#####################################################################################

### README OF OPTION PRICING MODULE

### AUTHOR: ALAN

### DATE: 2018/12/17

### CONTACTS: [SHAOLUN.DU@GMAIL.COM](mailto:SHAOLUN.DU@GMAIL.COM)

#####################################################################################

PART ZERO: HOW TO USE & MAINTAIN

#####################################################################################

1. LOCATION: “\*\*\*\Shared Projects\Portfolio\_Management\option\”
2. TEST CASES: “\*\*\*\Shared Projects\Portfolio\_Management\option\option\_test\”
3. HOT TO RUN: 1, For stan alone module test please run “test\_option” in TEST CASE lcation 2, For portfolio level testing please make corresponding inputs and run “test\_portfolio” in TEST CASE of portfolio module

\*\*\* Will be detailed here after implementation \*\*\*

PART ONE: GENERAL OUTLINE

#####################################################################################

This is option pricing part of portfolio management system, we will implement the below option pricing within this module: Stock pricing option pricing, Stock indices option pricing, Currency option pricing and Interests rate option pricing.

Each of these types of options also has a variety of styles they are: European option which pays only at maturity, American option which can be exercised any time before maturity, and Bermuda option which can be exercised on predetermined dates that works pretty like a combination of American and European options.

For each module we will generate the pricing value (NPV) also the corresponding Greek letters (Delta, Theta, Gamma, Vega, Rho) Also, we are incorporating the situation when volatility is changing with respect to time.

Generally speaking, to price an option we have two methods to follow: 1, Black-Scholes model with a given (market implied) volatility and 2, Monte-Carlo simulation and take the average value times discount factor. Within this module, we will implement both of them and do a side-by-side comparison as a validation test.

There are a few numerical methods involved: 1, Binomial trees, 2, Time-dependent parameters, 3, Monte-Carlo simulations, 4, Variance reduction procedures, 5, Finite difference methods,

PART TWO: MODULE OUTLINE

#####################################################################################

Module objects outline:

1. Monte-Carlo generator object, which should be flexible to take time dependent variable as input to generate a random path of a given scenario.
2. Option pricing object, which should take an option style as input (European, American or Bermuda option) and can generate NPV of a given input option and can choose different mode (B-S formula OR Monte-Carlo simulation) to calculate the task. Also, it has Greek-Generating method which output all Greek-letters at the same time. Option pricing object {Option style object, Option type object holds, Statistic collector object}
3. Option style object holds all option information given a option style name like: European, American or Bermuda option.
4. Option type object holds all option types information as like: Stock indices option, Currency option and Interests rate option.
5. Statistic collector object collects valuable information like the NPV, std. along the way of Monte-Carlo simulation. Having this object can significantly reduce the memory usage of MC simulation. Also, all time-consuming calculation should be done in this object.

Some of the numerical procedure mentioned above, like Finite difference methods will be implemented within MC generation object as a different mode to generate the random sample path which we will not list details information here please refer to corresponding “\*\*\*.py” files.

Since the inputs of an option is totally different from the other market derivatives we had before like swaps and fxs. We also have to modify the reader object which is in the module of IO named with reader.py to incorporate it. Potentially we will use reader.read\_opt() as the new method name for option pricing instruments.

The general outline for option pricing module is very similar to other derivative object outline. We will have: market\_instruments as the general inputs and pricing or cal\_NPV function as the pricing as well as Greek letters calculation. At the end of pricing we will return a dictionary with outline below:

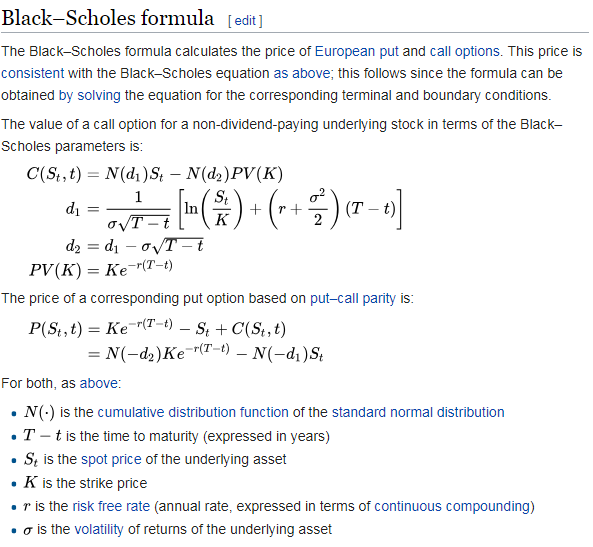
* Answer dictionary = {“Option position name {\*\*\*} “: { {“NPV”:\*\*\*}, {“Greek letters”:\*\*\*} }}

PART THREE: KEY FUNCTIONS OUTLINE

#####################################################################################

Some KEY function to be implemented:

* B-S formula: For different type of options it varies a little bit, but in general it looks like below:



* Binomial tree and MC simulation:

