Assignment 3 Guidance. -25/11/2022

You can write your own test file to test whether your class file is correct or not. But you should only submit the class file: q1.py, q2.py, q3.py, and *strictly follow the given format.* To import your own class in the test file. You can simply put the class file and the test file under the same folder.

Question 1:

A. All the following requirements should be involved in the codes:

1. q1.py: the file containing the class
2. Class "Flower" should be contained in q1.py.
3. An initializer and methods for setting the value of each type should be in the program.
4. Input order: Name, number of petals, and price.

B. The following instructions explain how we would test your codes:

The basic test framework is given by:

```
from q1 import *
# test one
flower = Flower('juice', 5, 7.82)
result_one = flower.Information()
```

>>>Here is the information of your flower. Name: juice, Number of petals: 5, price: 7.82

Your program should be robust enough to handle possible inappropriate inputs:

1. The 1st input is not a string

```
flower = Flower(666, 5, 7.82)
result_one = flower.Information()
```

>>>The input of the flower name is incorrect. A string is required.

2. The 2nd input is not an integer

```
flower = Flower('juice', 5.21, 7.82)
flower.Information()
```

>>>The input of the number of petals is incorrect. An integer is required.

Question 2:

A. All the following requirements should be involved in the codes:

1. q2.py: the file containing the class
2. Class "process_derivative" should be contained in q2.py.
3. A method "get_first_derivative" for calculating the first derivative should be involved in the program.
4. Something should be noted (You can also read through the Assignment 3 paper):

a) There is only one variable and can be any English letter.
b) The output order should be the same as the input terms order.
c) The degree of each input term should be a non-negative integer (0 should be considered).
d) You can not import any other existing libs to calculate the first derivative.

B. The following instructions explain how we would test your codes:

1. Test one:

```
from q2 import *
# test one
test_one = process_derivative('2*x^3+3*x^2+5*x+1')
result_one = test_one.get_first_derivative()
```

>>> The first derivative is: '6*x^2+6*x+5'

2. Test two:

```
from q2 import *
# test two
test_two = process_derivative('2*y^3+3*y^2+5*y+1')
result_two = test_two.get_first_derivative()
```

>>> The first derivative is: '6*y^2+6*y+5'

3. Test three:

```
from q2 import *
# test three
test_three = process_derivative('0*x^3+3*x^2+5*x+1^0')
result_three = test_three.get_first_derivative()
```

>>> The first derivative is: '6*x'

4. Test four:

```
from q2 import *
# test four
test_four = process_derivative('2*x^6-3*x^8+5*x+1')
result_four = test_four.get_first_derivative()
```

>>> The first derivative is: '12*x^5-24*x^7+5'

5. Test five:

Test five is to input only one integer, e.g., '10'.

>>> The first derivative is: '0'

Question 3:

A. All the following requirements should be involved in the codes:

1. q3.py: the file containing the class
2. Class "ecosystem" should be contained in q3.py.
3. An initializer should be involved in the program to assign each input value.
4. A method "simulation" to simulate the system in several steps should be involved in the program. The input order of this function is supposed to be: river length, the number of fish, the number of bears, number of steps for simulation.
5. Something should be noted (You can also read through the Assignment 3 paper):
   a) Fish (F) and bears (B) are allocated randomly before simulation.
   b) Each agent (F/B) from left to right would take one random action in each step.
   c) Fish would be eliminated once they run into a bear position.
   d) Two same creatures running into each other would create the third creature in any empty position 'N'. The newly generated animals will NOT take any action in the current step.

B. The following instructions explain how we would test your codes:

1. Test one:

```
from q3 import *
# test one (river length, num of fish, num of bears, steps)
eco = ecosystem(5, 2, 1, 3)
result_one = eco.simulation()
```

```
>>>  The ecosystem at the beginnning of the step 1:
['N', 'B', 'F', 'N', 'F']
Animal: B, Action: 1
The current ecosystem after the action:
['N', 'N', 'B', 'N', 'F']
Animal: F, Action: 0
The current ecosystem after the action:
['N', 'N', 'B', 'N', 'F']
The ecosystem at the beginnning of the step 2:
['N', 'N', 'B', 'N', 'F']
Animal: B, Action: 0
The current ecosystem after the action:
['N', 'N', 'B', 'N', 'F']
Animal: F, Action: 1
The current ecosystem after the action:
['N', 'N', 'B', 'N', 'F']
The ecosystem at the beginnning of the step 3:
['N', 'N', 'B', 'N', 'F']
Animal: B, Action: -1
The current ecosystem after the action:
['N', 'B', 'N', 'N', 'F']
Animal: F, Action: 1
The current ecosystem after the action:
['N', 'B', 'N', 'N', 'F']
```

2. Test two:

```
from q3 import *
# test two (river length, num of fish, num of bears, steps)
eco = ecosystem(5, 3, 0, 3)
result_two = eco.simulation()
```

```
>>>
The ecosystem at the beginnning of the step 1:
['F', 'N', 'F', 'F', 'N']
Animal: F, Action: 0
The current ecosystem after the action:
['F', 'N', 'F', 'F', 'N']
Animal: F, Action: 1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'N']
Animal: F, Action: -1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
The ecosystem at the beginnning of the step 2:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: 0
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: -1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: 1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: 0
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: -1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
The ecosystem at the beginnning of the step 3:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: 1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: 0
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: -1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: 1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
Animal: F, Action: -1
The current ecosystem after the action:
['F', 'F', 'F', 'F', 'F']
```

Note that the newly generated animals will NOT take actions in the current step.

3. Test three:

```
from q3 import *
# test three (river length, num of fish, num of bears, steps)
eco = ecosystem(5, 2, 3, 3)
result_three = eco.simulation()
```

```
>>>    The ecosystem at the beginnning of the step 1:
       ['B', 'F', 'F', 'B', 'B']
       Animal: B, Action: 1
       The current ecosystem after the action:
       ['N', 'B', 'F', 'B', 'B']
       Animal: F, Action: -1
       The current ecosystem after the action:
       ['N', 'B', 'N', 'B', 'B']
       Animal: B, Action: 1
       The current ecosystem after the action:
       ['N', 'B', 'B', 'B', 'B']
       Animal: B, Action: 0
       The current ecosystem after the action:
       ['N', 'B', 'B', 'B', 'B']
       The ecosystem at the beginnning of the step 2:
       ['N', 'B', 'B', 'B', 'B']
       Animal: B, Action: 1
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
       Animal: B, Action: 1
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
       Animal: B, Action: 0
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
       Animal: B, Action: 0
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
       The ecosystem at the beginnning of the step 3:
       ['B', 'B', 'B', 'B', 'B']
       Animal: B, Action: -1
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
       Animal: B, Action: -1
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
       Animal: B, Action: 1
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
       Animal: B, Action: -1
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
       Animal: B, Action: -1
       The current ecosystem after the action:
       ['B', 'B', 'B', 'B', 'B']
```