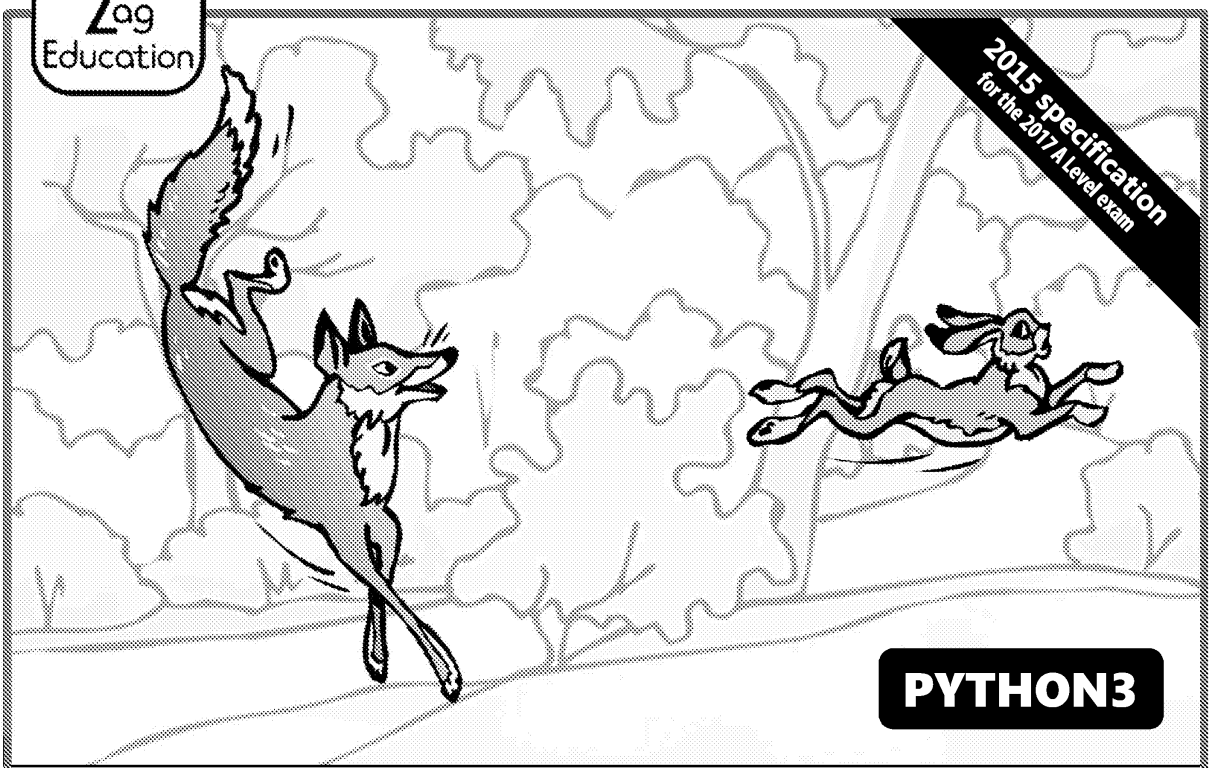
Zig Zag Education

Computer Science A Level | AQA | 7517



PYTHON3

AQA PAPER 1 EXAM RESOURCE PACK 2017

Rabbits and Foxes

for A Level AQA Computer Science

zigzageducation.co.uk

POD 7224

Contents

Teacher's Introduction

This pack is designed to help you support your students taking the A Level Computer Science Paper 1 examination. It is based on the 'Rabbits & Foxes' preliminary material (Python3) – for examination June 2017.

It consists of the following:

① **Pre-release Commentary** (for teachers)
A detailed overview of the skeleton program, describing all Python3 code elements and routines.

This section is designed to help you get to grips with the program, so that you can feel confident helping your students. This commentary is not designed to be given to students before they have explored the code for themselves, and if used in this way could lead to misconceptions of how the program works.

② **UML Diagram Activity**
A partially incomplete UML class diagram for students to complete while getting to grips with the skeleton program. Any missing operations and attributes must be added to the diagram. A completed version is provided in the solutions section at the back of the resource.

③ **Programming Theory Questions**
Theory questions test students' understanding of the 'Rabbits & Foxes' code, like Section C in the exam. These are provided in both write-on and non-write-on format.

④ **Programming Exercises**
Modification exercises put students' programming skills to the test, like Section D in the exam. An Electronic Answer Document (EAD) and the modified Python3 code are provided on the CD.

Answers and solutions for the UML Diagram activity, theory questions and programming exercises are provided from page 23 onwards. Note that for the programming exercises in particular, these are example solutions and you must use your discretion to award marks accordingly where there are valid alternative solutions.

The **Appendices** contains some additional resources, including:

- Further modifications worksheet: a template for brainstorming further enhancements to the skeleton program. This is suggested as a group activity, so that students (and the teacher) can share their ideas, thus increasing the likelihood of covering every area that will come up in the exam.
- Electronic Answer Document (EAD) printout: hard copy version of the file on CD (for reference).

Enter the URL **zzed.uk/7226** in your web browser to download a folder containing the following:

- **MODIFIED_PY3_CODE.txt** – text file containing the new and/or modified program code as shown in the mark scheme for section ④ (from page 26).
- **PAPER1_EAD.docx** – Electronic Answer Document for completing sections ③ and ④

RABBITS AND FOX

Description of the Program

The program is a simulation of rabbit population over time and how it is affecte

The world is represented by a grid in which each squa[illegible] ontain a rabbit wa[illegible] rabbits live) or a fox, or both. F designates a f[illegible]x, [illegible]nd [illegible]umber designates a rab[illegible] rabbits are in the warren).

The menu holds the f[illegible]g options:

- Ru[illegible]mulation with default settings
- Run [illegible]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 o[illegible] 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 fo[illegible] have eaten compared to what they need, and whether t[illegible]y have reproduced. If[illegible] have reproduced, the location of the new fox[illegible]s i[illegible] dis[illegible]yed at the bottom.

Rabbits and Foxes

Description of Program Classes

This program contains multiple classes used to simulate f[illegible] an [illegible]bits in their natural environmen
The classes have been listed below, along wit[illegible] bri[illegible]escription of their purpose.

Class	Description
Location	A class that creates an object corresponding to a location on the g
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 prog
Animal	An abstract class used for creating foxes and rabbits. It contains al
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 p[illegible]rti[illegible] la[illegible]s. For each of the classes abo

Location — Instance variables	T[illegible]	Description
self.Fox	Fox	This value is equal to None when the simulat This value will hold a Fox object, if there is a
self.Warren	Warren	This value is equal to None when the simulat This value will hold a Warren object, if there

Simulation — Instance variables	Type	Description
self.__ViewRabbits	String	Variable that should either have the value 'y'
self.__TimePeriod	Integer	Counter to store ho[illegible]any iterations of the s
self.__WarrenCount	Integer	Variab[illegible] co[illegible]s the number of warrens.
self.__FoxCount	Integer	[illegible]able that counts the number of foxes.
self.__ShowDetail	[illegible]	If this is true, more detail will be shown abou
self.__Landsca[illegible]	Integer	Value that stores the size of the Landscape (t
self.__Variabil[illegible]	Integer	Value that determines how differently the sir other variable values.
self.__FixedInitialLocations	Boolean	If True, the warrens and foxes will start in a f
self.__Landscape	List	The variable used to store the space in which

Warren — Instance variables	Type	Description
self.MAX_RABBITS_IN_WARREN	Integer	Value that stores the maximum number of ra
self.__RabbitCount	Integer	The value that stores the number of rabbits w
self.__PeriodsRun	Integer	This variable sto[illegible]y many periods have p
self.__AlreadySpread	Boolean	Boole[illegible]va[illegible]e[illegible]t to determine whether a r[illegible]e).
self.__Variability	[illegible]tege[illegible]	Value that determines how differently the sir other variable values.
self.__Rabbits	List	A list containing the rabbits that are currentl

Genders — Ins[illegible] variables
This class is slightly different from the others – the purpose of this class is to be able to state that an anim or 2, as meaning is given to the number.

Animal — Instance variables	Type	Description
self._NaturalLifespan	Integer	Integer value stating how long (in iteratio
self._ProbabilityOfDeathOtherCauses	Real	Real value used f alculating the chance
self._IsAlive	Boolean	Bo val hat states whether an anim
self._ID	Inte e	nteger value given to uniquely identify th
self._Age	I ger	Value used to store the age of an animal (
Animal._ID	*Integer*	*Value used to make sure that each new inst VARIABLE, shared by every instance of the c*

Fox — Instance variables	Type	Description
self.__DEFAULT_LIFE_SPAN	Integer	Value used for calculating the lifespan of the variability variable in the Simulation c
self.__DEFAULT_PROBABILITY_DEATH_OTHER_CAUSES	Real	Probability used for calculating the chanc in the Animal class using the variability va
self.__FoodUnitsNeeded	Integer	Number of food units needed to stop the
self.__FoodUnitsConsumedThisPeriod	Integer	Number of food units that have been cons

Rabbit — Instance variables	Type	De ion
self.__DEFAULT_LIFE_SPAN	Inte e	alue used for calculating the lifespan of the variability variable in the Simulation c
self.__DEFAULT_PROBABILIT HER_CAUSES	Real	Probability used for calculating the chanc in the Animal class using the variability va
self.__Reprod e	Real	Probability used for calculating the chanc
self.__Gender	Genders	The gender of the rabbit. Equal to either G

Description of Class Methods

Along with class variables, each class has a number of methods unique to that class. For each class, its f

Location — Methods	Description
__init__ Ⓟ	Input: Self Output: Nothing

Note: In each of the classes the variabl[illegible] is p[illegible]d into every function. This refers to the instance o

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 sim 1. Creates a Landscap 2. Adds fox 3. Draws th 4. Starts the inspect a
__InputCoordinate Ⓕ	Input: Self, Coordinate name ('x' or 'y') Output: Coordinate (Integer)	Asks the user (x or y). Returns an in
__AdvanceTimePeriod Ⓟ	Input: Self Output: None	Updates the s 1. For each a. If ther they a b. If the create c. The w d. If the 2. For each a. If ther b. Check i. If i ii. If i c. Reset 3. If new fox

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 la 1. If the loca fixed loca 2. Otherwise determine
__CreateNewWarren Ⓟ	Input: Self Output: None	Creates a new 1. Find a spo 2. Create a n
__CreateNew	Input: Self Output: None	Creates a new 1. Find a spo 2. Create a n
__FoxesEatRabbitsInWarren Ⓟ	Input: Self, warren's x-coordinate (Integer), warren's y-coordinate (Integer) Output: None	Function that 1. For each a. If there a warr b. OTHE away c. OTHE
__DistanceBetween Ⓕ	Input: Self, two sets of x- and y-coordinates Output: Distance between the points (Real)	Calculates the
__DrawLandscape Ⓟ	Input: Self Output: None	Draws the lan It checks each

Warren — Methods	Description	
__init__ Ⓟ	Input: Self, variability (Integer), number of foxes in warren (Integer) Output: None	Creates a ne 1. Creates 2. If the nu initial n variabil 3. It adds
__CalculateRandomValue Ⓕ	Input: Self, base value (Integer), variability (Integer) Output: Random value (Real)	Provides a r variability is

Warren — Methods (cont.)	Description	
GetRabbitCount Ⓕ	Input: Self Output: Number of rabbits in warren (Integer)	Returns the
NeedToCreateNewWarren Ⓕ	Input: Self Output: Whether a [illegible] warren needs to be created (Boolean)	1. Checks [illegible] split up 2. If this is
WarrenHasDiedOut Ⓕ	Input: Self Output: Whether a warren is empty or not (Boolean)	This functio 1. If there 2. Otherwi
AdvanceGene	Input: Self, whether you should show detail (Boolean) Output: None	Advances th 1. If there 2. If there 3. If there males, t 4. Otherwi rabbits
EatRabbits Ⓕ	Input: Self, number of rabbits that need to be eaten (Integer) Output: Updated number of rabbits to be eaten (Integer)	Removes a 1. Finds a 2. Remove 3. Repeats 4. Compre
__KillByOtherFactors Ⓟ	Input: Self, whether you should show d[illegible] (Boolean) Output: None	Kills rabbits randomly d 1. Goes th 2. Checks 3. Remove 4. Compre
__AgeRabbits Ⓟ	Input: Self, whether you should show detail (Boolean) Output: None	Makes each 1. Goes th 2. Determi a. If the and

Warren — Methods (cont.)	Description	
__MateRabbits Ⓟ	Input: Self, whether you should show detail (Boolean) Output: None	Func 1. G 2. If a b c
__CompressRabbitList Ⓟ	Input: Self, number of dead rabbits (Integer) Output: None	Shifts
__ContainsMales Ⓟ	Input: Self Output: Whether a warren contains males (Boolean)	Chec 1. It 2. If
Inspect Ⓟ	Input: Self Output: None	Prints
ListRabbits Ⓟ	Input: Self Output: None	Prints

Animal — Methods	Description	
__init__ Ⓟ	Input: Self, average lifespan (Integer), average probability of dying from other causes (Real), variability (Integer) Output: None	Cons
CalculateNewAge Ⓟ	Input: Self Output: None	Incre
CheckIfDead Ⓕ	Input: Self Output: Boolean	Whet
Inspect Ⓟ	Input: Self Output: None	Prints
CheckIfKilled[illegible]act[illegible] Ⓕ	Input: Self Output: Boolean	Deter
_CalculateRandomValue Ⓕ	Input: Self, base value (Integer), variability (Integer) Output: Real	Calcu

Fox — Methods	Description	
__init__ (P)	Input: Self, variability (Integer) Output: None	Constructor
AdvanceGeneration (P)	Input: Self, whether detail should be (Boolean) Output: None	Determines
ResetFoodConsumed (P)	Input: Output: None	Resets this
ReproduceThisPeriod (F)	Input: Self Output: Boolean	Determines
GiveFood (P)	Input: Self, number of food units (Integer) Output: None	Adds the nu
Inspect (P)	Input: Self Output: None	Prints out th

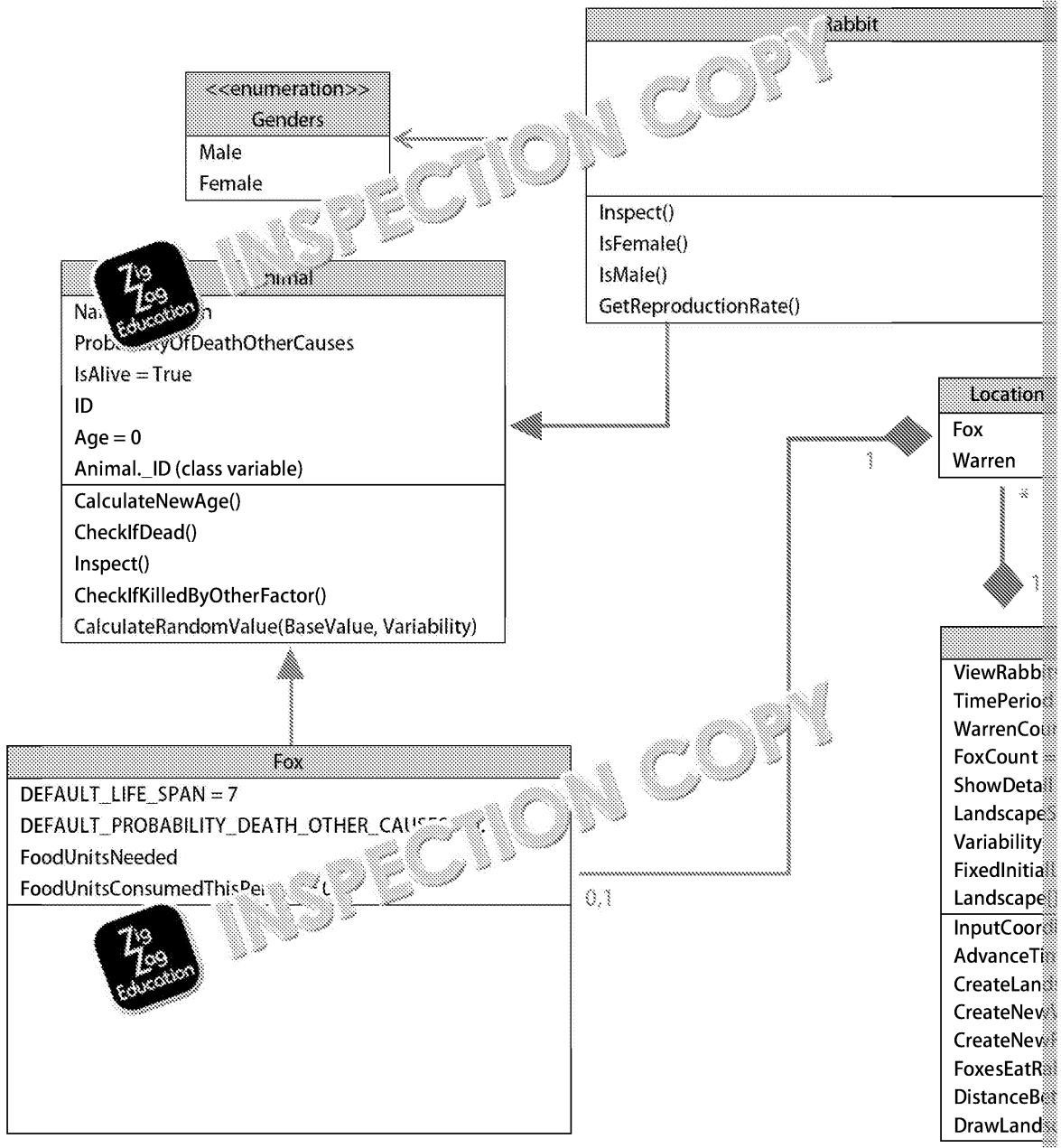
Genders Methods
The Genders class does not contain any functions or procedures. It is only used with rabbits, not foxes or

Rabbit — Methods	Description	
__init__ (P)	Input: Self, variability (Integer), reproduction rate (Real) Output: None	Constructor
Inspect (P)	Input: Output: None	Print out the
IsFemale (F)	Input: Self Output: Boolean	Returns whe
GetReproduct (F)	Input: Self Output: Reproduction rate (Real)	Returns the

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

Rabbits and Foxes

Add the missing operations and attributes to the UML diagram



Programming Theory Questio

These questions refer to the Preliminary Material and require you to loa
but do not require any additional programming.

1. Give an example of instantiation from the skeleton program.

.....

2. State the name of an identifier(s) for the following:

   a. A list variable

   .....

   b. A subclass

   .....

   c. A parent class

   .....

   d. A constant that stores a float

   .....

   e. A class variable

   .....

   f. An accessor method

   .....

   g. A mutator method

   .....

   h. A variable that is used to store a whole number.

   .....

3 a. Two classes that have a [illegible]on aggregation relationship.

   .....

   b. Why is Warren to Rabbit not an example of association aggregation?

   .....

   .....

4. Are there any examples of polymorphism in the skeleton code?

.....

5. State the name of an identifier for a procedure or function that is overridde

6. Look at the EatRabbits subroutine in the Warren class in the skeleton progr
Why does the generation of a random rabbit need to be inside a repetition

7. Look at the Warren class. Why has a named const[illegible]n used instead of

8. State the name of an identifier for an enumerated data type.

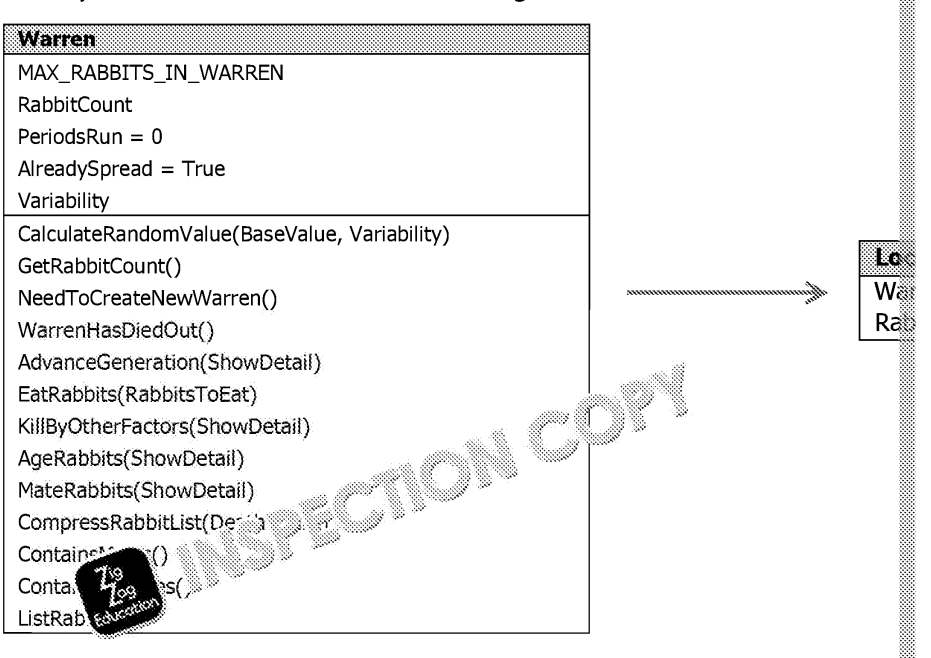
9. How could the Fox class be changed to make the foxes live longer?

10. What is the purpose of the variable AlreadySpread in the Warren class and h

11. What is the purpose of the method CompressRabbitList?

12. Why is [illegible]ess[illegible] to store the gender of the rabbits?

13. Identify six errors in the section of UML diagram below.



1 .....

2 .....

3 .....

4 .....

5 .....

6 .....

14. Create a UML diagram to show the relationship between rabbits, foxes and . All variables and methods must be shown.

15. What conditions are needed for a new warren to be created?

.....

.....

.....

16. State the name of an identifier for a variable that holds:

a. An integer value

.....

b. A string value

.....

c. A real value

.....

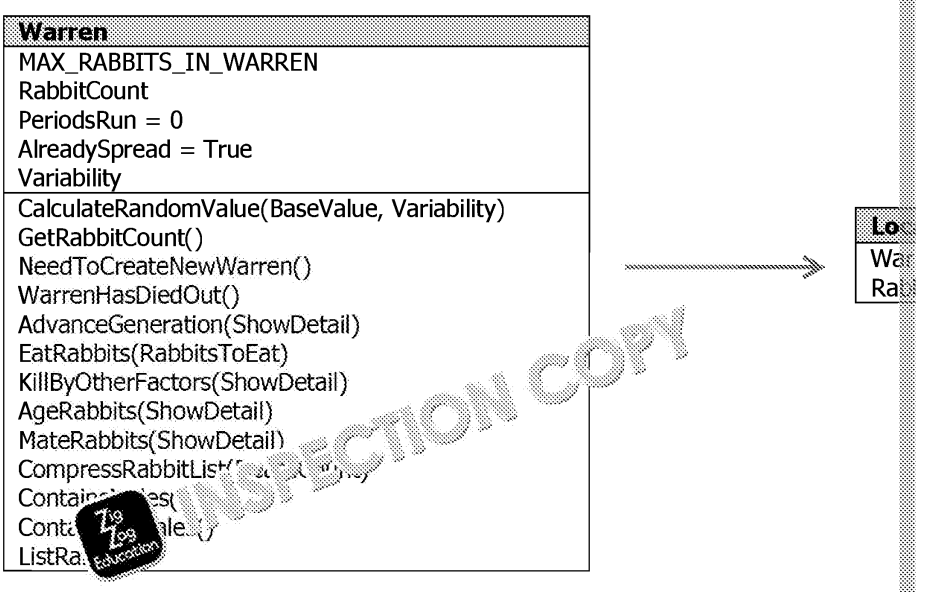
d. A Boolean value

.....

Programming Theory Questio

These questions refer to the Preliminary Material and require you to load
but do not require any additional programming.

1. Give an example of instantiation from the skeleton program.
2. State the name of an identifier(s) for the following:
   a. A list variable [1 mark]
   b. A subclass [1 mark]
   c. A parent class [1 mark]
   d. A constant that stor
   e. A class variable [1 mark]
   f. An accessor method
   g. A mutator method [1 mark]
   h. A variable used to st
3. a. Name two classes that have a c... ition aggregation relationship.
   b. Why is Warren to Ra... no... a example of association aggregation?
4. Are ther... y e... ... of polymorphism in the skeleton code?
5. State t... e of an identifier for a procedure or function that is overridde
6. Look at the EatRabbits subroutine in the Warren class in the skeleton progr
   Why does the generation of a random rabbit needs to be inside a repetition
7. Look at the Warren class. Why has a named constant been used instead of
8. State the name of an identifier for an enumerated data type.
9. How could the Fox class be changed to make the foxes live longer?
10. What is the purpose of the variable AlreadySpread in the Warren class and h
11. What is the purpose of the method CompressRabbitList?
12. Why is it necessary to store the gender of the rabbits?
13. Identify six errors in the section of UML diagram below.



14. Create a UML diagram to show the relationship between rabbits, foxes and
    All variables and methods must be shown.
15. What conditions are needed for a new warren to be created?
16. State the name of an identifier for a variable that holds the following values:
    a. An integer value [1 mark]
    b. A string value [1 ma
    b. A real value [1 mark]
    d. A Boolean value [1 m

Programming Exercises

The following require you to open the skeleton program and make modifications. Th
and illustrate how you should prepare your answer

Question 1

This task refers to the Main procedure

Alter how the menu displays so that:

- There is a new option '3. Rabbit Paradise'
- The 'Exit' option is now numbered 4

Evidence you need to provid

- Copy of your a
- Scree it executing

Question 2

This task refers to the Main procedure

Code option 3 so that when it is selected the simulation is run with the following

- A landscape size of 20
- 20 warrens
- 0 foxes
- Locations are not fixed
- Variability is 1

Evidence you need to provide:

- Copy of your amended code
- Screen capture of it executing

Question 3

This task refers to the Simulation clas

Add an option to the q
'0. Advanc ods hiding detail'

Code this option.

Evidence you need to provide:

- Copy of your amended code
- Screen capture of it executing

Question 4

This task refers to the Rabbit class

Change *Rabbit's* constructor so that it receives in an extra variable that will allow rabbits to be altered. Use the identifier *genderRatio* for the new variable.

Set the default value to 50 so that the constructor can be called without specifying

Evidence you need to provide:

- Copy of your amended code

Question 5

This task re... the Fox class

Add *Gender* to the *Fox* class.

Make the ratio of males to females 1 : 2.

Alter the *Inspect* method so that the gender of a fox is reported.

Change *ReproduceThisPeriod* so that only female foxes can reproduce.

Evidence you need to provide:

- Copy of your amended code
- Screen capture of an inspection of the Fox at 2,10

Question 6

A new subclass must be created for this task, as well as changes to the createLand... in Simulation

Create a subclass of *Warren* called a *GiantWarren*.

- A giant warren has a maximum capacity of 20... and can always spawn a ne... already.
- A giant warren h... default rabbit.
- Add ... warren to the default game at position (11,4) with a starting p...

Evidence you need to provide:

- Copy of your amended code
- Screen capture of a default simulation executing

Question 7

A new subclass must be created for this task, as well as changes to the Location c
createLandscapeAndAnimals, drawLandscape and AdvanceTimePeriod procedu

Create a *Den* class that can exist in a location.

- The den will spawn 1 new fox per 3 time periods.
- The den will store how many foxes it has created as a private instance var
- The fox will appear at a random position.
- If there is already a fox in this location, it is replaced by the new fox.
- Position the den at (2,3) in a default game.
- The den will be displayed on the map as a D plus the number of foxes it ha

Evidence you need to provide:

- Copy of your amended code
- Screen capture at time period 3 of a default game running

Question 8

This tasks refers to the Fox class

The average age of death of foxes needs to be known.

- Create a class variable called *_TotalDeadFoxes* to store the total foxes wh
- Create a class variable called *_TotalFoxAge* to store the sum of the ages o
- When a fox dies, the *_TotalDeadFoxes* needs to be incremented and its ag
- An accessor method in Fox called *getLifeExpect* will return the average ag
- A message stating 'The average life expectancy of a fox stands at X' shoul each time it is displayed.
- If no foxes have yet died, the default lifespan should be returned.

Evidence you need to provide:

- Copy of your amended code
- Screen capture of default sim ime period 0
- Screen capture of d ation at time period 4

Question 9

This task refers to the Simulation class

Create a menu option in the simulation: '6. Find biggest warren'.

The coordinates of the biggest warren will then be displayed: 'Biggest warren at

Create a new procedure called findBiggest to search the warren array in a linear
message.

Evidence you need to provide:
- Copy of your amended code
- Screen capture of option 6 running

Question 10

This task refers to the Rabbit class

Make rabbit death probability go up by 10% with age.

Evidence you need to provide:
- Copy of your amended code
- Screen capture of a warren inspected (showing individual rabbits) at time p

Question 11

This task requires changes to Warren, Rabbit and Simulation classes

Create a menu option: '7. Inspect all rabbits'.

It should display a list of all rabbits in all warrens, showing their details.

The rabbits must be shown in age order, oldest to you

Bubble-sort the rabbits after adding the
one list.

An accessor method to
bbits list out of a warren must be created.

An accesso
d to get a rabbit's age out of a rabbit must be created.

Evidence you need to provide:
- Copy of your amended code
- Screen capture of option 7 running

Question 12

This task requires changes to Simulation as well as creation of new classes

Beneath the warrens are secret tunnels connecting them. Not every warren is c
warren is connected to more than two other warrens. This data must be stored i

WarrenGraph
-nodes[]
+addNode(theNode) +adjList()
Node
-selfX -selfY -leftBranch -leftBranchY -rightBranchX -rightBranchY
+getCoord(l/r/s)

Each warren connected to another has the coordinates of itself and its connecti
WarrenGraph contains a list of all nodes. The procedure *getCoord* returns the x-
based on arguments (l)eft, (r)ight and (s)elf.

The *adjList* method displays an adjacency list and should be executed by a new o

The following data should be used to initially populate the graph.

self	left	right
(1,1)	(2,8)	(9,7)
(2,8)	(13,4)	(1,1)
(9,7)	(1,1)	(13,4)
(13,4)	(9,7)	(2,8)

Evidence you need to pr
- Copy o a code
- Screen e of option 8 running

Question 13

This task requires changes to Simulation and WarrenGraph

Create a new procedure in *WarrenGraph* called *adjMatrix*. It will display the gra
will be executed by '9. Display adjacency matrix'. A 1 should be used to indicate

Evidence you need to provide:
- Copy of your amended code
- Screen capture of option 9 running

Question 14

This task re changes to WarrenGraph

Amend your solution for task 13 to replace the '1' with the actual distance betwe

Use Pythagoras' theorem to calculate the distance between the two points.

Distances should be rounded to 1 decimal place.

Evidence you need to provide:
- Copy of your amended code for adjMatrix
- Screen shot of option 9 running

Question 15

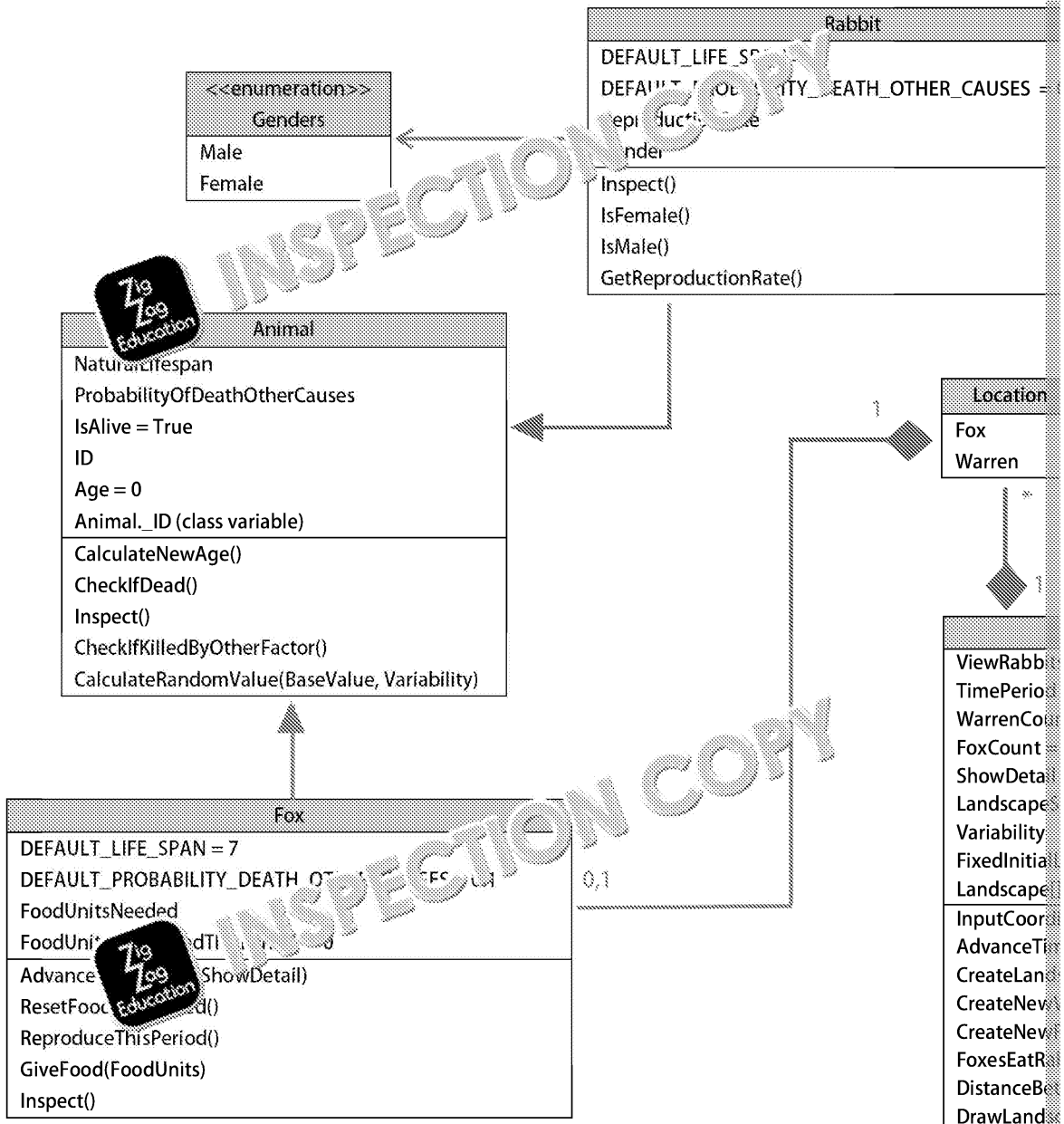
This task requires changes to Simulation and WarrenGraph

Create a procedure to find whether there is a route between two warrens.

It will be executed by Option 10.

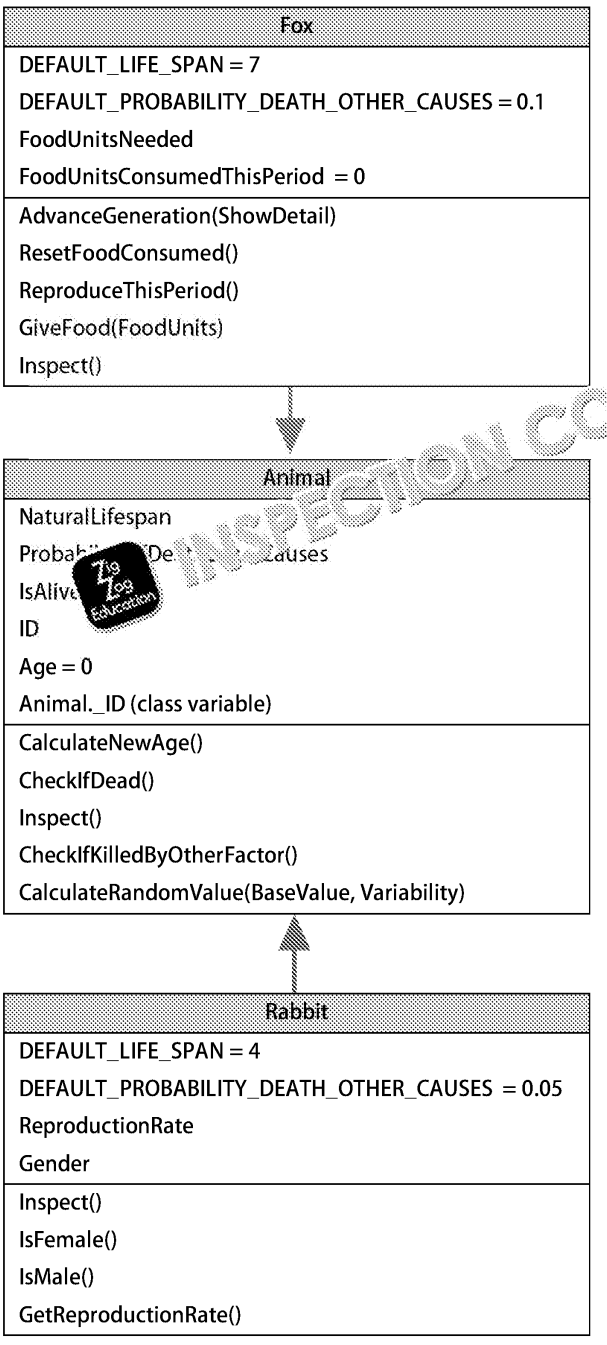
Evidence you need to provide:
- Copy of your code
- Screen capture of o ning showing no route between warrens
- Screen re on 10 running showing a route between warrens

Rabbits and Foxes

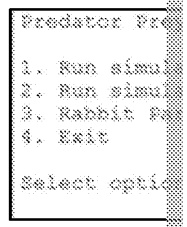
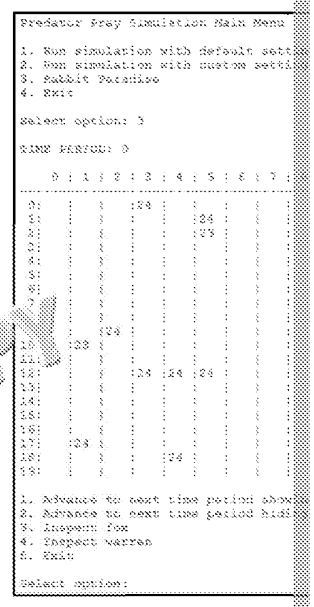


Programming Theory Questions (Suggested Answers)

Q	Marking Guidance
1	Sim = Simulation(LandscapeSize, InitialWarrenCount, InitialFoxCount, Variabili self.__Landscape[1][1].Warren = Warren(self.__Variability, 38) self.__Landscape[2][10].Fox = Fox(self.__Variability) self.__Rabbits[r] = Rabbit(self.__Variability)
2a	Landscape / LandscapeRow / Rabbits
2b	Fox/Rabbit
2c	Animal
2d	DEFAULT_PROBABILITY_DEATH_OTHER_CAUSES REPRODUCTION_PROBABILITY DEFAULT_PROBABILITY_DEATH_OTHER_CAUSES
2e	Animal.ID
2f	Any pr…es with Get at the start of the identifier
2g	Any procedures with Set at the start of the identifier
2h	TimePeriod / WarrenCount / FoxCount / NewFoxCount / PeriodsRun / RabbitCo…
3a	Location to Fox or Location to Warren or Warren to Rabbit (any correct pair for …
3b	Rabbits objects cannot exist unless they have an associated Warren
4	There are none
5	Inspect
6	To keep selecting a different rabbit at random until the required number of rab…
7	Makes the program code easier to understand / improves readability Makes it easier to update the program Makes it easier to change the maximum number of rabbits in a warren ANY 2 UP TO A MAX OF 2
8	Gender
9	The DEFAULT_LIFE_SPAN needs to be increased from 7
10	It stores whether or not the warren has already created a new warren It stops the warren creating more than 1 new warren It is set to False by default It is set to True when a new warren is created
11	When rabbits are eaten or die they are removed from random positions in the r… Compressing rabbits list removes the gaps
12	Only female rabbits can reproduce This therefore affects the calcul…fo…w many new baby rabbits are born
13	Type and direction o…ing Warren d… n… …from location Locati… so…iated to Warren Locatio…es warrens and/or foxes Location cannot store rabbits AlreadySpread should be set to False as default The constant MAX_RABBITS_IN_WARREN has a default value of 99 Warren should contain a list of rabbits The inspect() procedure is missing There is no function called ContainsFemales() in Warren ANY 6 FOR 6 marks

Q	Marking Guidance
14	 1 mark for correct class name (×3) 1 mark for correct instance variables (×3) 1 mark for correct methods (×3) 1 mark for correct inheritance arr[illegible] (×1)
15	The number of rabbi[illegible]ren must have reached the maximum allowed The wa[illegible]ar [illegible]e already created a new warren
16	a: Time[illegible], WarrenCount, FoxCount, MenuOption b: ViewRabbits c: Dist d: ShowDetail Or any other suitable answer DO NOT ACCEPT CONSTANTS (THEIR IDENTIFIERS ARE MADE UP OF BLOCK C

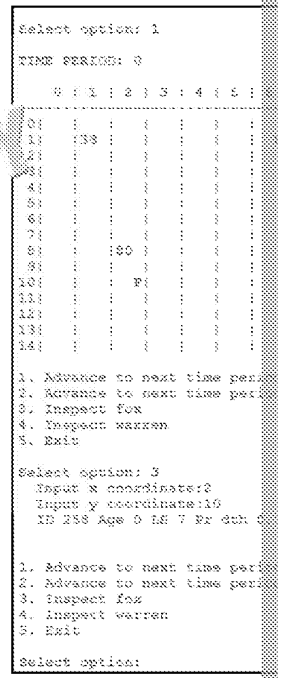
Programming Exercises (Solutions)

Q	Example Solution
1	def Main(): MenuOption = 0 while MenuOption != 4: print("Predator Prey Simulation Main Menu") print() print("1. Run simulation with default settings") print("2. Run simulation with custom settings") print("3. Rabbit Paradise") print("4. Exit") print() MenuOption = int(input("Select option: ")) 
2	MenuOption = int(input("Select option: ")) if MenuOption == 1 or MenuOption == 2 or MenuOption == 3: if MenuOption == 1: LandscapeSize = 15 InitialWarrenCount = 5 InitialFoxCount = 5 Variability = 0 FixedInitialLocations = True elif MenuOption == 3: LandscapeSize = 20 InitialWarrenCount = 20 InitialFoxCount = 0 Variability = 1 FixedInitialLocations = False 

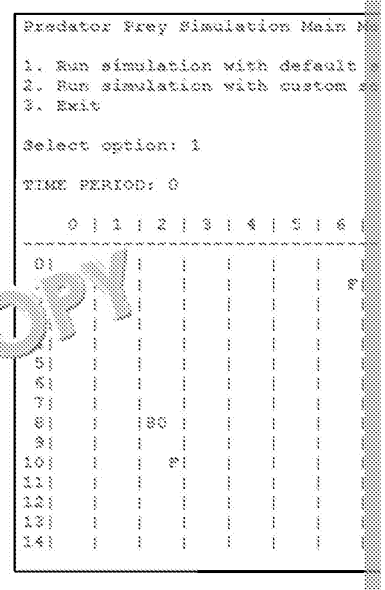
Q	Example Solution
3	while (self.__WarrenCount > 0 or self.__FoxCount > 0) and MenuOption != 5: print() **print("0. Advance 10 time periods hiding detail")** ... ... MenuOption = int(input("Select option: ")) **if MenuOption == 0:** **self.__TimePeriod += 10** **self.__ShowDetail = False** **for i in range(10):** **self.__AdvanceTimePeriod()**
4	def __init__(self, Variability, ParentsReproductionRate = 1.2, **genderRatio = 50**): self.__DEFAULT_LIFE_SPAN = 4 .. .. if random.randint(0, 100) < **genderRatio**: self.__Gender = Genders.Male ...
5	class Fox(Animal): ... ... self.__FoodUnitsConsumedThisPeriod = 0 **if (random.randint(0, 3) < 2):** **self.__Gender = Genders.Female** **else:** **self.__Gender = Genders.Male** def R[illegible]ce [illegible]self): **if [illegible]de [illegible] Genders.Male:** **r[illegible]e** **else:** REPRODUCTION_PROBABILITY = 0.25 ...

5 (cont.)

```
def Inspect(self):
  super(Fox, self).Inspect()
  print("Food needed", self.__FoodUnitsNeeded, "", end = "")
  print("Food eaten", self.__FoodUnitsConsumedThisPeriod, "", end = "")
  if self.__Gender == Genders.Female:
    print("Gender Female")
  else:
    print("Gender Male")
  print()
```



Q	Example Solution
6	**class GiantWarren(Warren):** def __init__(self, Variability, RabbitCount = **50**): self.__MAX_RABBITS_IN_WARREN = **200** **super()** self.__RabbitCount = RabbitCount ... def __CreateLandscape(self, InitialWarrenCount, InitialFoxCount, FixedInitialLocations): ... if FixedInitialLocations: ... self.__Landscape[10][3].Warren = Warren(self.__Variability, 52) **self.__Landscape[11][4].Warren = GiantWarren(self.__Variability, 115)** self.__WarrenCount = **6** self.__Landscape[2][10].Fox = Fox(self.__Variability) ***Plus:*** *Warren instance variables need to be protected and not private (done by changing __ to _):* self._MAX_RABBITS_IN_WARREN = 99 self._RabbitCount = RabbitCount self._PeriodsRun = 0 self._AlreadySpread = False self._Variability = Variability self._Rabbits = []



Q	Example Solution
7	**class Den:** **def __init__(self):** **self.__foxes = 0** **def spawn(self):** **self.__foxes += 1** **return Fox(50)** **def getSymbol(self):** **return ("D" + str(self.__fo**
	clas def _(self): self.Fox = None self.Warren = None **self.Den = None**
	def __DrawLandscape(self): ... for x in range (0, self.__LandscapeSize): if not self.__Landscape[x][y].Warren is None: if self.__Landscape[x][y].Warren.GetRabbitCount() < 10: print(" ", end = "") print(self.__Landscape[x][y].Warren.GetRabbitCount(else: print(" ", end = "") if not self.__Landscape[x][print("F", end = "") **pe[x][y].Den is None:** **f.__Landscape[x][y].Den.getSymbol(), end = "")** print(" ", end = "") print("|", end = "") print()

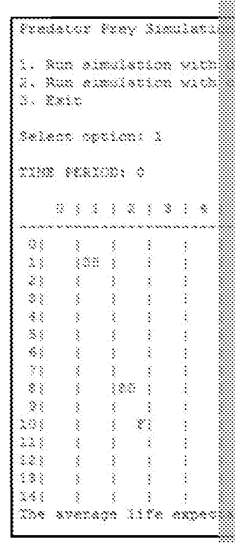
7 (cont.)

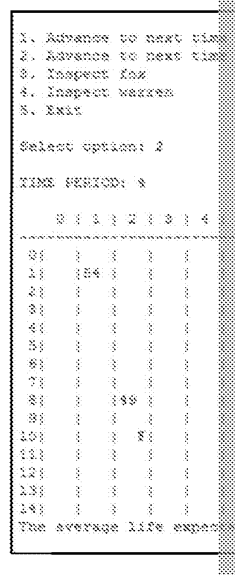
```
def __CreateLandscapeAndAnimals(self, InitialWarrenCount, InitialFoxCount, FixedInitialLocation
  ...
  if FixedInitialLocations:
    self.__Landscape[1][1].Warren = Warren(self.__Variability, 38)
    self.__Landscape[2][8].Warren = Warren(self.__Variability, 80)
    self.__Landscape[9][7].Warren = Warren(self.__Variabili
    self.__Landscape[10][3].Warren = Warren(s
    self.__Landscape[13][4].Warren = W
    self.__Landscape[11][4].W
    self._ Warren
    s
    s
    se
    self.__Landscape[11][13].Fox = Fox(self.__Variability)
    self.__Landscape[12][4].Fox = Fox(self.__Variability)
    self.__FoxCount = 5
    self.__Landscape[2][3].Den = Den()
  else:
    ...
```

```
def __AdvanceTimePeriod(self):
  NewFoxCount = 0
  if (self.__TimePeriod % 3) == 0:
    x = random.randint(0, self.__LandscapeSize - 1)
    y = random.randint(0, self.__LandscapeSiz
    self.__Landscape[x][y].Fox = De
    print("Fox spawned at
  if self.__ShowD
  fo
```

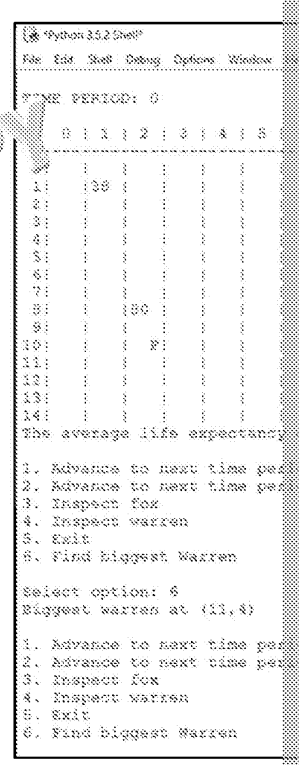
```
Select option: 2
Fox spawned at 13,3

TIME PERIOD: 3

    0 | 1 | 2 | 3 | 4 |

 0|   |   |   |   |   |
 1|   |42 |   |   |   |
 2|   |   | 80|   |   |
 3|   |   | 038   |   |
 4|   |   |   |   |   |
 5|   |   |   |   |   |
 6|   |   |   |   |   |
 7|   |   |   |   |   |
 8|   |136|   |   |   |
 9|   |   |   |   |   |
10|   |   |   |   | 90|
11|   |   |   |   |   |
12|   |   |   |   |   |
13|   |   |   |   |   |
14|   |   |   |   |   |

1. Advance to next time
2. Advance to next time
3. Inspect fox
4. Inspect warren
5. Exit
```

Q	Example Solution
8	class Fox(Animal): **_TotalDeadFoxes = 0** **_TotalFoxAge = 0** def __init__(self, Variability): ... ... if (random.randint(0, 3) < 2) self.__Gender = G... el... ...ler = Genders.Male **def getLifeExpect(self):** **if Fox._TotalDeadFoxes > 0:** **return float(Fox._TotalFoxAge/Fox._TotalDeadFoxes)** **else:** **return self.__DEFAULT_LIFE_SPAN** --- def __DrawLandscape(self): ... ... if not self.__Landscape[x][y].Fox is None: print("F", end = "") lifeExpect = self.__Landscape[x][y].Fox.getL...pec... else: print(" ", end = "") print("|", end = "") ... **pr...verage life expectancy of a fox stands at " + str(lifeExpect))**

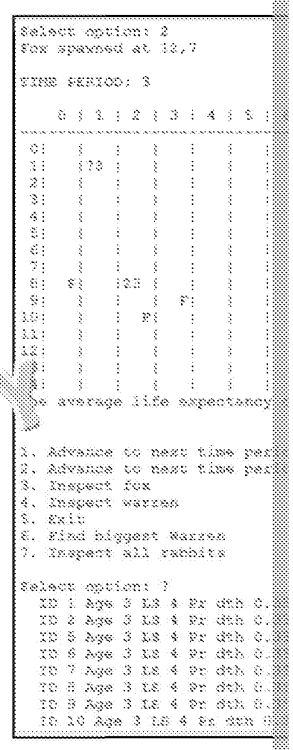




Q	Example Solution
9	class Simulation: ... ... print("5. Exit") **print("6. Find biggest warren")** print() ... if not self.__Landscape[x][y].Warren is None: self.__Landscape[x][y].Warren.Inspect() ViewRabbits = input("View individual rabbits (y/n)? ") if ViewRabbits == "y": self.__Landscape[x][y].Warren.ListRabbits() **if MenuOption == 6:** **self.findBiggest()** input() **def findBiggest(self):** **biggestX = -1** **biggestY = -1** **biggestSize = -1** **for x in range (0, self.__LandscapeSize):** **for y in range (0, self.__LandscapeSize):** **if not self.__Landscape[x][y].Warren is None:** **if biggestSize < self.__Landscape[x][y].Warren.GetRabbitCount():** **biggestSize = self.__Landscape[x][y].Warren.GetRabbitCount()** **biggestX = x** **biggestY = y** **print("Biggest warren at (" + str(biggestX) +"," + str(biggestY) + ")")**



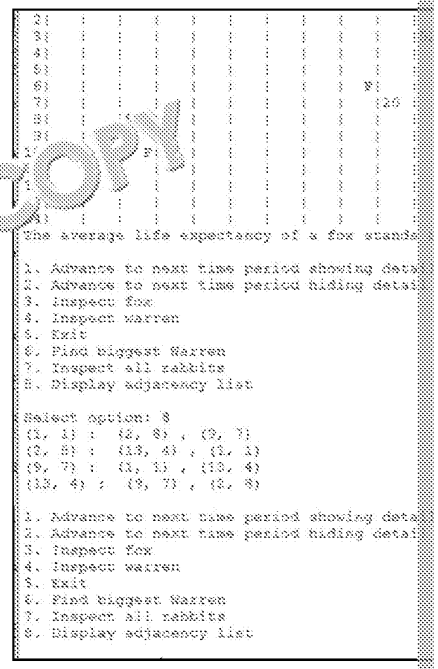
Q	Example Solution
10	def CalculateNewAge(self): self._Age += 1 if self._Age >= self._NaturalLifespan: self._IsAlive = False **self._ProbabilityOfDeathOtherCauses += 0.1**
11	***Chan... Warren:*** **def getRabbits(self):** **return self._Rabbits** ***Changes to class Rabbit:*** **def getAge(self):** **return self._Age** class Simulation: ... ... print("6. Find biggest Warren") **print("7. Inspect all rabbits")** print() ... ...ption = ...iggest() **if ...option == 7:** **self.sortAndListRabbits()** input()

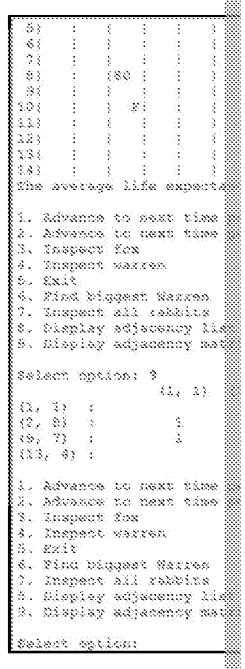
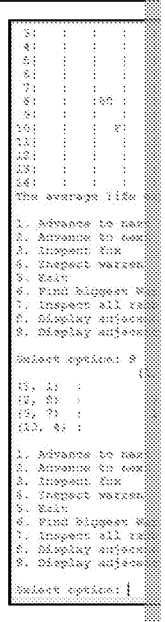


```
def sortAndListRabbits(self):
    #create a list to store all rabbits
    allRabbits = []
    theRabbits = []
    #get all the rabbits from all the warrens and add to the list
    for x in range (0, self.__LandscapeSize):
        for y in range (0, self.__LandscapeSize):
            if not self.__Landscape[x][y].Warren is None:
                allRabbits.extend(self.__Landscape[x][y].Warren.getRabbits())
    #remove "none" values
    for x in range(0, len(allRabbits)):
        if allRabbits[x] is None:
            allRabbits.append(allRabbits[x])
    #bubble sort the rabbits list
    for passnum in range(len(theRabbits)-1,0,-1):
        for i in range(passnum):
            if theRabbits[i].getAge() < theRabbits[i+1].getAge():
                temp = theRabbits[i]
                theRabbits[i] = theRabbits[i+1]
                theRabbits[i+1] = temp
    #display all the rabbits
    for x in range(len(theRabbits)):
        theRabbits[x].Inspect()
```

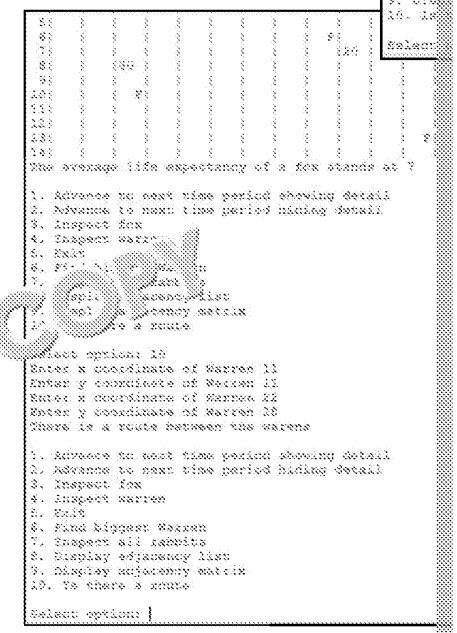
Q	Example Solution
12	class Simulation: ... ... self._CreateLandscapeAndAnimals(InitialWarrenCount, InitialF...n, self._FixedInitialLoca **self._Landscape = []self._WarrenGraph = WarrenGraph()** **theNode = Node(1,1,2,8,9,7)** **self._WarrenGraph.addNode(the...** **theNode = Node(2,8,13,...** **self._Warren...de(theNode)** **th...Node(1,2,1,13,4)** **se...nGraph.addNode(theNode)** **ther...e = Node(13,4,9,7,2,8)** **self._WarrenGraph.addNode(theNode)** self._DrawLandscape() ... ... print("7. Inspect all rabbits") **print("8. Display adjacency list")** print() MenuOption = int(input("Select option: ")) **if MenuOption == 8:** **self._WarrenGraph.adjList()** **class Node:** **def __init__(self, selfX, selfY, leftBranch...Br...ightBranchX, rightBranchY):** **self.selfX = selfX** **self.selfY = selfY** **sel...nc...anchX** **s...ch... = leftBranchY** **sel...anchX = rightBranchX** **self.rightBranchY = rightBranchY**

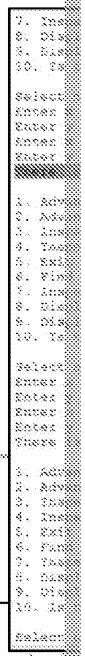
Q	Example Solution
12 (cont.)	def getCoord(self, arg): X = -1 Y = -1 if arg == "s": X = self.selfX Y = self.selfY if arg == "l": X = self.leftBranchX Y = self.leftBranchY if ar[illegible]": [illegible]htBranchX Y [illegible]htBranchY return (X,Y) class WarrenGraph: def __init__(self): self.__nodes = [] def addNode(self, theNode): self.__nodes.append(theNode) def adjList(self): for Node in self.__nodes: print(Node.getCoord("s") ,": ", Node.getCoord("l") ,",", Node.getCoord("r"))
13	class Simulation: ... ... print("8. Display adjacency [illegible]") print("9. Display [illegible]") pri[illegible] [illegible]on = int(input("Select option: ")) if [illegible]tion == 9: self.__WarrenGraph.adjMatrix()



Q	Example Solution
13 (cont.)	*Added to class WarrenGraph:* `def adjMatrix(self):` `  #column heading` `  headingList = []` `  heading = "\t"` `  for Node in self.__nodes:` `  headingList.append(Node.getCo...` `  heading += "\t" + str(No... or...),` `  print(heading)` `  #r...` `  fo... sel.__nodes:` `  ro...` `  row += (str(Node.getCoord("s")) + "\t:\t")` `  for item in headingList:` `  if (Node.getCoord("l") == item) or (Node.getCoord("r") == item):` `  row += " 1"` `  row += "\t"` `  print(row)` 
14	def adjMatrix(self): ... ... for item in headingList: if (Node.getCoord("l") == item) or (Node.getCoord("r") == item): x1, y1 = Node.getCoord("s") if (Node.getCoord("l") == item): x2, y2 = Node.getCoord("l") else: x2, y2 = Node ... di... = ... ow(x1 - x2, 2) + pow(y1 - y2, 2))) ... {round(distance,1)} print(row) 

Q	Example Solution
15	class Simulation: ... ... print("9. Display adjacency matrix") **print("10. Is there a route")** print() MenuOption = int(input("Select ... **if MenuOption == 10:** **self.__Warren... ...ute()** ... **def isRoute(self):** **#get coordinates of warrens** **startX = int(input("Enter x coordinate of Warren 1"))** **startY = int(input("Enter y coordinate of Warren 1"))** **finishX = int(input("Enter x coordinate of Warren 2"))** **finishY = int(input("Enter y coordinate of Warren 2"))** **route = False** **#find start** **for Node in self.__nodes:** **checkX, checkY = Node.getCoord("s")** **if (checkX == startX) and (checkY == startY):** **checkX, checkY = Node.getCoord("l")** **if (checkX == finishX) and (checkY == finishY):** **route = True** **checkX, checkY = Node.getCoord("r")** **if (checkX == finishX) and (checkY == finishY):** **route = True** **if route == True:** **print("There is a route between the warrens")** **else:** **print("There is no route between the warrens")**





Ideas for modifications	How to implement them

Name	

ZigZag Education supporting

A Level AQA Computer Science Pap

Summer 2017: RABBITS AND FOXE

Electr Answer Document (EAD)

Instructions

- Enter your name in the box at the top of this page
- Answer **all** questions by entering your answers into this document
- Remember to **save** this document regularly
- Save and print this document and any additional pages

- Answer **all** questions
- The marks available for each question are shown in brackets

- You will need:
  - access to a computer
  - access to a printer
  - access to appropriate software
  - electronic copies of the required skeleton code
  - EAD (Electronic Answer Document)

Total marks:

Programming Theory Question

Answer all questions.
Remember to save this document regularly.

Q		Answer
1		
2	(a)	
	(b)	
	(c)	
	(d)	
	(e)	
	(f)	
	(g)	
	(h)	
3	(a)	
	(b)	
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16	(a)	
	(b)	
	(c)	
	(d)	

Programming Exercises

Answer all questions.
Remember to save this document regularly.

Q	Answer
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	