Python for Finance

Exercises

Matteo Sani

Quants Staff - MPS Capital Services
matteo.sani@mpscapitalservices.it

Contents

Chapter 1

Introduction to python

Exercise 1.1

What is the built-in function that python uses to iterate over a number sequence ? Write an example that uses it.

Exercise 1.2

What is a string in python ? Declare one string variable and try to manipulate it (concatenate, make uppercase, capitalize, replace characters, split...).

Exercise 1.3

What does the continue do in python ? Show an example of its usage printing all the odd numbers between 0 and 10.

Exercise 1.4

When should you use the break in python ? Show an example of its usage.

Exercise 1.5

Which python function will you use to convert a number to a string ? Show an example.

Exercise 1.6

Import the math module and compute the logarithm of 2.09, the exponential of 1.57 and the area of a circle of radius 6 cm (circle area = $\pi \cdot r^2$).

Exercise 1.7

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'.
Hint: 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, the name of the function is `norm`.

Chapter 2

Data Containers

Exercise 2.1

What is a dictionary in python programming ? Create a dictionary, modify it and then print all its items.

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
```

Exercise 2.3

Write a python program to convert a list of tuples into a dictionary where the keys are the first elements of each tuples and the values the second. Input:

```
l = [("x", 1), ("x", 2), ("x", 3), ("y", 1), ("y", 2), ("z", 1)]
```

Exercise 2.4

Write a python program to replace the last value of each tuples in a list. Input:

```
l = [(10, 20, 40), (40, 50, 60), (70, 80, 90)]
```

Exercise 2.5

Write a python program to count the elements in a list until an element is a tuple. Input:

```
{[1, 5, 'a', (1,2), {'test':1}]}
```

Exercise 2.6

Write a python script to concatenate following dictionaries to create a new single one. Input:

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50, 6:60}
```

Exercise 2.7

Write a `python` script to check whether a given key already exists in a dictionary.

Exercise 2.8

Write a `python` program to combine two dictionary adding values for common keys. Input:

```
d1 = {'a': 100, 'b': 200, 'c':300}
d2 = {'a': 300, 'b': 200, 'd':400}
```

Exercise 2.9

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.

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

Chapter 3

Date and Time

Exercise 3.1

Write code that:

- print the day of the week of your birthday
- print the weekday of your birthdays for the next 120 years

Exercise 3.2

Write code to determine whether a given year is a leap year and test it with 1800, 1987 and 2020.
Hint: a leap year is divisible by 4, by 100 and by 400.

Exercise 3.3

Write code to print next five days starting from today.

Exercise 3.4

Build again dates as in Exercise 3.1 (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. (expected output something like: `{6: 10, 0: 10, 2: 9, 3: 10, 4: 10, 5: 10, 1: 9}`)

Exercise 3.5 (Date Generator)

In the next lessons we will create many contracts (e.g. swaps) which take in input lists of dates like for example the payment dates. Since it would be very boring to write long list of dates for each of these contracts, the goal of this exercise is to write code which given a start date and a number of months, returns a list of dates of **annual** frequency from the start date to the ending of the period after the specified number of months.
For example

- 2019-11-10 start date 12 months → 2019-11-10, 2020-11-10
- 2019-11-10 start date 24 months → 2019-11-10, 2020-11-10, 2021-11-10

Note that if the number of months is not a multiple of 12, the last period should simply be shorter than 12 months. For example:

- 2019-11-10 start date 9 months → 2019-11-10, 2020-08-10
- 2019-11-10 start date 15 months → 2019-11-10, 2020-11-10, 2021-02-10

Once you have done save this code in a file called `finmarkets.py`, this will become our financial library and will be extended and used later on.

Chapter 4

Function and Classes

Exercise 4.1

Take the code for the Black-Scholes formula from Exercise 1.7 and wrap it in a function. Then, use this function to calculate the prices of calls with various strikes, using the following data.

```
s = 800
# strikes expressed as % of spot price
moneyness = [ 0.5, 0.75, 0.825, 1.0, 1.125, 1.25, 1.5 ]
vol = 0.3
ttm = 0.75
r = 0.005
```

The output should be a dictionary mapping strikes to call prices.

Exercise 4.2

Write two classes, `Circle` and `Rectangle` that given the radius and height, width respectively allow to compute area and perimeter of the two shapes. Test them with the following:

```
a_circle = Circle(5)
print ("My circle has an area of {} m**2".format(a_circle.area()))

a_rectangle = Rectangle(3, 6)
print ("My rectangle has a perimeter of {} m and an area of {} m**2" \
    .format(a_rectangle.perimeter(), a_rectangle.area()))
```

Exercise 4.3

Define a class `Songs`, its `__init__` should take as input a dictionary (`lyrics` that contains lyrics line by line). Define a method, `sing_me_a_song` that prints each element of the lyrics in his own line. Also test it with the follwing input.

```
lyrics = {"Wonderwall":["Today is gonna be the day",
                        "That they're gonna throw it back to you",
                        "By now you should've somehow", "..."],
          "Wish you were here":  ["So, so you think you can tell",
                                  "Heaven from hell",
                                   "Blue skies from pain", "..."]}
```

Exercise 4.4

Define a Point2D class that represent a point in a plane. Its `__init__` method should accept the point coordinates `x` and `y`. Write a method `distanceTo` that compute the distance of the point to another passed as input. Test the class by printing the distance of the point $P = (4, 5)$ to the origin $P = (0, 0)$ and to $P = (3, 4)$.
Hint: in the Cartesian plane the distance between two points is: $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Exercise 4.5

Write a class `Student` which inherits from `Person` defined during Lesson 6. This new class should have two new attributes: `grade` which keeps the type of school and `votes` a dictionary which will record the student's votes and the corresponding course. Then add two methods, one to add votes and another to compute the average vote. Instantiate a "student" add some votes and show how good it has been.
Hint: this is the `Person` class already developed.

```
class Person:
    def __init__(self, name, birthday):
        self.name = name
        self.birthday = birthday
        self.employment = None

    def age(self, d=date.today()):
        age = (d - self.birthday).days/365
        print ("{} is {:.0f} years old".format(self.name, age))

    def mainOccupation(self, occupation):
        self.employment = occupation
        print ("{}'s main occupation is: {}".format(self.name, self.employment))
```

Chapter 5

Data Manipulation and Its Representation

Exercise 5.1

Using `pandas` import data stored in `stock_market.xlsx` (click on the name to see and download it). With the resulting dataframe determine:

1. remove duplicates and missing data (how many rows are left ?)
2. stocks with positive variation;
3. the first five stocks with the lowest price.

Exercise 5.2

Given the following discount factors plot the resulting discount curve, possibly adding axis labels and legend.

```
dfs = [1.0, 1.0014907894567657, 1.0031038833235129, 1.0047764800189012,
       1.0065986105304596, 1.014496095021891, 1.022687560553011,
       1.0303585751965112, 1.0369440287181253, 1.0422287558021188,
       1.0461834022163963, 1.0489228953047331, 1.0505725627906783,
       1.0513323539753632, 1.0513777790851995, 1.0508768750534248,
       1.049935905228433, 1.0486741093761602, 1.047175413484517,
       1.0455115431993336, 1.0437147446170034, 1.0418294960952215,
       1.0398823957504923, 1.0378979499878478, 1.0358789099539805,
       1.0338409767365169, 1.031791178324756, 1.0297378455884902,
       1.0276772747965244, 1.0256154380560942, 1.0235543974485939,
       1.0214974135391857, 1.0194401540150835, 1.0173862951028778]

pillars = [datetime.date(2020, 8, 3), datetime.date(2020, 11, 3),
           datetime.date(2021, 2, 3), datetime.date(2021, 5, 3),
           datetime.date(2021, 8, 3), datetime.date(2022, 8, 3),
           datetime.date(2023, 8, 3), datetime.date(2024, 8, 3),
           datetime.date(2025, 8, 3), datetime.date(2026, 8, 3),
           datetime.date(2027, 8, 3), datetime.date(2028, 8, 3),
           datetime.date(2029, 8, 3), datetime.date(2030, 8, 3),
           datetime.date(2031, 8, 3), datetime.date(2032, 8, 3),
           datetime.date(2033, 8, 3), datetime.date(2034, 8, 3),
           datetime.date(2035, 8, 3), datetime.date(2036, 8, 3),
           datetime.date(2037, 8, 3), datetime.date(2038, 8, 3),
           datetime.date(2039, 8, 3), datetime.date(2040, 8, 3),
           datetime.date(2041, 8, 3), datetime.date(2042, 8, 3),
           datetime.date(2043, 8, 3), datetime.date(2044, 8, 3),
           datetime.date(2045, 8, 3), datetime.date(2046, 8, 3),
           datetime.date(2047, 8, 3), datetime.date(2048, 8, 3),
           datetime.date(2049, 8, 3), datetime.date(2050, 8, 3)]
```

Chapter 6

Interpolation, Discount Factors and Forward Rates

Exercise 6.1 (Assertion)

Python has a useful command called `assert` which can be used for checking that a given condition is satisfied, and raising an error if the condition is not satisfied.
The following line does not cause an error, in fact it does nothing since 1 is lower than 2, hence the condition is met.

```
assert 1 < 2
```

This causes an error (the condition is evaluated to false.

```
assert 1 > 2
```

`assert` can take a second argument with a message to display in case of failure.

```
assert 1 > 2, "Two is greater than one"
```

Now takes the `df` function from Chapter 6 of the Lecture Notes and modify it by adding some assertions to check that:

- the pillar date list contains at least 2 elements;
- the pillar date list has the same length as the discount factor one;
- the first pillar date is equal to the today's date;
- the value date (first argument `d`) is greater or equal to the first pillar date and also less than or equal to the last pillar date.

Then try using the function with some invalid data to make sure that your as sertions are correctly checking the desired conditions

Exercise 6.2 (Black-Scholes Again)

Copy into the file `finmarkets.py` the function used to compute Black Scholes formula used in Ex. 4.1. This is another utility for our financial library. Then repeat Ex. 4.1 now using the version of the Black and Scholes formula in the `finmarkets` module.

Exercise 6.3 (Discount Curves)

Following the steps outlined in Chapter 6 of the Lecture Notes, implement a `DiscountCurve` class and add it to `finmarkets` module. The class should have as attributes the pillar dates and the corresponding discount factors and two methods, one to interpolate discount factors and another to calculate forward rates. Finally using that class compute the forward 6M LIBOR coupon using the curves given below in pre and post 2008 crisis way.

Input:

```
observation_date = date (2020, 1, 1)
t1 = date(2020,4, 1)
t2 = date(2020, 10, 1)

# for EONIA
pillar_dates_eonia = [date(2020 , 1 ,1),
                      date(2021, 1, 1),
                      date(2022, 10 ,1)]
discount_factors_eonia = [1.0, 0.97, 0.72]

# for LIBOR 6M
pillar_dates_libor = [date(2020, 1 ,1),
                      date(2020, 6, 1),
                      date(2020, 12 ,1)]
discount_factors_libor = [1.0, 0.95, 0.90]
```

Exercise 6.4 (Forward Rate Curve)

Write a ForwardRateCurve class (for EURIBOR/LIBOR rate curve) which doesn't compute discount factors but only interplatates forward rates; then add it to the `finmarkets` module (this function is used to define the LIBOR curve needed throughout future lessons).