Technical University of Denmark

Course name	Introduction to programming and data processing
Course number	02633
Aids allowed	All Aids
Exam duration	2 hours
Weighting	All exercises have equal weight

Contents

Submission details

You must hand in your solution electronically:

1. You can upload your solutions individually on CodeJudge (`dtu.codejudge.net/prog-jan18/assignment`) under *Afleveringer/Exam*. When you hand in a solution on CodeJudge, the test example given in the assignment description will be run on your solution. If your solution passes this single test, it will appear as *Submitted*. This means that your solution passes on this single test example. You can upload to CodeJudge as many times as you like during the exam.

2. You must upload your solutions on CampusNet. Each assignment must be uploaded as one separate .py file, given the same name as the function in the assignment:

   (a) `health.py`

   (b) `imageCrop.py`

   (c) `chaos.py`

   (d) `threshold.py`

   (e) `checkParentheses.py`

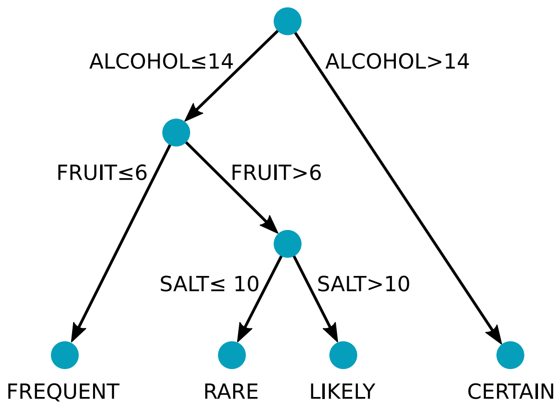
   The files must be handed in separately (*not* as a zip-file) and must have these exact filenames.

After the exam, your solutions will be automatically evaluated on CodeJudge on a range of different tests, to check that they work correctly in general. The assessment of your solution is based only on how many of the automated tests it passes.

- Make sure that your code follows the specifications exactly.
- Each solution shall not contain any additional code beyond the specified function.
- Remember, you can check if your solutions follow the specifications by uploading them to CodeJudge.
- Note that all vectors and matrices used as input or output must be numpy arrays.

Assignment A Health risk

To aid in assessing the health risk of patients, a doctor has created the decision tree shown in the figure. The tree is based on 3 risk factors:



ALCOHOL: Number of alcohol units per week.

FRUIT: Number of pieces of fruit per week.

SALT: Amount of salt in grams per day.

To assess a patient's likelihood of developing a health problem, the doctor starts at the top of the tree. If the patient drinks more than 14 units of alcohol per week, the doctor moves down the right branch of the tree; otherwise, she moves down the left branch. This process continues until the doctor reaches one of the bottom nodes in the tree. The bottom nodes are labelled with the corresponding likelihood of developing a health problem.

Problem definition

Create a function named `health` that takes as input the three risk factors and outputs the corresponding likelihood of a health problem. The output must be in the form of a string, written in capital letters as in the figure.

Solution template

```
def health(ALCOHOL, FRUIT, SALT):
  #insert your code
  return RISK
```

Input	
`ALCOHOL`	Number of alcohol units per week (whole number).
`FRUIT`	Number of pieces of fruit per week (whole number).
`SALT`	Amount of salt in grams per day (whole number).
Output	
`RISK`	Likelihood of developing a health problem (string).
	Takes one of the following values: `FREQUENT`, `RARE`, `LIKELY`, `CERTAIN`.

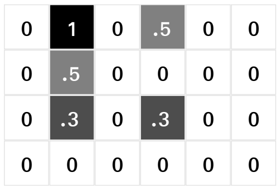
Example

Consider a patient who drinks 8 units of alcohol per week, eats 9 pieces of fruit per week and eats 10 grams of salt per day. Following the decision tree, we first move left because ALCOHOL ≤ 14; we next move right because FRUIT > 6; and we finally move left because SALT ≤ 10. Thus, we arrive at the node labelled `RARE`, which is what the function must return.

A

Assignment B Image crop

A black and white image can be represented by an array where the values in the array denote the grey scale level, e.g. white=0, black=1, and grey scales are numbers between 0 and 1. When an image has a fixed background color, it can be useful to crop the image by removing the outer parts of the image which have constant color.



0	1	0	.5	0	0
0	.5	0	0	0	0
0	.3	0	.3	0	0
0	0	0	0	0	0

Problem definition

Create a function named `imageCrop` that takes as input an array representing a black/white image, as well as a number between 0 and 1 representing the background color. The function must return a cropped image, where all leading and trailing rows and columns with value equal to the given background color are removed. You may assume that the image contains at least one non-background colored pixel.

Solution template

```python
def imageCrop(img_in, background):
  #insert your code
  return img_out
```

Input

`img_in`	Input greyscale image (array of decimal numbers between 0 and 1).
`background`	Background greyscale value (decimal number between 0 and 1).

Output

`img_out`	Cropped greyscale image (array of decimal numbers between 0 and 1).

Example

Consider the image illustrated in the figure above. It can be represented by the following array:

$$\begin{bmatrix} 0 & 1 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

If the background color is given as 0 (white), the first column and the two last columns as well as the last row must be removed, since these columns and rows are constant equal to the background color 0. The result should thus be:

$$\begin{bmatrix} 1 & 0 & 0.5 \\ 0.5 & 0 & 0 \\ 0.3 & 0 & 0.3 \end{bmatrix}$$

Assignment C Chaos

The socalled *logistic equation* is a simple example that can lead to chaotic behavior. It can be expressed by the formula

$$x^* = k \cdot x \cdot (1 - x) \tag{1}$$

The formula takes in a number, x (between zero and one) and computes a new number, x^*. The new number can then be put back into the formula again as x, and so on, thus producing a sequence of numbers.

The number k in the formula is called a *parameter*, and it determines the behavior of the sequence. The parameter k must be in the range $0 \leq k \leq 4$. After some iterations, the sequence settles into a pattern, which depending on k can be constant, periodic, or chaotic.

Problem definition

Create a function named `chaos` that takes as input an initial x value, a value for the parameter k as well as a number of iterations T. The function must return the sequence of numbers that results from iterating the logistic equation T steps.

Solution template

```
def chaos(x, k, T):
  #insert your code
  return seq
```

Input

`x`	Initial value of x (decimal number between 0 and 1).
`k`	Parameter k for the logistic equation (decimal number between 0 and 4).
`T`	Number of iterations (positive whole number).

Output

`seq`	Number sequence from iterating the logistic equation (vector of decimal numbers).

Example

Consider the following input: $x = 0.5$, $k = 3.6$, $T = 10$. With the initial condition $x = 0.5$, we must now compute a sequence of $T = 10$ numbers. The first number can be computed as:

$$x^* = 3.6 \cdot 0.5 \cdot (1 - 0.5) = 0.9.$$

This number is then fed back into the formula as x to compute the second number:

$$x^* = 3.6 \cdot 0.9 \cdot (1 - 0.9) = 0.324.$$

The final sequence of 10 numbers can be computed as:

$$0.9000 \quad 0.3240 \quad 0.7885 \quad 0.6004 \quad 0.8637 \quad 0.4238 \quad 0.8791 \quad 0.3827 \quad 0.8505 \quad 0.4578$$

C

Assignment D Fail prediction

A university teacher wants to make an effort to help students who are in risk of failing her master level course. She decides to analyze how the students' bachelor level grade point average (GPA) is related to whether or not they fail her master level course. She has access to data from N former students: For each of these N students, she knows their bachelor level GPA and whether they failed her master level course. An example of such a dataset with $N = 7$ students is given here:

Grade point average (GPA)	4.20	3.52	9.44	12.00	10.50	7.30	9.44
Failed	1	1	0	0	0	0	1

She decides to make a special effort for students with GPA less than some threshold value, X

$$\text{GPA} \leq X.$$

To determine an appropriate threshold, she uses the following approach:

1. She extracts all the unique GPA values, and sorts them.
2. She only considers all midpoints (averages) between consecutive unique GPA values as possible thresholds.
3. For each possible threshold, she computes the F-score defined as:

$$\text{F-score} = 2 \cdot \frac{[\text{Number of failed students with GPA} \leq X]}{[\text{Total number of students with GPA} \leq X] + [\text{Total number of failed students}]}$$

4. Finally chooses the threshold which yield the highest F-score. If there is more than one threshold that yields the same highest F-score, she chooses the lowest of these thresholds.

Problem definition

Create a function named `threshold` that takes as input a vector containing GPAs for N students and a vector of same length containing ones or zeros indicating whether the students have failed or passed. The function must return the threshold X computed as described above.

```
def threshold(GPA, fail):
  #insert your code
  return X
```

Input

`GPA`	Grade point average for each student (vector with decimal numbers).
`fail`	Fail or pass (coded as 1 and 0) for each student (vector with zeros and ones).

Output

`X`	Threshold (decimal number).

Example

Consider the data in the table above. There are six unique values (9.44 occurs twice), and sorting the unique values gives list below. Next, the five possible threshold values can be computed as the average between the consecutive unique GPA-values. For each threshold the F-score is then computed.

Sorted unique GPA values	3.52	4.20	7.30	9.44	10.50	12.00
Possible threshold X	3.86	5.75	8.37	9.97	11.25	
Number of failed students with GPA $\leq X$	1	2	2	3	3	
Total number of students with GPA $\leq X$	1	2	3	5	6	
Total number of failed students	3	3	3	3	3	
F-score $\approx$	0.5	**0.8**	0.667	0.75	0.667	

The highest F-score is 0.8, and the function must return the corresponding threshold which is $X = 5.75$.

Assignment E Matching parentheses

Parentheses can be used to group terms in an arithmetic expression in a text string, and one can us different types of parentheses (round, square, and curly) to make complex arithmetic expressions easier to read. An example could be `1/(a + 1/[b + 1/{c + 1/d}])`. The algorithm below can check if the parentheses in an expression are balanced, i.e. that each opening symbol has a matching closing symbol of the same type, and that the pairs of parentheses are properly nested.

Algorithm	Description
see below	The algorithm reads each character of a sring of length N (all characters except `([{)]}` are ignored). When the algorithm encounters a start-parenthesis, it adds it to the end of a list. When the algorithm encounters an end-parenthesis, it checks that the end-parenthesis is of the same type as the most recent start-parenthesis, which it then removes from the list in case of a match. If the type of parenthesis does not match, the algorithm returns the index in the string where the error occured. If all parentheses match, the list should be empty after going through all characters in the string: In that case, the algorithm returns zero. Otherwise, if the list is not empty, the algorithm must return N+1.

```
input: A string, str, with N characters. Let
       str[i] denote the i'th character in the
       string.
Let list denote a list of characters. Initially list
 is empty.
for i=1 to N
    if str[i] is one of the three possible
     start-parentheses
        Append str[i] to list.
    else if str[i] is one of the three possible
     end-parentheses
        if list is not empty and str[i] matches
         the type of the last character in list
            Remove the last character in list
        else
            return i
if list is empty
    return 0
else
    return N+1
```

Problem definition

Create a function named `checkParentheses` that takes a text string as input, and returns a non-zero index where a parenthesis error is discovered according to the algorithm above, or zero if parentheses match. The function must handle round, square and curly parentheses.

Solution template

```python
def checkParentheses(str):
  #insert your code
  return err
```

Input	
`str`	Input string (text-string).

Output	
`err`	Index of parenthesis error or zero if no error detected (whole number).

Example

Consider the following text string:

`([a)]{`

The fourth character `)` in the string does not match the most recent start-parenthesis, which is `[`. Thus the algorithm must return the value 4.