FISHING SIMULATION

Assignment 2: ITECH1400

JUNE 4, 2022

Raju Lamsal
304109

Contents

Fishing Simulation

I added six classes to a module called fish species: AustralianBass, ShortFinnedEel, EelTailedCatfish, GippslandPerch, Branzino, and Tilapia.

Fish Species Class Module

In this class module, I have created all six classes of fish species which contains all the variable and constants like MAX_WEIGHT, MAX_EATING_WEIGHT, NAME, LATIN_NAME. Also, it contains the constructor function that initialize weight attribute. After that, I have created the function 'is_good_eating' that return True if weights is range 500 g and 2500g. Finally, __str__ function is used that returns string representation of fish object as shown below:

Class AustralianBass

```
class AustrailianBass:
        #constant variables
    MAX_WEIGHT = 4000
    MAX_EATING_WEIGHT = 2500
    NAME = 'Australian Bass'
    LATIN_NAME = 'Macquaria Novemaculeata'

        #construtor with one arguments
    def __init__(self,weight):
        if weight > self.MAX_WEIGHT:  #weight is greater than max weight
            pass   #return false
        else:
            self.weight=weight   #weight is initialised
    def is_good_eating(self):
        if self.weight>500 and self.weight<self.MAX_EATING_WEIGHT:  #if wieight is between 500 and Max eating weight
            return True   #return true
        else:
            return False   #return false

    def __str__(self):

            #return string
        msg=self.NAME+"("+self.LATIN_NAME+"), "+str(self.weight)+" "+str(self.is_good_eating())
        return msg
```

Fig 1: AustralianBass class

Class ShortFinnedEel

```python
#class ShortFinnedEel
class ShortFinnedEel:
        #constant variables
    MAX_WEIGHT = 3000
    MAX_EATING_WEIGHT = 1500
    NAME = 'ShortFinnedEel'
    LATIN_NAME = 'Anguilla australis'

        #construtor with one arguments
    def __init__(self,weight):
        if weight > self.MAX_WEIGHT:  #weight is greater than max weight
            pass   #return false
        else:
            self.weight=weight   #weight is initialised

    def is_good_eating(self):
        if self.weight>500 and self.weight<self.MAX_EATING_WEIGHT:  #if wieight is between 500 and Max eating weight
            return True   #return true
        else:
            return False   #return false

    def __str__(self):

            #return string
        msg=self.NAME+"("+self.LATIN_NAME+"), "+str(self.weight)+" "+str(self.is_good_eating())
        return msg
```

Fig 2: ShortFinnedEel class

Class EelTailedCatfish

```
#class EelTailedCatfish
class EelTailedCatfish:
        #constant variables
    MAX_WEIGHT = 5000
    MAX_EATING_WEIGHT = 3500
    NAME = 'EelTailedCatfish'
    LATIN_NAME = 'Tandanus tandanus'

        #construtor with one arguments
    def __init__(self,weight):
        if weight > self.MAX_WEIGHT:  #weight is greater than max weight
            pass   #return false
        else:
            self.weight=weight   #weight is initialised

    def is_good_eating(self):
        if self.weight>500 and self.weight<self.MAX_EATING_WEIGHT:  #if wieight is between 500 and Max eating weight
            return True   #return true
        else:
            return False   #return false

    def __str__(self):

            #return string
        msg=self.NAME+"("+self.LATIN_NAME+"), "+str(self.weight)+" "+str(self.is_good_eating())
        return msg
```

Fig 3: EelTailedCatfish class

Class GippslandPerch

```
#class GippslandPerch
class GippslandPerch:
        #constant variables
    MAX_WEIGHT = 4000
    MAX_EATING_WEIGHT = 2000
    NAME = 'GippslandPerch'
    LATIN_NAME = 'Percichthyidae'

        #construtor with one arguments
    def __init__(self,weight):
        if weight > self.MAX_WEIGHT:  #weight is greater than max weight
            pass   #return false
        else:
            self.weight=weight   #weight is initialised

    def is_good_eating(self):
        if self.weight>500 and self.weight<self.MAX_EATING_WEIGHT:  #if wieight is between 500 and Max eating weight
            return True   #return true
        else:
            return False   #return false

    def __str__(self):

            #return string
        msg=self.NAME+"("+self.LATIN_NAME+"), "+str(self.weight)+" "+str(self.is_good_eating())
        return msg
```

Fig 4: GippslandPerch class

Class Branzino

```
#class Branzino
class Branzino:
        #constant variables
    MAX_WEIGHT = 2000
    MAX_EATING_WEIGHT = 1000
    NAME = 'Branzino'
    LATIN_NAME = 'Dicentrarchus labrax'

        #construtor with one arguments
        #construtor with one arguments
    def __init__(self,weight):
        if weight > self.MAX_WEIGHT:  #weight is greater than max weight
            pass   #return false
        else:
            self.weight=weight   #weight is initialised

    def is_good_eating(self):
        if self.weight>500 and self.weight<self.MAX_EATING_WEIGHT:  #if wieight is between 500 and Max eating weight
            return True   #return true
        else:
            return False   #return false

    def __str__(self):

            #return string
        msg=self.NAME+"("+self.LATIN_NAME+"), "+str(self.weight)+" "+str(self.is_good_eating())
        return msg
```

Fig 5: Branzino Class

Class Tilapia

```
#class Tilapia
class Tilapia:
        #constant variables
    MAX_WEIGHT = 4000
    MAX_EATING_WEIGHT = 2000
    NAME = 'Tilapia'
    LATIN_NAME = 'Oreochromis niloticus'

        #construtor with one arguments
    def __init__(self,weight):
        if weight > self.MAX_WEIGHT:  #weight is greater than max weight
            pass   #return false
        else:
            self.weight=weight   #weight is initialised

    def is_good_eating(self):
        if self.weight>500 and self.weight<self.MAX_EATING_WEIGHT:  #if wieight is between 500 and Max eating weight
            return True   #return true
        else:
            return False   #return false

    def __str__(self):

        #return string
        msg=self.NAME+"("+self.LATIN_NAME+"), "+str(self.weight)+" "+str(self.is_good_eating())
        return msg
```

Fig 6: Tilapia Class

Module fishing

After completing creating the classes for six fish species, I created another module called **fishing.py** that import **fish_species.py**

```
import pickle
import fish_species  #imported the module in the same folder
import random
import time
from matplotlib import pyplot as plt
#calling AustralianBass object, contstrutor(__init__(weight)) and __str__
#__str__ return the string to AustrailianBass

def start_fishing():
    #print(fish)
    #f_list=[]
    basket=[]
    all_fish=[]
    b_w=0
    while b_w < 25:
        time.sleep(0.1)
        fish = random.randint(0,5)
        if fish ==0:
            try:
                w = random.randint(500,4000)
                AustrailianBass=fish_species.AustrailianBass(w)
                #print(AustrailianBass)
                fs=str(AustrailianBass)
                x = fs.split(', ')
                xx=x[1]
                s=xx.split(' ')
                ww=float(s[0])
                weight=float(ww)/1000
                #print(weight)
                if s[1]=='True':

                    st=x[0]+" wights "+str(weight)+" kg - added to the basket."
                    st1=x[0]+" wights "+str(weight)+" kg."
                    all_fish.append(st)
                    basket.append(st1)
                    b_w+=weight
                else:

                    st=x[0]+" wights "+str(weight)+" kg - released."
                    all_fish.append(st)
```

Fig 8: fishing.py

Start_fishing () Function :

Finally, I developed a method called **start_fishing()** that similates fishing based on weight and good eating fish of different species. It randomly chooses any fish species and generates a weight according to the defined range. Lastly, if the fish **'is_good_eating'** , then the selected fish is added to basket as a fish object. For this working, I utilize the **time.sleep(1)** function by importing time to hold function execution for 1 second. Similarly, I imported all necessary libraries such as **pickle, random and matplotlib** which will be used in this function.

```
import pickle
import fish_species  #imported the module in the same folder
import random
import time
from matplotlib import pyplot as plt
```

Fig 9: importing libraries

```
def start_fishing() :
  print('Fishing started')
  choosenFish = fish_categories[choose_category()]
  if choosenFish == 'Australian Bass' :
    fish = AustralianBass(randrange(0, AustralianBass.MAX_WEIGHT))
  elif choosenFish == 'Short Finned Eel' :
    fish = ShortFinnedEel(randrange(0, ShortFinnedEel.MAX_WEIGHT))
  elif choosenFish == 'Eel Tailed Catfish' :
    fish = EelTailedCatfish(randrange(0, EelTailedCatfish.MAX_WEIGHT))
  elif choosenFish == 'Gippsland Perch' :
    fish = GippslandPerch(randrange(0, GippslandPerch.MAX_WEIGHT))
  elif choosenFish == 'Yellowtail Amberjack' :
    fish = YellowtailAmberJack(randrange(0, YellowtailAmberJack.MAX_WEIGHT))
  elif choosenFish == 'Barramundi' :
    fish = Barramundi(randrange(0, Barramundi.MAX_WEIGHT))
  else :
    return

  if fish.is_good_eating() :
    basket["fishes"].append(fish)
    basket["total_weight"] += fish.weight
    basket["categorized_fishes"][fish.NAME] += fish.weight
    print('{} - added to the basket'.format(fish))
    time.sleep(1)
  else :
    print('{} - released'.format(fish))
    time.sleep(1)
```

Fig 10. Start_fishing function

Function print_basket :

I created a **print_basket** function to print the fishes present in the basket. Once the fishing session ends, all the fishing contents are printed using this function.

```
def print_basket(basket):
    print("Content of the basket:")
    for item in basket:
        print("\t\t",item)
```

Fig 11: print_basket function

Function plot_basket :

I built a **plot_basket** function that import the pyplot library from matplotlib that is used to plotting the graph. The below image demonstrates the plot basket function that is used for visualization of fish simulation.

```
def plot_basket(basket):
    label=["Australian ","Finned","EelTailed", "Gippsland","Branzino","Tilapia"]
    wt=[0,0,0,0,0,0]
    for j in basket:
        if "Australian" in j:
            x = j.split(' ')
            wt[0]+=float(x[4])
        if "Short" in j:
            x = j.split(' ')
            #print(x)
            wt[1]+=float(x[3])
        if "Catfish" in j:
            x = j.split(' ')
            wt[2]+=float(x[3])
        if "Gippsland" in j:
            x = j.split(' ')
            wt[3]+=float(x[2])
        if "Branzino" in j:
            x = j.split(' ')
            wt[4]+=float(x[3])

        if "Tilapia" in j:
            x = j.split(' ')
            wt[5]+=float(x[3])
    plt.figure()
    plt.bar(label,wt)
    plt.ylabel("Weights in kg")
    plt.show

def save_basket(basket, file_name):
```

Fig 12. Plot_basket function

Function save_basket :

The figure below shows how I have created save_basket function which is used to save the good eating fish objects. For this function to work, I used the pickle library.

```
def save_basket(basket, file_name):
    basketfile = open(file_name, 'basket')

    # source, destination
    pickle.dump(basket, basketfile)
    basketfile.close()
```

Fig 13. save basket function

Function load_basket :

The figure below shows how I have created load_basket function which is used to save the good eating fish objects. For this function to work, I used the pickle library.

```
def load_basket(file_name):
    basketfile = open(file_name,'basket')
    db = pickle.load(basketfile)
    for i in db:
        print(i)
    basketfile.close()
```

Fig 14. Load basket function

Output:

The figures below are the final out of the fishing simulation which shows the fishing process and data visualization in bar graph.

```
In [107]: runfile('C:/Users/rlams/OneDrive/Desktop/python/fishing.py.py',
wdir='C:/Users/rlams/OneDrive/Desktop/python')
Reloaded modules: fish_species
Fishing Started!
GippslandPerch(Percichthyidae) wights 3.612 kg - released.
Tilapia(Oreochromis niloticus) wights 3.028 kg - released.
Tilapia(Oreochromis niloticus) wights 0.884 kg - added to the basket.
Branzino(Dicentrarchus labrax) wights 1.714 kg - released.
Australian Bass(Macquaria Novemaculeata) wights 2.036 kg - added to the basket.
Tilapia(Oreochromis niloticus) wights 2.041 kg - released.
GippslandPerch(Percichthyidae) wights 1.537 kg - added to the basket.
Branzino(Dicentrarchus labrax) wights 1.012 kg - released.
ShortFinnedEel(Anguilla australis) wights 1.037 kg - added to the basket.
Tilapia(Oreochromis niloticus) wights 3.806 kg - released.
Branzino(Dicentrarchus labrax) wights 0.636 kg - added to the basket.
Tilapia(Oreochromis niloticus) wights 2.959 kg - released.
EelTailedCatfish(Tandanus tandanus) wights 1.112 kg - added to the basket.
Tilapia(Oreochromis niloticus) wights 2.87 kg - released.
Tilapia(Oreochromis niloticus) wights 0.769 kg - added to the basket.
GippslandPerch(Percichthyidae) wights 2.539 kg - released.
EelTailedCatfish(Tandanus tandanus) wights 2.361 kg - added to the basket.
Branzino(Dicentrarchus labrax) wights 1.111 kg - released.
ShortFinnedEel(Anguilla australis) wights 2.23 kg - released.
GippslandPerch(Percichthyidae) wights 2.789 kg - released.
Australian Bass(Macquaria Novemaculeata) wights 2.239 kg - added to the basket.
Australian Bass(Macquaria Novemaculeata) wights 1.784 kg - added to the basket.
EelTailedCatfish(Tandanus tandanus) wights 2.936 kg - added to the basket.
EelTailedCatfish(Tandanus tandanus) wights 1.504 kg - added to the basket.
EelTailedCatfish(Tandanus tandanus) wights 2.325 kg - added to the basket.
EelTailedCatfish(Tandanus tandanus) wights 1.617 kg - added to the basket.
Branzino(Dicentrarchus labrax) wights 0.539 kg - added to the basket.
Tilapia(Oreochromis niloticus) wights 3.983 kg - released.
Australian Bass(Macquaria Novemaculeata) wights 2.905 kg - released.
GippslandPerch(Percichthyidae) wights 3.438 kg - released.
GippslandPerch(Percichthyidae) wights 1.752 kg - added to the basket.
Basket is full. End of fishing session.
```

Fig 18 : Output of start_fishing

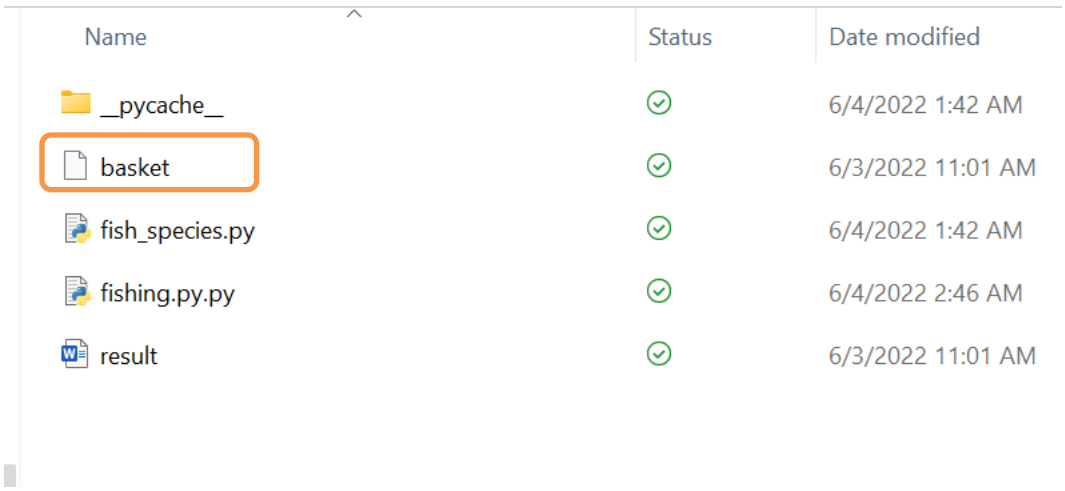

Fig 20: File creation

After the fish species are added to the basket, the plot_basket function plots the data graphically as shown below:

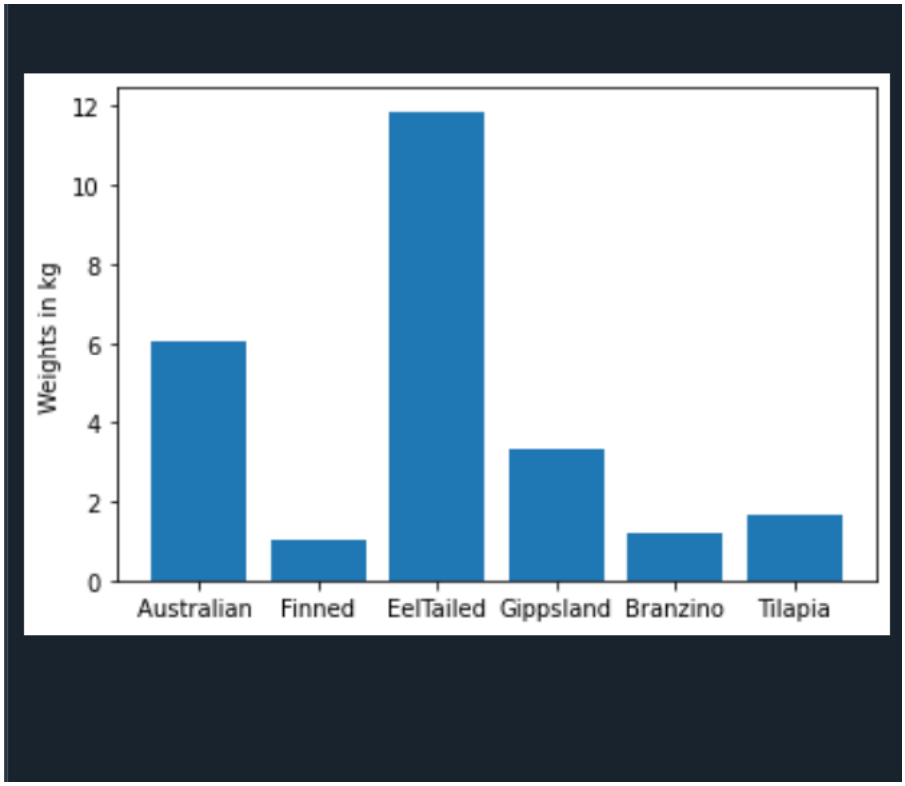

Fig 21: Bar graph of fishes in the basket