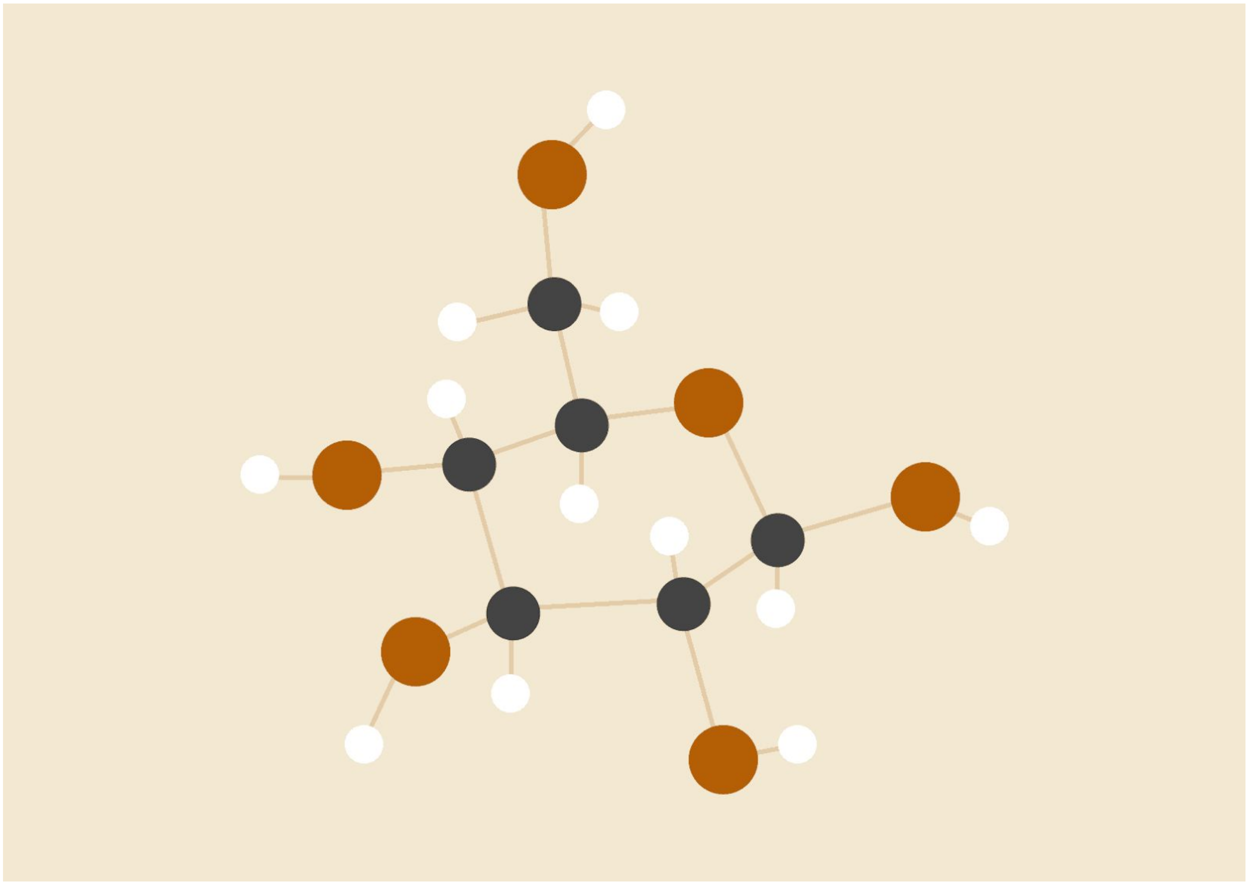
Final Project Report

Option Price Calculator



Yifu He (190003956)

Junjian Yao (197005736)

Junyi Yuan (197005740)

05.08.2020
Object Oriented Programming II

Table of Contents

INTRODUCTION

In this project, we create an option price calculator. The reason we choose to do this application is that it is very hard to find an effective option price calculator combining 4 models into a calculator and we hope to integrate them together. With this calculator, users can achieve the option price of European or American option in different models by providing the related inputs such as current stock price, strike price of option, time to maturity, risk free rate and volatility. In order to increase the accuracy of option price, we provide users several different models to choose, like Monte Carlo, Binary Tree, Ternary Tree and Black Scholes methods. Meanwhile, Greeks letters, which are a good reference of relationship among option price and other basic parameters like stock price,are also offered to users. For investors who want to calculate the option price, they can take advantage of 4 models and find the best model to calculate the option price.

IMPLEMENTATION PLAN

YIFU HE was responsible for the Monte Carlo model and the Tree Model. Junjian Yao was responsible for the Black Scholes model. Junyi Yuan was responsible for the graphic user interface, and implementation of models.

OPTION PRICE CALCULATOR

Options are important financial instruments that are derivatives based on the value of the underlying asset such as stock. Options contract provides the buyer the opportunity to buy or sell the underlying asset, which depends on the type of the options they have. Each option will define a specific expiration date by which the holder must exercise the option, while if it is a American option, the holder can exercise it before the expiration date.

In order to calculate the option price, usually we need some basic parameters. For example, we need stock price and strike price of option because the main value of option is from the difference between stock price and strike price, for a call option, the difference calculating by stock price minus the strike price accounts for a large part of option value. Besides, since the time to maturity decides how much time is left we can hold the option contract, it is also crucial in calculating the option price. And volatility of stock also can affect the fluctuation of option value.

Besides the option price, this calculator can also provide the greeks letters to users for reference. Greeks is a term that is used in the option market to describe the different dimensions of risk involved in taking an options position. Greeks composes of many variables, which include delta, theta, gamma, vega and rho. Each of these variables has a number that tells the trader something about how the option moves or the risk associated with that option.

This calculator provides 4 models for users to choose to calculate the option price. First is the Monte Carlo, which is good for path of option payoff is path dependent. This method usually first calculates the potential future price of the underlying asset, and then calculates the payoff of the option for each of the potential price paths. Last, it discounts the payoffs back to today and averages them to the expected price.

Second model is the Binary tree. With binary tree option price models, it assumes two possible outcomes, which are a move up or a move down. This model provides the calculation of the option for multiple periods along with the range.

The third model is the Ternary tree. The difference between this method and Binary tree is it incorporates another possible value in one time period. It considers a third possible value, which is a zero change in value over a time period.

The fourth model is the Black Scholes, which assumes the price of assets follows a geometric brownian motion with constant drift and volatility. It's used to calculate the theoretical value of options using current stock prices, expected dividends, the option's strike price, expected interest rates, time to expiration and expected volatility.

GRAPHICAL USER INTERFACE

In order to make our system more user friendly, we hope to use Tkinter as the GUI frame work.Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

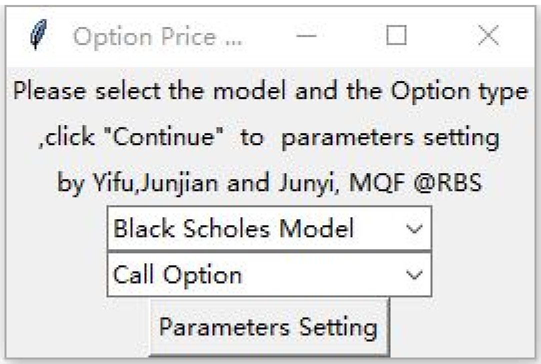
Moreover, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that's functional and cross-platform quickly.

Model Selection Page:

On the first page we have to two combobox. The first combobox allow the user to select the model they would like to use. we will have four different model for our option calculator. They are Black

Scholes, Monte Carlo, Binary tree, and Trigeminal tree. In the second combobox the user can select the Option type. There are usually two main types of options call options and Put options.

After the user selected the Model, they can click the Parameters Setting button, and they will go to the next page.


Please select the model and the Option type ,click "Continue" to parameters setting

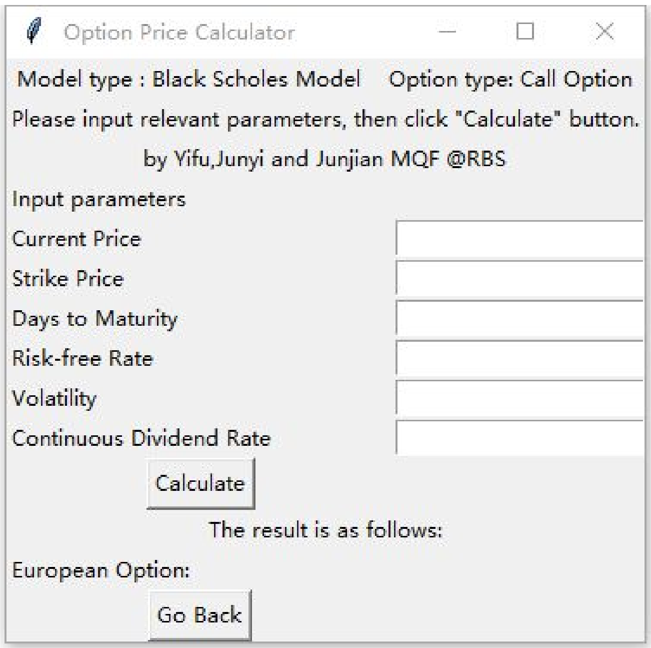
by Yifu,Junjian and Junyi, MQF @RBS

Black Scholes Model

Call Option

Parameters Setting Page:

If the user select the Black Schole model, the parameters Setting Page will ask the user to input current price, strike price, day to maturity, risk-free rate, volatility, and continuous dividend rate to the system.


Model type : Black Scholes Model   Option type: Call Option

Please input relevant parameters, then click "Calculate" button.

by Yifu,Junyi and Junjian MQF @RBS

Input parameters

Current Price

Strike Price

Days to Maturity

Risk-free Rate

Volatility

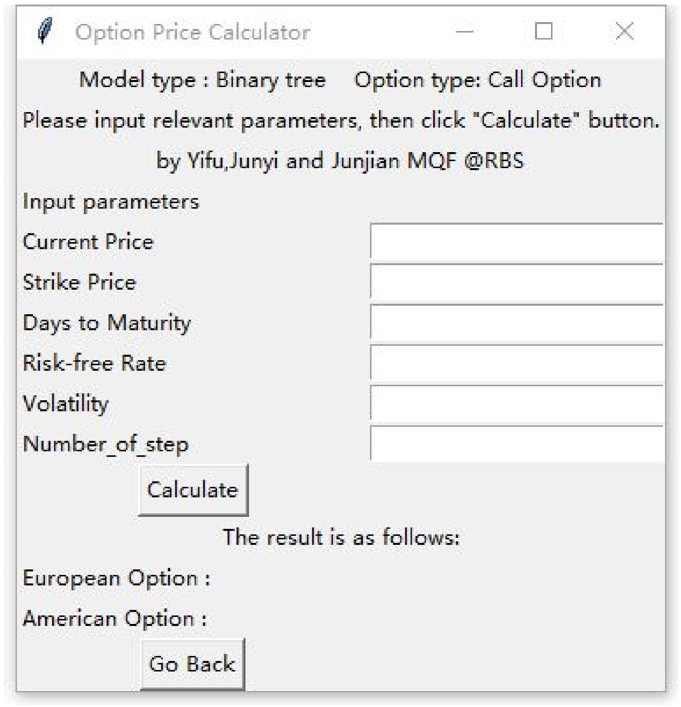
Continuous Dividend Rate

The result is as follows:

European Option:

After the user input parameters and click the Calculate button, the system will output the Option price.

From the figure above we can see that the model we selected is Black Scholes Model and the Option type is call option.



Option Price Calculator

Model type : Binary tree   Option type: Call Option

Please input relevant parameters, then click "Calculate" button.

by Yifu,Junyi and Junjian MQF @RBS

Input parameters

Current Price

Strike Price

Days to Maturity

Risk-free Rate

Volatility

Number_of_step

Calculate

The result is as follows:

European Option :

American Option :

Go Back

If the user select the Binary Tree model , the user will be asked to input one more parameter: number of step. Moreover, the system will output both the price for European Option and the price for American Option.

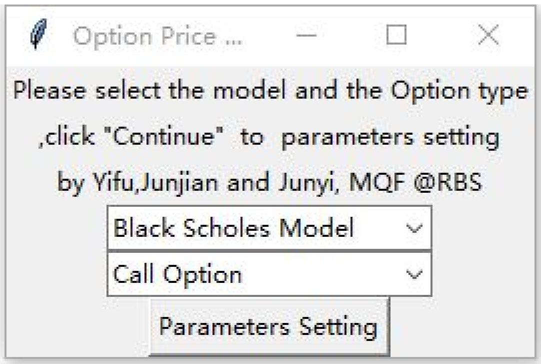
Performance analysis

The object for us is to make the option price calculation easy to use. It will not ask users to input the formula, the only thing users need to do is fill in parameters. Then the calculator will show the information of the model and the price directly on the next page.

To start the option calculator, the user need to run the GUI.py file first. The window will pop up.

Suppose that we need to calculate a European option using the Black-Scholes Model and parameters we need to use are:

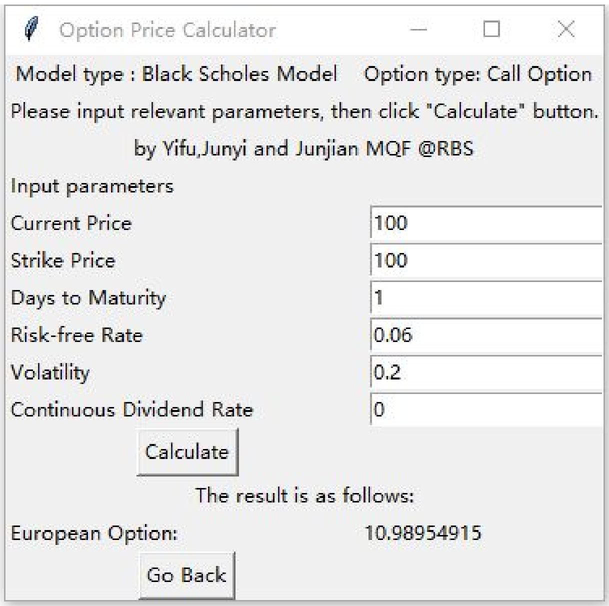
Option Type	Call
Current Price	100
Strike Price	100
Days to maturity	1
Risk-Free rate	0.06
Votatility	0.2
Continuous Dividend Rate	0



Please select the model and the Option type
,click "Continue" to parameters setting
by Yifu,Junjian and Junyi, MQF @RBS

Black Scholes Model
Call Option

We need to select the Black Scholes Model and Set the option type as a call option at first. After we click Parameters Setting button next page will pull up.By using the calculator we get to know that the Option price will be 10.99


Option Price Calculator

Model type : Black Scholes Model Option type: Call Option

Please input relevant parameters, then click "Calculate" button.

by Yifu,Junyi and Junjian MQF @RBS

Input parameters

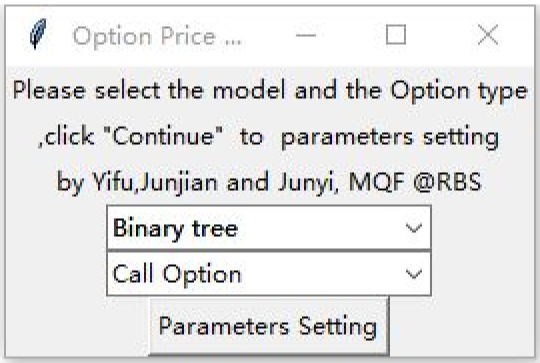
Current Price	100
Strike Price	100
Days to Maturity	1
Risk-free Rate	0.06
Volatility	0.2
Continuous Dividend Rate	0

The result is as follows:

European Option: 10.98954915

Then we test the system by using a Binary Tree model. Parameters we need to input are:

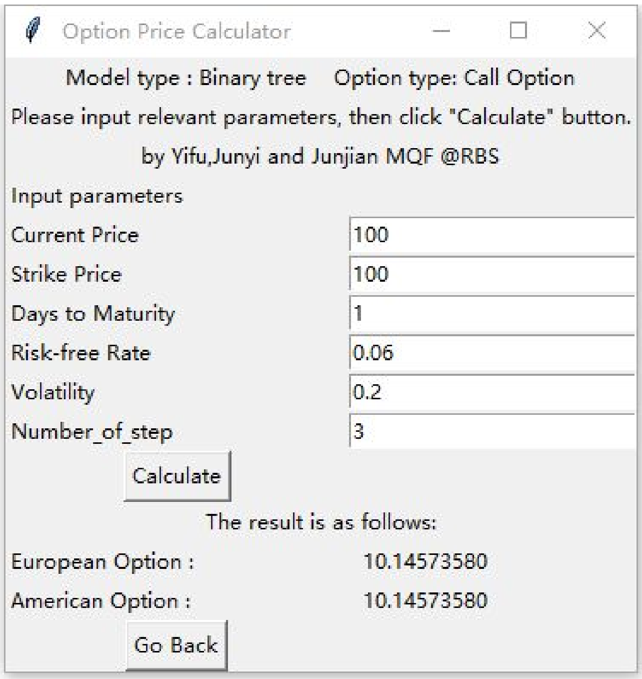
Option Type	Call
Current Price	100
Strike Price	100
Days to Maturity	1
Risk-Free Rate	0.06
Votatility	0.2
Continuous Dividend Rate	0
Number of Steps	3


Option Price ...

Please select the model and the Option type
,click "Continue" to parameters setting
by Yifu,Junjian and Junyi, MQF @RBS

Binary tree

Call Option


Option Price Calculator

Model type : Binary tree Option type: Call Option
Please input relevant parameters, then click "Calculate" button.
by Yifu,Junyi and Junjian MQF @RBS

Input parameters

Current Price	100
Strike Price	100
Days to Maturity	1
Risk-free Rate	0.06
Volatility	0.2
Number_of_step	3

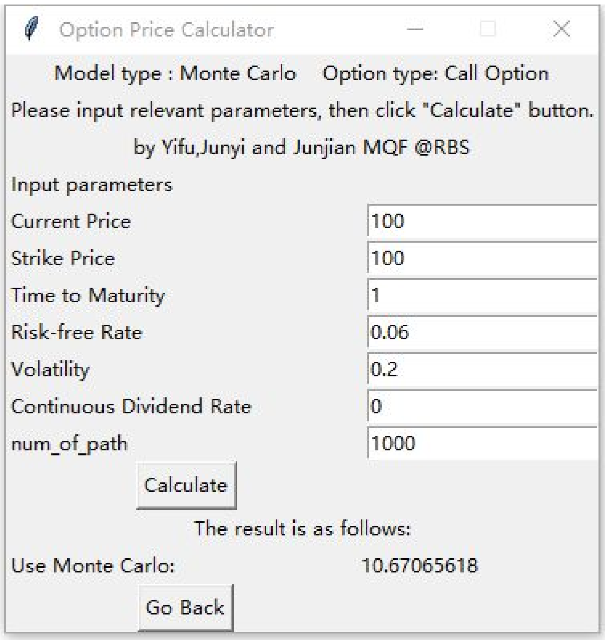
The result is as follows:

European Option :	10.14573580
American Option :	10.14573580

At first the Binary tree model was selected, then on the next page we input parameters for the calculation.

Then system will tell us that the price for both European Option and American Option are $10.15

Finally we test the Monte Carlo Model. The main difference between this model and other model is that it will simulate the payoff. We need to set the how many times we need to do the simulation. So it’s time complexities is O(n).


Option Price Calculator

Model type : Monte Carlo   Option type: Call Option

Please input relevant parameters, then click "Calculate" button.

by Yifu,Junyi and Junjian MQF @RBS

Input parameters

Current Price	100
Strike Price	100
Time to Maturity	1
Risk-free Rate	0.06
Volatility	0.2
Continuous Dividend Rate	0
num_of_path	1000

The result is as follows:

Use Monte Carlo: 10.67065618

In this test, we will simulate 1000 times, and finally we get to know that the price for the call option is likely to be $10.67.

Possible future improvement

1.We only encapsulated the payoff class. And pass it as a parameter into the Monte Carlo Simulation class. But in the real-world, pay off is not a concrete concept or object ,we should use the Option object as an alternative choice. In the option object, we need to include the way we calculate payoff and get a method function to provide time to maturity.

2.More model. I actually want to include more models to compute option prices such as Finite Difference Method and Crank Nicolson Method. I have already done that in C++, just have no time to finish it in python.

3.Mode Option Type. Similar to 2, I want to add more complicated option type such as Power Option and Path Dependent Option, including Barrier Option and Asian Option.

4.User Defined Payoff. I also defined a UI to deal with input string and convert it into a User Defined form of Payoff in C++. By doing it, we can name the format of payoff and calculate it.

Conclusions and remarks

In this project, we learned how to get started with Python GUI programming. Tkinter is a compelling choice for a Python GUI framework because it’s built into the Python standard library, and it’s relatively painless to make applications with this framework. Moreover, we combined things we learned from the Object Oriented Programming Class and Derivatives Class together. This is a very good exercise, which prepares us for our future career.

APPENDIX

The source code for GUI (Users need to run GUI.py to start the calculator):

```
import tkinter
from tkinter import *
from tkinter import ttk
from core.Tree_Model_Base import tree_model
from core.Black_Schcoles_Models_concrete import *
from core.MonteCarlo import *
class StartPage(object):
    def __init__(self):
        self.root=Tk()
        self.root.wm_title(' Option Price Calculator')
        Label(self.root, text='Please select the model and the Option type').pack()
        Label(self.root, text=',click "Continue"  to  parameters setting').pack()
        Label(self.root, text='by Yifu,Junjian and Junyi, MQF @RBS').pack()
        comvalue = StringVar()  # initialize the value
        self.comboxlist = ttk.Combobox(self.root, textvariable=comvalue)  # initialize
        self.comboxlist["values"] = ("Black Scholes Model", "Monte Carlo", "Binomial tree", "Trinomial tree")
        self.comboxlist.current(0)  # select the first one to show
        self.comboxlist.pack()
        comvalue2 = StringVar()
        self.comboxlist2 = ttk.Combobox(self.root, textvariable=comvalue2)  # initialize
        self.comboxlist2["values"] = ("Call Option","Put Option")
        self.comboxlist2.current(0)  # select the first one to show
        self.comboxlist2.pack()
        Button(self.root, text="Parameters Setting", command=self.go_calculator).pack()
        self.root.resizable(0, 0)
        self.root.mainloop()
    def go_calculator(self):
        modelType=self.comboxlist.get()
        optionType=self.comboxlist2.get()
        print(modelType,optionType)
        self.root.destroy()
        CalculatorPage(modelType,optionType)

class CalculatorPage(object):
    def __init__(self,modelType,optionType):
        self.modelType=modelType
        self.optionType=optionType
        self.vlist=[]
        self.root = Tk() # Create an object
        self.root.wm_title(' Option Price Calculator')
        Label(self.root, text='Model type : '+self.modelType+'    Option type: 
'+self.optionType).grid(row=0, column=0, columnspan=3)
        Label(self.root, text='Please input relevant parameters, '
                    'then click "Calculate" button.').grid(row=1, column=0, columnspan=3)
        Label(self.root, text='by Yifu,Junyi and Junjian MQF @RBS').grid(columnspan=3)
        self.cp = self.optionType
        Label(self.root, text='Input parameters').grid(row=4, column=0, sticky=W)
        if self.modelType=="Black Scholes Model":
            self.plist = ['Current Price', 'Strike Price', 'Time to Maturity',
                    'Risk-free Rate', 'Volatility', 'Continuous Dividend Rate']
            self.elist = [] # for parameters
            r = 5 # r start from 0 and now is 3
            for param in self.plist:
                Label(self.root, text=param).grid(column=0, sticky=W)
                e = Entry(self.root)
                e.grid(row=r, column=1, columnspan=2, sticky=W+E)
                self.elist.append(e)
                r += 1
            Button(self.root, text='Calculate',command = self.get_parameter_BS).grid(row=r)
            r += 1
            self.answ = Label(self.root, text='The result is as follows:')
            self.answ.grid(row=r, columnspan=3)
            r += 1
            self.BS = Label(self.root)

            self.BS.grid(row=r, columnspan=2, sticky=E)

            Label(self.root, text='European Option: ').grid(row=r, sticky=W)

            Button(self.root, text='Go Back', command=self.go_startPage).grid(row=r + 1)
            self.root.resizable(0, 0)

        elif self.modelType=="Monte Carlo":
            self.plist = ['Current Price', 'Strike Price', 'Time to Maturity',
                    'Risk-free Rate', 'Volatility', 'Continuous Dividend Rate','num_of_path']
            self.elist = [] # for parameters
            r = 5 # r start from 0 and now is 3
            for param in self.plist:
                Label(self.root, text=param).grid(column=0, sticky=W)
                e = Entry(self.root)
                e.grid(row=r, column=1, columnspan=2, sticky=W+E)
                self.elist.append(e)
                r += 1
            Button(self.root, text='Calculate',command = self.get_parameter_MC).grid(row=r)
            r += 1
            self.answ = Label(self.root, text='The result is as follows:')
            self.answ.grid(row=r, columnspan=3)
            r += 1
            self.MC = Label(self.root)
            self.MC.grid(row=r, columnspan=2, sticky=E)
            Label(self.root, text='Use Monte Carlo: ').grid(row=r, sticky=W)
            Button(self.root, text='Go Back', command=self.go_startPage).grid(row=r+1)
            self.root.resizable(0, 0)
        elif self.modelType=="Binomial tree":
            self.plist = ['Current Price', 'Strike Price', 'Time to Maturity',
                    'Risk-free Rate', 'Volatility','Number_of_step']
            self.elist = [] # for parameters
            r = 5 # r start from 0 and now is 3
            for param in self.plist:
                Label(self.root, text=param).grid(column=0, sticky=W)
                e = Entry(self.root)
                e.grid(row=r, column=1, columnspan=2, sticky=W+E)
                self.elist.append(e)
                r += 1
            Button(self.root, text='Calculate',command = self.get_parameter_BT).grid(row=r)
            r += 1
            self.answ = Label(self.root, text='The result is as follows:')
            self.answ.grid(row=r, columnspan=3)
            r += 1
            self.EU = Label(self.root)
            self.AM = Label(self.root)
            self.EU.grid(row=r, columnspan=2, sticky=E)
            self.AM.grid(row=r+1, columnspan=2, sticky=E)
            Label(self.root, text='European Option : ').grid(row=r, sticky=W)
            Label(self.root, text='American Option : ').grid(row=r+1, sticky=W)
            Button(self.root, text='Go Back', command=self.go_startPage).grid(row=r+2)
            self.root.resizable(0, 0)
        elif self.modelType=="Trinomial tree":
            self.plist = ['Current Price', 'Strike Price', 'Time to Maturity',
                    'Risk-free Rate', 'Volatility','Number_of_step']
            self.elist = [] # for parameters
            r = 5 # r start from 0 and now is 3
            for param in self.plist:
                Label(self.root, text=param).grid(column=0, sticky=W)
                e = Entry(self.root)
                e.grid(row=r, column=1, columnspan=2, sticky=W+E)
                self.elist.append(e)
                r += 1
            Button(self.root, text='Calculate',command = self.get_parameter_TT).grid(row=r)
            r += 1
            self.answ = Label(self.root, text='The result is as follows:')
            self.answ.grid(row=r, columnspan=3)
            r += 1
            self.EU = Label(self.root)
            self.AM = Label(self.root)
            #self.mc = Label(self.root)
            self.EU.grid(row=r, columnspan=2, sticky=E)
            self.AM.grid(row=r+1, columnspan=2, sticky=E)
            Label(self.root, text='European Option : ').grid(row=r, sticky=W)
            Label(self.root, text='American Option : ').grid(row=r+1, sticky=W)
            Button(self.root, text='Go Back', command=self.go_startPage).grid(row=r+2)
            self.root.resizable(0, 0)

        def get_parameter_BS(self):
        vlist = []
        for e in self.elist:
            try:
                p = float(e.get())
                vlist.append(p)
            except:
                answ.config(text='Invalid Input(s). Please input correct parameter(s)', fg='red')
                e.delete(0, len(e.get()))
                return 0

        self.answ.config(text='The result is as follows:', fg='black')
        if self.optionType=="Call Option":
            option=European_Call_BS(vlist[0], vlist[1], vlist[2], vlist[3], vlist[4])
            self.BS.config(text=str("%.8f" % (option.get_Option_Price())))
        else:
            option = European_Put_BS(vlist[0], vlist[1], vlist[2], vlist[3], vlist[4])
            self.BS.config(text=str("%.8f" % (option.get_Option_Price())))
    def get_parameter_MC(self):
        vlist = []
        for e in self.elist:
            try:
                p = float(e.get())
                vlist.append(p)
            except:
                answ.config(text='Invalid Input(s). Please input correct parameter(s)', fg='red')
                e.delete(0, len(e.get()))
                return 0
        optionPrice = vlist[0] + vlist[1] + vlist[2] + vlist[3] + vlist[4]
        self.answ.config(text='The result is as follows:', fg='black')
        if self.optionType=="Call Option":
            European_call = Pay_Off_Vanilla.European_Pay_Off("call", int(vlist[1]))
            optionPrice = MonteCarlo(European_call, vlist[0], vlist[2], vlist[3],
vlist[4]).get_MonteCarlo_Price(int(vlist[6]))
            self.MC.config(text=str("%.8f" % (optionPrice)))

        else:
            European_put = Pay_Off_Vanilla.European_Pay_Off("put", int(vlist[1]))
            optionPrice = MonteCarlo(European_put, vlist[1], vlist[2], vlist[3],
vlist[4]).get_MonteCarlo_Price(int(vlist[6]))
            self.MC.config(text=str("%.8f" % (optionPrice)))
    def get_parameter_BT(self):
        vlist = []
        for e in self.elist:
            try:
                p = float(e.get())
                vlist.append(p)
            except:
                answ.config(text='Invalid Input(s). Please input correct parameter(s)', fg='red')
                e.delete(0, len(e.get()))
                return 0
        option=tree_model(vlist[0],vlist[1],vlist[2],vlist[3],vlist[4])
        self.answ.config(text='The result is as follows:', fg='black')
        if self.optionType=="Call Option":
            self.EU.config(text=str("%.8f" % (option.European_Binomial_Multiplicative('call',
int(vlist[5]), 1.1, 1 / 1.1))))
            self.AM.config(text=str("%.8f" % (option.American_Binomial_Multiplicative('call',
int(vlist[5]), 1.1, 1 / 1.1))))
        else:
            self.EU.config(text=str("%.8f" % (option.European_Binomial_Multiplicative('put', int(vlist[5]), 1.1, 1 / 1.1))))
            self.AM.config(text=str("%.8f" % (option.American_Binomial_Multiplicative('put', int(vlist[5]), 1.1, 1 / 1.1))))
    def get_parameter_TT(self):
        vlist = []
        for e in self.elist:
            try:
                p = float(e.get())
                vlist.append(p)
            except:
                answ.config(text='Invalid Input(s). Please input correct parameter(s)', fg='red')
                e.delete(0, len(e.get()))
                return 0
        option=tree_model(vlist[0],vlist[1],vlist[2],vlist[3],vlist[4])
        self.answ.config(text='The result is as follows:', fg='black')
        if self.optionType=="Call Option":
            self.EU.config(text=str("%.8f" % (option.European_Trinomial('call', int(vlist[5]), 0.2))))
            self.AM.config(text=str("%.8f" % (option.American_Trinomial('call', int(vlist[5]), 0.2))))
        else:
            self.EU.config(text=str("%.8f" % (option.European_Trinomial('put', int(vlist[5]), 0.2))))
            self.AM.config(text=str("%.8f" % (option.American_Trinomial('put', int(vlist[5]), 0.2))))
    def go_startPage(self):
        self.root.destroy()
        StartPage()


if __name__ == '__main__':
    StartPage()
```

Reference

[1] https://realpython.com/python-gui-tkinter/

[2] https://www.tutorialspoint.com/python/python_gui_programming.htm

[3] https://www.investopedia.com/terms/b/blackscholes.asp

[4] https://www.investopedia.com/terms/o/option.asp

[5] https://www.investopedia.com/terms/g/greeks.asp