Random numbers

- see the `random` module for details and extent of creating random numbers

Random numbers

- the method `random.randint` can be used to generate random integers
- any easy example usage is:
- ```
  #!/usr/bin/env python

  import random

  print "your die roll was", random.randint (1, 7)
  ```


Sounds

- ```
  #!/usr/bin/env python

  sound = pygame.mixer.Sound ("mysound.wav")
  sound.play ()
  ```
- please see the pygame ⟨`https://www.pygame.org/docs/ref/mixer.html`⟩ web site for more detail

Adding Cities to your game

- the tutorial asked you to add cities and guns to your program
- ```
  class city:
      def __init__ (self, pos):
          self._pos = pos
          self._epicenter = [pos[0] + city_length/2, pos[1]
          self._exploding = False
          self._explosion = None
          self.draw_city ()
  ```
  ```

Adding Cities to your game

```
    def draw_city (self):
        pygame.draw.rect (screen, wood_light, (self._pos[
    def update (self):
        pass
    def ignite (self, p):
        return self._exploding
    def erase (self):
        pygame.draw.rect (screen, black, (self._pos[0], s
```


Adding Cities to your game

```
    def check (self, p, radius):
        if (not self._exploding) and sqr (radius) > sqr (
            self._exploding = True
            createExplosion (p, grey)
            createExplosion (self._epicenter, light_grey)
            globalRemove (self)
```


Using cities in your code

```
city_list = []
city_length =  90
city_height =  25
attack_number = 5
rate_of_attack = 400

def make_cities ():
    global city_list

    for p in [[100, 768-city_height], [200, 768-city_heig
              [600, 768-city_height], [700, 768-city_heig
        c = city (p)
        city_list += [c]
```


Using cities in your code

```
def check_cities_guns (pos, radius):
    for c in city_list:
        c.check (pos, radius)
```

Using cities in your code

```
def spawn_attack ():
    global attack_number
    if attack_number > 0:
        if random.randint (1, rate_of_attack) == 1:
            attack_number -= 1
            c = city_list [random.randint (0, 5)]
            createMissile ([random.randint (1, 1000), 0],
                           c._epicenter)
```


Using cities in your code

```
def no_of_cities ():
    n = 0
    for c in city_list:
        if not c._exploding:
            n += 1
    return n
```


Using cities in your code

```
def check_finished ():
    if attack_number == 0 and len (allObjects) == 0:
        n = no_of_cities ()
        if n == 0:
            print "you lost!"
        elif n == 1:
            print "you survived with 1 city left"
        else:
            print "you survived with", n, "cities left"
        sys.exit (0)
```


Using cities in your code

```
def main ():
    global screen
    pygame.init ()
    screen = pygame.display.set_mode ([width, height])
    make_cities ()
    wait_for_event ()
```

Guns

- guns in missile command are rather similar to cities
- we need another class, with similar methods
  - it might be possible to inherit code - but that is left for another day
-
```
ammo_per_silo = 20
gun_length = 90
gun_height = 25
gun_list = []
```


Guns

-
```
class gun:
    def __init__ (self, pos):
        global screen
        self._ammo = ammo_per_silo
        self._pos = pos
        self._epicenter = [pos[0] + gun_length/2, pos[1]-
        self._exploding = False
        self._explosion = None
        self.draw_gun ()
```


Guns

-
```
    def draw_gun (self):
        global screen
        print "rect", self._pos, gun_length, gun_height
        pygame.draw.rect (screen, dark_blue, (self._pos[0
    def fire (self):
        if self._ammo > 0 and (not self._exploding):
            self._ammo -= 1
            createMissile (self._epicenter, pygame.mouse.
```


Guns

-
```
    def update (self):
        pass
    def ignite (self, p):
        return self._exploding
    def erase (self):
        pygame.draw.rect (screen, black, (self._pos[0], s
    def check (self, p, radius):
        if (not self._exploding) and sqr (radius) > sqr (
            self._exploding = True
            createExplosion (p, grey)
            createExplosion (self._epicenter, light_grey)
            globalRemove (self)
```

Linking up guns to the game

```
def check_cities_guns (pos, radius):
    for c in city_list:
        c.check (pos, radius)
    for g in gun_list:
        g.check (pos, radius)
```


Linking up guns to the game

```
def spawn_attack ():
    global attack_number
    if attack_number > 0:
        if random.randint (1, rate_of_attack) == 1:
            attack_number -= 1
            c = city_list [random.randint (0, 5)]
            createMissile ([random.randint (1, 1000), 0],
                           c._epicenter)
```


Linking up guns to the game

```
        if random.randint (1, rate_of_attack) == 1:
            attack_number -= 1
            g = gun_list [random.randint (0, 2)]
            createMissile ([random.randint (1, 1000), 0],
                           g._epicenter)
```


Linking up guns to the game

```
def make_guns ():
    global gun_list
    for p in silos:
        g = gun (p)
        gun_list += [g]
```

Linking up guns to the game

```
def main ():
    global screen
    pygame.init ()
    screen = pygame.display.set_mode ([width, height])
    make_cities ()
    make_guns ()
    wait_for_event ()
```


Tutorial

- add this code to your game
- comment the code
- observe the similarity between guns and cities
- add scoring, sounds and features