CodeAdvantage®
Learn Create Master

Python
Animal Expedition – Lesson 1-2
Game Setup and Backend Design

Student Objectives

- Create the base game screen
- Create utility functions
- Setup backend design structure
  - Global control Variables
  - The 3 x 3 Grid design (gameDesign)
  - The animals per biome (biomes)

Concepts Covered

1. Backend Setup
2. Dictionaries
3. Functions
4. Global Variables
5. Turtle (Game) Screen

Today' s Activity

Today, we will set up the game screen and the backend mechanics (dictionaries) to keep track of the game map and animals to spawn there. We will also add the final 2 lines of code for the game and write all remaining code above it.

Classroom Discussion

Create the basic game screen and utility functions for random numbers (rnum) and the distance formula (distForm). Then, create the 2 dictionaries for backend management (grid design and animal population) as well as the global variables and the animal objects (list and for loop for turtle creation).

Class Activity- Guided Work

```
# GAME FILE
import os, sys, math, random, turtle

def rnum(n0,n1): return random.randint(n0,n1)
def distForm(a,b): return math.sqrt((a.xcor()-b.xcor())**2 + (a.ycor()-b.ycor())**2)

tpool = {}

gridX = 1
gridY = 1
current_biome = 'plains'

gameDesign = {
    '1,1':'plains',
    '1,2':'river',
    '1,3':'ocean',
    '2,1':'forest',
    '2,2':'rain_forest',
    '2,3':'glacier',
    '3,1':'mountain',
    '3,2':'valleys',
    '3,3':'desert'
}

biomes = {
    'ocean':['goldfish','shark'],
    'plains':['lion','hyenna'],
    'rain_forest':['sloth','fire_ant'],
    'river':['bass','salmon'],
    'glacier':['penguin','polar_bear'],
    'mountain':['goat','wolf'],
    'valleys':['lizard','komodo_dragon'],
    'desert':['vulture','rattle_snake'],
    'forest':['jackrabbit','jaguar']
}

current_animals = biomes['plains']
move_animals = False
animal_house = []
```

```
s = turtle.Screen()
s.bgcolor('black')
s.title('Animal Expedition')
s.screensize(1000,500); s.setup(1050,550)

for biome, animals in biomes.items():

    for a in animals:
        tpool[a] = turtle.Turtle()
        tpool[a].hideturtle()
        tpool[a].penup()
        tpool[a].shapesize(rnum(1,5),rnum(1,5))
        tpool[a].setpos(rnum(-300,300),rnum(-200,200))


        # Tie animal shapes into the game
        s.register_shape(f"{a}.gif")
        tpool[a].shape(f"{a}.gif")

# Last 2 Lines
s.listen()
s.mainloop()
```

1. Import needed resources for the game at first, which should always be at the top of the script. Then, define the 2 utility functions rnum() (a wrapper for random.randint() – for less character typing) and distForm() (for determining the distance between 2 turtle objects).
2. Create the tpool dictionary, which will be used to hold the player and animal turtle objects, by name for easy referencing (which also allows any number of animals to be created for the game in the future).
3. The gameDesign dictionary will hold the 9 biome types, named by the grid coordinates 1,1 through 3,3. These biomes will be what the player can travel to/from and be used to select which animal types to create. The grid numbers (as keys) will be used to reference the proper information an background images as well during runtime.
4. The biomes dictionary will hold each biome' s respective animals (in a list, by name, as many as desired) that will be showing and active while the player is in that biome. Also create global variables for the current animals to show, whether or not the animals are allowed to move and a list to hold the current animals to move.
5. Create the game screen and populate the tpool dictionary with animals algorithmically. **NOTE: Until all images are created (for the animals), have the students comment out the code for registering and setting the images, and replace it with a simple >> tpool[a].shape(' turtle') << call. This will be changed when all images can be supplied by the last class.
6. Also, add the final 2 lines of code for button/click binding and keeping the script alive. NOTE: Image creation for animals and background images should be made each day (and for students in their free time if they wish) so that all images are ready by the last class so the complete game can be finished.

Independent Play/ Game Customization

Allow your students to enjoy their new game! Give them the last 10 minutes of class to play their game, or explore what they can do to customize their world. Celebrate creativity!

Building Relationships

Share what your students can do! Take screenshots of your student' s work, or pictures of them (absolutely no faces please!) with their computer screens, and use the handle @CodeAdvantage and the handle #CodeAdvantage to let others see and be inspired by your students' success with coding!

(Tweet or Instagram example: " My students completed @CodeAdvantege #Lets Float Around today! We designed and coded games! I' m so proud of them! My students rock!)

Consider the plans for your next lesson and start preparing for how the kids are going to have a great time with @CodeAdvantage creating #(next lesson name)!

We value your opinion! Please send any lesson suggestions, edits, comments, or questions to feedback@codeadvantage.org.