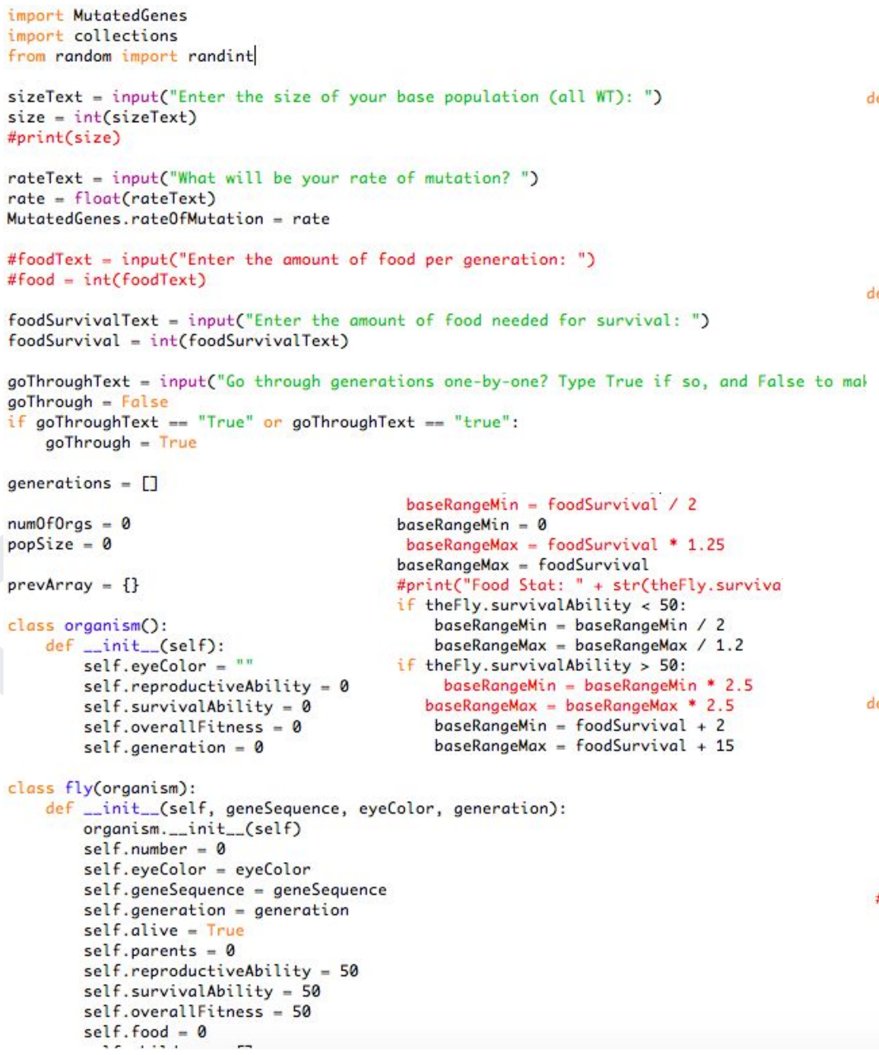
Background

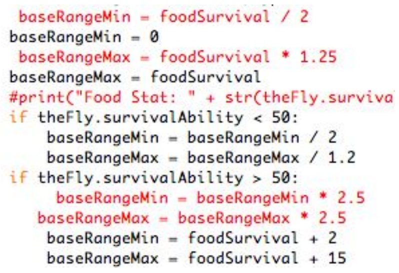
- Evolution is defined as the genetic change in a population over time. Rahul and I hoped to create a model of this by writing our own computer program to display the growth of an arbitrary population through natural or sexual selection. To explain how this correlates to the real world, each individual part of a population has its own specific traits that make itself up due to DNA and genetics. Genes, sequences of DNA that code for specific proteins, are the basis which make up these traits. However, there are different alleles, or gene sequences that code for the same protein in a different way, that come along with traits. For example, a human may have a gene for eye color, but there would be two different alleles that code for blue eye color and brown eye color.
- These genes/alleles will code for the traits of an organism, and they will be passed down during reproduction so that offspring keep similar traits of their parent. This idea of heritability is an important factor in why evolution occurs, as traits being passed down to offspring can increase the frequency of an allele in a population, and therefore cause a genetic change. However, populations start out with only one allele, called the wild type. Mutations, or random changes in the coding of a gene creating a never before seen allele, occur during reproduction to cause variation as they bring up new traits that differ from the wild type. This is what kickstarts evolution, as when advantageous traits are brought up by mutations, selective pressures cause the allele to increase in frequency around the population. For example, a trait that made a organism survive longer would mean it would reproduce more, therefore causing the allele to be passed down in abundance throughout the generations. Eventually, the trait would take up most of the population, as those without the trait would be at a disadvantage to reproduce, also called less fit.
- This process is called selection and comes in the form of natural selection and sexual selection among others. As the selection causes an advantageous trait to spread throughout the population, evolution occurs as the population evolves to be better. Rahul and I attempt to model this with organisms which have varying eye colors, as you will see in the procedure section.

Hypothesis

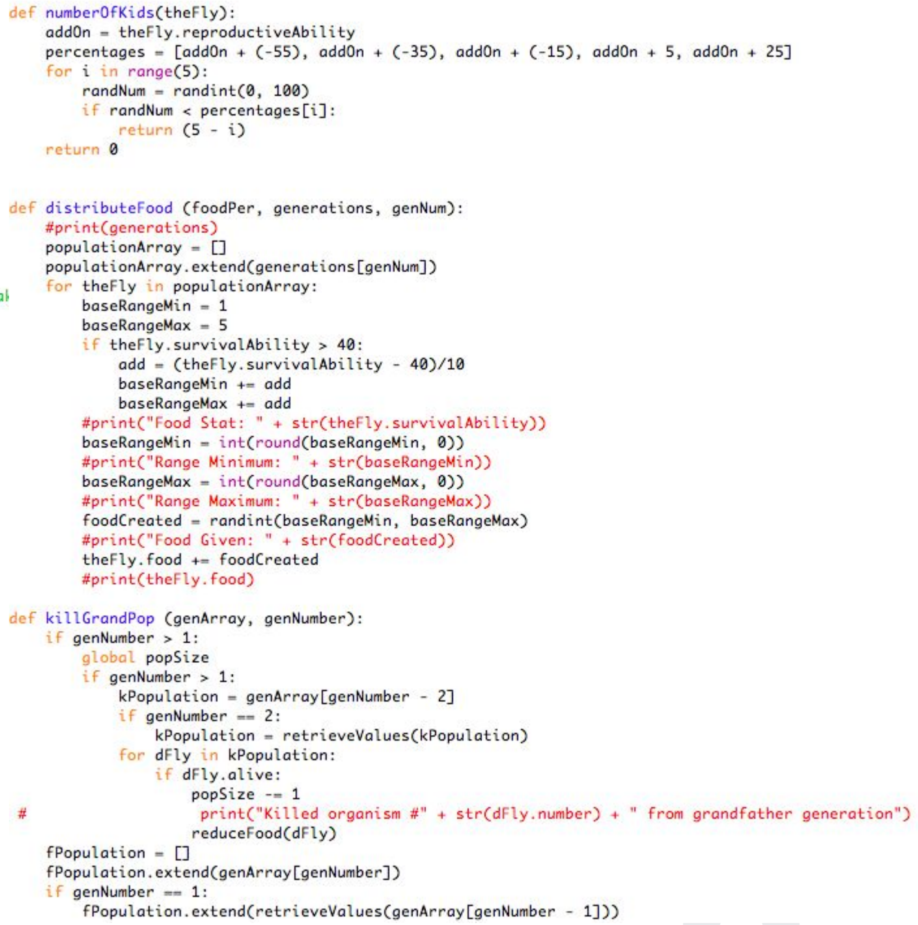
- This is a **discovery-based** experiment and does not have a clear hypothesis. However, we believe the simulation will be able to correctly model evolution and end up with the advantageous alleles having more rapid population growth even with the variety of different inputs.



```
import MutatedGenes
import collections
from random import randint

sizeText = input("Enter the size of your base population (all WT): ")
size = int(sizeText)
#print(size)

rateText = input("What will be your rate of mutation? ")
rate = float(rateText)
MutatedGenes.rateOfMutation = rate

#foodText = input("Enter the amount of food per generation: ")
#food = int(foodText)

foodSurvivalText = input("Enter the amount of food needed for survival: ")
foodSurvival = int(foodSurvivalText)

goThroughText = input("Go through generations one-by-one? Type True if so, and False to mal
goThrough = False
if goThroughText == "True" or goThroughText == "true":
    goThrough = True

generations = []

numOfOrgs = 0
popSize = 0

prevArray = {}

class organism():
    def __init__(self):
        self.eyeColor = ""
        self.reproductiveAbility = 0
        self.survivalAbility = 0
        self.overallFitness = 0
        self.generation = 0

class fly(organism):
    def __init__(self, geneSequence, eyeColor, generation):
        organism.__init__(self)
        self.number = 0
        self.eyeColor = eyeColor
        self.geneSequence = geneSequence
        self.generation = generation
        self.alive = True
        self.parents = 0
        self.reproductiveAbility = 50
        self.survivalAbility = 50
        self.overallFitness = 50
        self.food = 0
```



```
 baseRangeMin = foodSurvival / 2
baseRangeMin = 0
 baseRangeMax = foodSurvival * 1.25
baseRangeMax = foodSurvival
#print("Food Stat: " + str(theFly.surviva
if theFly.survivalAbility < 50:
    baseRangeMin = baseRangeMin / 2
    baseRangeMax = baseRangeMax / 1.2
if theFly.survivalAbility > 50:
     baseRangeMin = baseRangeMin * 2.5
   baseRangeMax = baseRangeMax * 2.5
    baseRangeMin = foodSurvival + 2
    baseRangeMax = foodSurvival + 15
```



```
def numberOfKids(theFly):
    addOn = theFly.reproductiveAbility
    percentages = [addOn + (-55), addOn + (-35), addOn + (-15), addOn + 5, addOn + 25]
    for i in range(5):
        randNum = randint(0, 100)
        if randNum < percentages[i]:
            return (5 - i)
    return 0


def distributeFood (foodPer, generations, genNum):
    #print(generations)
    populationArray = []
    populationArray.extend(generations[genNum])
    for theFly in populationArray:
        baseRangeMin = 1
        baseRangeMax = 5
        if theFly.survivalAbility > 40:
            add = (theFly.survivalAbility - 40)/10
            baseRangeMin += add
            baseRangeMax += add
        #print("Food Stat: " + str(theFly.survivalAbility))
        baseRangeMin = int(round(baseRangeMin, 0))
        #print("Range Minimum: " + str(baseRangeMin))
        baseRangeMax = int(round(baseRangeMax, 0))
        #print("Range Maximum: " + str(baseRangeMax))
        foodCreated = randint(baseRangeMin, baseRangeMax)
        #print("Food Given: " + str(foodCreated))
        theFly.food += foodCreated
        #print(theFly.food)

def killGrandPop (genArray, genNumber):
    if genNumber > 1:
        global popSize
        if genNumber > 1:
            kPopulation = genArray[genNumber - 2]
            if genNumber == 2:
                kPopulation = retrieveValues(kPopulation)
            for dFly in kPopulation:
                if dFly.alive:
                    popSize -= 1
#                    print("Killed organism #" + str(dFly.number) + " from grandfather generation")
                    reduceFood(dFly)
    fPopulation = []
    fPopulation.extend(genArray[genNumber])
    if genNumber == 1:
        fPopulation.extend(retrieveValues(genArray[genNumber - 1]))
```

Materials:

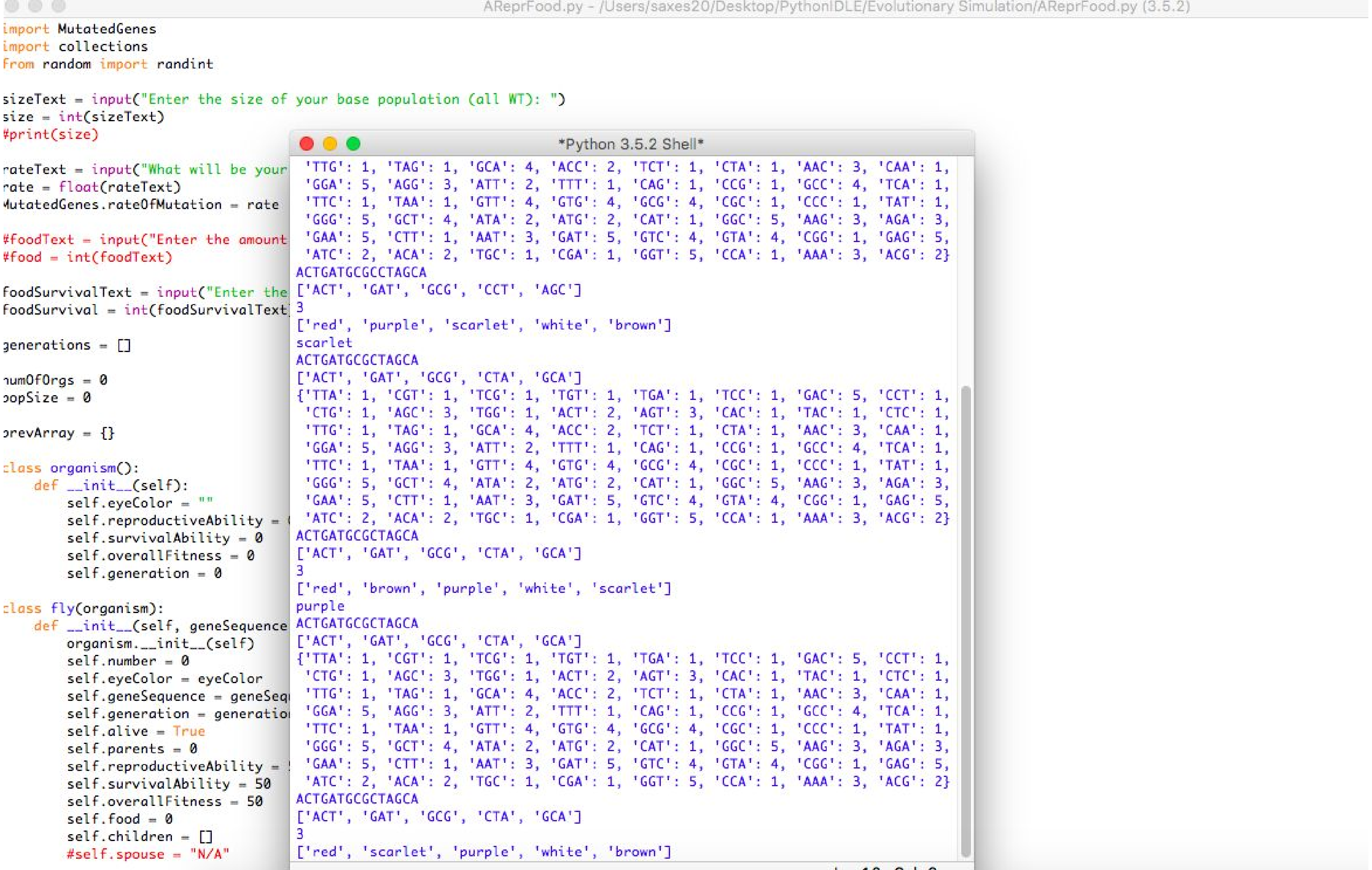
- ❑ Computer (MacBook Air)
- ❑ IDLE (software application)
- ❑ Internet Access for APIs and frameworks

Procedure:

- ❑ Write the program in Python
- ❑ Run the simulation
- ❑ Record data into Microsoft Excel spreadsheet

Procedure for Writing the Code

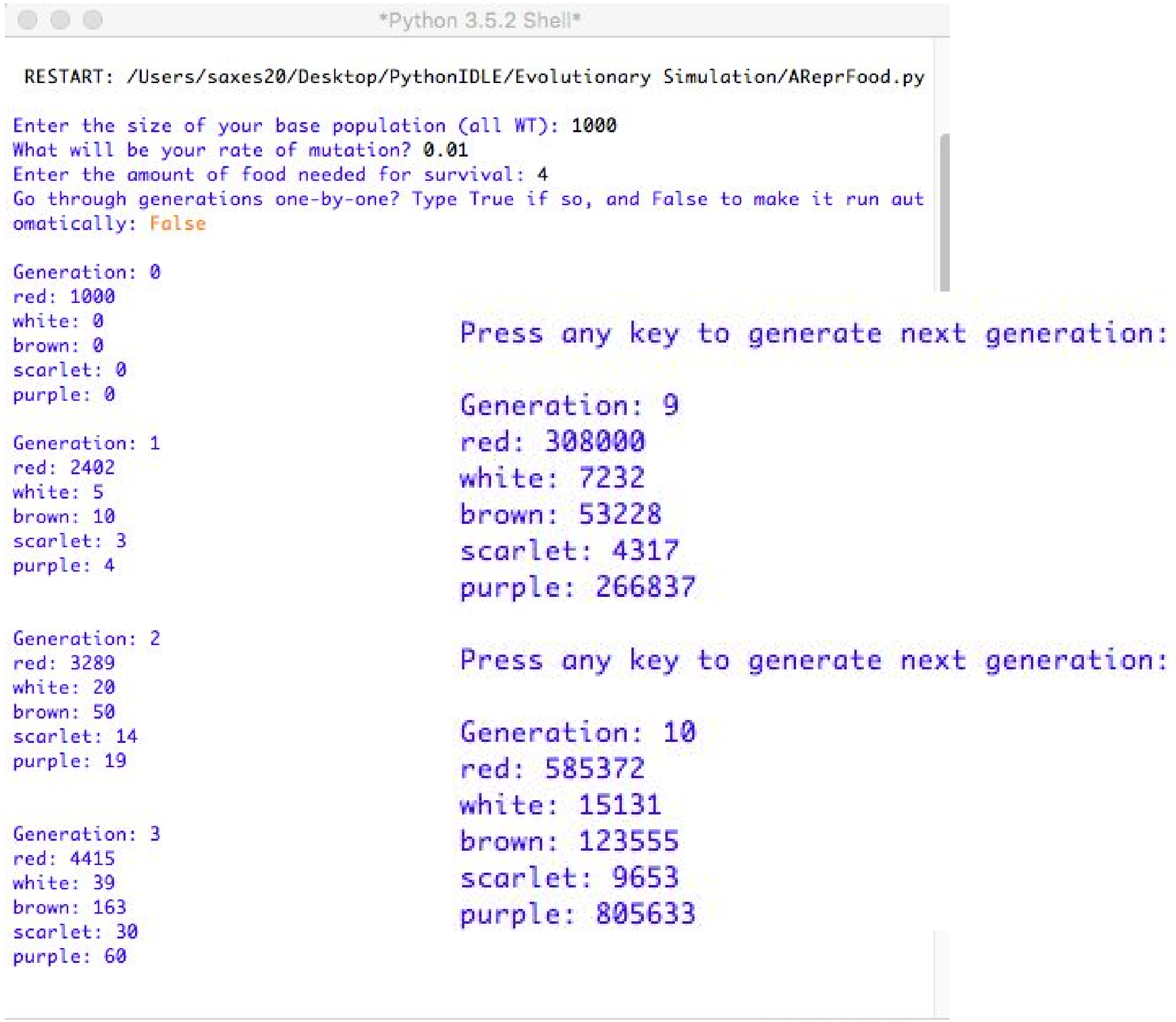
1. Firstly we had to perform research on evolution. We had to understand the processes of natural and sexual selection, along with selective pressures pushing evolution in a certain direction. Furthermore, mutations were a necessary topic to comprehend as mutations cause evolution by bringing up a never before seen advantageous trait. We utilized the processes of DNA transcription and DNA translation in our mutation algorithm to create a successful simulation. Without the evolution unit and further research on the processes which end up in populations evolving, this model would never be created.
2. As we now knew the ins-and-outs of evolution, we had to apply our knowledge to outlining a simulation to model this. However, similar to every computer simulation, we needed to abstract, or simplify, some processes and factors. Nobody would ever be able to create a complete model of evolution exactly similar to real life. As programmers we needed to deduce the essential parts of evolution to include in the simulation. Examples of parts we left out were a visual display, various organisms, various traits, artificial selection, and creating three dimensional proteins out of our mutated gene sequences.
3. We started with the mutation algorithm. This set of functions was meant to take an input of a gene sequence and randomly change it to create a new gene sequence, using insertion, deletion, and substitution. To accomplish this, firstly the gene sequence was broken up into codons, and the codons were formed into a codon table with every possible codon correlating to a specific number (1-5 as we had 5 traits). After mutating the gene sequence, we would break up the new gene sequence into codons and match them to the numbers on the codon table. Taking an average of that number brought us to our final result, a number which was matched up with a trait. Through the mutation algorithm, reproduction could lead to new alleles.
4. The reproduction algorithm was obviously the main part of our simulation. Our goal was to create one species of organisms that had only one varying trait, eye color. Different eye colors would offer various advantages and disadvantages, but based on certain inputs the simulation would hopefully model how the population changes based on these traits. We started off generating a base population of *x* amount of organisms, all with the wild type, a red eye color. Reproduction would occur asexually by these organisms, and they would replicate themselves a certain amount of times based on their reproductive ability. However, organisms would only be able to reproduce if they had enough food to survive. This is how advantages to survival ability came into play, as the traits which led to more food received would most likely increase in allele frequency as food becomes harder to get. All of this along with many other factors and processes coded for led to a simulation where reproduction was looped and stats regarding allele frequencies for each generation were displayed. After almost 600 lines of code and hours of debugging, we had a working simulation that accomplished this.


AReprFood.py - /Users/saxes20/Desktop/PythonIDLE/Evolutionary Simulation/AReprFood.py (3.5.2)

```
import MutatedGenes
import collections
from random import randint

sizeText = input("Enter the size of your base population (all WT): ")
size = int(sizeText)
#print(size)

rateText = input("What will be your
rate = float(rateText)
utatedGenes.rateOfMutation = rate

#foodText = input("Enter the amount
#food = int(foodText)

foodSurvivalText = input("Enter the
foodSurvival = int(foodSurvivalText

generations = []

umOfOrgs = 0
opSize = 0

revArray = {}

lass organism():
    def __init__(self):
        self.eyeColor = ""
        self.reproductiveAbility =
        self.survivalAbility = 0
        self.overallFitness = 0
        self.generation = 0

lass fly(organism):
    def __init__(self, geneSequence
        organism.__init__(self)
        self.number = 0
        self.eyeColor = eyeColor
        self.geneSequence = geneSeq
        self.generation = generatio
        self.alive = True
        self.parents = 0
        self.reproductiveAbility =
        self.survivalAbility = 50
        self.overallFitness = 50
        self.food = 0
        self.children = []
        #self.spouse = "N/A"
```

Python 3.5.2 Shell

```
 'TTG': 1, 'TAG': 1, 'GCA': 4, 'ACC': 2, 'TCT': 1, 'CTA': 1, 'AAC': 3, 'CAA': 1,
 'GGA': 5, 'AGG': 3, 'ATT': 2, 'TTT': 1, 'CAG': 1, 'CCG': 1, 'GCC': 4, 'TCA': 1,
 'TTC': 1, 'TAA': 1, 'GTT': 4, 'GTG': 4, 'GCG': 4, 'CGC': 1, 'CCC': 1, 'TAT': 1,
 'GGG': 5, 'GCT': 4, 'ATA': 2, 'ATG': 2, 'CAT': 1, 'GGC': 5, 'AAG': 3, 'AGA': 3,
 'GAA': 5, 'CTT': 1, 'AAT': 3, 'GAT': 5, 'GTC': 4, 'GTA': 4, 'CGG': 1, 'GAG': 5,
 'ATC': 2, 'ACA': 2, 'TGC': 1, 'CGA': 1, 'GGT': 5, 'CCA': 1, 'AAA': 3, 'ACG': 2}
ACTGATGCGCCTAGCA
['ACT', 'GAT', 'GCG', 'CCT', 'AGC']
3
['red', 'purple', 'scarlet', 'white', 'brown']
scarlet
ACTGATGCGCTAGCA
['ACT', 'GAT', 'GCG', 'CTA', 'GCA']
{'TTA': 1, 'CGT': 1, 'TCG': 1, 'TGT': 1, 'TGA': 1, 'TCC': 1, 'GAC': 5, 'CCT': 1,
 'CTG': 1, 'AGC': 3, 'TGG': 1, 'ACT': 2, 'AGT': 3, 'CAC': 1, 'TAC': 1, 'CTC': 1,
 'TTG': 1, 'TAG': 1, 'GCA': 4, 'ACC': 2, 'TCT': 1, 'CTA': 1, 'AAC': 3, 'CAA': 1,
 'GGA': 5, 'AGG': 3, 'ATT': 2, 'TTT': 1, 'CAG': 1, 'CCG': 1, 'GCC': 4, 'TCA': 1,
 'TTC': 1, 'TAA': 1, 'GTT': 4, 'GTG': 4, 'GCG': 4, 'CGC': 1, 'CCC': 1, 'TAT': 1,
 'GGG': 5, 'GCT': 4, 'ATA': 2, 'ATG': 2, 'CAT': 1, 'GGC': 5, 'AAG': 3, 'AGA': 3,
 'GAA': 5, 'CTT': 1, 'AAT': 3, 'GAT': 5, 'GTC': 4, 'GTA': 4, 'CGG': 1, 'GAG': 5,
 'ATC': 2, 'ACA': 2, 'TGC': 1, 'CGA': 1, 'GGT': 5, 'CCA': 1, 'AAA': 3, 'ACG': 2}
ACTGATGCGCTAGCA
['ACT', 'GAT', 'GCG', 'CTA', 'GCA']
3
['red', 'brown', 'purple', 'white', 'scarlet']
purple
ACTGATGCGCTAGCA
['ACT', 'GAT', 'GCG', 'CTA', 'GCA']
{'TTA': 1, 'CGT': 1, 'TCG': 1, 'TGT': 1, 'TGA': 1, 'TCC': 1, 'GAC': 5, 'CCT': 1,
 'CTG': 1, 'AGC': 3, 'TGG': 1, 'ACT': 2, 'AGT': 3, 'CAC': 1, 'TAC': 1, 'CTC': 1,
 'TTG': 1, 'TAG': 1, 'GCA': 4, 'ACC': 2, 'TCT': 1, 'CTA': 1, 'AAC': 3, 'CAA': 1,
 'GGA': 5, 'AGG': 3, 'ATT': 2, 'TTT': 1, 'CAG': 1, 'CCG': 1, 'GCC': 4, 'TCA': 1,
 'TTC': 1, 'TAA': 1, 'GTT': 4, 'GTG': 4, 'GCG': 4, 'CGC': 1, 'CCC': 1, 'TAT': 1,
 'GGG': 5, 'GCT': 4, 'ATA': 2, 'ATG': 2, 'CAT': 1, 'GGC': 5, 'AAG': 3, 'AGA': 3,
 'GAA': 5, 'CTT': 1, 'AAT': 3, 'GAT': 5, 'GTC': 4, 'GTA': 4, 'CGG': 1, 'GAG': 5,
 'ATC': 2, 'ACA': 2, 'TGC': 1, 'CGA': 1, 'GGT': 5, 'CCA': 1, 'AAA': 3, 'ACG': 2}
ACTGATGCGCTAGCA
['ACT', 'GAT', 'GCG', 'CTA', 'GCA']
3
['red', 'scarlet', 'purple', 'white', 'brown']
```

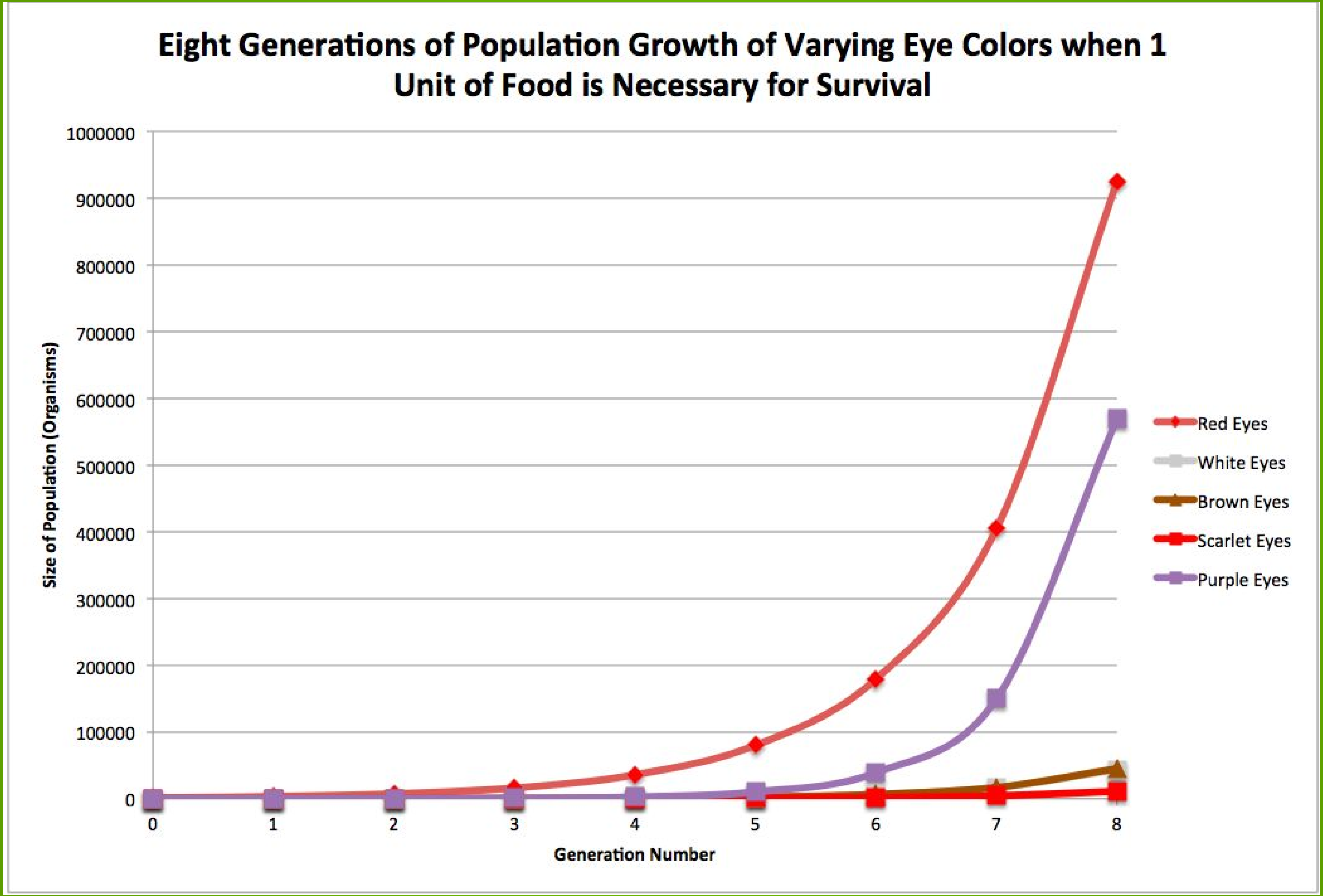


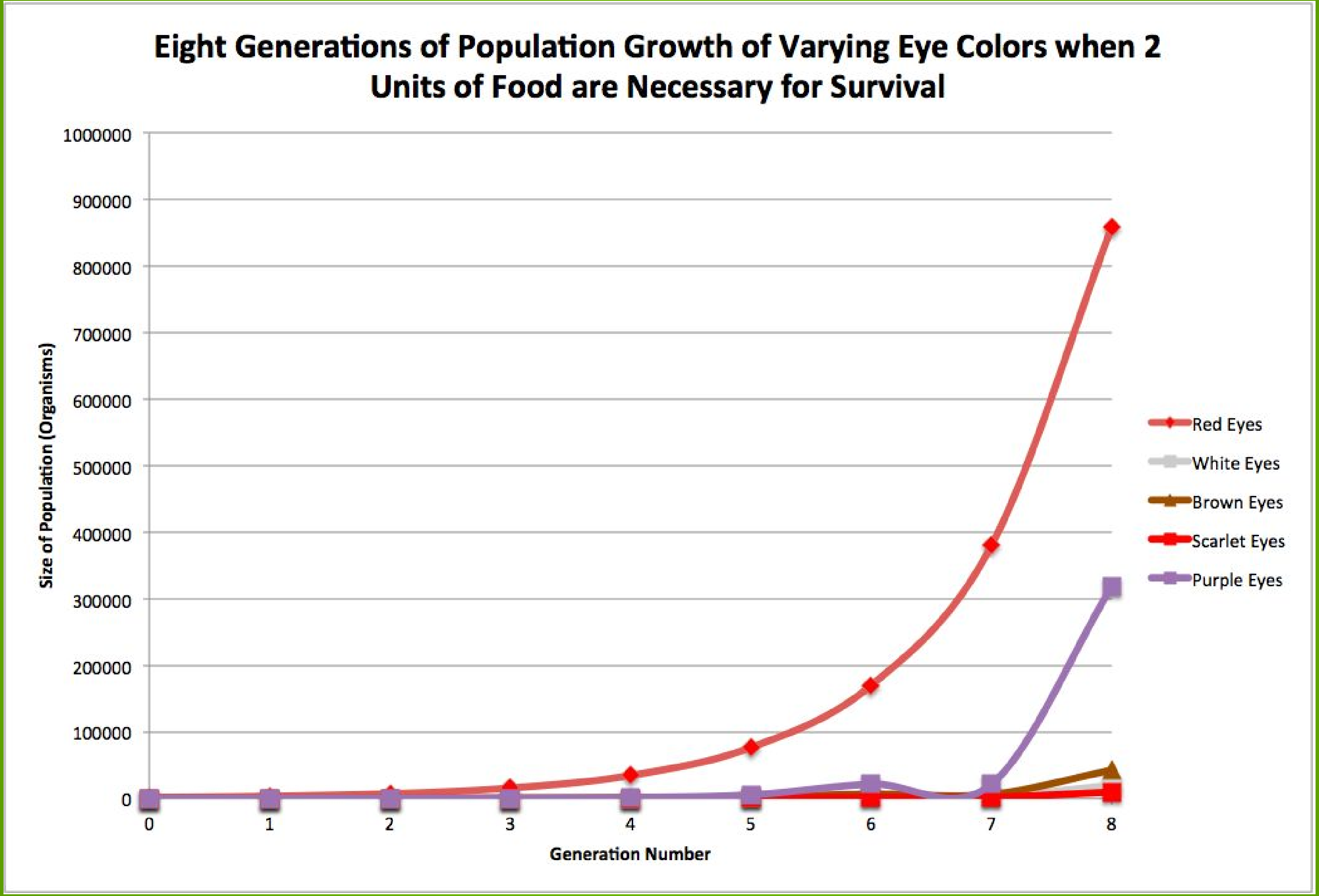
```
 RESTART: /Users/saxes20/Desktop/PythonIDLE/Evolutionary Simulation/AReprFood.py

Enter the size of your base population (all WT): 1000
What will be your rate of mutation? 0.01
Enter the amount of food needed for survival: 4
Go through generations one-by-one? Type True if so, and False to make it run aut
omatically: False

Generation: 0
red: 1000
white: 0
brown: 0
scarlet: 0
purple: 0

Generation: 1
red: 2402
white: 5
brown: 10
scarlet: 3
purple: 4


Generation: 2
red: 3289
white: 20
brown: 50
scarlet: 14
purple: 19


Generation: 3
red: 4415
white: 39
brown: 163
scarlet: 30
purple: 60
```

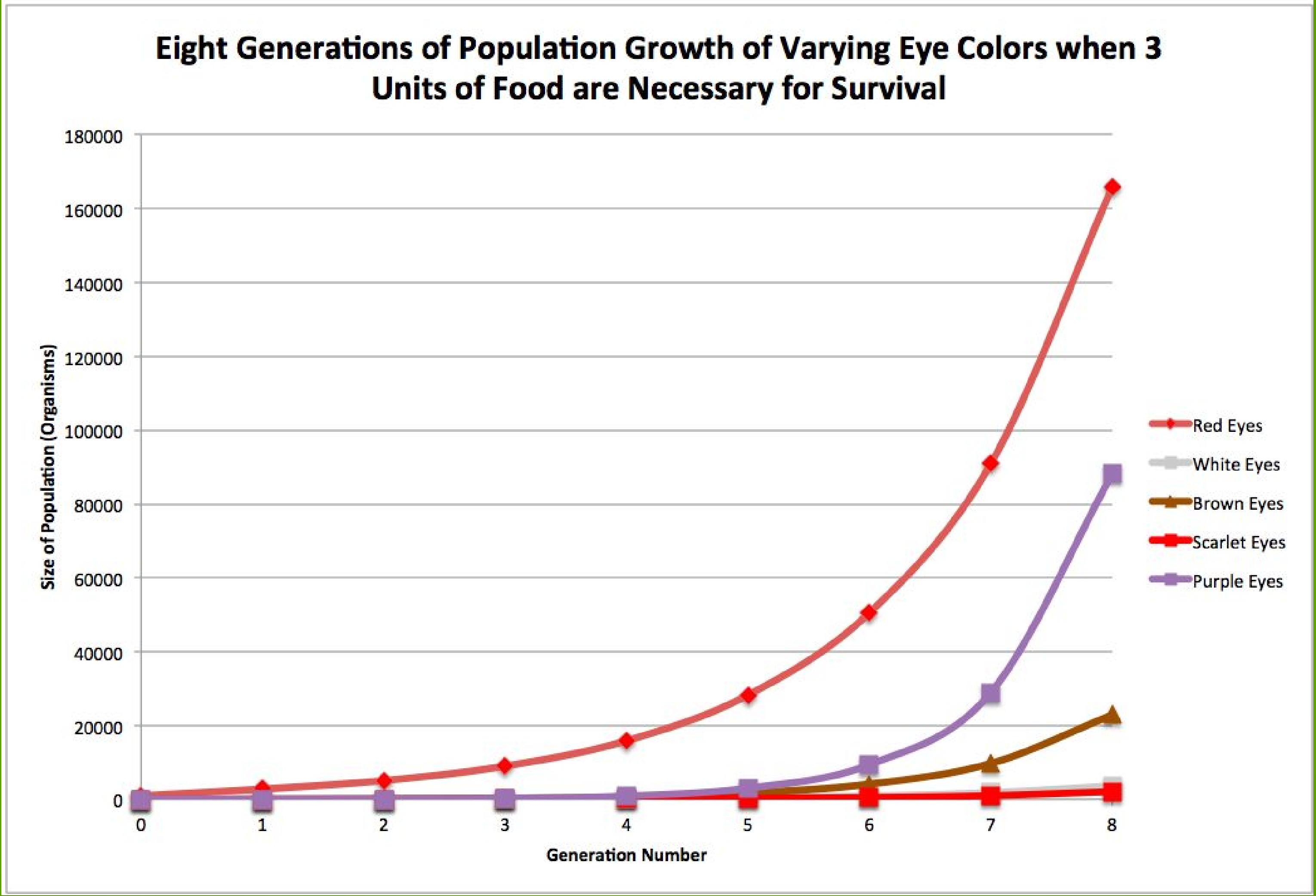
```
Press any key to generate next generation:

Generation: 9
red: 308000
white: 7232
brown: 53228
scarlet: 4317
purple: 266837

Press any key to generate next generation:

Generation: 10
red: 585372
white: 15131
brown: 123555
scarlet: 9653
purple: 805633
```

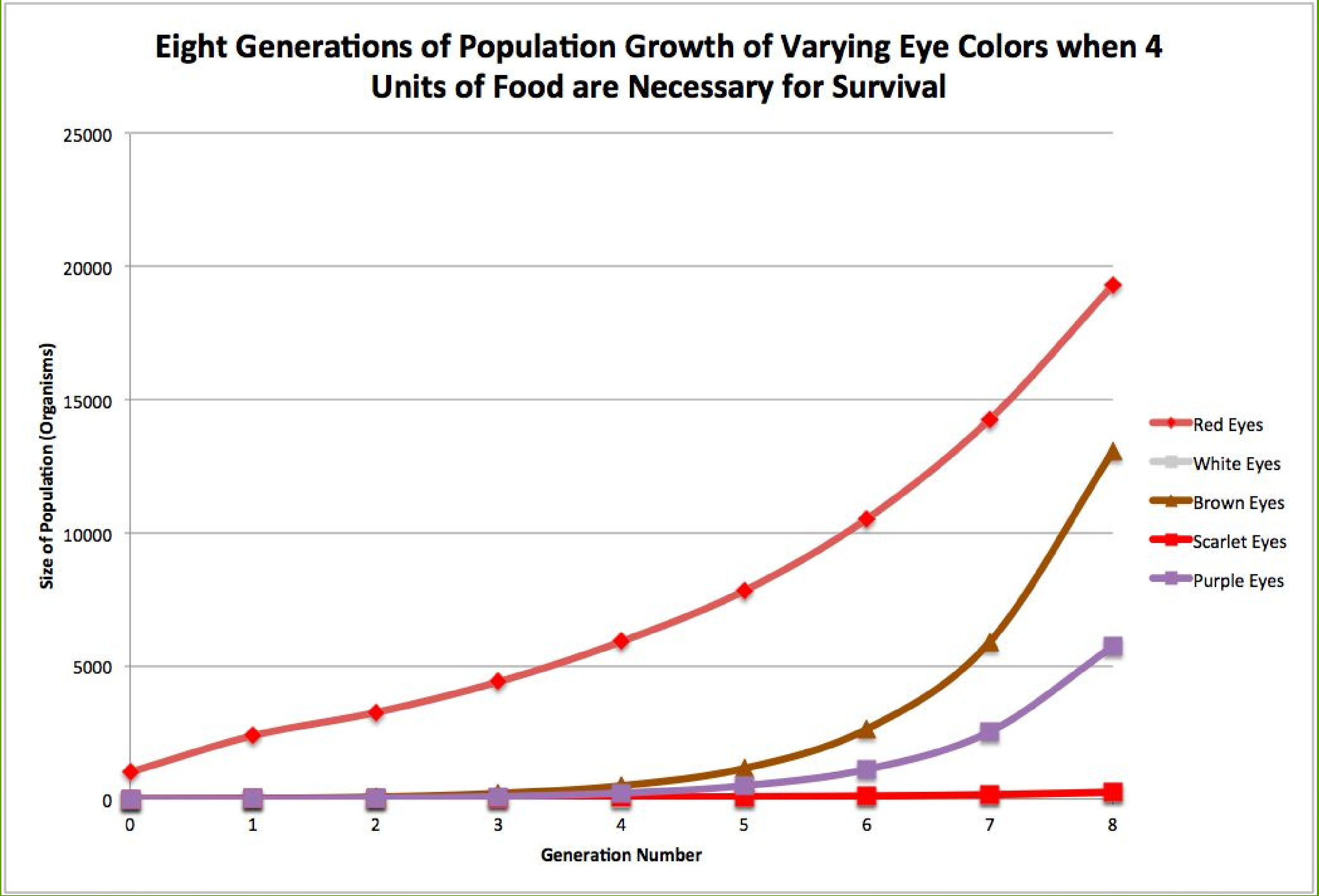
Results

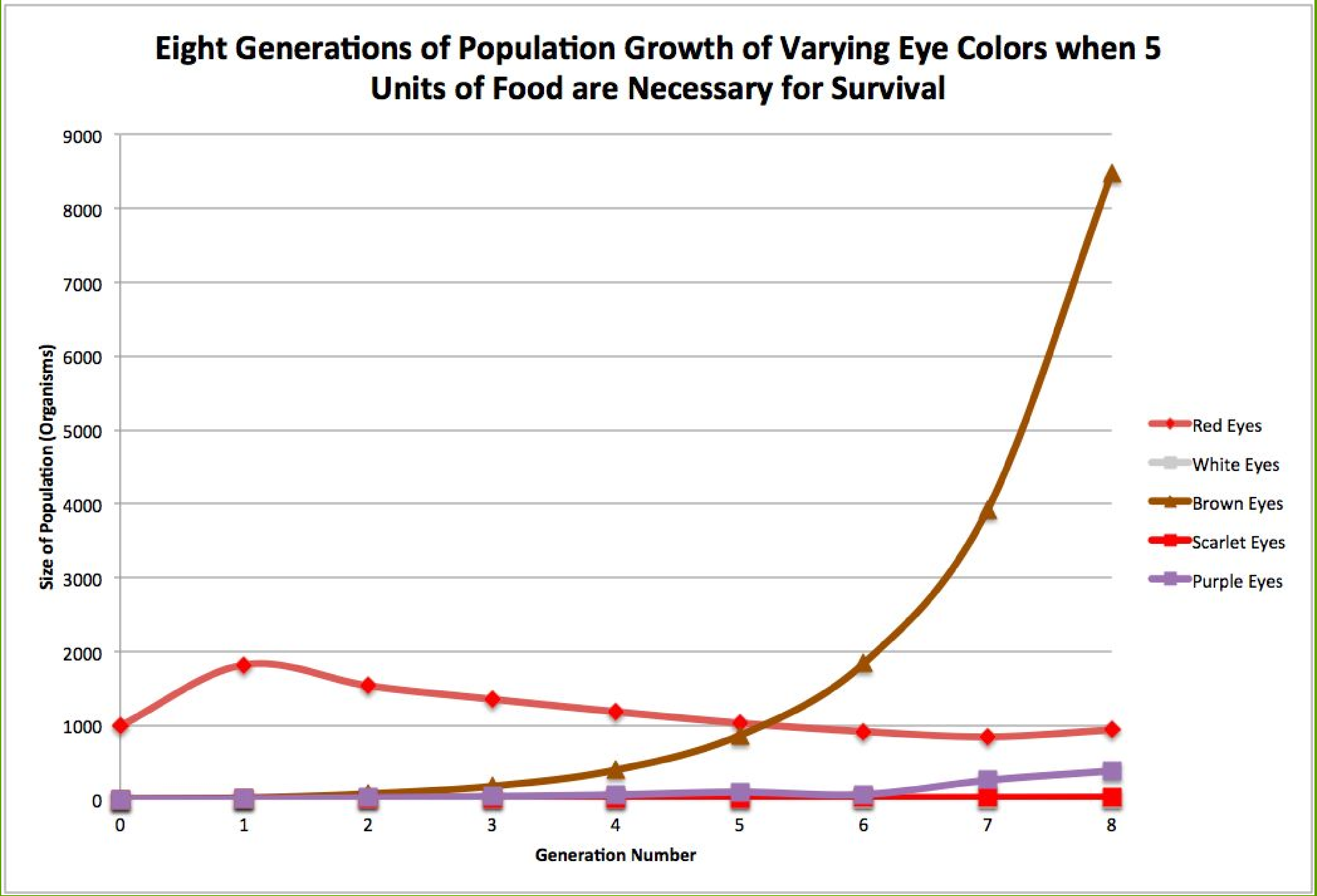
- Below is our data table for all the results of our simulation. We ran our simulation for eight generations to see how the population changed based on how much food was needed for survival. As we saw, when the amount of food needed was very low, the traits with the higher reproductive rates survived. When this changed, however, and the amount of food needed became a problem, those traits with the higher survival ability thrived and increased in frequency.
- Note: The red eye-colored organisms always seem to be at the top because the base population is all red organisms with no organisms of another eye color. It takes time, more than eight generations, to see a drastic change.
- Though we had wished to run more generations, we were partly limited by the speed of the simulation. When population sizes get past the millions, with all the algorithms running in the simulation it tends to take a while. However, simulation speeds began to get faster as the amount of food necessary became higher. This is because population sizes began to get much smaller as not enough organisms were able to maintain the food necessary to survive
- Though it is not shown, there was some important data to note past just eight generations. When the food necessary was **three units,** the purple eye-colored population overtook red by more than 200,000 organisms by the tenth generation. When the food necessary was **four or more units** the brown eye-colored organisms began to increase rapidly. When it was **four units** necessary, the brown population overtook the red population by more than 2,000 organisms in the next (ninth) generation. In the other instances, the brown population more than doubled its size each time a generation passed, and it became especially large by the tenth and succeeding generations.

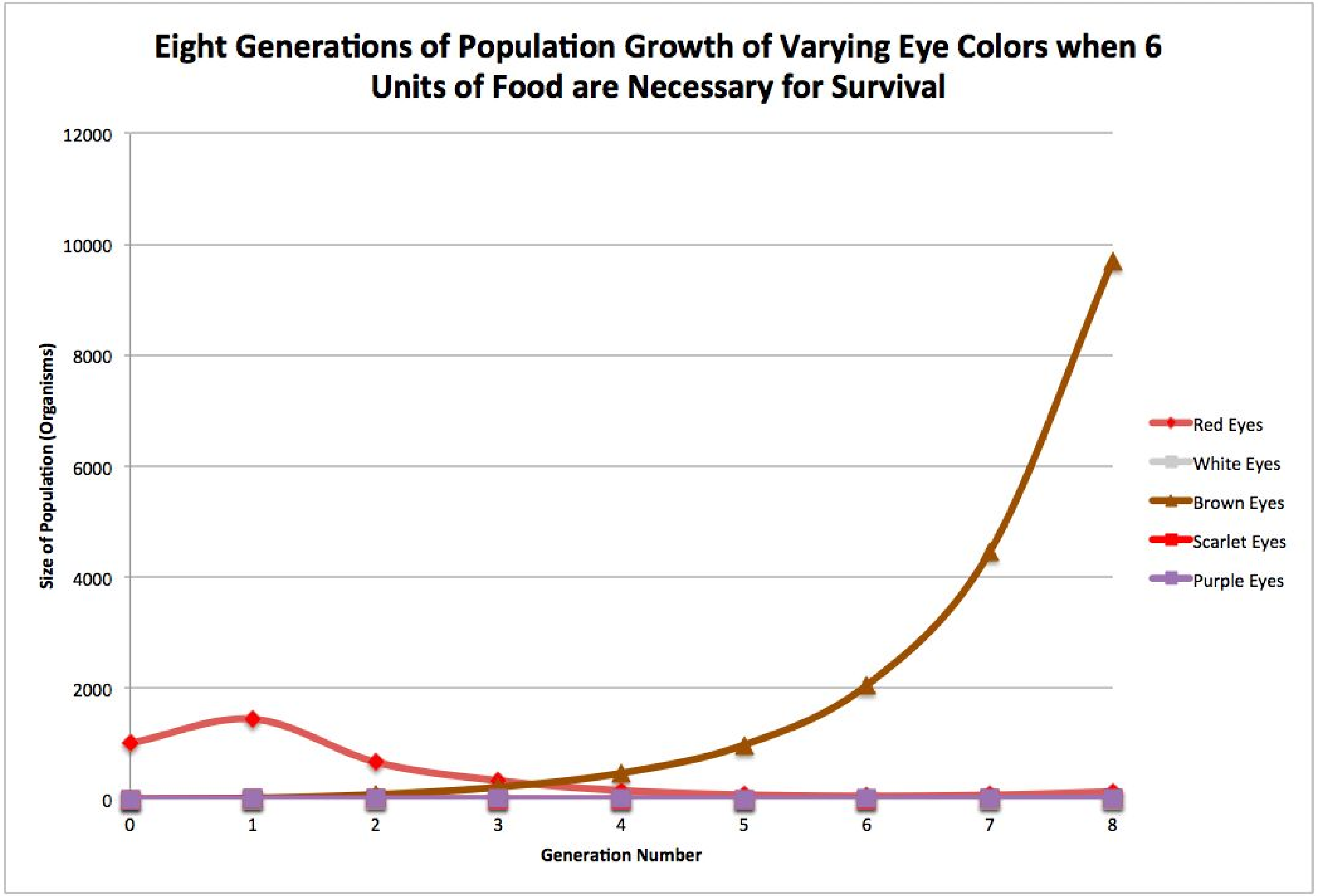

Eight Generations of Population Growth of Varying Eye Colors when 1 Unit of Food is Necessary for Survival
1000000
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
Size of Population (Organisms)
0
1
2
3
4
5
6
7
8
Generation Number
Red Eyes
White Eyes
Brown Eyes
Scarlet Eyes
Purple Eyes


Eight Generations of Population Growth of Varying Eye Colors when 2 Units of Food are Necessary for Survival
Size of Population (Organisms)
1000000
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
4
5
6
7
8
Generation Number
Red Eyes
White Eyes
Brown Eyes
Scarlet Eyes
Purple Eyes


Eight Generations of Population Growth of Varying Eye Colors when 3 Units of Food are Necessary for Survival
Size of Population (Organisms)
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
2
3
4
5
6
7
8
Generation Number
Red Eyes
White Eyes
Brown Eyes
Scarlet Eyes
Purple Eyes


Eight Generations of Population Growth of Varying Eye Colors when 4 Units of Food are Necessary for Survival
Size of Population (Organisms)
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
6
7
8
Generation Number
Red Eyes
White Eyes
Brown Eyes
Scarlet Eyes
Purple Eyes


Eight Generations of Population Growth of Varying Eye Colors when 5 Units of Food are Necessary for Survival
Size of Population (Organisms)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
6
7
8
Generation Number
Red Eyes
White Eyes
Brown Eyes
Scarlet Eyes
Purple Eyes


Eight Generations of Population Growth of Varying Eye Colors when 6 Units of Food are Necessary for Survival
12000
10000
8000
6000
4000
2000
0
Size of Population (Organisms)
0
1
2
3
4
5
6
7
8
Generation Number
Red Eyes
White Eyes
Brown Eyes
Scarlet Eyes
Purple Eyes

Tracking 8 Generations of Population Growth (Organisms) of Varying Eye Col

	Amount of Food (unit														
	1					2					3				
Generation	Red Eyes	White Eyes	Brown Eyes	Scarlet Eyes	Purple Eyes	Red Eyes	White Eyes	Brown Eyes	Scarlet Eyes	Purple Eyes	Red Eyes	White Eyes	Brown Eyes	Scarlet Eyes	Purple Eyes
0	1000	0	0	0	0	1000	0	0	0	0	1000	0	0	0	0
1	3250	9	15	14	16	3206	8	15	9	4	2813	8	17	6	12
2	7216	61	86	125	125	7100	42	76	43	45	5094	37	89	39	72
3	16095	264	287	150	603	15656	158	265	114	272	9020	100	266	92	275
4	35848	778	855	336	2539	34592	454	800	273	1282	15899	225	687	159	947
5	80065	2211	2351	785	10091	76619	1238	2313	671	5389	28272	488	1748	274	3030
6	179519	6163	6355	1854	39118	169920	3243	6299	1674	21452	50550	968	4142	519	9419
7	405254	16442	16921	4371	149908	380360	3243	6299	1674	21452	90955	1844	9743	1032	28977
8	926123	43417	44928	10914	569644	859182	19562	42638	9664	318413	165927	3603	22909	2085	88173

lors when Different Amounts of Food are Necessary for Survival

ts) Necessary for Survival

4					5					6				
Red Eyes	White Eyes	Brown Eyes	Scarlet Eyes	Purple Eyes	Red Eyes	White Eyes	Brown Eyes	Scarlet Eyes	Purple Eyes	Red Eyes	White Eyes	Brown Eyes	Scarlet Eyes	Purple Eyes
1000	0	0	0	0	1000	0	0	0	0	1000	0	0	0	0
2375	10	19	9	7	1821	4	16	4	5	1443	0	19	5	3
3249	26	71	18	30	1536	9	70	16	18	667	0	81	6	6
4402	35	208	33	87	1354	8	175	15	36	332	0	214	1	5
5915	44	500	63	219	1184	7	396	7	58	154	0	468	0	3
7822	56	1147	82	499	1031	7	865	11	94	76	0	975	0	1
10531	97	2631	108	1115	917	7	1842	19	63	53	0	2054	1	3
14265	160	5908	157	2530	844	10	3929	22	256	71	0	4460	3	12
19298	261	13047	249	5732	939	13	8481	25	383	130	0	9702	4	17

Conclusion/Analysis

- Fortunately, we had successfully been able to model a simulation of evolution, as the data is structured as population growth in the real world would play out.
- **When the food necessary was 1 Unit:** the purple population grew as the purple organisms had a higher reproductive ability.
- **When the food necessary was 2 Units and 3 Units:** Again the purple eyed organisms increase in frequency throughout the population as their reproductive rat is higher than any other organisms
- **When the food necessary was 4 Units:** Instead of purple, now the brown eyed organisms start to rapidly increase in population growth as brown eyed organisms have a higher survival ability
- **When the food necessary was >4 Units:** The brown population begins to more than double in size every generation
- THIS TELLS US that when the need for food is low, and organisms do not need much food to survive, evolution will stray toward making a population with those traits most suited for reproduction. This is called sexual selection. We saw that the purple massively increased when the amount of food necessary was less than 3 units, meaning it was easy to get food so the purple eyed organisms got the advantage.
- HOWEVER when the food necessary to survive becomes higher, such as 4 or more units, then those traits most suited for survival ability, or gathering food, will increase in frequency throughout the population. Evolution will cause the population to evolve to be better suited to get food, as we saw by the later generations the brown population grew so quickly it overtook the red population. This is called natural selection

Bibliography

- Brain, Marshall. "How Evolution Works." *The Basic Process of Evolution - How Evolution Works | HowStuffWorks*, HowStuffWorks, 2016, science.howstuffworks.com/life/evolution/evolution1.htm.
- Phelan, Jay. "Evolution and Natural Selection: Darwin's Dangerous Idea." *What Is Life?: A Guide to Biology*, 3rd ed., New York, W H Freeman, 2015, pp. 315-63.
- Understanding Evolution. 2016. University of California Museum of Paleontology. 23 October 2016 <http://evolution.berkeley.edu/>.
- http://www.slidescarnival.com/iras-free-presentation-template/1350