i Forside (viktig informasjon!)

UNIVERSITY OF OSLO
Faculty of mathematics and natural sciences
Exam in: IN1900 og INF1100
Exam date: December 17th, 2020
Time for exam: 15.00-19.00 (4 hours)
Attachments: ODESolver.pdf in question 22.
Permitted aids: All, except communication with others.

Important note:
This was the final exam in 2020, and was given as a home exam. During the exam some questions were drawn randomly from a pool of questions. This version contains all the questions, and therefore contains four questions more than what each candidate got on the exam. The maximum total score is also higher. On the exam it was 75 points.

This page contains important information. It is recommended to read the entire page before you start working on the questions.

Rules for digital home exam
The exam shall be an independent work. You can use all kinds of aids (books, web pages, notes, etc.), but it is *not* allowed to collaborate or communicate with others during the exam or to share your work with others. *All forms of communication and collaboration will be considered as cheating.*

Obligatory oral questioning
Approximately 2/3 of the candidates will be subject to an oral questioning after the exam. The purpose of the questioning is to verify ownership to your submitted answers. The questioning will not affect the grading of the exam, but it can potentially lead to the University initiating a cheating case. *This questioning is obligatory for those who are requested to take part in it and it will take place in the first two weeks of January.*

Questions during the exam
The lecturers responsible for the course are available on email during the exam. Send email to sundnes@ifi.uio.no if you believe that the question text is unclear or if you suspect there are errors in the questions. Answers to questions will be made available for everyone on this web page:
https://www.uio.no/studier/emner/matnat/ifi/IN1900/h20/eksamen/eksamensrunde.html
We will only answer questions regarding errors and unclear formulations in the question text.

Contact point
User support for home exams

Information about the questions
The exam contains multiple choice questions, and text questions where you shall write short programs or read programs and write the output from the program. If you are missing information you can make your own reasonable assumptions, as long as they are in line with the "nature" of the question. In text questions you should then specify the assumptions you made, for instance in comments to the code.

All code in the question texts is written in Python 3. You can write your answers in either Python 2 or Python 3, but you should avoid using a mix.

Most of the questions lead to short code with little need for comments, unless you do something complicated or non-standard. (which is not recommended; but in this case the comments shall explain the ideas behind the program to make it easier to evaluate the code).

A question may ask you to write a function. A main program which calls the function is in this case not needed, unless it is specifically asked for in the question text.

1.1 Hvilken påstand er riktig?

One of the following statements is correct. Which one?
Select one alternative:

- A function must always have at least one input argument
- A function can have multiple return statements ✔
- A function must always have at least one return statement
- A function can not have multiple return statements

Maximum marks: 2

1.2 Hvilken påstand er riktig?

One of the following statements is correct. Which one?
Select one alternative:

- Vectorization of computations happens automatically when we use lists in Python.
- A for loop can always be written as a while loop. ✔
- A while loop will normally give shorter and simpler code than a for loop.
- If x is a list, then the assignment y = x will result in a copy of the list object.
- An if test must always be accompanied by else.

Maximum marks: 2

1.3 Lister

If a is an integer ('int'), which of the following lines will *not* result in a list of length a:
Select one alternative:

- b = list(range(a-1)) + [a]
- b = [0]*a
- b = [3*e for e in range(1,a+1)]
- b = [3*e for e in range(a+1)] ✔
- b = list(range(a))

Maximum marks: 2

1.4 Hvilket funksjonsargument?

The function **find(text, arg)** is defined as follows:

```
def find(text, arg):
    for j in range(len(text)):
        if text[j:j+len(arg)] == arg:
            return j
    return False
```

Aassume that we perform the function call **find(txt, a)** where the argument **txt** is a string. You are informed that the call did *not* result in an error message, and that it did *not* return the value False. What can we then say about the argument **a** in the function call?

Select one alternative:

- **a** is a string ('str') with length 1.
- **a** is a string ('str') having the same length as **txt**.
- **a** is a list having the same length as **txt**.
- **a** is an integer ('int').
- **a** is a string ('str') which is not longer than **txt**. ✔

Maximum marks: 2

1.5 Hvilket funksjonskall?

The following function uses Newton's method to find a solution of the equation f(x) = 0.

```
def Newton(f, dfdx, x0, tol= 1e-5):
    f0 = f(x0)
    while abs(f0) > tol:
        x1 = x0 - f0/dfdx(x0)
        x0 = x1
        f0 = f(x0)
    return x0
```

The arguments to the function are a python-funksjon f that implements the mathematical function f(x), a python function dfdx that implements the derivative (f'(x)), a start value x0, and a tolerance tol. We want to use the function to solve the equation

$x^3 + 2x - 1 = 0$

with x0 = 0 as start value. Which function call is correct?

Select one alternative:

- x = Newton(lambda x: x**3+2*x-1,lambda x: 3*x**2+2,0) ✔
- x = Newton(lambda x: f(x), lambda x: dfdf(x), 0)
- x = Newton(x**3+2*x-1,3*x**2+2,0)
- x = Newton(f = lambda x: x**3+2*x-1,dfdx = 3.0*x**2+2+,x0=0)
- x = Newton(f = x**3+2*x-1, dfdx= 3*x**2+2,0)
- x = Newton(f(x)=x**3+2*x-1, dfdx(x)=3*x**2+2, x0=0, tol=1e-5)

Remember that a lambda-funksjon is a compact way to define a function. For instance, the following line will define a function that returns x^2+y^2:

```
func = lambda x,y: x**2 + y**2
```

This line is equivalent to the following code:

```
def func(x,y):
    return x**2 + y**2
```

Maximum marks: 2

1.6 Hva gjør funksjonen?

The function **f(arg)** is defined as follows. The argument **arg** is assumed to be a list.

```
def f(arg):
    n = len(arg)
    result = []
    for i in range(n,0,-1):
        result.append(arg[i-1])
    return result
```

Which of the following statements gives the most precise description of what the function **f(arg)** does?
Select one alternative:

- The function returns the smallest element in arg.
- The function returns a list containing the elements of arg, but in opposite order. ✔
- The function returns a sorted version of the list arg, from smallest to largest value.
- The function returns the largest element in arg.

Maximum marks: 2

1.7 Hva gjør funksjonen?

The function **f(arg)** is defined as follows. The argument **arg** is assumed to be a list.

```
def f(arg):
    result = arg.copy()
    for i in range(len(result)):
        min_i = i
        for j in range(i + 1, len(result)):
            if result[j] < result[min_i]:
                min_i = j
        result[i], result[min_i] = result[min_i], result[i]
    return result
```

Which of the following statements gives the most precise description of what f(x) returns when x is a list of numbers?

Select one alternative:

- The largest element in x.
- A sorted version of x, from smallest to largest value. ✔
- A sorted version of x, from smallest to largest value and with all duplicates removed.
- The smallest element in x.
- A list with the elements from x, but in the opposite order.

Maximum marks: 2

2.1 Hva gjør koden?

The file altitude.py contains the following code:

```
from math import exp

h = input('Input the altitude (in meters):')
h = float(h)

p0 = 100.0    # Sea level pressure (kPa)
h0 = 8400     # Scale height (m)
p = p0 * exp(-h/h0)
print(p)
```

Explain what happens when this program is run in the terminal window with the command
Terminal> python altitude.py

Also explain what can go wrong when this code runs.

Fill in your answer here

Format B I U x_2 x^2 I_x Ω Σ

Words: 0

Maximum marks: 4

2.2 Hvilken linje mangler?

The file counties.txt contains the following:

```
F-30;Viken;    1241165; 24593
F-03;Oslo;    693494; 454
F-34;Innlandet;    371385; 52072
F-38;Vestfold og Telemark; 419396; 17466
F-42;Agder; 307231; 16434
F-11;Rogaland; 479892; 9377
F-46;Vestland; 636531; 33871
F-15;Møre og Romsdal; 265238; 14356
F-50;Trøndelag; 468702; 42202
F-18;Nordland; 241235; 38155
F-54;Troms og Finnmark; 243311; 74830
```

Each line contains the following information about a county in Norway: county number, name, population, and area. There are no blank lines in the file.

We want the code below to read this file and store the information about each county in a nested dictionary named **counties.** The keys in the dictionary shall be the county number, and each value shall be a dictionary with keys 'name', 'pop', and 'area', and associated values read from the file. We want the population and the area to be integers (int). One line is mising in the code.

```
counties = {}

with open('counties.txt') as infile:
  for line in infile:
    w = line.split(';')
    ## Missing line goes here ##
    counties[w[0]] = data

print(counties['F-03'])
```

We want this code to work and print the following:

{'name': 'Oslo', 'pop': 693494, 'area': 454}

Write the missing line in the code, to make the program work as desired!

It is possible to split the missing line into two or more shorter lines if you want (this gives the same score), but you. must keep all the code written above and you can only insert code where it says that a line is missing.

Fill in your answer here

Format

Words: 0

Maximum marks: 3

2.3 Hva gjør funksjonen?

Give a brief explanation of what the function **find(text,args)** does. You can assume that the argument **text** is a string and **args** is a list of strings.

```
def find(text, args):
    result = {}
    for a in args:
        result[a] = 0
        for j in range(len(text)):
            if text[j:j+len(a)] == a:
                result[a] += 1
    return result
```

Fill in your answer here

Format

Words: 0

Maximum marks: 3

3.1 Numerisk derivasjon

The derivative of a mathematical function f(x) can be approximated with the forward difference

$$f'(x) \approx \frac{f(x+h)-f(x)}{h}$$

for a small number h.

a) Write a Python function **forward(f,x,h),** which uses this formula to estimate the derivative of a function f in the point x. The function shall return the value of the estimated derivative. The argument f can be an arbitrary mathematical function implemented as a Python function, which takes one input argument and returns one value.

b) Add a line to the code you wrote in part a), where you call the function to estimate the derivative of cos(x) in the point x=0, for h=0.001, and write the value to the screen as a formatted number with four decimals.

Fill in your answer here

Maximum marks: 5

3.2 Funksjon av to variable

Write a Python function **f(x,y),** where **x** is a number and **y** is a list of numbers. The function shall calculate the value of the mathematical expression

$$f(x,y) = \begin{cases} 3x^2y - 3xy, & y \leq 0 \\ 3x^2y + 3xy, & y > 0 \end{cases}$$

for the given value of **x** and each of the values in the list **y**, and return the answer as a list of the same length as **y**.

Fill in your answer here

```
1
```

Maximum marks: 5

3.3 Implementasjon av en sum

Write a Python function **log_approx(x,n)** which calculates the sum

$$f(x) = \sum_{k=1}^{n} (-1)^{k+1} \frac{x^k}{k}$$

and returns the result. Here x can be either a floating point number (float) or a numpy array of floats, while n is a positive integer.

Fill in your answer here

```
1
```

Maximum marks: 5

3.4 Stykkvis lineær funksjon

Write a Python function **piecewise(x, a, b)**, which implements the mathematical expression

$$f(x) = \begin{cases} 0.0 & \text{for } x \leq a \\ \frac{x-a}{b-a} & \text{for } a < x \leq b \\ 1.0 & \text{for } x > b \end{cases}$$

The arguments x, a and b er scalars (numbers), not lists or arrays.

Include code to call the function and print the result, for a = 0, b = 1, and x = 0.5.

Fill in your answer here

```
 
```

Maximum marks: 5

3.5 Testfunksjon

The function below is a Python implementation of the following mathematical function:

$$f(x) = \begin{cases} 0 & \text{for } x \leq a \\ sin(x-a) & \text{for } a < x \leq a+\pi \\ 0 & \text{for } x > a+\pi \end{cases}$$

```
def f(x,a):
    if 0 <= x-a <= pi:
        return sin(x-a)
    else:
        return 0
```

Write a test function for this function. In the test function you shall choose a value for the parameter a, and then test the function for three values of x in the three intervals of the function definition.

Fill in your answer here

1

Maximum marks: 5

4.1 Finn den lengste listen

Write a function **longest(a)** which takes a nested list **a** as argument, and returns the longest element. In other words, **a** is a list of lists, and **longest(a)** shall return the longest of the lists in **a**. You can assume that all the lists in **a** have different length.

Fill in your answer here

Maximum marks: 5

4.2 Primtall

A prime number is an integer larger than 1 which is only divisible by 1 and itself.

Assume that you have already written a function **is_prime(k)**, which returns **True** if the argument **k** is a prime number and **False** otherwise. Write a function **primes(n)** which takes a number **n** as input og returns a list of all prime numbers smaller than or equal to **n**. You can assume that the two functions are written in the same file, so you can use **is_prime(k)** without importing it.

Fill in your answer here

Maximum marks: 5

4.3 Palindrom

A palindrome is a word that reads the same backwards as forwards, for instance "eye" and "racecar". Write a function **is_palindrome(word),** which takes a string **word** as input and returns **True** if the string is a palindrome and otherwise **False**.

Fill in your answer here

Maximum marks: 5

4.4 Lesing av fil

The file counties.txt contains population numbers for the counties in Norway, in the following format:

County   Population

Viken 1241165
Oslo 693494
Innlandet 371385
Vestfold og Telemark 419396
Agder 307231
Rogaland 479892
Vestland 636531
Møre og Romsdal 265238
Trøndelag 468702
Nordland 241235
Troms og Finnmark 243311

There are no blank lines in the file.

Write a Python program which reads this file and stores its contents in a dictionary. Each key. in the dictionary shall be a string with the name of county (for instance "Vestfold og Telemark"), and the value shall be the population number.

Hint: The Python method **join** can be used to join a list of strings into a single string. For instance, the code
string_list = ['Hello','world']
hello = ' '.join(string_list)
will result in a string **hello** with the value "Hello world".
Fill in your answer here

```
1
```

Maximum marks: 5

4.5 Klasse for en funksjon

This question consists of two parts.

a) Write a Python class **F** which implements the function
$f(x; a, b, c) = ax^3 + bx^2 + cx + d$
The parameters a, b, c and d shall be attributes, and the class shall be possible to use in the following way

f = F(a=0.0, b=1.0, c=2.0, d=0.0)
x = 2.0
print(f(x))   # prints the function value (8.0)

b) Extend the class **F** with a special method **__str__(self)** which returns the mathematical expression with the parameter values inserted. The following code shall then work:

f = F(1.0, 2.0, 3.0,0.0)
print(f)   # Should output the following: 1.0*x^3 + 2.0*x^2 + 3.0*x + 0.0
Fill in your answer here

Maximum marks: 5

5.1 ODE-løser, funksjon

Write a Python function **ralston(f, U0, T, n)**, which uses Ralston's method to solve an ordinary differential equation (ODE) given by:

$$u' = f(u,t), \qquad u(0) = u_0.$$

Ralston's method is given by:

$$u_{k+1} = u_k + \Delta t(\tfrac{1}{4}k_1 + \tfrac{3}{4}k_2)$$
$$k_1 = f(u_k, t_k)$$
$$k_2 = f(u_k + \tfrac{2}{3}\Delta t k_1, t_k + \tfrac{2}{3}\Delta t)$$

The arguments to the function shall be a callable function f, which defines the right hand side of the ODE, the initial condition U0, the final time T, and the number of time steps n. The function shall return two numpy arrays u and t, where u contains the solution and t contains the time points where the solution is approximated. In this question you can assume that we are solving a scalar ODE, where the solution has a single component. The solution array u can therefore be a one-dimensional array. Include necessary import.

Fill in your answer here

```
 
```

Maximum marks: 5

5.2 ODESolver, arv

Implement the method from the previous question (Ralston's method) as a sub class **Ralston(ODESolver)**. The base class ODESolver is defined in the attached file. Use inheritance to reuse as much code as possible from the base class. You can assume that you write the class in the same file as the base class, so you don't need to import it. The class **Ralston** must support the following use:

```
from ODESolver import *
import numpy as np
rhs = lambda u,t: -u
solver = Ralston(rhs)
solver.set_initial_condition(1.0)
time = np.linspace(0,10,101)
u, t = solver.solve(time)
```

Fill in your answer here

Maximum marks: 5

5.3 SIRD-modell

This question presents a so-called SIRD-model for modeling infectious diseases. The model is a modification of the classical SIR-model, where the population is divided in four groups: those who can be infected (S), those who are infected and can infect others (I), those who have recovered and are immune (R), and those who have died from the disease (D). Let S(t), I(t), R(t), and D(t) be the number of people in each category. The following system of differential equations describes how S(t), I(t), R(t) and D(t) evolve over a time interval [0,T]:

$$S'(t) = -\alpha(t)\frac{S(t)I(t)}{N},$$
$$I'(t) = \alpha(t)\frac{S(t)I(t)}{N} - \beta I(t) - \gamma I(t)$$
$$R'(t) = \beta I(t)$$
$$D'(t) = \gamma I(t)$$

At time t=0 we have the initial conditions S(0) = S0, I(0) = I0, R(0) = D(0) = 0. The function $\alpha(t)$ and the constants β og γ are assumed to be known. *N* = S0+I0 is the total population (including those who die from the disease, therefore constant). All constants and functions are >0.

Write a Python-function **SIRD(S0,E0,alpha,beta, gamma,T)**, which takes initial values S0, I0, the function $\alpha(t)$ (alpha), parameters β, γ (beta, gamma), and the end-time T as input arguments. The function shall solve the equations of the SIRD-model and return the solution. Use the Ralston class from the previous question to solve the differential equations. Let the time be given in days. Use ten time steps per day, so that the total number of time steps N for a simulation over the interval [0,T] is 10T+1. The function **SIRD** shall return 5 arrays:

t, which contains the time points tk where the numerical solution is calculated,
S, which contains S(0),S(t1), ... S(T),
I, which contains I(0),I(t1), ... I(T),
R, which contains R(0),R(t1), ... R(T),
D, which contains D(0),D(t1), ... D(T).

We want to solve the model with the following parameters; S0 = 370000, I0 = 30, beta = 0.025, gamma = 0.25, and alpha(t) = 1.0 (constant). Write the code to call the **SIRD** function with the given parameters and T=100. Also include code to plot S(t), I(t), R(t), and D(t) in the same window, with a legend for each curve.

Fill in your answer here

```
1
```

Maximum marks: 5

Question 22
Attached

```python
import numpy as np

class ODESolver:
    def __init__(self, f):
        # Wrap user's f in a new function that always
        # converts list/tuple to array (or let array be array)
        self.f = lambda u, t: np.asarray(f(u, t), float)

    def set_initial_condition(self, U0):
        if isinstance(U0, (float,int)):  # scalar ODE
            self.neq = 1                 # no of equations
            U0 = float(U0)
        else:                            # system of ODEs
            U0 = np.asarray(U0)
            self.neq = U0.size           # no of equations
        self.U0 = U0

    def solve(self, time_points):
        self.t = np.asarray(time_points)
        N = len(self.t)
        if self.neq == 1:  # scalar ODEs
            self.u = np.zeros(N)
        else:              # systems of ODEs
            self.u = np.zeros((N,self.neq))

        # Assume that self.t[0] corresponds to self.U0
        self.u[0] = self.U0

        # Time loop
        for n in range(N-1):
            self.n = n
            self.u[n+1] = self.advance()
        return self.u, self.t

class ForwardEuler(ODESolver):
    def advance(self):
        u, f, n, t = self.u, self.f, self.n, self.t

        dt = t[n+1] - t[n]
        unew = u[n] + dt*f(u[n], t[n])
        return unew

class ExplicitMidpoint(ODESolver):
    def advance(self):
        u, f, n, t = self.u, self.f, self.n, self.t
        dt = t[n+1] - t[n]
        dt2 = dt/2.0
        k1 = f(u[n], t[n])
        k2 = f(u[n] + dt2*k1, t[n] + dt2)
        unew = u[n] + dt*k2
        return unew
```

IN1900 - solutions final exam fall 2020

Question 2.1

This question has many correct answers. Here's one fairly detailed alternative: When the program is run the execution will stop at the line `h = input(...)` and display the message `Input the altitude (in meters):` on the screen. The user inputs a number and presses Enter to make the program continue. The input number is converted from text to a floating point variable, which is then used in the formula to compute `p`. Finally, the value of `p` is written in the terminal.

The program may fail if the user does not provide a number when requested, for instance by just pressing Enter or by typing text that cannot be converted to a number. This will result in the program stopping with a `ValueError` on the next line. Another potential error is an overflow error in the formula for `p`, if a very large number is input by the user.

Question 2.2

Code with the missing line included:

```
counties = {}
with open('counties.txt') as infile:
    for line in infile:
        w = line.split(';')
        data = {'name':w[1],'pop':int(w[2]),'area':int(w[3])}
        counties[w[0]] = data

print(counties['F-03'])
```

Other alternatives are of course possible, but the added code needs to define a dictionary named `data` with the indicated structure and content.

Question 2.3

This question has many correct answers. Here's one alternative:

The function takes a string (`text`) and a list of strings (`args`) as arguments. The function first creates an empty dictionary, and then traverses the list og strings `args`. For each string `a` in `args`, an entry is created in the dictionary, with key `a` and value 0. The string `text` is then traversed character by character, and checked for occurence of the string `a`. Each time `a` is found in `txt` the corresponding value in the dictionary is increased by one. The result is a dictionary counting the number of times each string in `args` occurs in `text`.

Question 3.1

Suggested solution:

```
from math import cos
```

```
def forward(f,x,h):
    return (f(x+h)-f(x))/h

print(forward(cos,0,0.001))
```

Question 3.2

Suggested solution:

```
def f(x,y):
    res = []
    for y_ in y:
        if y_ <= 0:
            res.append(3*x**2*y_-3*y_)
        else:
            res.append(3*x**2*y_+3*y_)
    return res
```

Question 3.3

Suggested solution:

```
def log_approx(x,n):
    s = 0
    for k in range(1,n+1):
        s += (-1)**(k+1)*(x**k)/k
    return s
```

Question 3.4

Suggested solution:

```
def piecewise(x,a,b):
    if x <= a:
        return 0
    elif x <= b:
        return (x-a)/(b-a)
    else:
        return 1

print(piecewise(0.5,0,1))
```

Question 3.5

Suggested solution:

```
def test_f():
    a = 1
    x = [0,pi/2+1,2*pi]
    expected = [0,1,0]
    tol = 1e-10
    for x_,e in zip(x,expected):
        assert abs(f(x_,a)-e) < tol
```

Question 4.1

The formulation of this question left room for different interpretations, and it raised some questions about the dimensions of the nested list, and whether one could assume that the list is only two-dimensional. The original intention of the question was for `a` to have any dimension, but for the function to return the longest of the lists in `a`, without diving deeper into these lists to search for longer lists. For instance, if we have

```
a = [[1,2,3],[4],[5,[6,7,8,9]]]
```

the function shall return `[1,2,3]`, since this is the longest of the lists in the first dimension of `a`. The solution in this case becomes rather simple:

```
def longest(a):
    long = a[0]

    for e in a[1:]:
        if len(e) > len(long):
            long = e
    return long
```

If the lists in `a` contain only one data type it is possible to use the builtin function `max`. So the simple solution

```
def longest(a):
    return max(a)
```

works if the lists in `a` contain, for instance, only numbers, such as

```
a = [[1,2],[1,2,3],[3,4,5],[5,6,7,8]]
```

However, this solution fails if the lists in `a` contain any mix of data types such as strings, numbers, and lists, and was therefore not given full score on the exam.

Another valid interpretation of the question is to search for the longest list in `a` regardless of the dimension, which for the example above would return `[6,7,8,9]` since this is the longest of all the lists contained in `a`. With this interpretation the question becomes far more complicated and is best solved using a combination of a for loop and recursion.

Question 4.2

The question assumed that the function `is_prime` already existed, so this did not have to be written as part of the answer. A primitive implementation of this function is included here for completeness:

```
from math import sqrt

#not part of the question, included here for completeness:
def is_prime(k):
    if k == 1:
        return False
    for i in range(2,int(sqrt(k))+1):
        if k%i == 0:
            return False
```

```python
    return True

#suggested answer to Q4.2:
def primes(n):
    p = []
    for k in range(2,n+1):
        if is_prime(k):
            p.append(k)
    return p
```

Question 4.3

This question does not specify whether to distinguish between uppercase and lowercase letters. The simplest solution is to treat these as different, so that "racecar" is a palindrome but "Racecar" is not. Both this solution and versions that do not consider letter case were given full score on the exam. Suggested alternative solutions:

```python
def is_palindrome(word):
    for i in range(int(len(word)/2)):
        if word[i] != word[-1-i]:
            return False
    return True

def is_palindrome2(word):
    return word == word[::-1]

"""
Both functions can easily be modified to ignore letter case, for instance
by adding the line word=word.lower() as the first line of the function.
"""
```

Question 4.4

Suggested solution:

```python
with open('counties.txt') as infile:
    infile.readline()
    infile.readline()

    info = {}

    for line in infile:
        words = line.split()
        name = ' '.join(words[:-1])
        pop = int(words[-1])
        info[name] = pop

print(info)
```

Question 4.5

Suggested solution:

```python
class F:
    def __init__(self,a,b,c,d):
```

```
        self.a = a
        self.b = b
        self.c = c
        self.d = d

    def __call__(self,x):
        return self.a*x**3+self.b*x**2+self.c*x+self.d

    def __str__(self):
        return f'{self.a}*x^3 + {self.b}*x^2 + {self.c}*x + {self.d}'
```

Question 5.1

Suggested solution:

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

def ralston(f,U0,T,n):
    t = np.linspace(0,T,n+1)
    u = np.zeros_like(t)
    u[0] = U0
    dt = t[1]-t[0]

    for i in range(n):
        k1 = f(u[i],t[i])
        k2 = f(u[i]+(2/3)*dt*k1,t[i]+(2/3)*dt)
        u[i+1] = u[i]+dt*(0.25*k1+0.75*k2)
    return u, t

#test code, not requested as part of answer:
def rhs(u,t):
    return -u

steps = 100
u,t = ralston(rhs,1,5,steps)
plt.plot(t,u)
plt.plot(t,np.exp(-t))
plt.show()
```

Question 5.2

Suggested solution:

```
class Ralston(ODESolver):
    def advance(self):
        u, f, n, t = self.u, self.f, self.n, self.t
        dt = t[n+1] - t[n]
        k1 = f(u[n],t[n])
        k2 = f(u[n]+(2/3)*dt*k1,t[n]+(2/3)*dt)
        return u[n]+dt*(0.25*k1+0.75*k2)
```

Question 5.2

Suggested solution:

```
def SIRD(S0,I0,alpha,beta,gamma,T):
    N = S0 + I0
```

```
    def rhs(u,t):
        s,i,r,d = u
        ds = -alpha*s*i/N
        di = -ds-beta*i-gamma*i
        dr = beta*i
        dd = gamma*i
        return [ds,di,dr,dd]

    solver = Ralston(rhs)
    solver.set_initial_condition([S0,I0,0,0])
    time = np.linspace(0,T,int(10*T)+1)
    u,t = solver.solve(time)

    s,i,r,d = u[:,0],u[:,1],u[:,2],u[:,3]
    return t,s,i,r,d

t,S,I,R,D = SIRD(370000,30,1.0,0.025,0.25,100)
plt.plot(t,S,label='S')
plt.plot(t,I,label='I')
plt.plot(t,R,label='R')
plt.plot(t,D,label='D')

plt.legend()
plt.show()
```