Pro. 1

Format of your code should follow the template:

1. q1.py: the file containning the class

2. the class named as Flower

3. the order of the input for the class: Name, number of petals, price

Here is the example how we are going to test yout codes. Based on differen errors of your input, the output is suppose to point out the description of the wrong input.

Hint:

1. Please review the initializer of the 'class' in python.

2. Your program should be robust enough.

In [17]:

```
from q1 import *
```

In [18]:

```
flower = Flower('juice', 5, 7.82)

flower.Information()
```

```
Here is the information of your flower. Name: juice, Number of petals: 5, Price: 7.8
2
```

In [19]:

```
flower = Flower('juice', 5.66, 7.82)

flower.Information()
```

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

In [20]:

```
flower = Flower(56, 5, 7.82)

flower.Information()
```

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

In [21]:

```python
flower = Flower('juice', 5, 7)

flower.Information()
```

```
The input of the price is incorrect. A type of float is required.
```

Pro. 2

From the standard definition of polynomial, no need to consider the negative power.

The name of .py file is q2, the class name is process_derivative.

In the class, there should be a function named get_first_derivative that return the result.

I will check and call your code as follows:

In [ ]:

```python
from q2 import process_derivative

# test one
test_one = process_derivative('2*x^3+3*x^2+5*x+1')
result_one = test_one.get_first_derivative()
print(result_one)
# the output result should be: 6*x^2+6*x+5.
```

In [ ]:

```python
# test two
test_two = process_derivative('2*x^3+3*y^2+5*x+1')
result_two = test_two.get_first_derivative()
print(result_two)
# the output result should be: invalid input!
```

In [ ]:

```python
# test three
test_three = process_derivative('2*y^3+3*y^2+5*y+1')
result_three = test_three.get_first_derivative()
print(result_three)
# the output result should be: 6*y^2+6*y+5.
```

In [ ]:

```
# test four
test_four = process_derivative('2*y^6-3*y^8+5*y+1')
result_four = test_four.get_first_derivative()
print(result_four)
# the output result should be: 12*y^5-24*y^7+5.
```

In [ ]:

```
# test five
test_five = process_derivative('10')
result_five = test_five.get_first_derivative()
print(result_five)
# the output result should be: 0.
```

Pro. 3

Please carefully read the material of this problem.

Format of your code should follow the template:

1. q3.py: the file containning the class

2. the class named as ecosystem

3. the class has the function which is named as simulation

3. the order of the input for the fuction "simulatuion": the length of the river, the number of fish, the number of bears, number of steps in the simulation

Here is a test sample for your to compare your code.

Since the creature takes random action, the output is not unique, even if your use identical inputs with the test sample. However, the logic is supposed to be same

Your code ishould be able to handle all different situations.

Here is the hide from me:

The action order of creatures is defined at the beginning of each step.

If the fish which was in the action list at the beginning of the step is eaten by the bear in the step, then the defined action by the fish can not be excuted since it is dead.

The new creature can be randomly allocated to the new place, only if there is an avaliable empty space for the new instance.

In [1]:

```
from q3 import *
```

In [8]:

```
# ecosystem(river length, number of the fish, number of the bear, steps)
eco = ecosystem(5,2,1,3)
```

In [9]:

```
river = 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']
```

In [12]:

```
# ecosystem(river length, number of the fish, number of the bear, steps)
eco = ecosystem(5,3,0,3)
```

In [13]:

```
river = 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']
```

In [14]:

```
# ecosystem(river length, number of the fish, number of the bear, steps)
eco = ecosystem(5,2,3,3)
```

In [15]:

```
river = 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']
```