Exam Scientific Programming 101

Date: ...

Name:

.....

- This is a digital exam. The exam consists of 3 assignments in which you have to write a short python program.
- Create one file for all your solutions called `sp101_exam0.py`. This is the file you'll hand in at the end of your exam.
- This is a **closed-book exam**, meaning:
  - Close all programs on your laptop, except:
    - The Pulsar editor (with only the python file for this exam open)
    - Your terminal
    - The submit page for this exam: (practice; no submit page)
  - You're not allowed to use any other webpage.
  - You're not allowed to look at any existing code you've written before this exam.
  - You cannot get any help with programming during the exam.
- You're only evaluated based on the *correctness* of your solutions, code design is not important. So, **you don't have to worry about comments or style guides**.
- You're not allowed to use `numpy`, `csv` or any other external Python module.
- You can test your code using checkpy. First download the tests for the exam:

  ```
  checkpy -d /spcourse/exam-tests
  ```

  Run checkpy:

  ```
  checkpy sp101_exam0
  ```

- Submit your solutions on the website when you're done. **Check with the teacher present if you handed in your assignment correctly before leaving the exam venue.**
- After submitting, hand in this exam paper showing your student card or ID.

Time submitted (to be filled out by teacher):

.....

1 Distance

We have a list of successive distances from a trip and we would like to compute the cumulative distances from the start. Write a function called `cumulative(distances)`. This function takes a list of distances and returns a list with cumulative distances (i.e., the distances summed from the start up to that point).

Example usage:

```
distances = [2, 2, 1]
cumulative_distances = cumulative(distances)
print(cumulative_distances)
```

Expected output:

```
[2, 4, 5]
```

Example usage:

```
distances = [19, 32, 7, 1, 5, 1]
cumulative_distances = cumulative(distances)
print(cumulative_distances)
```

Expected output:

```
[19, 51, 58, 59, 64, 65]
```

2 Starting letter

Write a function `filter_words_starting_with(text, letter)`. This function has two input strings: `text` and `letter`. The function finds all words from `text` that start with `letter` and it returns those words in a list. You may assume that the string `letter` indeed always contains a single letter.

Example usage:

```
example_text = "David Donald Doo dreamed a dozen doughnuts and a duck-dog, too."
print(filter_words_starting_with(example_text, "d"))
```

Expected output:

```
['David', 'Donald', 'Doo', 'dreamed', 'dozen', 'doughnuts', 'duck-dog']
```

Tips:

- You can split a text into a list of words using the `text.split(" ")` method.
- You can convert a word to lowercase by using `word.lower()`
- You can remove punctuation with `word.strip(",.?! ")`

3 Expense

You're writing a program that keeps track of your expenses. You're using a dictionary that keeps track of the monthly expenses in euros per category (*food*, *rent*, *internet*, *utilities*, *social activities*, etc.). Now you would like to know what percentages of you monthly expenses these categories represent.

Write a function `euros_to_percentage(expenses)` that accepts a dictionary containing the expenses in euros. It should create a new dictionary containing the expenses in percentages.

Have a look a this example:

```
expenses_january_in_euros = {'rent': 735, 'utlities': 221,
                             'food': 167, 'social activities': 185,
                             'internet + netflix + spotify': 58, 'phone': 25}
expenses_january_in_percentages = euros_to_percentage(expenses_january_in_euros)
print(expenses_january_in_percentages)
```

This should produce the output:

```
{'rent': 52.83968368080517, 'utlities': 15.88785046728972, 'food': 12.005751258087706,
'social activities': 13.299784327821712, 'internet + netflix + spotify': 4.169662113587347,
'phone': 1.7972681524083394}
```

Note: the order in which this result is printed does not need to be the same as the example above. Check whether each category has the right value. If this is the case, your code probably works!