespective of whether you solved this problem or not.)

Fill in your answer here

```
1
```

Maximum marks: 3

18

4a) 10 poeng

This problem consists of several subproblems in which you implement components of a bigger system. If you skip any of the subproblems, you should still read the whole problem text. (A PDF is included with each subproblem.)

Write the answer to subproblem a) from the appendix here:

```
1
```

Maximum marks: 10

19

4b) 5 poeng

Write your answer to subproblem b) from the appendix here:

```
1
```

Maximum marks: 5

20

4c) 10 poeng

Write your answer to subproblem c) from the appendix here:

Note! The menu returned from the method hentRedusertMeny shall be represented as a dictionary with categories, not as an object of the class Meny.

```
1
```

Maximum marks: 10

21

4d) 8 poeng

Write your answer to subproblem d) from the appendix here:

```
1
```

Maximum marks: 8

22

4e) 7 poeng

Write your answer to subproblem e) from the appendix here:

```
1
```

Maximum marks: 7

23

4f) 5 poeng

Write your answer to subproblem f) from the appendix here:

```
1
```

Maximum marks: 5

24

Oppgave 5) 6 poeng

Write a function **godkjenn(aldre)** which receives as argument a list of lists with ages of family members. Each family is represented as a list of the ages of its members in no particular order. A family with three members of age 30, 10 and 2 years may, for instance, be represented by a list *[10,2,30]*. All ages are integers. The function godkjenn shall receive a list of such lists (i.e., of several families). If, for example, we have a family with ages 20 and 1 years in addition to the family mentioned previously, these two families could be represented as *[[10,2,30], [20,1]]*.

Write the function godkjenn so that it checks whether every family has at least one member of legal age. In other words, it checks that every family list has at least one value 18 or above. If all families have at least one member of legal age, the function shall return True. If at least one family is without a member of legal age (i.e., for at least one family, all values are below 18), it shall return False. For example, the call *godkjenn([[10,2,30], [20,1]])* shall return *True*, while the call *godkjenn([[10,2,30], [10,1]])* shall return *False*.

Fill in your answer here

```
1
```

Maximum marks: 6

Problem 4 Take-away service (45 points)

You shall write (parts of) a program for a take-away service which receives orders from customers on the Internet using a simple terminal-based system. The program shall keep track of a number of regular customers with phone numbers and what kind of food/ingredients (substances) the customer does *not* want (like gluten, dairy products or pork).

Furthermore, the program shall know which dishes that be delivered in several categories – like, for instance, hors d'œuvres, main courses or desserts. For every dish, it must store which substances that may be problematic for the customer. In this problem, you may assume that the terms used for substances which the customer wants to avoid, and the terms used for ingredients in a dish, come from the same vocabulary, so that comparisons will be easy.

The program uses these data to present an adapted menu for each customer, one in which every dish is according to the customer's demands regarding ingredients.

In addition to **TakeAway**, you need to know the classes **Rett, Kategori, Meny** og **Kunde**. You shall write all methods described for each class unless explicitly told not to. For your own use, a structure diagram may be useful.

a) 10 points. Write the class Rett (dish)

The class has a name (text string), a price (floating-point number) and a list (which may be empty) of ingredients (text strings). All instance variables get their initial value from arguments to the constructor **__init__**. In addition to the constructor, the class has the following methods in its interface:

- **sjekkInnholdOK** (checkContentOK) has as argument a list of substances. The method checks whether any of the substances in the argument are found in the content list of the dish. If the method finds a hit, it returns False. If none of the ingredients in the argument are found on the dish, the method shall return True.
- **__str__** returns a text string with the name of the dish as well as the price and all ingredients in a readable form.

b) 5 points. Write the class Kategori (category)

The class has two instance variables: a category name and a list of references to **Rett**-objects. Both get their value from arguments to the constructor. The interface also has this method:

- **hentOkRetter** (fetchOkDishes) with one argument: a list of ingredients a customer wants to avoid. The method traverses the category's list of dishes, and makes a new list of references to dishes that do not contain any unwanted ingredients. This new list is returned. (You do not have to create copies of the Rett-objects in the new list; you may refer to the same objects as the instance variable.)

c) 10 points. Write the class Meny (menu)

This class has one instance variable which represents the whole menu in a dictionary with all category names as keys and references to Kategori-objects as values. The class constructor has one parameter: a list of all category names in the menu. All data for a category is to be read from a file whose name is the category name followed by ".txt". The constructor shall construct the menu by calling the private (non-public) method _lesKategoriFil which has a file name as parameter and returns a complete category object. You shall *not* write the method _lesKategoriFil.

Apart from the constructor, the class has one method:

- **hentRedusertMeny** (fetchReducedMenu) receives one parameter: a list of ingredients to be avoided. The method traverses the whole menu, category by category, and constructs a new reduced menu in which no dishes contain unwanted substances. Categories without any suitable dishes are not included in the reduced menu.

d) 8 poeng. Skriv klassen Kunde (customer)

The class has two instance variables: the customer's phone number (a text string) and a list of ingredients (text strings) which the customer wishes to avoid (because of allergies or something else). Both instance variables get their value from the constructor arguments. In addition, the class has a method

- **velgRetter**, which takes a Meny-object as parameter and calls hentRedusertMeny on this object to obtain a bespoke menu for the customer. Then, the reduced menu is presented to the customer on the screen, one category at a time. The customer selects a dish from the category by entering the name of the dish, or an empty line to skip this category. Every non-empty line entered by the customer is stored as a text string, and the method returns these text strings in a list. The customer input is not checked against dishes on the menu in case the customer wants to add messages to the chef about sizes or preparation.

e) 7 points. Write the class TakeAway

The class **TakeAway** has two instance variables: a reference to an object of the class Meny and a dictionary of customers in which the phone number is the key and the values are references to Kunde-objects. The constructor has two parameters; a list of category names and the name of a customer file; it shall construct a menu and a customer catalogue. The method _lesKundefil returns a customer catalogue with all customers; it is called by the constructor of TakeAway, but you shall *not* write it. Apart from the constructor, the class interface is the following:

- **betjenKunde** (serveCustomer) with one parameter: the phone number of a customer who has contacted the system. The method calls the method velgRetter for the right customer. (You may assume that all customers are already registered.) Then, it calls the private (non-public) method _lagOgLeverMat with the order from velgRetter. **_lagOgLeverMat** has one parameter: the order, and in this version of the program, it shall only print the customer's order (names of all the dishes) on the computer screen. You shall write the method _lagOgLeverMat.

f) 5 points. Write a main program

The main program shall do the following:

- Start a take-away service with the categories "Hors d'œuvres", "Main courses" and "Desserts" on the menu, and a customer catalogue on the file "Kunder.txt". You may assume that all required data files exist.
- Ask for a phone number (on the terminal) and service the corresponding customer until a customer gives an empty text string as a phone number; this will terminate the program.