← Zurück

File write (8 pt)

Frage 3 von 5 (8 Punkte)
Nicht beantwortet

Note:

> This is an independent task. That means, the "Logger" class in this task has nothing (!!!) to do with any other task of the exam!

Tasks to solve:

Write a simple logger class (class name: "**Logger**") that creates an append-only file when a new log-object is initiated.
The logger comprises:

- a private entry number generator starting with 1 and increments with each new log entry
- one function '**add_entry()**' to append a new log entry to the file 'log.txt' starting with the entry number at the begining.
- each entry number is written with minimum 5 characters and enclosed in square brackets.

Two typical log entries look like:

```
[    1] First entry.
[    2] Second entry.
...
[  125] This is the 125th entry.
```

With this 'logger test script' you can test your Logger class:

```
logger = Logger('log.txt')
logger.add_entry('First entry.')
logger.add_entry('Second entry.')
logger.add_entry('The PDS MEP is running!')
```

Important:

- Copy your whole solution (with the 'logger test script') into the text box below.
- Copy also the content of your generated 'log.txt' File on the end of your solution.

P

Loop (8 pt)

Frage 2 von 5 (8 Punkte)
Nicht beantwortet

The given 2D list contains some empty (NA = no value is available) elements. Write a function called **fixMatrix(....)** that locates missing elements and fills them up by linear interpolation of the previous and next (adjacent = benachbart) elements in the same row. So, if the element E22 is missing for instance, it has to be filled up using the following formula:

```
E22 = (E21+E23)/2
```

The 2D list looks like:

```
[[E00, E01, E02, E03],
 [E10, E11, E12, E13],
 [E20, E21, E22, E23],
 [E30, E31, E32, E33]]
```

Tasks to solve:

Program a function **fixMatrix(...)** with the following rules for edge conditions:

- If there is no previous element (i.e. the missing element is the first of the row) use the last element of previous row. Example: `E10 = (E03+E11)/2`
- If there is no next element (i.e. the missing element is the last of the row) use the first element of next row. Example: `E23 = (E22+E30)/2`
- The first element (i.e. E00) and the last element (i.e. Enn) are never 'NA's.
- There are never multiple 'NA' next to each other (=nebeneinander).

Write the function as generic as possible and try it with the following two 2D lists:

```
missingEl1 = [[   1,'NA',   3,   4],
              ['NA',   6,'NA',   8],
              [   9,  10,  11,  12],
              [  13,'NA',  15,  16]]


missingEl2 = [[   1,'NA',   3,   4,   5],
              [  11,'NA',  13,  14,'NA'],
              [  21,  22,'NA',  24,  25],
              [  31,  32,  33,'NA',  35],
              ['NA',  42,  43,  44,  45],
              ['NA',  52,'NA',  54,  55]]
```

Some Code Hints:

```
def fixMatrix(matrix):
    rows = len(matrix)           # number of rows
    cols = len(matrix[0])        # number of cols
    ...
print(fixMatrix(missingEl1))
print(fixMatrix(missingEl2))
```

Important:

Copy your whole solution into the text box below.

Iterator & Iterable (8 Pt)

Frage 1 von 5 (8 Punkte)
Nicht beantwortet

Given is following class:

```
class DownSizeMutable:
    def __init__(self, start, downsize = 1):
        self.current = start
        self.downsize = downsize

    def __iter__(self):
        return self

    def __next__(self):
        if self.current <= 0:
            raise StopIteration
        else:
            self.current -= self.downsize
            return self.current + self.downsize
```

Tasks A:

Think about: *Are objects of the class* `DownSizeMutable` *'iterable'? Answer: YES!!!*

Program a function `isIterable(object) -> bool` that proves if the passed object is iterable.
Call the function **isIterable(...)** with the following short python script:

```
dsm = DownSizeMutable(70, 7)
print(isIterable(dsm))
```

Task B:

Demonstrate with a **for-loop** that the object `dsm` is iterable. Save the values of `DownSizeMutable(70, 7)` in a list with the name **myList**.
Print out the content of **myList**.
The printed result will be: `[70, 63, 56, 49, 42, 35, 28, 21, 14, 7]`

Task C:

The result of the task C is the same as in TaskB.
BUT: Solve the same task with a **while-loop**.

Important:

- Copy your solution for Task A, B and C into the text box below.
- The different parts of these 3 tasks (Task A, Task B and Task C) must be **clearly labeled** in the text box below.
  -> **In short:** It must be very clear, which code belongs to which task (use comments, titles, lables, etc.)

← Zurück

Inheritance (8 pt)

Frage 4 von 5 (8 Punkte)
Nicht beantwortet

Tasks to solve:

- Create the class **MyStr** which extends the built-in class **str** with a property **wordCount** that returns the word count of the string.

Note:

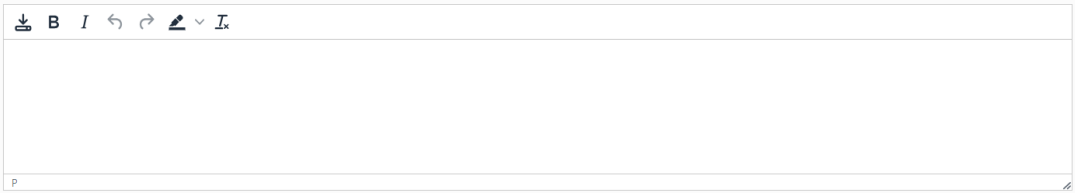
Words are separated by spaces. That means, numbers in text also count as words.

The following 'inheritance test script' should print the number 69 with your implementation of **MyStr**.

```
# 'inheritance test script':
txt = MyStr("Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.")
print(txt.wordCount)          # prints the number 69 to console (there are 69 words in the text!)
```

Important:

Copy your whole solution (with the class Mystr and the 'inheritance test script') into the text box below.



P

Fragenliste aus | Bearbeitungsstand | Test beenden

← Zurück

Lambda function & list comprehension (8 pt)

Frage 5 von 5 (8 Punkte)
Nicht beantwortet

Given is following code:

```
myList = [5.69, 9, 8.5, 7.36, 9.52, 7, 6.01, 6.68, 2.28, 9.78, 6, 9.33, 2.05, 3.40, 7.76, 6.1, 5.68]

def myfilter(mylist, lamFunc):
    return [lamFunc(x) for x in mylist]   # returns the list resulting from the list comprehension
```

Tasks A:

program a **lambda function** 'decimal_part' with

```
decimal_part = lambda x: ...# your task ! ..............
```

which is called in the following script:

```
result = myfilter(myList, decimal_part)
print(result)
```

The lambda function should return all decimal parts of `myList`.
The result of the `print(result)` should be something like:

```
[0.69, 0, 0.5, 0.36, 0.52, 0, 0.01, 0.68, 0.28, 0.78, 0, 0.33, 0.05, 0.4, 0.76, 0.1, 0.68]      # 9, 7 and 6 have NO decimal places (=Dezimalstellen)
```

Tasks B:

Change the **list comprehension** in the function `myfilter(mylist, lamFunc)` such that the resulting list for the given `myLlist` only contains decimal values >0.

The final list of `print(result)` should look as follow:

```
[0.69, 0.5, 0.36, 0.52, 0.01, 0.68, 0.28, 0.78, 0.33, 0.05, 0.4, 0.76, 0.1, 0.68]      # all 0 values removed
```

Here are some code snippets that might be of some help to you... ;-):
Check out the solutions and benefit!

```
print("5 != int(5):", 5 != int(5))
print("5.1 != int(5.1):", 5.1 != int(5.1))
print("int(5.1):", int(5.1))
print("int(0.33):", int(0.33))

print("5 % 3 = ", 5%3)
print("5.1 % 3 = ", 5.1%3)
print("round(5.1 % 3, 2): ", round(5.1 % 3, 2))
print("round(5.666 % 1, 2): ", round(5.666% 1, 2))
```

Important:

Copy your solution of task A and task B individually or (also possible) together in ONE solution into the text box below.