Operator Functions Set 1 & Set 2
Data Types
Introduction
Arrays Set 1, Set 2
String Methods Set 1, Set 2, Set 3
String Template Class & String Formatting using %
List Methods Set 1, Set 2, Set 3
Tuples & Sets
Dictionary Methods Set 1, Set 2
ChainMap
Explore More...
Control Flow
Loops and Control Statements
Counters & Accessing Counters
Iterators & Iterator Functions Set 1, Set 2
Generators
Explore More...
Functions
Function Decorators
Returning Multiple Values
Yield instead of Return
Python Closures & Coroutine
Explore More...
Modules
Introduction

Numeric Functions & Logarithmic and Power functions
Calender Functions Set 1, Set 2
Complex Numbers Introduction & Important functions
Explore More...
Object Oriented Concepts
Class, Object and Members
Data Hiding and Object Printing
Inheritance, Subclass and super
Class method vs static method & Class or Static Variables
Explore More...
Exception Handling
Exception Handling
User-Defined Exceptions
Built-in Exceptions
Libraries and Functions
Timeit
Numpy Set 1, Set 2
Get and Post
import module & reload module
Collection Modules Deque, Namedtuple & Heap
Explore More...
Machine Learning with Python
Classifying data using Support Vector Machines(SVMs) in Python

K means Clustering
How to get synonyms/antonyms from NLTK WordNet in Python?
Explore More...
Misc
Sql using Python & MongoDB and Python
Json formatting & Python Virtual environment
Metaprogramming with Metaclasses in Python
Python Input Methods for Competitive Programming
Explore More...
Applications and Projects
Creating a proxy webserver Set 1, Set 2
Send Messsage to FB friend
Twitter Sentiment Analysis & Whatsapp using Python
Desktop Notifier & Junk File Organizer
Explore More...

Understanding Code Reuse and Modularity in Python 3

What is Object Oriented Programming(OOP)?

OOP is a programming paradigm based on the concept of “objects”, which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.Learn more here, or just Google “OOP”.

Objects have characteristics and features, known as attributes, and can do various things, through their methods. The biggest feature of OOP is how well objects can be interacted with, and even molded in the future, which makes them very friendly to developers, scale, change over time, testing, and much more.

What is Modularity?

Modularity refers to the concept of making multiple modules first and then linking and combining them to form a complete system (i.e, the extent to which a software/Web application may be divided into smaller modules is called modularity). Modularity enables re-usability and will minimize duplication.

Flow of the Article

Aim: To learn object oriented programming – Modularity. How can we turn some portions of our code into a library so that it can be used by anyone for future reference. Making the code modular will enable re-usability and minimizes duplication.

Dependencies: pygame

Summary: We are going to make a small game(not really a game) but just an environment and some objects in it. We will try to make the environment static and the objects(blobs in our case) modular. We will make use of PyGame, since it gives us a simple way to actually visualize what we're doing and building, so we can see our objects in action. What we're going to do is build Blob World, which is a world that consists of actors, known as blobs. Different blobs have different properties, and the blobs need to otherwise function within their Blob World environment. With this example, we'll be able to illustrate modularity.
We are dividing our learning process into two phases.

1. Creating the Environment and the Blobs
2. Understanding Modularity

Repository(Github): source

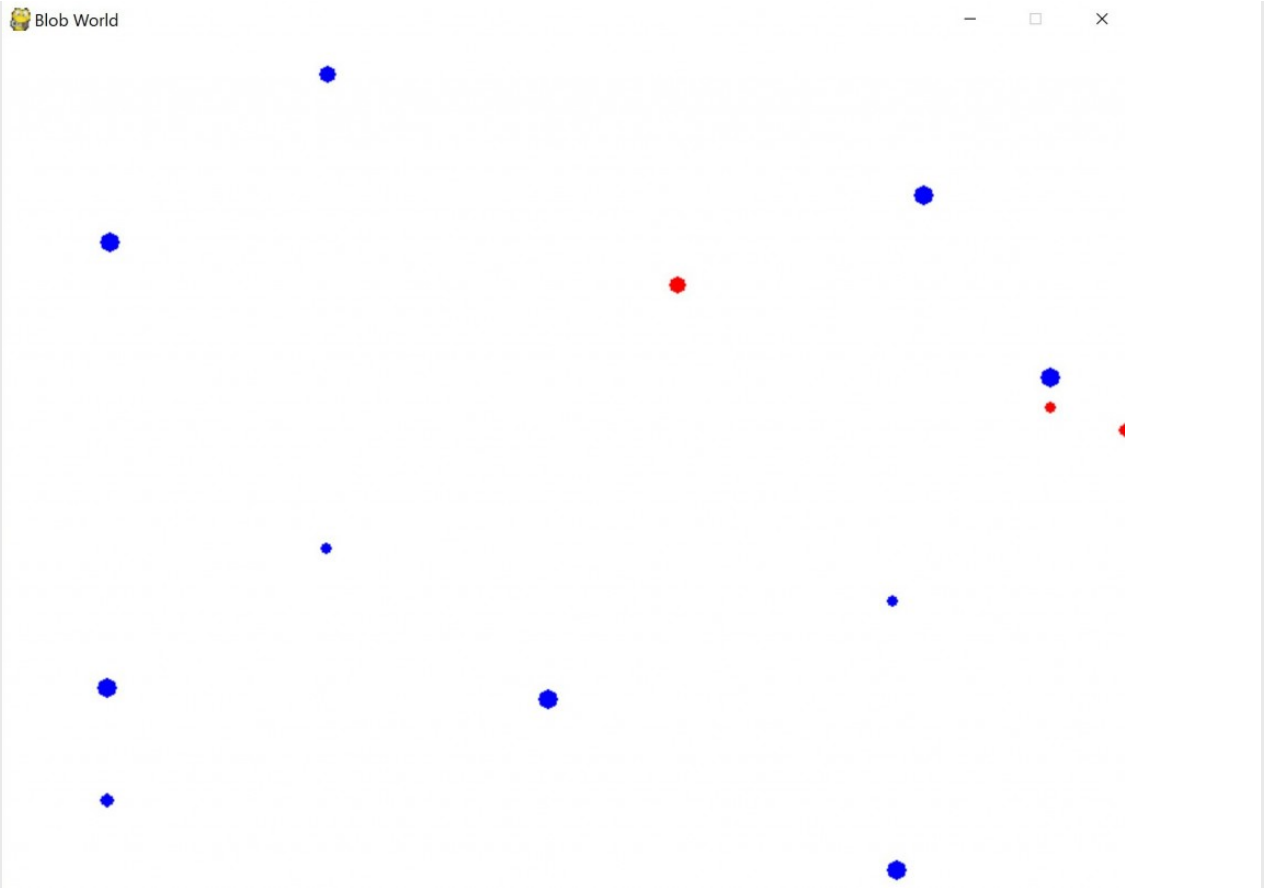
BLOB WORLD (Python Code)

```
import pygame
import random

STARTING_BLUE_BLOBS = 10
STARTING_RED_BLOBS = 3

WIDTH = 800
HEIGHT = 600
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
RED = (255, 0, 0)

game_display = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Blob World")
clock = pygame.time.Clock()

class Blob:

    def __init__(self, color):
        self.x = random.randrange(0, WIDTH)
        self.y = random.randrange(0, HEIGHT)
        self.size = random.randrange(4,8)
        self.color = color

    def move(self):
        self.move_x = random.randrange(-1,2)
        self.move_y = random.randrange(-1,2)
        self.x += self.move_x
        self.y += self.move_y
```

```
        if self.x < 0: self.x = 0
        elif self.x > WIDTH: self.x = WIDTH

        if self.y < 0: self.y = 0
        elif self.y > HEIGHT: self.y = HEIGHT


def draw_environment(blob_list):
    game_display.fill(WHITE)

    for blob_dict in blob_list:
        for blob_id in blob_dict:
            blob = blob_dict[blob_id]
            pygame.draw.circle(game_display, blob.color, [blob.x, blob.y], blob.size)
            blob.move()

    pygame.display.update()


def main():
    blue_blobs = dict(enumerate([Blob(BLUE) for i in range(STARTING_BLUE_BLOBS)]))
    red_blobs = dict(enumerate([Blob(RED) for i in range(STARTING_RED_BLOBS)]))
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
        draw_environment([blue_blobs,red_blobs])
        clock.tick(60)

if __name__ == '__main__':
    main()
```

Run on IDE

Output:



PART(1/2): Blob World In this part, we are creating a simple game environment and some objects in it because visualizing what we have created is an exceptional way of learning programming. The explanation for the creation of the blob world (i.e, its environment and its objects) using pygame is explained here. All we need to understand is how to make our code modular.

PART(2/2): Modularity In this second part, we are going to understand an essential feature of Object Oriented Programing, i.e Modularity. So far, we've not introduced anything that will make this (BLOB WORLD code) too hard to maintain or scale over time, at least within the scope of what we can do with PyGame. What about making it modular? There's a really easy test for this, let's try to import it!
To do this, let's have two files. Let's copy the Blob class and random, and make a new file: blob.py

```
import random

class Blob:

    def __init__(self, color):
        self.x = random.randrange(0, WIDTH)
        self.y = random.randrange(0, HEIGHT)
        self.size = random.randrange(4,8)
        self.color = color

    def move(self):
        self.move_x = random.randrange(-1,2)
```

```
        self.move_y = random.randrange(-1,2)
        self.x += self.move_x
        self.y += self.move_y

        if self.x  WIDTH: self.x = WIDTH

        if self.y  HEIGHT: self.y = HEIGHT
```

Back to our original file, let's remove the Blob class, and then import Blob from blob.py.

```
import pygame
import random
from blob import Blob

STARTING_BLUE_BLOBS = 10
...
```

Immediately, we're given an error in the blob.py file, regarding our Blob class, where we have some undefined variables. This is definitely a problem with writing classes, we should try to avoid using constants or variables outside of the class. Let's add these values to the __init__ method, then modify all of the parts where we used the constants.
So, here is our new Blob class file :blob.py
Next, within our original file, when we call the Blob class, it's expecting some values for those arguments, so you'd add those in the main function:

```
def main():
    blue_blobs = dict(enumerate([Blob(BLUE,WIDTH,HEIGHT) for i in range(STARTING_BLUE_BLOBS)]))
    red_blobs = dict(enumerate([Blob(RED,WIDTH,HEIGHT) for i in range(STARTING_RED_BLOBS)]))
    while True:
        ...
```

Great, so now our Blob class can at least be imported, so it's modular by nature already! Another good idea is to attempt to give the developer that is using your code as much power as possible, and to make your class as generalize-able as possible. At least one example where we could definitely give more to the programmer using this class is in the definition of the blob's size:

```
  self.size = random.randrange(4,8)
```

Is there any reason why we wouldn't want to give the programmer an easy way to change these? I don't think so. Unlike x_boundary and y_boundary, however, we do not necessarily *need* the programmer to provide us a value for the size, since we can at least use a reasonable starting default. Thus, we can do something like:

```
class Blob:

    def __init__(self, color, x_boundary, y_boundary, size_range=(4,8)):
        self.x_boundary = x_boundary
        self.y_boundary = y_boundary
```

```
        self.x = random.randrange(0, self.x_boundary)
        self.y = random.randrange(0, self.y_boundary)
        self.size = random.randrange(size_range[0],size_range[1])
        self.color = color
```

Now, if the programmer wants to change the size, they can, otherwise they don't have to. We might also want to allow the programmer to modify the speed of the blob if they want to:

```
import random


class Blob:

    def __init__(self, color, x_boundary, y_boundary, size_range=(4,8), movement_range=(-1,2)):
        self.size = random.randrange(size_range[0],size_range[1])
        self.color = color
        self.x_boundary = x_boundary
        self.y_boundary = y_boundary
        self.x = random.randrange(0, self.x_boundary)
        self.y = random.randrange(0, self.y_boundary)
        self.movement_range = movement_range

    def move(self):
        self.move_x = random.randrange(self.movement_range[0],self.movement_range[1])
        self.move_y = random.randrange(self.movement_range[0],self.movement_range[1])
        self.x += self.move_x
        self.y += self.move_y

        if self.x  self.x_boundary: self.x = self.x_boundary

        if self.y  self.y_boundary: self.y = self.y_boundary
```

Now we've opened up the class quite a bit. Does anything else jump out at us? Yes, the line where we force the blob to remain in-bounds. Might there be examples where we'd like the blobs to be able to roam freely out of view? Certainly! Is this bounding code useful though? Is it likely that programmers will want to make use of this quite often? Certainly! however, that it makes more sense to either not have the code at all, or to give it its own method, like so:

```
import random

class Blob:

    def __init__(self, color, x_boundary, y_boundary, size_range=(4,8), movement_range=(-1,2)):
        self.size = random.randrange(size_range[0],size_range[1])
        self.color = color
        self.x_boundary = x_boundary
        self.y_boundary = y_boundary
        self.x = random.randrange(0, self.x_boundary)
        self.y = random.randrange(0, self.y_boundary)
        self.movement_range = movement_range

    def move(self):
        self.move_x = random.randrange(self.movement_range[0],self.movement_range[1])
```

```
        self.move_y = random.randrange(self.movement_range[0],self.movement_range[1])
        self.x += self.move_x
        self.y += self.move_y

    def check_bounds(self):
        if self.x  self.x_boundary: self.x = self.x_boundary

        if self.y  self.y_boundary: self.y = self.y_boundary
```

Now, the programmer can decide to use it or not. You could also give some sort of argument in the move method, where, if True, then boundaries would be enforced.
So we got a glimpse of how we can make our python code Modular.

Resources:

- Original Video Series (by pythonprogramming.net)
- pythonprogramming.net
- Harrison Kinsley (Thank you H.Kinsley)

This article is contributed by **Amartya Ranjan Saikia**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

Tracking bird migration using Python-3

OpenCV Python Program to analyze an image using Histogram
Comparing Intel i3, i5 and i7 processors
Quickhull Algorithm for Convex Hull
Difference between various Implementations of Python

(Login to Rate and Mark)

2 Average Difficulty : **2/5.0**
Based on **1** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!