Python Programming

Programming Exercises 03 (Class)

Aug/Sep, 2019

Dr. Ram P Rustagi
Professor, CSE Dept
KRP, KSGI
rprustagi@ksit.edu.in

Resources and Acknowledgements

- Intro to Programming in Python
  - Sedgewick, Wayne, Dondero (Princeton Univ)
- A first course in programming
  - https://introcs.cs.princeton.edu/python/home/
  - https://introcs.cs.princeton.edu/java/home/
- Python for everybody
  - https://www.py4e.com
- Web Applications for everybody
  - https://www.wa4e.com
- Turtle Graphics
  - https://docs.python.org/3/library/turtle.html
- https://www.w3schools.com/python/
  - Basic Python Tutorial

Instructions

- Install library module stdlib-python.zip
  - https://introcs.cs.princeton.edu/python/code/
  - Follow the installation instructions.
- Ensure installation is proper
  - Check for import of following module i.e.

```
import stddraw
import stdio
import stdarray
```

Stop Watch

- Ex 01: Implement stopwatch
  - Design a class called `Stopwatch` providing following function
  - `elapsedTime`():
    - returns time since stopwatch is started.
    - Note: use time.time() function to get system time.

    - For a given N (e.g. 100), use stop watch to compute
      - sum $1^2+2^2+...+N^2$ using `i**2`
      - sum $1^2+2^2+...+N^2$ using `i*i`
      - Using Stopwatch class, compare the time computations.
      - Repeat the process for computing cube i.e.
        - » compare time computations for `i**3` vs `i*i*i`

Stop Watch

- Ex 02: Define sub class `Stopwatch2` which inherts `Stopwatch` and provides following additional methods
  - `reset()`
    - reset the stop watch
  - `pause()`
    - pause the stop watch
  - `resume()`
    - resume the stopwatch
- Use this class to resume just before computation `i**2` (or `i*i`) and pause just after that.
- At the end use `elapsedTime()`
- Compare the two times.

Bit Arithmetic

- Ex 03:
  - Define a class for conducting bitwise operations. The class should support the following.
    - Initialize number of bits (limited to 8, 16, 32 and 64)
    - `Reset()`: Reset all bits to zero.
    - `SetBit(n)`: Set the `n`th bit to `1`.
    - `ChkBit(n)`: Check if `n`th bit to `1`. Returns `True` or `False`
    - `Not()`: implement 1's complement on bits
    - `And(o)`: Perform bitwise AND operation with bits of object `o` and return the result.
    - `Or(o)`: Perform bitwise OR operation with bits of object `o` and return the result.

Complex Numer Arithmetic

- Ex 04:
  - Define a class for complex number arithmetic and perform following operations
    - Add another complex number to it
    - Subtract another complex number from it
    - Multiply it by another complex numbers
    - Divide it by another complex numbers
    - Conjugate this complex number
    - Compare this complex number with other
    - Return absolute value

Rational Numbers

- Ex 05:
  - Define a class for representing fractions as rational number P/Q, Q!=0, and P and Q are relatively prime.
  - Define following operations
    - Add another rational number to it
    - Subtract another rational number from it
    - Multiply it by another rational numbers
    - Divide it by another rational numbers
    - Compare this rational number with other

Rational Number Equivalence

- Ex 06:
  - Using the rational number class as programmed in exercise 02, do the following,
  - Read a text file where each line contains two rational numbers with some mathematical operation
    - e.g. `5/6 + 3/4`
  - Read line and create a rational number for it.
  - Find all the rational numbers which occur more than once e.g.
    - `5/6 + 3/4` and `2/3 + 11/12` represent same rational number

Questions