Outline of the Talk

1. Part 1: Preliminaries of Python
2. Part 2: Scientific Libraries
3. Part 3: Object Oriented Programming

Programming in Practice: Simulations

- We typically use simulations to understand the performance/behavior of a system
- Examples:
  1. manufacturing process simulation
  2. stock market simulation
  3. statistical processes simulation

Programming in Practice: Simulations

- We typically use simulations to understand the performance/behavior of a system
- Examples:
  1. manufacturing process simulation
  2. stock market simulation
  3. statistical processes simulation
- Typical steps of simulation:
  - Create a computational environment that mimics the real world
  - Generate (synthetic) or load data from sources
  - Test hypotheses

Programming in Practice: Simulations

- We typically use simulations to understand the performance/behavior of a system
- Examples:
  1. manufacturing process simulation
  2. stock market simulation
  3. statistical processes simulation
- Typical steps of simulation:
  - Create a computational environment that mimics the real world
  - Generate (synthetic) or load data from sources
  - Test hypotheses
- Examples:
  1. simulating a 2D random walk
  2. simulation of a discrete random variable

`object` and `class`

Why do we need an object-oriented code?

- Purpose: reuse a code that we need quite often

`object` and `class`

Why do we need an object-oriented code?

- Purpose: reuse a code that we need quite often
- data and function/method bound together, e.g., a `list`
  - list instance `l`, method `l.append()` or `l.sort()`
  - when the `object` is passed, the methods are passed too

`object` and `class`

Why do we need an object-oriented code?

- Purpose: reuse a code that we need quite often
- data and function/method bound together, e.g., a `list`
  - list instance `l`, method `l.append()` or `l.sort()`
  - when the `object` is passed, the methods are passed too
  - `object` is an **instance** of a `class`

`object` and `class`

Why do we need an object-oriented code?

- Purpose: reuse a code that we need quite often
- data and function/method bound together, e.g., a `list`
  - list instance `l`, method `l.append()` or `l.sort()`
  - when the `object` is passed, the methods are passed too
  - `object` is an **instance** of a `class`
  - example: class `rectangle`, create an object of class rectangle, has parameters/data length and height, function of that data is area, longer side, shorter side, length of diagonal, area of the largest ellipse inscribed inside it

`object` and `class`

Why do we need an object-oriented code?

- Purpose: reuse a code that we need quite often
- data and function/method bound together, e.g., a `list`
  - list instance `l`, method `l.append()` or `l.sort()`
  - when the `object` is passed, the methods are passed too
  - `object` is an **instance** of a `class`
  - example: class `rectangle`, create an object of class rectangle, has parameters/data length and height, function of that data is area, longer side, shorter side, length of diagonal, area of the largest ellipse inscribed inside it
- abstract data types – defined by a `class`

`object` and `class`

Why do we need an object-oriented code?

- Purpose: reuse a code that we need quite often
- data and function/method bound together, e.g., a `list`
  - list instance `l`, method `l.append()` or `l.sort()`
  - when the `object` is passed, the methods are passed too
  - `object` is an **instance** of a `class`
  - example: class `rectangle`, create an object of class rectangle, has parameters/data length and height, function of that data is area, longer side, shorter side, length of diagonal, area of the largest ellipse inscribed inside it
- abstract data types – defined by a `class`
- `object`: collection of data and function, as defined in the `class`

`object` and `class`

Why do we need an object-oriented code?

- Purpose: reuse a code that we need quite often
- data and function/method bound together, e.g., a `list`
  - list instance `l`, method `l.append()` or `l.sort()`
  - when the `object` is passed, the methods are passed too
  - `object` is an **instance** of a `class`
  - example: class `rectangle`, create an object of class rectangle, has parameters/data length and height, function of that data is area, longer side, shorter side, length of diagonal, area of the largest ellipse inscribed inside it
- abstract data types – defined by a `class`
- `object`: collection of data and function, as defined in the `class`
- **Example:** simulation of LUDO game

Simulation using OOP

- discrete random variable with finite state space

Simulation using OOP

- discrete random variable with finite state space
- define the class of discrete random variables
- an instance of the class is a random variable, which has a distribution – probability masses

Simulation using OOP

- discrete random variable with finite state space
- define the class of discrete random variables
- an instance of the class is a random variable, which has a distribution – probability masses

how to draw a random variable of a given distribution

- inverse CDF method – true for discrete random variable too

Summary

This series of lectures discussed

- The basic syntaxes of python and its standard data types, loops and conditionals
- some implementations of numerical techniques

Summary

This series of lectures discussed

- The basic syntaxes of python and its standard data types, loops and conditionals
- some implementations of numerical techniques
- Four important set of scientific libraries

Summary

This series of lectures discussed

- The basic syntaxes of python and its standard data types, loops and conditionals
- some implementations of numerical techniques
- Four important set of scientific libraries
- Object-oriented programming in python

References

- **[online course materials]** Python for scientific computing, by Swaprava Nath: `http://scientificcomputing.is-great.net/`
- **[book]** Introduction to Computation and Programming using Python, by John Guttag, Prentice Hall of India.
- **[online course materials]** Lectures in Quantitative Economics, Thomas J. Sargent and John Stachurski: `https://lectures.quantecon.org/py/index.html`