Farming

Release 1.0.0

Armaghan

Apr 28, 2023

CONTENTS:

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

CHAPTER

ONE

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.

class farming.farm1.**Cow**(*breed: str*, *feed_type: str*, *milk_yield: float*, *ghg_emission: float*)

Bases: object

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.

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) → 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) → 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.

get_cost_estimation() → None
Calculate and print the estimated cost on the farm.

Returns None

get_ghg_emissions() → None
Calculate and print the expected greenhouse gas emissions on the farm.

Returns None

get_milk_yield() → None
Calculate and print the expected milk yield on the farm

Returns None

get_pen_allocation() → 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**(*farm: farming.part2.Farm*) → 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: str
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*) → 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

INDEX