Solutions - Practical Lesson 2

Matteo Sani
matteosan1@gmail.com

October 8, 2019

1 Solutions

1.1 Exercises

1.1.1 Exercise 2.2

Write code which, given the following list

```
input_list = [3, 5, 2, 1, 13, 5, 5, 1, 3, 4]
```

prints out the indices of every occurrence of

```
y = 5
```

Solution:

```
In [1]: input_list = [3, 5, 2, 1, 13, 5, 5, 1, 3, 4]
        y = 5
        for i, value in enumerate(input_list):
            if value == 5:
                print (i)
```

```
1
5
6
```

1.1.2 Exercise 2.3

Given the following variables

```
S_t = 800.0 # spot price of the underlying
K = 600.0 # strike price
vol = 0.25 # volatility
r = 0.01 # interest rate
ttm = 0.5 # time to maturity, in years
```

write out the Black Scholes formula and save the value of a call in a variable named 'call_price' and the value of a put in a variable named 'put_price'

Solution: The BS equation for the price of a call is:

$$C(S,t) = S_t N(d_1) - Ke^{-r(T-t)} N(d_2)$$

where

- S_t is the spot price of the underlying
- K is the strike price
- r is the risk-free interest rate (expressed in terms of continous compounding)
- $N(\cdot)$ is the cumulative distribution function of the standard normal distribution
- $T - t$ is the time to maturity
- σ is the volatility of the underlying

$$d_1 = \frac{\ln(\frac{S_t}{K}) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}$$

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

Remember that there are many modules available in python that let you save a lot of time. In this case we need the cumulative distribution function of the standard normal distribution which can be found in `scipy.stats` module.

```
In [8]: from math import log, exp, sqrt
        # You'll need the Gaussian cumulative distribution function
        from scipy.stats import norm

        S_t = 800.0
        ttm = 0.5
        K = 600.0
        vol = 0.25
        r = 0.01

        d1_num = (log(S_t/K)+(r+0.5*pow(vol, 2))*ttm)
        d1_den = vol*sqrt(ttm)
        d1 = d1_num /d1_den
        d2 = d1 - d1_den

        call_price = S_t * norm.cdf(d1) - K * exp(-r*ttm)*norm.cdf(d2)
        put_price = - S_t * norm.cdf(-d1) + K * exp(-r*ttm)*norm.cdf(-d2)

        print ("{:.3f} {:.3f}".format(call_price, put_price))
```

```
205.472 2.480
```

1.1.3 Exercise 2.4

Given the following dictionary mapping currencies to 2-year zero coupon bond prices, build another dictionary mapping the same currencies to the corresponding annualized interest rates.

```
discount_factors = {
'EUR': 0.98,
'CHF': 1.005,
'USD': 0.985,
'GBP': 0.97
}
```

Solution: The price of a n-years zero coupon bond is:

$$P = \frac{M}{(1+r)^n} = M \cdot D$$

where * M is the value of the bond at the maturity * r is the risk-free rate * n is the number of years untill maturity

Hence:

$$D = \frac{1}{(1+r)^n} \Longrightarrow r = \left(\frac{1}{D}\right)^{\frac{1}{n}} - 1$$

```
In [5]: from math import exp

        # initialize an empty dictionary in which to store result
        rates = {}

        maturity = 2
        discount_factors = {
            'EUR': 0.98,
            'CHF': 1.005,
            'USD': 0.985,
            'GBP': 0.97
        }

        # loop over the input dictionary to get the currencies
        for currency, df in discount_factors.items():
            # calculate the rate and store it in the output dictionary
            rates[currency] = pow(1/df, 1/maturity) - 1

        for r in rates.items():
            print (r)

('EUR', 0.010152544552210818)
('CHF', -0.002490663892367073)
('USD', 0.007585443719756668)
('GBP', 0.015346165133619083)
```

1.1.4 Exercise 2.5

Build again dates as in Exercise 2.1 (i.e. the weekday of your birthdays for the next 120 years) and count how many of your birthdays is a Monday, Tuesday, . . . , Sunday until 120 years of age. Print out the result using a dictionary.

Solution:

```
In [7]: import datetime
        from dateutil.relativedelta import relativedelta

        name_of_day = {0:"Mon", 1:"Tue", 2:"Wed", 3:"Thu", 4:"Fri", 5:"Sat", 6:"Sun"}
        birthday_weekdays = {}
        birthday = datetime.date(1974, 10, 20)

        for i in range(121):
          next_birthday = birthday + relativedelta(years=i)
          wd = next_birthday.weekday()
          if wd not in birthday_weekdays.keys():
            birthday_weekdays[wd] = 1
          else:
            birthday_weekdays[wd] += 1

        print (birthday_weekdays)

        # if you want to be more precise you can map integers
        # from 0 to 6 to the day name
        print ()
        for k in sorted(birthday_weekdays.keys()):
          print (name_of_day[k], birthday_weekdays[k])
```

```
{6: 17, 0: 18, 2: 18, 3: 17, 4: 17, 5: 17, 1: 17}

Mon 18
Tue 17
Wed 18
Thu 17
Fri 17
Sat 17
Sun 17
```