Scientific Programming 1; Practice Exam

Name:

....

- This is a digital exam consisting of 4 assignments where you will write short Python programs.
- You will complete the exam using the online editor linked on the course website.
- Use a single file for all solutions called sp1_exam_practice.py. This file is already open in the online exam editor.
- You may use the course website (sp.proglab.nl) as a recource. Keep in mind that:
  - you cannot use any other website,
  - you cannot use rely on any existing code you've written before this exam,
  - you cannot get assistance with the programming during the exam.
- You will be **evaluated solely on the correctness** of your solutions. Code design, comments, and style are not important, so you do not need to worry about them.
- Do not use external modules such as numpy, csv, or others unless the assignment specifically says you can.

- **Submitting the exam**. When you have completed all assignments:
  - use the **submit button** on the website to ensure your final edits are committed,
  - **go to the teacher** with your **student card or ID** and this exam sheet,
  - have the teacher mark your work as completed,
  - hand in this exam sheet.

* This practice exam contains 5 questions in stead of 4 to have a bit more practice material. The actual exam will contain 4 questions.

Assignment 1a: Collatz

Write a function `collatz(number)`. This function should return the collatz sequence for the passed `number`. The Collatz sequence for a number is found by using the following steps:

- If the number is **odd** it should be multiplied by 3, after which 1 is added
- If the number is **even** it should be divided by 2

The Collatz sequence of a number will eventually return to 1, after which you can stop.

Example usage:

```
print(collatz(3))
print(collatz(16))
```

Expected output:

```
[10, 5, 16, 8, 4, 2, 1]
[8, 4, 2, 1]
```

Tip: To prevent your program from converting integers to floats, make sure to use integer division (//)!

Assignment 1b: Bounce

Write a Python function named `bounce(n)`. The function produces a sequence of numbers. This sequence starts with the number 1.0 and follows a pattern where it multiplies each number by 4 until it reaches a value of 100 or higher, after this it divides the number by 2 until it reaches a value lower than 2. This process continues based on the specified count `n`, alternating between multiplication by 4 and division by 2 to create a list of length `n`.

Example usage:

```
    print(bounce(20))
```

Expected output:

```
    [1.0, 4.0, 16.0, 64.0, 256.0, 128.0, 64.0, 32.0, 16.0, 8.0, 4.0, 2.0,
    1.0, 4.0, 16.0, 64.0, 256.0, 128.0, 64.0, 32.0]
```

Assignment 2: Swap

Create a Python function called `swap_words(text)` that takes a string text as its parameter. The function goes over all the words in the `text` and returns a string where the first word of the input is swapped with the second, the third is swapped with the fourth, etc. If the input text contains an odd number of words, the last word of the text will not be swapped with anything.

Example usage:

```
print(swap_words("Why is a raven like a writing desk?"))
print(swap_words("You can always take more than nothing."))
```

Example output:

```
is Why raven a a like desk? writing
can You take always than more nothing.
```

Assignment 3: Gregory-Leibniz

The Gregory-Leibniz series is a way to approximate π The series follows this pattern:

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \frac{4}{13} - \frac{4}{15} + \frac{4}{17} - \dots$$

So the series alternates between addition and subtraction of a term that gets smaller every step.

- Step 1: take 4
- Step 2: subtract $\frac{4}{3}$
- Step 3: add $\frac{4}{5}$
- Step 4: subtract $\frac{4}{6}$
- etc. (the divisor increases by 2 at each step).

Create a function `gregory_leibniz(n)`, that approximates π in `n` number of steps. Example usage:

```
print(gregory_leibniz(1))
print(gregory_leibniz(10))
print(gregory_leibniz(1000000))
```

Expected output:

```
4.0
3.0418396189294032
3.1415916535897743
```

Assignment 4: Barca

For this assignment you need to read the files `barca.txt` and `barca\_short.txt`. This contains the results for football matches of F.C. Barcelona (from seasons 11/12 to 13/14). The file contains the following data:

```
29/08/11,Villarreal,won,5,0,home
10/09/11,Sociedad,draw,2,2,away
17/09/11,Osasuna,won,8,0,home
...
03/05/14,Getafe,draw,2,2,home
11/05/14,Elche,draw,0,0,away
17/05/14,Ath Madrid,draw,1,1,home
```

As you can see, the data fields are separated by a comma and contain the following information:

1. Date of the match
2. The opponent
3. The result: won/lost/draw
4. The number of goals for Barcelona
5. The number of goals for the opponent 6. The location: away/home

Playing at home is considered an advantage for any football club. Let's see if this is true for Barcelona. Write a function home_advantage(filename). This function computes the difference between the amount of home matches won and the amount of away matches won. (So, a positive number means that more home matches were won than away matches.)

Example usage 1:

```
advantage = home\_advantage('barca\_short.txt')
print(advantage)
```

Expected output:

```
2
```

Example usage 2:

```
advantage = home\_advantage('barca.txt')
print(advantage)
```

Expected output:

```
15
```