Introduction to Scientific Computing, PSCB57, Fall 2018
Assignment 1
Floating Point Numbers

Instructions

- You must submit the assignment electronically via Quercus. The deadline for this assignment is Monday, September 24th, 9am. Late assignments will not be accepted unless accompanied by supportive documentation.
- This assignment comes in multiple parts. Submit all your answers in one Jupyter Notebook file with the file type `ipynb`.
- Use mark-down cells to add your name, student number. Also use mark-down cells and python comments to describe your code! Well documented code might help you with the quiz.
- Do not use any packages or libraries in this assignment.
- You must be present at the tutorial on Tuesday where you will be quizzed about your assignment. If you do not show up or fail to pass the quiz, your assignment might be marked as 0% even if it was correct.
- Plagiarism is taken very seriously. However, you are not expected to work in solitude and are encouraged to talk to your classmates. But keep in mind that if you submit an assignment, you have to fully understand it in order to pass the quiz.

Part 1

This part is about the precision of floating point numbers. Have a look at the following function.

```
def f(x):
    return x/x
```

The return value is obviously 1, no matter what the function argument is. Your task is to modify the function to get the following behaviour:

```
>>> f(1e-16)                          >>> f(1.12e-16)
0.0                                   1.9825411154020653
>>> f(2.5e-16)                        >>> f(-1e15)
0.8881784197001251                    1.0
>>> f(1e-13)                          >>> f(1e-15)
0.9992007221626409                    1.1102230246251565
```

You are not allowed to change the mathematical nature of the function. In other words, you should be able to simplify the expression mathematically to always get exactly 1. Thus, to get the desired effect in your python function, you need to make use of the fact that computers use floating point numbers with a finite precision. To make things a little more interesting, you are only allowed to add the following characters (multiple times if you want) to the body of the function `f`:

```
(    )    +    -    1.
```

Part 2

This part is about the finite range of floating point numbers. Write a function `g(x)` that returns `inf` if x is any arbitrary positive floating point number, `-inf` if x is any arbitrary negative floating point number, and 0 if x is exactly 0. Calculate the return value by multiplying the argument x with a constant multiple times. Create a for loop to do this multiplication and hard code the number of iterations. Thus, your function should include a loop of the following form:

```
for i in range(...):
```

Think about how many iterations you need such that your function works with any double precision IEE754 floating point number, i.e. what argument requires the most iterations? Write down your answer in the notebook.

Finally, when you pass your function an integer rather than a floating point number, it should return a finite number. Make sure you understand why this happens (there might be a question about this on the quiz). Hint: look up what dynamically typed means and think about what data types python uses when it executes your function.