Lab01: Calculating π
CSE2050 Fall 2017
Instructor: Wei Wei
TA: Zigeng Wang & Param Bidja

1. Introduction

In this exercise, we will complete several pieces of Python programs that calculate the value of π and compare their performance.

2. Objectives

The purpose of this assignment is to give you experience in:

- Using mathematical formulas in Python.
- Converting mathematical expressions into Python expressions.
- Refreshing knowledge on function and loops in Python.
- Doing a little bit study on series based π approximation.

***Note**: Before you start, if you are not familiar with if/else statement, for loop, while loop or function in Python, you are recommended to review the sample codes we covered in lecture first.*

3. Background

3.1. Calculating π using series

π is the ratio of a circle's circumference to its diameter. From the early old days, mathematicians have been trying different methods to calculate the value of π. Calculating π by using an infinite series is one popular category for π approximation.

In this lab assignment, we will be implementing two types of series for π calculation, namely, the Gregory-Leibniz series and the Nilakantha series.

- Gregory-Leibniz series is one of the simplest series to calculate the value of π. Here is the formula to apply:

$$\pi = \left(\frac{4}{1}\right) - \left(\frac{4}{3}\right) + \left(\frac{4}{5}\right) - \left(\frac{4}{7}\right) + \left(\frac{4}{9}\right) - \left(\frac{4}{11}\right) + \left(\frac{4}{13}\right) - \left(\frac{4}{15}\right) \dots$$

- Nilakantha series is a somewhat more complicated series to calculate the value of π. Here is the formula to apply:

$$\pi = 3 + \left(\frac{4}{2*3*4}\right) - \left(\frac{4}{4*5*6}\right) + \left(\frac{4}{6*7*8}\right) - \left(\frac{4}{8*9*10}\right) + \left(\frac{4}{10*11*12}\right) \dots$$

These series, with every iteration, will get closer and closer to π. In other words, the more terms being considered in the series, the closer we can get π.

However, the speeds at which these series approach to π are different. Formally, we call this type of speed, *rate of convergence*. By including the same number of terms in these series, the approximation accuracy can be quite different. Let's have an example of including 4 terms, where we use $\hat{\pi}$ to denote the approximation for π,

- Gregory-Leibniz series

$$\hat{\pi} = \left(\frac{4}{1}\right) - \left(\frac{4}{3}\right) + \left(\frac{4}{5}\right) - \left(\frac{4}{7}\right) = 2.8952$$

- Nilakantha series (where 3 in the following equation is an offset, but not the first term)

$$\hat{\pi} = 3 + \left(\frac{4}{2*3*4}\right) - \left(\frac{4}{4*5*6}\right) + \left(\frac{4}{6*7*8}\right) - \left(\frac{4}{8*9*10}\right) = 3.1397$$

As we can see above, with both four terms, Nilakantha series provides a better estimation of π than Gregory-Leibniz series, since 3.1397 is much closer to π (3.1415927) than 2.8952.

4. Assignment

When you are reading this lab assignment, you must have already logged in the Mimir Classroom Website, joined the CSE2050-001 course and downloaded the skeleton zip file. In the zip file, together with this lab assignment, you can find a skeleton code. All you need to do is to complete the skeleton code based on the instructions and submit it to the Mimir system.

4.1. π calculation function

Now, open the skeleton code. You can find, at the very beginning of the skeleton code, two functions that you need to complete. These functions use the Gregory Leibniz series and Nilakantha series, respectively, to calculate π and return the result. The input parameter variable *n* denotes the number of terms that should be included to calculate π.

- For example, if we call function GregoryLeibniz as the following,

```
GregoryLeibniz(4)
```

  Then, the function should calculate π with the first 4 terms of Gregory Leibniz series like

$$\hat{\pi} = \left(\frac{4}{1}\right) - \left(\frac{4}{3}\right) + \left(\frac{4}{5}\right) - \left(\frac{4}{7}\right) = 2.8952 \ldots$$

And return value as 2.8952…

- For another example, if we call function Nilakantha like,

```
Nilakantha(2)
```

Then, the function should calculate π with the first 2 terms of Nilakantha series,

$$\hat{\pi} = 3 + \left(\frac{4}{2*3*4}\right) - \left(\frac{4}{4*5*6}\right) = 3.1333\ldots$$

Offset; 1^{st} Term; 2^{nd} Term

and return value as 3.1333… (Please note that the offset 3 in the equation above is not a term in the series)

After implementing the two functions, we can run the python script in Python Shell. The following piece of code in the skeleton code will call the two newly implemented functions, and print the number of terms followed by the estimated values of π.

```
n = 100
for i in range(n):
    print(i+1, GregoryLeibniz(i+1), Nilakantha(i+1))
```

In the Python Shell, you should see something show up similar to the following. If your output is different, please double check your code.

```
>>>
=== RESTART: C:\Users\Zigeng Wang\Dropbox\CSE2050\Zigeng and Wei\twopi.py ===
1 4.0 3.1666666666666665
2 2.666666666666667 3.1333333333333333
3 3.466666666666667 3.145238095238095
4 2.8952380952380956 3.1396825396825396
5 3.3396825396825403 3.1427128427128426
6 2.9760461760461765 3.1408813408813407
7 3.2837384837384844 3.142071817071817
…(lines of output omitted)
94 3.1309546566679236 3.1415923620424726
95 3.152118677831945 3.1415929361216497
96 3.131176269454982 3.1415923797055036
97 3.151901658056018 3.1415929191758516
98 3.13138888375431977 3.141592395970107
99 3.1516934060711166 3.1415929035585544
100 3.1315929035585537 3.1415924109719824
GregoryLeibniz 0 0
Nilakantha 0 0
```

4.2. Rate of convergence comparison

The second part of our lab is to further compare the rate of convergence of the two series. We should add numbers of lines of code into the following corresponding positions.

Go back to the skeleton code. The following piece of code will compute and print the minimum number of terms required in order to approximate π within an accuracy of 0.001 using Gregory Leibniz series (in other words, the approximated π value should be within 3.1415927-0.001 to 3.1415927+0.001), together with its corresponding approximated value of π. The value of the minimum number of terms should be stored in variable *i*, and its corresponding approximated value of π should be stored in variable *result*.

```
pi = 3.1415927
accuracy = 0.001
i = 0
diff = 1
result = 0
### add your code here
print("GregoryLeibniz", i, result)
```

Do the same calculation to Nilakantha series and complete the last piece of code. It should print the minimum number of terms and the approximated π value with the same accuracy with Nilakantha series.

```
i = 0
diff = 1
### add your code here
print("Nilakantha", i, result)
```

In this subsection, we only need to print, in total, two lines of information. Do not modify any exiting statement or add any extra print statement.

5. Submit your work to Mimir

Submit your code to Mimir after you complete your code. The Mimir will automatically check the output of your code and grade your submission. You can submit your code to Mimir up to **30 times** to refresh your existing score before the submission deadline.

6. Due date

This lab assignment is worth **2 points** in the final grade. It will be due by **11:59pm on Sep 14th 2017**. A penalty of **10% per day** will be deducted from your grade, starting at 12:00am.