Photocopiable/digital resources may only be copied by the purchasing institution on a single site and for their own use

2015 specification

for the 2017 A Level exam

AQA Paper 1 Exam Resource Pack 2017

**Rabbits and Foxes**

for A Level AQA Computer Science

Computer Science A Level | AQA | 7517



zigzageducation.co.uk

Publish your own work... Write to a brief...

Register atpublishmenow.co.uk

POD

7224



PYTHON3

# Pre-Release Commentary

**Rabbits and Foxes**



## Description of the Program

The program is a simulation of rabbit population over time and how it is affected by the foxes who hunt them.

The world is represented by a grid in which each square can contain a rabbit warren (a burrow where many rabbits live) or a fox, or both. F designates a fox, and a number designates a rabbit warren (and how many rabbits are in the warren).

The menu holds the following options:

 Run the simulation with default settings

 Run the simulation with custom settings

 Exit

The settings that can be changed in option 2 include:

 Landscape size

 Number or rabbit warrens at start

 Number of foxes at start

 Randomness (as a %)

During the simulation you can advance to the next time period showing detail or not, as well as inspect the current state of a fox or rabbit warren.

Each time a period runs, the rabbits can:

 Be eaten by a fox

 Be killed by something other than a fox

 Die of old age

 Increase in number (a number of new baby rabbits are born)

This information is displayed for each warren.

Each time a period runs there is a report on the foxes' age, how much food the foxes have eaten compared to what they need, and whether they have reproduced. If they have reproduced, the location of the new foxes is displayed at the bottom.



**Rabbits and Foxes**



## Description of Program Classes

This program contains multiple classes used to simulate foxes and rabbits in their natural environment.

The classes have been listed below, along with a very brief description of their purpose.

|  |  |
| --- | --- |
| Class | Description |
| Location | A class that creates an object corresponding to a location on the grid. |
| Simulation | The class that drives the main simulation. |
| Warren | A class that simulates a rabbit warren (where they live). |
| Genders (inherits enum.Enum) | A class that is used to track the gender of an animal – in this program only rabbits have a gender. |
| Animal | An abstract class used for creating foxes and rabbits. It contains all the variables and functions common to both animals. |
| Fox (inherits Animal) | The class used to model foxes. |
| Rabbit (inherits Animal) | The class used to model rabbits. |

## Description of Class Variables

Each class has a number of variables, only accessible in that particular class. For each of the classes above, their variables have been listed, along with a brief description.

|  |  |  |
| --- | --- | --- |
| Location – Instance variables | Type | Description |
| self.Fox | Fox | This value is equal to None when the simulation is started.  This value will hold a Fox object, if there is a fox in this particular location. |
| self.Warren | Warren | This value is equal to None when the simulation is started.  This value will hold a Warren object, if there is a warren in this particular location. |

|  |  |  |
| --- | --- | --- |
| Simulation – Instance variables | Type | Description |
| self.\_\_ViewRabbits | String | Variable that should either have the value ‘y’ or ‘n’. |
| self.\_\_TimePeriod | Integer | Counter to store how many iterations of the simulation have occurred. |
| self.\_\_WarrenCount | Integer | Variable that counts the number of warrens. |
| self.\_\_FoxCount | Integer | Variable that counts the number of foxes. |
| self.\_\_ShowDetail | Boolean | If this is true, more detail will be shown about the simulation. |
| self.\_\_LandscapeSize | Integer | Value that stores the size of the Landscape (the landscape is assumed to be square). |
| self.\_\_Variability | Integer | Value that determines how differently the simulation can vary. A high variability increases the range of possible other variable values. |
| self.\_\_FixedInitialLocations | Boolean | If True, the warrens and foxes will start in a fixed location. |
| self.\_\_Landscape | List | The variable used to store the space in which foxes can move, and the locations that have warrens. |

|  |  |  |
| --- | --- | --- |
| Warren – Instance variables | Type | Description |
| self.MAX\_RABBITS\_IN\_WARREN | Integer | Value that stores the maximum number of rabbits that can be stored inside a warren. |
| self.\_\_RabbitCount | Integer | The value that stores the number of rabbits who are alive. |
| self.\_\_PeriodsRun | Integer | This variable stores how many periods have passed since the start of the program. |
| self.\_\_AlreadySpread | Boolean | Boolean value used to determine whether a new warren needs to be created (if an existing one has become too large). |
| self.\_\_Variability | Integer | Value that determines how differently the simulation can vary. A high variability increases the range of possible other variable values. |
| self.\_\_Rabbits | List | A list containing the rabbits that are currently alive in the specified warren. |

|  |
| --- |
| Genders – Instance variables |
| This class is slightly different from the others – the purpose of this class is to be able to state that an animal is either Male or Female. This is better than giving it the value 1 or 2, as meaning is given to the number. |

|  |  |  |
| --- | --- | --- |
| Animal – Instance variables | Type | Description |
| self.\_NaturalLifespan | Integer | Integer value stating how long (in iterations) the animal will live for before dying of natural causes. |
| self.\_ProbabilityOfDeathOtherCauses | Real | Real value used for calculating the chance of death from other reasons. |
| self.\_IsAlive | Boolean | Boolean value that states whether an animal is alive or not. |
| self.\_ID | Integer | Integer value given to uniquely identify the animal. |
| self.\_Age | Integer | Value used to store the age of an animal (in iterations). |
| Animal.\_ID | *Integer* | *Value used to make sure that each new instance is given a unique identification number. Note: this is a CLASS VARIABLE, shared by every instance of the class.* |

|  |  |  |
| --- | --- | --- |
| Fox – Instance variables | Type | Description |
| self.\_\_DEFAULT\_LIFE\_SPAN | Integer | Value used for calculating the lifespan of the fox. The actual lifespan is calculated in the Animal class using the variability variable in the Simulation class. |
| self.\_\_DEFAULT\_PROBABILITY\_DEATH\_OTHER\_CAUSES | Real | Probability used for calculating the chance of dying from random causes. The actual probability is calculated in the Animal class using the variability variable in the Simulation class. |
| self.\_\_FoodUnitsNeeded | Integer | Number of food units needed to stop the fox from aging or dying. |
| self.\_\_FoodUnitsConsumedThisPeriod | Integer | Number of food units that have been consumed in one iteration of the simulation. |

|  |  |  |
| --- | --- | --- |
| Rabbit – Instance variables | Type | Description |
| self.\_\_DEFAULT\_LIFE\_SPAN | Integer | Value used for calculating the lifespan of a rabbit. The actual life span is calculated in the Animal class using the variability variable in the Simulation class. |
| self.\_\_DEFAULT\_PROBABILITY\_DEATH\_OTHER\_CAUSES | Real | Probability used for calculating the chance of dying from random causes. The actual probability is calculated in the Animal class using the variability variable in the Simulation class. |
| self.\_\_ReproductionRate | Real | Probability used for calculating the chance that any two rabbits reproduce. Inherited from the rabbits’ parents. |
| self.\_\_Gender | Genders | The gender of the rabbit. Equal to either Genders.Male or Genders.Female. |

## Description of Class Methods

Along with class variables, each class has a number of methods unique to that class. For each class, its functions Ⓕ and procedures Ⓟ have been described below.

|  |  |  |  |
| --- | --- | --- | --- |
| Location – Methods | Description | |  |
| \_\_init\_\_ Ⓟ | Input: Self  Output: Nothing | Creates a location instance:  1. Initially there are no foxes in the location  2. Initially there are no warrens in the location | |

Note: In each of the classes the variable 'self' is passed into every function. This refers to the instance of the class that is being manipulated.

| Simulation – Methods | Description | |  |
| --- | --- | --- | --- |
| \_\_init\_\_ Ⓟ | Input: Self, size of landscape (Integer), initial number of warrens (Integer), initial number of foxes (Integer), variability (Integer), whether fixed locations should be used or not (Boolean)  Output: Nothing | Creates a simulation instance:  1. Creates an array of Location instances according to the size of the Landscape.  2. Adds foxes and warrens to the Landscape.  3. Draws the landscape.  4. Starts the main simulation loop giving options to advance the generation, or inspect a fox/warren. | |
| \_\_InputCoordinate Ⓕ | Input: Self, Coordinate name (‘x’ or ‘y’)  Output: Coordinate (Integer) | Asks the user to enter a coordinate – depending on the supplied coordinate axis (x or y).  Returns an integer value corresponding to the specified coordinate axis. | |
| \_\_AdvanceTimePeriod Ⓟ | Input: Self  Output: None | Updates the simulation.  1. For each location:  a. If there is a warren in the space, and there are foxes that are alive, and they are near the warren, then they should eat some rabbits.  b. If the warren has reached its capacity, then a new warren needs to be created.  c. The warren should then advance to the next generation.  d. If the warren is now empty, then it should be removed from the landscape.  2. For each location:  a. If there is a fox in the space, advance to the next generation.  b. Check whether the fox has died.  i. If it has, remove it from the landscape, and jump to 3).  ii. If it has not died, check whether it should reproduce.  c. Reset the amount of food that it has consumed in this period.  3. If new foxes should be born, create and add them to the landscape. | |

| Simulation – Methods (cont.) | Description | |  |
| --- | --- | --- | --- |
| \_\_CreateLandscapeAndAnimals Ⓟ | Input: Self, initial number of warrens (Integer), initial number of foxes (Integer), whether fixed locations should be used or not (Boolean)  Output: None | Creates the landscape.  1. If the locations of each warren and fox have been fixed – create them in the fixed locations.  2. Otherwise, create new warrens and foxes randomly. The number is determined by the initial fox and warren count. | |
| \_\_CreateNewWarren Ⓟ | Input: Self  Output: None | Creates a new warren.  1. Find a spot that does not already contain a warren.  2. Create a new warren instance in that spot. | |
| \_\_CreateNewFox Ⓟ | Input: Self  Output: None | Creates a new fox.  1. Find a spot that does not already contain a fox.  2. Create a new fox instance in that spot. | |
| \_\_FoxesEatRabbitsInWarren Ⓟ | Input: Self, warren’s x-coordinate (Integer), warren’s y-coordinate (Integer)  Output: None | Function that lets foxes eat rabbits.  1. For each location:  a. If there is a fox in the location and they are less than 3.5 units away from a warren, 20% of the rabbits should be eaten.  b. OTHERWISE if there is a fox in the location and they are less than 7 units away from a warren, 10% of the rabbits should be eaten.  c. OTHERWISE no rabbits should be eaten. | |
| \_\_DistanceBetween Ⓕ | Input: Self, two sets of x- and y-coordinates  Output: Distance between the points (Real) | Calculates the distance between points – using Pythagoras’ theorem. | |
| \_\_DrawLandscape Ⓟ | Input: Self  Output: None | Draws the landscape shown in the simulation.  It checks each location and draws either a W for a warren or an F for a fox. | |

|  |  |  |
| --- | --- | --- |
| Warren – Methods | Description |  |
| \_\_init\_\_ Ⓟ | Input: Self, variability (Integer), number of rabbits in warren (Integer)  Output: None | Creates a new warren instance.  1. Creates spaces for the maximum number of allowed rabbits in a warren.  2. If the number of rabbits in the warren is not provided, it decides on an initial number of rabbits to have in the warren – dependent on the variability.  3. It adds that number of rabbits to the warren. |
| \_\_CalculateRandomValue Ⓕ | Input: Self, base value (Integer), variability (Integer)  Output: Random value (Real) | Provides a random number centred around the provided base value. If the variability is high, the range of possible values is higher. |

|  |  |  |
| --- | --- | --- |
| Warren – Methods (cont.) | Description |  |
| GetRabbitCount Ⓕ | Input: Self  Output: Number of rabbits in warren (Integer) | Returns the number of rabbits in the warren that the function is being called from. |
| NeedToCreateNewWarren Ⓕ | Input: Self  Output: Whether a new warren needs to be created (Boolean) | 1. Checks whether a warren has reached capacity, and hasn’t already been split up.  2. If this is true, then a new warren needs to be created. |
| WarrenHasDiedOut Ⓕ | Input: Self  Output: Whether a warren is empty or not (Boolean) | This function checks the number of rabbits in the warren.  1. If there are no rabbits it returns True.  2. Otherwise, it returns False. |
| AdvanceGeneration Ⓟ | Input: Self, whether you should show detail (Boolean)  Output: None | Advances the warren to the next generation.  1. If there are rabbits, kill some of them off from other factors.  2. If there are rabbits they should be aged.  3. If there are rabbits, and the warren is not overfull, and the warren contains males, then rabbits should breed.  4. Otherwise, a message will be printed if detail is shown saying that all of the rabbits are dead in that particular warren. |
| EatRabbits Ⓕ | Input: Self, number of rabbits that need to be eaten (Integer)  Output: Updated number of rabbits to be eaten (Integer) | Removes a fixed number of rabbits from the warren.  1. Finds a rabbit in the warren at random.  2. Removes it from the warren.  3. Repeats until enough rabbits have been eaten.  4. Compresses the list of rabbits. |
| \_\_KillByOtherFactors Ⓟ | Input: Self, whether you should show detail (Boolean)  Output: None | Kills rabbits at random depending on the percentage chance of a rabbit randomly dying from other causes.  1. Goes through the list of rabbits in the warren.  2. Checks whether they have died from other causes.  3. Removes them from the list of rabbits.  4. Compresses the list of the remaining rabbits. |
| \_\_AgeRabbits Ⓟ | Input: Self, whether you should show detail (Boolean)  Output: None | Makes each rabbit older.  1. Goes through the list of rabbits in the warren, incrementing their age.  2. Determines whether a rabbit has died of old age.  a. If they have, increase the death count, remove them from the rabbit list, and compress the list of living rabbits. |

|  |  |  |
| --- | --- | --- |
| Warren – Methods (cont.) | Description |  |
| \_\_MateRabbits Ⓟ | Input: Self, whether you should show detail (Boolean)  Output: None | Function that makes new rabbits.  1. Goes through the list of rabbits finding females.  2. If the rabbit is female and there is space for a baby rabbit:  a. Finds a male rabbit to breed with  b. Combines their reproduction rates  c. If it is greater than 1, a new rabbit is born |
| \_\_CompressRabbitList Ⓟ | Input: Self, number of dead rabbits (Integer)  Output: None | Shifts the rabbits so in the list there are no spaces between them. |
| \_\_ContainsMales Ⓕ | Input: Self  Output: Whether a warren contains males (Boolean) | Checks whether a warren has male rabbits in it.  1. It assumes that there are no males.  2. If it sees a male somewhere in the list, the function will return True. |
| Inspect Ⓟ | Input: Self  Output: None | Prints the age of the warren, and the number of rabbits that it contains. |
| ListRabbits Ⓟ | Input: Self  Output: None | Prints the status of each rabbit in the rabbits list. |

|  |  |  |  |
| --- | --- | --- | --- |
| Animal – Methods | Description | |  |
| \_\_init\_\_ Ⓟ | Input: Self, average lifespan (Integer), average probability of dying from other causes (Real), variability (Integer)  Output: None | Constructs a new instance of an animal. | |
| CalculateNewAge Ⓟ | Input: Self  Output: None | Increments the animal’s age and determines whether it is still alive. | |
| CheckIfDead Ⓕ | Input: Self  Output: Boolean | Whether the animal is dead or not. | |
| Inspect Ⓟ | Input: Self  Output: None | Prints out the animal’s current state. | |
| CheckIfKilledByOtherFactor Ⓕ | Input: Self  Output: Boolean | Determines whether the animal has been killed by another factor. | |
| \_CalculateRandomValue Ⓕ | Input: Self, base value (Integer), variability (Integer)  Output: Real | Calculates a random value. | |

|  |  |  |  |
| --- | --- | --- | --- |
| Fox – Methods | Description | |  |
| \_\_init\_\_ Ⓟ | Input: Self, variability (Integer)  Output: None | Constructor – creates a new instance of a fox. | |
| AdvanceGeneration Ⓟ | Input: Self, whether detail should be shown (Boolean)  Output: None | Determines whether the fox has died or by how much it ages. | |
| ResetFoodConsumed Ⓟ | Input: Self  Output: None | Resets this value to 0. | |
| ReproduceThisPeriod Ⓕ | Input: Self  Output: Boolean | Determines whether the fox should reproduce. | |
| GiveFood Ⓟ | Input: Self, number of food units (Integer)  Output: None | Adds the number of food units passed in to the food consumed. | |
| Inspect Ⓟ | Input: Self  Output: None | Prints out the fox’s current state (overrides method of same name in Animal). | |

|  |  |
| --- | --- |
| Genders Methods |  |
| The Genders class does not contain any functions or procedures. It is only used with rabbits, not foxes or animals. | |

|  |  |  |  |
| --- | --- | --- | --- |
| Rabbit – Methods | Description | |  |
| \_\_init\_\_ Ⓟ | Input: Self, variability (Integer), parents reproduction rate (Real)  Output: None | Constructor method to create a new instance of a rabbit. | |
| Inspect Ⓟ | Input: Self  Output: None | Print out the rabbit’s current state (overrides method of same name in Animal). | |
| IsFemale Ⓕ | Input: Self  Output: Boolean | Returns whether the rabbit is male or female. | |
| GetReproductionRate Ⓕ | Input: Self  Output: Reproduction rate (Real) | Returns the reproduction rate. | |

In addition to the functions and procedures found in the classes, there is also the main program. UML Activity