# **Farming**

Release 1.0.0

Armaghan

# **CONTENTS:**

1	farm	ing															3
	1.1	farmir	ng package	e		 	 	 	 		 			 			3
		1.1.1	Submo	dules .		 	 	 			 			 			3
		1.1.2	farming	g.farm1	module	 	 	 	 		 			 			3
			1.1.2.1	farm1	ру	 	 	 	 		 			 			3
		1.1.3	farming	g.part2 i	nodule	 	 	 			 			 			5
			1.1.3.1	part_2	.py	 	 	 	 		 			 			5
		1.1.4	farming														
			1.1.4.1														
		1.1.5	Module														
			1.1.5.1														
2	Indic	es and	tables														9
Рy	thon N	Module	Index														11

This documentation contains what you need to know about the **Farming** project.

CONTENTS: 1

2 CONTENTS:

#### **FARMING**

### 1.1 farming package

#### 1.1.1 Submodules

#### 1.1.2 farming.farm1 module

#### 1.1.2.1 farm1.py

This module contains classes for simulating a farm with multiple cows and pens. It includes the following classes:

- Cow: A class representing a cow with its breed, feed type, milk yield, and greenhouse gas emission.
- Farm: A class representing a farm with a list of cows and pens, and methods for adding cows and pens, getting pen allocation, milk yield, greenhouse gas emissions, and cost estimation.
- Pen: A class representing a pen with its breed and cow count.

This module also includes a test suite for testing the functionality of the Farm class.

This class represents a cow with its breed, feed\_type, milk\_yield, and ghg\_emission.

Initialize a new Cow object.

#### **Parameters**

- **breed** (*str*) The breed of the cow.
- **feed\_type** (*str*) The type of feed given to the cow.
- milk\_yield (float) The milk yield of the cow in gallons per day.
- **ghg\_emission** (*float*) The greenhouse gas emission of the cow in kg per day.

#### breed: str

The breed of the cow.

#### feed\_type: str

The type of feed given to the cow.

#### ghg\_emission: float

The greenhouse gas emission of the cow in kg per day.

#### milk\_yield: float

The milk yield of the cow in gallons per day.

4

class farming.farm1.Farm

```
Bases: object
     This class represents a farm with a list of cows and pens, and methods for adding cows (add_cow()) and pens
     (add_pen()), getting pen allocation (get_pen_allocation()), milk yield (get_milk_yield()), greenhouse
     gas emissions (get\_ghg\_emissions()), and cost estimation (get\_cost\_estimation()).
     Initialize a new Farm object with empty lists for cows and pens.
     add\_cow(cow: farming.farm1.Cow) \rightarrow None
           Add a cow to the list of cows on the farm.
               Parameters cow (Cow) – The Cow object to be added.
               Returns None
     add_pen(pen: farming.farm1.Pen) \rightarrow None
           Add a pen to the list of pens on the farm.
               Parameters pen (Pen) – The Pen object to be added.
               Returns None
     cows: List[farming.farm1.Cow]
           A list holding cows.
     \mathtt{get\_cost\_estimation}() \to \mathrm{None}
           Calculate and print the estimated cost on the farm.
               Returns None
     get\_ghg\_emissions() \rightarrow None
           Calculate and print the expected greenhouse gas emissions on the farm.
               Returns None
     get_milk_yield() \rightarrow None
           Calculate and print the expected milk yield on the farm
               Returns None
     get_pen_allocation() \rightarrow None
           Print the number of cows in each pen on the farm.
               Returns None
     pens: List[farming.farm1.Pen]
           A list holding pens.
class farming.farm1.Pen(breed: str, cow count: int)
     Bases: object
     This class represents a pen with its breed and cow_count.
     Initialize a new Pen object.
           Parameters
                 • breed (str) – The breed of cows in the pen.
                 • cow_count (int) – The number of cows in the pen.
     breed: str
           The breed of cows in the pen.
     cow_count: int
           The number of cows in the pen.
```

#### class farming.farm1.TestFarm(methodName='runTest')

Bases: unittest.case.TestCase

This class tests the methods of the Farm class.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

#### setUp()

Initialize objects required for testing.

Returns None

#### test\_add\_cow()

Test the add\_cow() method of the Farm class.

**Returns** None

#### test\_add\_pen()

Test the add\_pen() method of the Farm class.

Returns None

#### test\_get\_ghg\_emissions()

Test the *get\_ghg\_emissions()* method of the *Farm* class.

Returns None

#### test\_get\_milk\_yield()

Test the get\_milk\_yield() method of the Farm class.

Returns None

#### test\_get\_pen\_allocation()

Test the get\_pen\_allocation() method of the Farm class.

Returns None

#### 1.1.3 farming.part2 module

#### 1.1.3.1 part 2.py

This module contains classes for simulating a special kind of farm with cows. It includes the following classes:

- Cow: A class representing a cow with its breed, feed type, milk yield attributes.
- Farm: A class representing a farm with a list of cows.

This module also includes a test suite.

#### class farming.part2.Cow(breed: str, feed\_type: str, milk\_yield: int)

Bases: object

This class represents a cow with its breed, feed\_type, and milk\_yield attributes.

Creates instances of the class Cow.

#### **Parameters**

- **breed** (*str*) The breed of the cow.
- **feed\_type** (*str*) The type of feed given to the cow.
- **milk\_yield** (*int*) The milk yield of the cow in gallons per day.

breed: str

The breed of the cow.

feed\_type: str

The type of feed given to the cow.

milk\_yield: float

The milk yield of the cow in gallons per day.

class farming.part2.Farm(cows: List[farming.part2.Cow])

Bases: object

This class represents a farm with a list of cows.

Initialize a new Farm object with empty lists for cows and pens.

**Parameters cows** ((List[Cow])) – A list holding cows.

cows: List[farming.part2.Cow]

A list holding cows.

class farming.part2.TestCalculateMilkYieldByBreedAndFeedType(methodName='runTest')

Bases: unittest.case.TestCase

The test class to calculate milk yield by breed and feed type.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

test\_calculate\_milk\_yield\_by\_breed\_and\_feed\_type()

Define a unit test for the calculate\_milk\_yield\_by\_breed\_and\_feed\_type() function.

 $farming.part2.calculate\_milk\_yield\_by\_breed\_and\_feed\_type(\textit{farm:} farming.part2.Farm) \rightarrow Dict[str, Dict[str, int]]$ 

Initialize a dictionary to keep track of the total milk yield for each feed type

#### 1.1.4 farming.part3\_fertilizer module

#### 1.1.4.1 part3\_fertilizer.py

This module contains the following classes:

- Breed: A class for breed of cows.
- Farm: A class for a farm with a list of breeds.
- FoodType: An enumeration for food types.

this module also includes a test suite.

class farming.part3\_fertilizer.Breed(name: str, food\_effects: dict)

Bases: object

Define a class for breed of cows

The initializer to define the breed of cows.

#### **Parameters**

- name (str) The name of the breed.
- **food\_effects** (*dict*) This dictionary holds the effects of the foods.

```
food_effects: dict
          This dictionary holds the effects of the foods.
     get_fertilizer_production(food: farming.part3_fertilizer.FoodType) → float
          Return the fertilizer production for a specific food type and breed
               Parameters food (FoodType) – the food type.
               Returns food_effects[food].get('fertilizer', 0)
     name:
          The name of the breed.
class farming.part3_fertilizer.Farm(breeds: List/farming.part3_fertilizer.Breed/)
     Bases: object
     Define a class for a farm with a list of breeds
     Initialises an instance of the class Farm containing a list of breeds.
          Parameters breeds (List [Breed]) – A list holding the breeds of the farm.
     breeds: List[farming.part3_fertilizer.Breed]
          The list holding the breeds of the farm.
     calculate_food_production(food: farming.part3_fertilizer.FoodType, soil_size: int = 1000) \rightarrow dict
          This method finds the food production.
               Parameters
                   • food (FoodType) – Specifies the food type.
                   • soil_size (int) – Specifies the soil size.
               Returns food_production that is a dictionary.
class farming.part3_fertilizer.FoodType(value)
     Bases: enum. Enum
     Define an enumeration for food types
     CORN = 'corn'
          Represents corn
     GRAIN = 'grain'
          Represents grain
     GRASS = 'grass'
          Represents grass
class farming.part3_fertilizer.TestFarm(methodName='runTest')
     Bases: unittest.case.TestCase
     Define a test class for the Farm class
     Create an instance of the class that will use the named test method when executed. Raises a ValueError if the
     instance does not have a method with the specified name.
     setUp()
          Necessary setups for testing are done in this method.
               Returns None
     test_calculate_food_production()
          Define a test function for the farming.part3_fertilizer.calculate_food_production() method
          of the farming.part3_fertilizer.Farm class
```

#### Returns None

### 1.1.5 Module contents

#### 1.1.5.1 \_\_init\_\_.py

This is the \_\_init\_\_ of the module farming. Currently, this file is empty.

# CHAPTER

# TWO

# **INDICES AND TABLES**

- genindex
- modindex
- search

# **PYTHON MODULE INDEX**

```
f
farming, 8
farming.farm1, 3
farming.part2, 5
farming.part3_fertilizer, 6
```

# **INDEX**

\spxentryadd_cow()\spxextrafarming.farm1.Farm method, 4	\spxentryFoodType\spxextraclass in farming.part3_fertilizer, 7
\spxentryadd_pen()\spxextrafarming.farm1.Farm method, 4	\spxentryget_cost_estimation()\spxextrafarming.farm1.Farm method, 4
lem:lem:lem:lem:lem:lem:lem:lem:lem:lem:	\spxentryget_fertilizer_production()\spxextrafarming.part3_fertilizer.Breed method, 7
\spxentrybreed\spxextrafarming.farm1.Cow attribute, 3 \spxentrybreed\spxextrafarming.farm1.Pen attribute, 4	\spxentryget_ghg_emissions()\spxextrafarming.farm1.Farm method, 4
\spxentrybreed\spxextrafarming.part2.Cow attribute, 5	\spxentryget_milk_yield()\spxextrafarming.farm1.Farm
$\verb \spxentry  breeds \\   spxextrafarming.part 3\_fertilizer. Farm at-\\$	method, 4
tribute, 7	\spxentryget_pen_allocation()\spxextrafarming.farm1.Farm method, 4
$\verb \spxentrycalculate_food_production() \verb \spxextrafarming.part  $	t3sfxtilizegFgremission\spxextrafarming.farm1.Cow
method, 7	attribute, 3
\spxentrycalculate_milk_yield_by_breed_and_feed_type()\	ASPSYSTATEDGRAIN\spxextrafarming.part3_fertilizer.FoodType
module farming.part2, 6	attribute, 7
\spxentryCORN\spxextrafarming.part3_fertilizer.FoodType attribute, 7	S\spxentryGRASS\spxextrafarming.part3_fertilizer.FoodType attribute, 7
\spxentryCow\spxextraclass in farming.farm1, 3	,
\spxentryCow\spxextraclass in farming.part2, 5	\spxentrymilk_yield\spxextrafarming.farm1.Cow at-
\spxentrycow_count\spxextrafarming.farm1.Pen attribute,	tribute, 3
4	\spxentrymilk_yield\spxextrafarming.part2.Cow at-
\spxentrycows\spxextrafarming.farm1.Farm attribute, 4	tribute, 6
\spxentrycows\spxextrafarming.part2.Farm attribute, 6	\spxentrymodule
	\spxentryfarming, 8
\spxentryFarm\spxextraclass in farming.farm1, 3	\spxentryfarming.farm1, 3
\spxentryFarm\spxextraclass in farming.part2, 6	\spxentryfarming.part2, 5
\spxentryFarm\spxextraclass in farming.part3_fertilizer, 7	\spxentryfarming.part3_fertilizer, 6
\spxentryfarming	
\spxentrymodule, 8	\spxentryname\spxextrafarming.part3_fertilizer.Breed at-
\spxentryfarming.farm1	tribute, 7
\spxentrymodule, 3	
\spxentryfarming.part2	\spxentryPen\spxextraclass in farming.farm1, 4
\spxentrymodule, 5	\spxentrypens\spxextrafarming.farm1.Farm attribute, 4
\spxentryfarming.part3_fertilizer	\spxentrysetUp()\spxextrafarming.farm1.TestFarm
\spxentrymodule, 6	method, 5
\spxentryfeed_type\spxextrafarming.farm1.Cow attribute,	\spxentrysetUp()\spxextrafarming.part3_fertilizer.TestFarm
3	method, 7
\spxentryfeed_type\spxextrafarming.part2.Cow attribute,	
6	\spxentrytest_add_cow()\spxextrafarming.farm1.TestFarm method, 5
\spxentry100a_effects\spxextrafarming.part3_fertilizer.Bree	method, 5
attribute, 6	

method, 5

method, 7

method, 5

 $\verb|\spxentrytest_get_milk_yield()| spxextrafarming.farm1.TestFarm|$ 

method, 5

\spxentrytest\_get\_pen\_allocation()\spxextrafarming.farm1.TestFarm method, 5

\spxentryTestCalculateMilkYieldByBreedAndFeedType\spxextraclass in farming.part2, 6

\spxentryTestFarm\spxextraclass in farming.farm1, 4

\spxentryTestFarm\spxextraclass in farm-

ing.part3\_fertilizer, 7

Index 13