Technical University of Denmark

Course name	Introduction to programming and data processing
Course number	02631, 02632, 02633, 02634, 02692
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 test your solutions individually on CodeJudge

   https://dtu.codejudge.net/prog-aug20/assignments/

   under *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 all your solutions at DTU's Online exam site. Each assignment must be uploaded as one separate .py file, given the same name as the function in the assignment:

   (a) `double_sort.py`
   (b) `compile.py`
   (c) `pack_clips.py`
   (d) `navigate.py`
   (e) `nonpalindromic_prime.py`

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

After the exam, your solutions submitted to DTU Inside 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, though `import` statements can be included.
- 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 Double sort

Given a matrix, we want to sort the elements in each column in ascending order based on magnitude (absolute value) and after that sort the columns in ascending order based on the signed value of the element in the first row. If the elements in the first row are equal, then the columns with equal elements should be ordered according to the order in the original matrix.

Problem definition

Create a function called `double_sort` which takes as input $\mathbf{X}$ which is a matrix. The function should return a sorted matrix, $\mathbf{Y}$.

Solution template

```
def double_sort(X):
  #insert your code
  return Y
```

Input	
`X`	Matrix with one or more columns and one or more rows.

Output	
`Y`	Matrix with one or more columns and one or more rows.

Example

Consider the matrix

$$\mathbf{X} = \begin{bmatrix} 7 & -1 & 0 & 5 \\ 2 & 5.2 & 4 & 2 \\ 3 & -2 & 1 & 4 \end{bmatrix}$$

First sorting each of the rows individually based on magnitude

$$\begin{bmatrix} 2 & -1 & 0 & 2 \\ 3 & -2 & 1 & 4 \\ 7 & 5.2 & 4 & 5 \end{bmatrix}$$

Then sort the columns based on the elements in the first row

$$\mathbf{Y} = \begin{bmatrix} -1 & 0 & 2 & 2 \\ -2 & 1 & 3 & 4 \\ 5.2 & 4 & 7 & 5 \end{bmatrix}.$$

This matrix is returned.

A

Assignment B Compile

We want to construct a compiler that takes a string representing a small program as input and compiles and executes it. The string representing a program contains positive integers and four alrithmetic operations: "A", "S", "M" and "D" for add, subtract, multiply and divide, respectively.

A program may be given by a single integer, or multiple integers with one of the four operations in between. The integers and operations are separated by one space character. The operator precedence means that operations should be performed with the leftmost operation first so that, e.g., "2 A 3 M 4" means $(2 + 3) \times 4$.

It can be assumped that only correct programs are inputted.

Problem definition

Create a function called `compile` which takes one input: the program as a string. The function should return the numeric result with an evaluation of the program.

Solution template

```
def compile(program):
  # insert your code
  return result
```

Input	
`program`	String with the program

Output	
`result`	Decimal or integer number with the result of the computation.

Example

Regard the program "2 A 3 M 4 D 8". This should be interpreted as $((2 + 3) \times 4)/8$ and this should result in $(5 \times 4)/8 = 20/8 = 2.5$, so the decimal number 2.5 is returned.

B

Assignment C Pack clips

We want to fit audio clips of a certain duration into a specified time slot with a total duration. Each duration is given in integer minutes and seconds, and the total duration of the time slot is also given as minutes and seconds.

Problem definition

Create a function called `pack_clips` that takes as the first argument a $(N \times 2)$-matrix where each row is a duration for an audio clip and the first column specifies the number of minutes of the clip and the second column specifies the number of seconds for the clip. As the second argument the function takes the duration of the time slot as a 2-element vector with the first element specifying the minutes and the second element the seconds. The function should return the maximal number of clips where the sum of the duration is lower or equal to the total duration of the time slot. The clips should be selected sequentially starting from the first row. If none can fit, then the function should return zero. If all can fit, then the function should return the number N.

Solution template

```
def pack_clips(durations, total):
  # insert your code
  return clips
```

Input	
`durations`	$(N \times 2)$-matrix with durations. $N > 0$
`total`	2-element vector with total duration of the time slot.

Output	
`clips`	Integer scalar with the number of audio clips that fit in the time slot.

Example

Consider the total duration to be 10:20, and 6 audio clip durations to be

```
3:10 4:20 0:50 1:50 10:10 1:00
```

The durations are represented in a matrix

$$\begin{bmatrix} 3 & 10 \\ 4 & 20 \\ 0 & 50 \\ 1 & 50 \\ 10 & 10 \\ 1 & 0 \end{bmatrix} \tag{1}$$

while the total duration is represented in the vector $[10, 20]$.

1. First clip has duration 3:10 and fits the time slot.
2. First two clips have duration 7:30 and fit the time slot.
3. First three clips have duration 8:20 and fit the time slot.
4. First four clips have duration 10:10 and fit the time slot.
5. First five clips have duration 20:20 and do not fit the time slot.

Therefore, the function should return 4.

C

Assignment D Navigate

We want to navigate the DTU campus where we have a target building represented by a 3-digit number and an origin building also represented by a 3-digit number, each in one of four quadrants. Buildings in the 100s range (100-199) are in the Northeast quadrant, 200s buildings in the Northwest, 300s buildings in the Southwest and 400s buildings in the Southeast.

We want a direction description with the origin building number, the direction and the target building number. The specifiction of the direction may be S, N, W, E for South, North, West and East, or SW, NW, SE, NE for Southwest, Northwest, Southeast and Northeast.

If our origin is 413 and target 202, we want a string "413NW202", indicated a Northwest direction going from building 412 (Southeast) to building 202 (Northwest). An origin-target combination of 414 to 101 should be "414N101".

If the origin and target building are in the same quadrant then the direction should be "H", e.g., 303 and 306 should result in the direction description "303H306".

Problem definition

Create a function called `navigate` that takes two integer inputs representing building numbers, the first the origin, the second the target. It should return a string with a concatenation between the origin building as a string, the direction, and the target building as a string.

Solution template

```
def navigate(origin, target):
    #insert your code
    return direction
```

Input

`origin`	3-digit integer specifying the origin building.
`target`	3-digit integer specifying the target building.

Output

`direction`	String with origin building, direction (S, N, E, W, NW, NE, SW, SE or H), and target building concatenated.

Example

Consider the case with the origin building being 413 and target 202, then the string "413NW202" should be returned.

D

Assignment E Nonpalindromic prime

A palindromic prime is a prime number that is also a palindromic number. A palindromic number is a number that is the same when the digits are reversed. Examples of palindromic primes are 2, 3, 11 and 101. A nonpalindromic prime is a prime number that is not palindromic. Examples of nonpalindromic primes are 13, 17, 31 and 47. We want to have a function that returns the n'th nonpalindromic prime.

Problem definition

Create a function called `nonpalindromic_prime` which takes an integer that specifies the index in the nonpalindromic prime series and returns that prime.

Solution template

```
def nonpalindromic_prime(n):
  # insert your code
  return prime
```

Input	
`n`	Integer that specifies which n'th non-palindromic prime should be returned.

Output	
`prime`	Integer as the n'th nonpalindromic prime.

Example

For $n = 7$, we list the first primes

$$2\ 3\ 5\ 7\ 11\ 13\ 17\ 19\ 23\ 29\ 31\ 37\ 41\ 43\ \ldots$$

When the palindromes are removed, we get the list of nonpalindromic primes

$$13\ 17\ 19\ 23\ 29\ 31\ 37\ 41\ 43\ \ldots$$

The 7th nonpalindromic prime is 37 and this is returned.

E