CASS
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON
EST 1894

Changing more than a name

The Business School (formerly Cass) Coursework Test

Term 3 Elective Module

Module Code
SMM283

Module Title
Introduction to Python

Date
16th of July 2021

Time
09:00 to 11:00

Division of Marks:	**100 marks**
	Part (a) is worth 10 marks
	Part (b) is worth 15 marks
	Part (c) is worth 35 marks
	Part (d) is worth 40 marks
Instructions to students**:**	**Use Python IDLE (3.5 or above) only to code the Python solution as a Python file (*StudentID.py*), alongside any other output text or Excel file(s). Save frequently the file(s) in a *StudentID* named folder, and upload the last version of the saved file(s) to Moodle at the end of the test. There should be only one Python file created. This should contain the solution to all four parts. Refer to instruction at the end of the test paper for clarity on partial solutions.**

This paper contains contains **the test activity set made up of four interlinked sections (a, b, c, d)** and comprises **seven** pages including the title page.

Your work should be in your own words, it should **NOT** contain material copied straight from lecture notes, textbooks, or other resources. If you do rely on external sources within your answer, these should be properly referenced/cited.

Students are expected to show all necessary workings to obtain their final solution. If this is not done, then marks will be deducted even when the correct numerical solution is obtained.

Module Leader (s): Dr. Adrian Euler

TEST ACTIVITY SET

Create and test Python functions based on the specifications of sections (a), (b), (c), and (d). User - defined functions in sections (a), (b) should be coded with functional cohesion attribution, whereas (c) and (d) should follow procedural cohesion. The `main` function of part (d) should serve as the master control function of the entire program. The functions from (a), (b), and (c) should be coded as independent entities, free of any internal input or memory validation, and should perform only the tasks specified under each section. The `main` function's procedure is given as algorithmic sequence of instructions, which you must convert to Python code, so that it tests the other functions, based on requirements under section (d). The `main` function should be coded to include in-range and data type validation and should include exception handling. Instruction on partial solutions is included at the end of this paper.

(a)

A function called `maximum` that computes the maximum of two real numbers (numbers with decimal representation and precision) and returns the maximum.

The Python function prototype should be:

```
def maximum(x, y):
```

It receives any real numbers, `x` and `y` and returns the maximum of the two numbers. The maximum should retained through the use of control statements, and should not use any built-in functions, such as the built-in `max()` function.

(10 marks)

(b)

A function called `optionPayoff` that computes the payoff of a put option, based on a stock price and constant exercise price. The mathematical formula for the call and put option pay offs are

$$callpayoff = MAX[S_T - K,\ 0]$$

$$putpayoff = MAX[0,\ K - S_T]$$

and the Python function prototype should be:

```
def optionPayoff(opType, s, k):
```

It receives the following:

(1) `oType`, a <u>string</u> representing the option type i.e. “c“ for a call option and “p” for a put option.
(2) `s` the market price of the underlying stock and must be positive.
(3) `k`, the strike price or the exercise price at expiration and must be positive.

This function should compute the payoff of the option, where among other things, it should call the `maximum` function you have coded in section (a), and should also decide if the option if a call option or a put option, thus returning the payoff of the correct option type (call or put). There is only one `optionPayoff` function for both calls and puts.

(15 marks)

(c)

A function called `printPayoff` that prints out either to the monitor or to a file (txt or xls). The option of printing out to the monitor or in a file will depend on an `ip` parameter, which is specified by the programmer; `ip =1` would trigger printing to the monitor, and `ip =2` would trigger the printing to a file.

The Python function prototype should be:

```
def printPayoff(msg, s = [], payoff = [], ip = 1):
```

It receives the following:

(1) `msg`, a text message as a string
(2) `s`, an array or list of emulated stock prices.
(3) `payoff`, an array or list of option spread payoffs.
(4) `ip`, the choice of printing to monitor or file. The default value for `ip` is 1.

The data should be printed in a tabulated format, with the columns of data next to each other (not one under the other). The data printed on the monitor (`ip = 1`) should be formatted to include the GBP currency symbol and printed in a field of 13 within one penny precision (2 decimal places), and should be right aligned. Where as data printed to text *(txt)* or Excel (*xls*) file (`ip = 2`) should be printed in columns next to each other with no data field and precision specification. You can choose to write to a *txt* file or *xls* file. This should be specified when file is created in Python, not through prompt.

(35 marks)

(d)

Test the functions by coding a user defined `main` function based on the following instructions:

1. Variables

```
tic, is a timer variable; captures program start time.
toc, is a timer variable; captures program end time.

stype,  a string variable with data entered through input()
function, 'c' for call; 'p' for put.

size, an integer variable with data entered through input()
function in the (40.0 to 140.0) interval.

ip, an integer variable with data entered through input()
function, 1 for monitor; 2 for file.

x = 0.0
dx = 10.0
s = []
longOptionK1 = []
longOptionK3 = []
shortOptionK2 = []
butterfly = []
k1 = (size*dx)*0.3
k2 = (size*dx)*0.5
k3 = (size*dx)*0.7
```

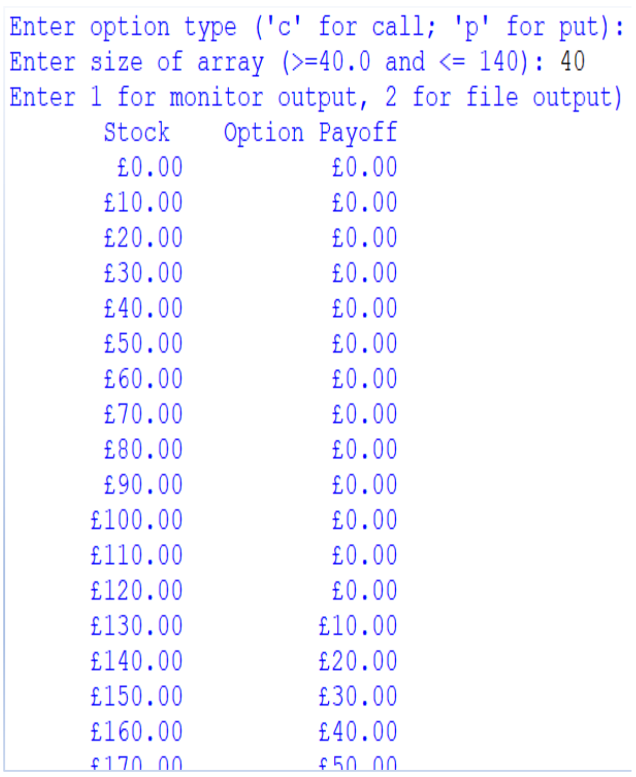
2. Prompt the use to enter the type of the option as a string. The input string should be validated and converted to lower case. Value should be stored on variable `stype`.
3. Prompt the user to enter size of the list. The input read from the keyboard should be validated in the proper range and data type with proper exception handling. Size of array should be between 40.0 and 140.0. Validate input to force user to input size above or equal 40.0 and below or equal to 140.0.
4. Prompt the user to enter choice for data printing; 1 to monitor, 2 to file. The input read from the keyboard should be validated and of the proper data type with proper exception handling.
5. Based on (4), data will be printed to monitor or file. Data printed should be the stock price and the call or put butterfly payoff in a formatted manner through the `printPayoff` function. You'd need to call function `optionPayoff()` and `printPayoff()` from the main function.
6. Print a message indicating the time it took to execute the program, through the use of data in variable `tic` and `toc`.

Note:

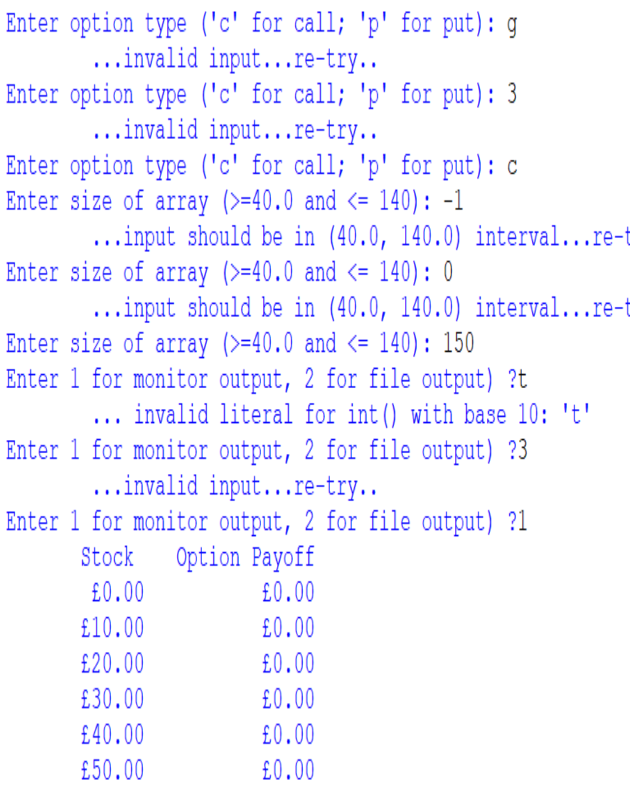
A butterfly spread is a neutral option strategy combining bull and bear spreads of four option contracts of the same option type and with the same expiration, but three different strike prices; a long option contract with exercise K_1, two short option contracts with exercise K2, and another long option contract with exercise K_3, where $K_1 < K_2 < K_3$.

CALL (BULL): $\text{Butterfly} = MAX(0, S-K_1) - 2*MAX(0, S-K_2) + MAX(0, S-K_3)$
PUT (BEAR): $\text{Butterfly} = MAX(0, K_1-S) - 2*MAX(0, K_2 - S) + MAX(0, K_3 - S)$

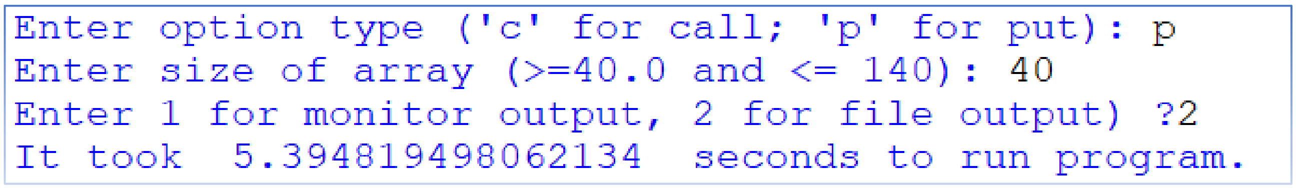
Output Scenarios:



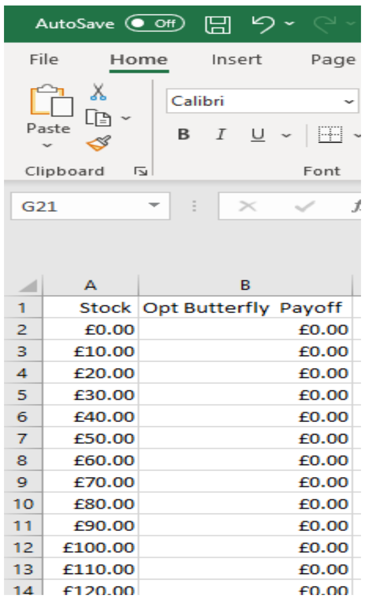
```
Enter option type ('c' for call; 'p' for put):
Enter size of array (>=40.0 and <= 140): 40
Enter 1 for monitor output, 2 for file output)
        Stock     Option Payoff
        £0.00              £0.00
       £10.00              £0.00
       £20.00              £0.00
       £30.00              £0.00
       £40.00              £0.00
       £50.00              £0.00
       £60.00              £0.00
       £70.00              £0.00
       £80.00              £0.00
       £90.00              £0.00
      £100.00              £0.00
      £110.00              £0.00
      £120.00              £0.00
      £130.00             £10.00
      £140.00             £20.00
      £150.00             £30.00
      £160.00             £40.00
      £170.00             £50.00
```



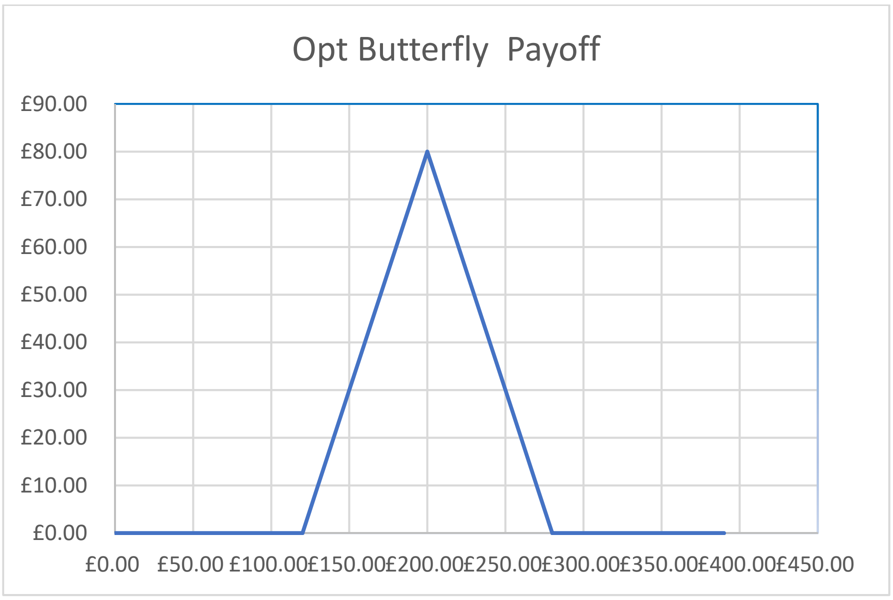
```
Enter option type ('c' for call; 'p' for put): g
        ...invalid input...re-try..
Enter option type ('c' for call; 'p' for put): 3
        ...invalid input...re-try..
Enter option type ('c' for call; 'p' for put): c
Enter size of array (>=40.0 and <= 140): -1
        ...input should be in (40.0, 140.0) interval...re-t
Enter size of array (>=40.0 and <= 140): 0
        ...input should be in (40.0, 140.0) interval...re-t
Enter size of array (>=40.0 and <= 140): 150
Enter 1 for monitor output, 2 for file output) ?t
        ... invalid literal for int() with base 10: 't'
Enter 1 for monitor output, 2 for file output) ?3
        ...invalid input...re-try..
Enter 1 for monitor output, 2 for file output) ?1
       Stock     Option Payoff
       £0.00             £0.00
      £10.00             £0.00
      £20.00             £0.00
      £30.00             £0.00
      £40.00             £0.00
      £50.00             £0.00
```



```
Enter option type ('c' for call; 'p' for put): p
Enter size of array (>=40.0 and <= 140): 40
Enter 1 for monitor output, 2 for file output) ?2
It took  5.394819498062134  seconds to run program.
```



	A	B
1	Stock	Opt Butterfly Payoff
2	£0.00	£0.00
3	£10.00	£0.00
4	£20.00	£0.00
5	£30.00	£0.00
6	£40.00	£0.00
7	£50.00	£0.00
8	£60.00	£0.00
9	£70.00	£0.00
10	£80.00	£0.00
11	£90.00	£0.00
12	£100.00	£0.00
13	£110.00	£0.00
14	£120.00	£0.00



(40 marks)

Note:

You are expected to produce the “yourStudentID.py” file and a file “butterfly.xls”. The program must run in order to be marked.

Your solution should contain Python implementation of parts (a), (b), (c), and (d), where you should not use the built-in function `max`. This solution will be marked from 0 to 100. The following scenarios are valid:

(i) If you find that you can not implement part (a), you may proceed to implement (b), (c), and (d), where you may use the built-in function `max`. This solution will be marked from 0 to 90. You loose 10 marks for not implementing part (a)

(ii) If you find that you can not implement part (b), you may proceed to implement (a), (c), and (d), where you may implement task (b) as code task inside the `main` function. You must use the `maximum` function of part (a). This solution will be marked from 0 to 85. You loose 15 marks for not implementing part (b).

(iii) If you find that you can not implement parts (a) and (b), you may proceed to implement (c), and (d), where you may implement the tasks of (a) and (b) as code inside the `main`, where the built-in function `max` may be used. You must use the `maximum` function of part (a). This solution will be marked from 0 to 75. You loose 25 marks for not implementing parts (a) and (b).

(iv) If you find that you can not implement part (c), you may proceed to implement (a), (b), and (d), where you may code the (c) task inside the `main` function. This solution will be marked from 0 to 70. You loose 30 marks for not implementing part (c).

(v) If you find that you can not implement parts (a) and (c), you may proceed to implement (b), and (d), where you may code the tasks of (a) and (c) inside the main function, where the built-in function max may be used. This solution will be marked from 0 to 65. You loose 35 marks for not implementing parts (a) and (c).

(vi) If you find that you can not implement parts (b) and (c), you may proceed to implement (a), and (d), where you may code the tasks of (b) and (c) in the main, where the built-in function `max` may be used. You must use the `maximum` function of part (a). This solution will be marked from 0 to 60. You loose 40 marks for not implementing parts (b) and (c).

(vii) If you find that you can not implement parts (a), (b) and (c), you may proceed to implement (d), where you may code the tasks of (a) , (b), and (c) in the `main`, where the built-in function `max` may be used. This solution will be marked from 0 to 55. You loose 45 marks for not implementing parts (a), (b) and (c).

(viii) If you choose to implement a program without any functions at all to perform the task, your work will be marked from 0 to 50.

END OF EXAMINATION