```
""" idea.py
    simplest possible pygame display
    demonstrates IDEA / ALTER model
    Andy Harris, 5/06
    """

#I - Import and initialize
import pygame
pygame.init()

#D - Display configuration
screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("Hello, world!")

#E - Entities (just background for now)
background = pygame.Surface(screen.get_size())
background = background.convert()
background.fill((0, 0, 255))

#A - Action (broken into ALTER steps)

    #A - Assign values to key variables
clock = pygame.time.Clock()
keepGoing = True

    #L - Set up main loop
while keepGoing:

    #T - Timer to set frame rate
    clock.tick(30)

    #E - Event handling
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            keepGoing = False

    #R - Refresh display
    screen.blit(background, (0, 0))
    pygame.display.flip()
```

```
""" moveBox.py
    illustrates basic motion
    in the IDEA/ALTER framework
    moves a rect across the screen """

#Initialize
import pygame
pygame.init()

#Display
screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption("move a box")

#Entities
#yellow background
background = pygame.Surface(screen.get_size())
background = background.convert()
background.fill((255, 255, 0))

#make a red 25 x 25 box
box = pygame.Surface((25, 25))
box = box.convert()
box.fill((255, 0, 0))

# set up some box variables
box_x = 0
box_y = 200

#ACTION

    #Assign
clock = pygame.time.Clock()
keepGoing = True

    #Loop
while keepGoing:

    #Time
    clock.tick(30)

    #Events
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            keepGoing = False

    #modify box value
    box_x += 5
    #check boundaries
    if box_x > screen.get_width():
        box_x = 0

    #Refresh screen
    screen.blit(background, (0, 0))
```

```
screen.blit(box, (box_x, box_y))
pygame.display.flip()
```

```
""" basicSprite.py
    works just like moveBox.py
    from chapter 4,
    but now uses a sprite """

import pygame
pygame.init()

screen = pygame.display.set_mode((640, 480))

class Box(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((25, 25))
        self.image = self.image.convert()
        self.image.fill((255, 0, 0))
        self.rect = self.image.get_rect()
        self.rect.centerx = 0
        self.rect.centery = 200
        self.dx = 10
        self.dy = 0

    def update(self):
        self.rect.centerx += self.dx
        if self.rect.right > screen.get_width():
            self.rect.left = 0

def main():
    pygame.display.set_caption("basic sprite demo")

    background = pygame.Surface(screen.get_size())
    background = background.convert()
    background.fill((255, 255, 0))
    screen.blit(background, (0,0))

    box = Box()
    allSprites = pygame.sprite.Group(box)

    clock = pygame.time.Clock()
    keepGoing = True
    while keepGoing:
        clock.tick(30)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                keepGoing = False

        allSprites.clear(screen, background)
        allSprites.update()
        allSprites.draw(screen)

        pygame.display.flip()

if __name__ == "__main__":
```

main()

```
""" boxes.py
    demonstrate multiple boxes,
    adding parameters """

import pygame, random
pygame.init()

screen = pygame.display.set_mode((640, 480))

class Square(pygame.sprite.Sprite):
    """ makes a box with a random starting
        position and the given color.
        To make a red square, use
        redSquare = Square((255, 0, 0))

        requires screen be predefined and import random """

    def __init__(self, color):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((50, 50))
        self.image.fill(color)
        self.rect = self.image.get_rect()
        self.rect.centerx = random.randrange(0, screen.get_width())
        self.rect.centery = random.randrange(0, screen.get_height())

def main():
    pygame.display.set_caption("lotsa boxes")

    background = pygame.Surface(screen.get_size())
    background.fill((255, 255, 255))
    screen.blit(background, (0, 0))

    boxes = []
    for colorName in pygame.color.THECOLORS:
        boxes.append(Square(pygame.color.Color(colorName)))

    allSprites = pygame.sprite.Group(boxes)

    keepGoing = True
    clock = pygame.time.Clock()
    while (keepGoing):
        clock.tick(30)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                keepGoing = False

        allSprites.clear(screen, background)
        allSprites.update()
        allSprites.draw(screen)

        pygame.display.flip()

if __name__ == "__main__":
```

main()

```
""" collisionObjects.py
    A class library of objects used in the collision demos
    in chapter 6"""

import pygame, random

class Square(pygame.sprite.Sprite):
    """ makes a box with a random starting
        position and the given color.
        To make a red square, use
        redSquare = Square((255, 0, 0))

        requires screen be predefined and import random """

    def __init__(self, color, screen):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((50, 50))
        self.image.fill(color)
        self.rect = self.image.get_rect()
        self.rect.centerx = random.randrange(0, screen.get_width())
        self.rect.centery = random.randrange(0, screen.get_height())

class Circle(pygame.sprite.Sprite):
    """ makes a blue circle that
        follows the mouse. """

    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((50, 50))
        self.image.fill((255, 255, 255))
        pygame.draw.circle(self.image, (0, 0, 255), (25, 25), 25, 0)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.center = pygame.mouse.get_pos()

class Label(pygame.sprite.Sprite):
    """ Label Class (simplest version)
        Attributes:
            font: any pygame font object
            text: text to display
            center: desired position of label center (x, y)
    """
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.font = pygame.font.SysFont("None", 30)
        self.text = ""
        self.center = (320, 240)

    def update(self):
        self.image = self.font.render(self.text, 1, (0, 0, 0))
        self.rect = self.image.get_rect()
        self.rect.center = self.center
```